Developing a System to Help Programmers Achieve a Good Coding Style

Haijie Zhao

D05103686

Haijie.zhao1@student.dit.ie

A dissertation submitted in partial fulfillment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Information Technology)

September 2009

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Information Technology), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____

Date: _____

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ABSTRACT

Software developers tend to believe that coding standards are important in programming but do not tend to comply with them, thus we need some new strategies for the software developers to keep them focusing on the style of the code as well as the development of the application.

This research looks at some common challenges to coding styles are facing and also a possible solution to help programmers to achieve good coding styles. Simply, the solution is based on two main factors, improving the existing coding standards and developing a plug-in in develop toolkit to help programmers following the standard.

This research takes Java for example and detailed analysis the current coding standards. Based on the research of existing coding standards, a new coding standard is developed. The new coding standard can be transformed into a configuration file used by the development toolkit. To achieve this, a new plug-in called Code-Check is developed and it helps the developers to gain that new coding standard and thus helps them to write better source code.

Keywords: coding style, coding standard, programmer habit, programming style, programming standard

ACKNOWLEDGEMENTS

Foremost, I am deeply indebted to my supervisor, Damian Gordon of the School of Computing, Dublin Institute of Technology for many insightful conversations during the development of ideas of this thesis. Some stimulating suggestions, constructive criticisms and encouragement helped me in all the time of research.

I would also like to thank Bryan Duggan for his advice and feedback on my thesis.

I would also like to express my gratitude to friends who give advice in the project.

Finally I would like to thank the developers from CSDN developer forum for participating in the survey and offering feedback for the generated Java coding standard.

1. INTRODUCTION

1.1 PROJECT BACKGROUND

Coding standards refers to a set of rules or guidelines used when writing source code for a computer program. The idea is that all code is written in the same style and that can help a software developer to understand the code better.

> *"Coding standards are important because they lead to greater consistency within your code and code of your team mates"* [1].

With the help of a good coding style, the code can show better consistency. Greater consistency makes the code easier to understand and to follow. It also means it is easier to develop and maintain the source code which will reduce the overall cost of the application that the developers have created.

Sometimes, code may be used for a long time with bad coding style and that makes it hard to maintain. An important goal during the development process is to ensure that developers can transition their work to another developer or another team of developers, so that they can continue to maintain and enhance it without having any problems in understanding the code the former developer writes. Good coding style can help programmers greatly to maintain and understand without introducing any inconvenience or errors.

Following a particular coding style when creating an application will help the programmers to read and understand source code conforming to the style, and help to avoid introducing errors, besides writing code with a good style can also provide the following extra benefits. [2]

1. It improves the readability, consistency, and homogeneity of the code, which makes it easier to understand and maintain.
2. It makes the code easier to trace and debug, because it's clear and consistent.
3. It allows the programmers to continue more easily where some other programmers stopped working on the code, particularly after a long period of time.
4. It increases the benefit of a peer code review, because the participants can focus more on what the code is doing.

The benefits listed above clearly shows that write out code with good style helps ideas and the thoughts expresses exceptionally clear.

1.2 CONTRIBUTION OF THIS PROJECT

Almost all programmers agree that coding standards are important; however, not all of them can follow coding standards when they develop applications. It also takes people to make extra effort to gain the code with nice and strict coding standards.

There can be many reasons like coding standards hard to follow or developer's bad coding habit. There are even some problems with the language standards. For example there are several versions of C++ language standards and there are some different standards even for one definition. Programmers are confused with those standards. Thus a new common coding style can help with this.

Take the Java programming language for example. There are many resources that are talking about Java coding standards however most of these resources are books and articles. When the programmers who want to learn a new programming language they need to do the approaches as showing below:

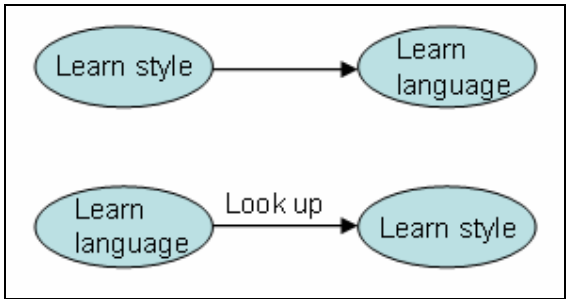


FIGURE 1: APPROACHES TO LEARN CODING STANDARDS

The first approaches is the best way, however few developers and students can learn all the coding style before they learn other things in this programming language.

Most people may have the experience with the second approaches. The traditionally process of this approach is to look up programming style from a book or search the rules on internet, and the developers can learn, remember and follow the coding standard when they are writing source code. In this paper, we will focus on this approach and improve the learning-approach through the new strategy.

This project is tending to help to solve this problem. The key of the solution is to work out a simple useful coding standard and an auto-check tool that can be used by programmers. When the developers write source code the new coding standard and the tools could help to improve the code quality.

This project focuses on Java programming language. The code style analyses will be done based on Java language and presents a new plug-in (Coding standard checker) and configuration files which can help the developers to analyze their source code and make improvements.

1.3 PROJECT DESCRIPTION

Most developers believe that coding standards are very important but they do not really tend to comply with them. Here are several reasons why people don't comply with coding standards:

1. It's hard to remember all the programming style.
2. It's very easy to confuse standard for the programmers who can use several programming languages.
3. Problems with the coding editors.
4. It takes a long time to comply with a coding standard for a short program.

Many developers have the problem mentioned above. However it's not easy to resolve those problems for a programmer without any external help.

Professor Xiaosong Li and Christine Prasad from New Zealand have present effectively teaching coding standards in programming. They list current strategies that we use for teaching coding standards, and evaluate them for effectiveness. They found most of the strategies need the students to remember the coding standard initiatively.[3] This means the programmer not only need to learn programming language but also need to take a long time to remember complex coding styles. This is not effectively enough and also that's a bad news for most of the programmers. Professor Xiaosong Li and Christine Prasad also propose strategies that are likely to be effective in teaching coding standards and present suggestions for further studies. Most of their strategies are very helpful for the students of information technology but also need take a long time for the people who want to learn programming.

Robert Cecil Martin [4] also did a lot of work in the coding standard area. He found some particular programming styles that can not only reduce the cost of the application but also greatly improve the efficiency of the program.

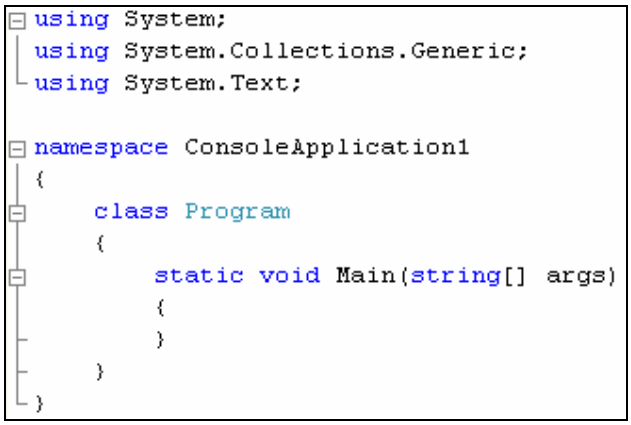
Coding standards can reduce the cost of software maintenance is the most often cited reason. In the introduction to code conventions for the Java Programming Language, Sun Microsystems provides the following rationale: [5]

1. 80% of the lifetime cost of a piece of software goes to maintenance.
2. Hardly any software is maintained for its whole life by the original author.
3. Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

From the results above we can presume when most of the programmers can handle those coding standard, there will be a great improvement in information technology.

Microsoft also did some research in coding standard area. They published a lot of guidelines for DotNet developers which can greatly help the DotNet developers to deeply understand DotNet technology. Microsoft also implemented some particular

coding standard functions in their products. For example, Visual Studios DotNet 2007 can handle some common the coding styles by itself. This makes most of the DotNet application have the similar coding standard which make the DotNet application easier to understand and greatly reduce the maintain cost.



```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

FIGURE 2: C# DOTNET CODING HANDLER

This function is a huge step in improving software development. DotNet software developers don't need to pay much attention on the coding standard. They can apply themselves to other parts of the application. Coding style then takes less time.

The research of this paper is based on existing coding standard but not fully carrying it. The consideration will take place for the ease of use and also the efficiency of the current coding standard and attempt to get a balanced coding style.
For example:

```
int getMaxIterations()
int get_max_interations()
```

These two definitions both are allowed according to the C++ coding standard, however most of the programmers suggest using the first definition as it's easy to read. So the new coding style for C++ will choose the first one and suggest the user to use the first definitions

The technology used in Visual Studio DotNet has not yet refined. The technology will be improved in the new coding standard. For example:

1. Provide several different coding structures for the developments which may improve the performance.
2. Provide different naming structures which are commonly used and easy to read.

All of these functions are in the implementation of the new coding style.

Most of those functions are improvements for the coding style, developers don't need to remember so many coding style rules, the rules in the plug-in will remind or hint the developers and the developers can pick up those rules when they use their develop toolkit with this plug-in.

The idea in this project is much different from that in most of software like Visual Studios DotNet. The implementation of the new coding style not only helps the developer to deal with the coding styles it also help the developer to remember the rules. So the developer can handle these rules well without the plug-in in further developing works.

1.4 RESEARCH AIM AND OBJECTIVES

1.4.1 PROJECT AIM

This project is aimed at helping programmers to achieve a better coding style. This is achieved by 2 main part of work. First, a large amount of research and comparison will be required to work out a coding style that can be generally accepted by most programmers. Second, a hint/reminder plug-in (Coding-checker) will be implemented to examine the coding style within the develop toolkit as a guide to make the programmer use the style in coding.

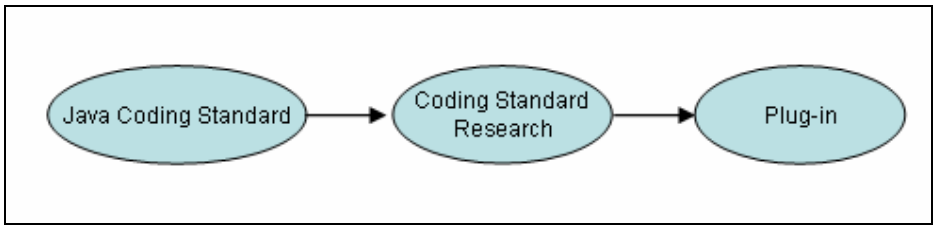


FIGURE 3: PROJECT IMPLEMENTATION

1.4.2 PROJECT OBJECTIVES

- To research existing Java coding standards, analyze those coding standards from the elements of coding standards. Generate a new version of the Java coding standards, all of those rules should be based on the existing valid Java coding standards. Developers should easily pick up rules.

- To design and develop a coding-checker (Eclipse plug-in), the coding-checker follows the rules generated from above. This coding-checker is a lightweight,

cross-platform Eclipse based plug-in which can give warning and advice when the developer writes source code.

- Do evaluations for the new Java coding standard and the coding-checker.

In order to achieve the aim, the following approach will be taken:

1. Gathering coding styles for Java
2. Compare different Java coding standards
3. Work out a possible coding style which can be commonly used.
   a) Easy to form a habit for the developer.
   b) Help the developer pick up the code quickly.
4. Work out some particular coding style which can improve the performance of the application.
5. Get the coding standard transformed to something that can be implemented in the development toolkit.
6. Craft a paper based on the research and implementation

1.5 INTELLECTUAL CHALLENGE

In this section the intellectual challenge of the dissertation is discussed on these challenges: technical, scientific, evaluation and personal challenges.

TECHNICAL CHALLENGE

The technical challenge here is that it requires to a mastery of the Java programming language, to setup and test the coding-checker and finally to deploy the application. It also requires well known Java coding standards to be gathered, analyzed and then generates a new version of Java coding standard.

SCIENTIFIC CHALLENGE

There are many scientific articles and books dealing with coding standards and particularly in the Java coding standards, however dealing which part of a specified programming standard to use is a significant challenge. Most people learn Java coding standards from the official web site of Sun, and nearly all the Java coding standards articles are talking about Sun version standards, so this dissertation will contribute to the scientific literature on Java coding standards.

EVALUATION CHALLENGE

The existing coding standards have supplied an efficient way of programming, however, there are not easy to follow or showing great usability in the real world coding. This brings additional difficult for coding when there is no way to force the programmers to follow them strictly. So most of the developers know it is very hard

to follow a new coding standard and it is hard to find many experienced developers who would like to evaluate the coding-checker and programming standards.

PERSONAL CHALLENGE

The analysis of coding standards and forming a new easy-to-follow coding style is a challenge and what makes this project attractive is that it is looking for a balancing point between better standardized and easier to follow styles.

The research in this part requires large amount of work in analysis, discussing and judging and making decision. And then the final version of coding style also needs the plug-in to help to become real and useable in the development. This is the opportunity to get coding standards knowledge and combine it with Java application development.

The research on coding style for this dissertation plans several areas.

- Explore the importance of coding standard for the programmer.
- Analysis, discussing and judging the existing coding standards for different programming language.
- Investigate the usage and popularity of several different Java coding standards.
- Investigate the appropriateness and usefulness of some common used Java coding conversion for Java coding standards.
- Investigate the advantages that come from the software code reflecting.
- Get communities from both Java developers as well as other domain experts to evaluate new version of Java coding standard.

1.6 RESEARCH METHODOLOGY

Both primary and secondary research will be performed throughout the duration of this dissertation. The secondary research comprises of a literature review of material pertaining to three topics:

- Real-world Coding Standards: its history, importance and how it works in the reality IT industry.
- Coding Standards: Analyse and compare on different coding standards for different programming languages.
- The Java Coding Standards: The definition, formats and syntax of Java coding standards and the role of Java coding standards in the Java programming language.

The varying sources what will be used to complete the literature review topics: ACM Digital Library, IEEE Electronic Library, books and journals from DIT library.
The primary research of this project involves many different programming language standards. A research focus on the existing Java coding standards will be performed.

A specified new Java coding standard based on the existing coding standards will be developed. A secondary survey regarding the usefulness and appropriateness of the new Java coding standards was conducted in order to exam whether the new coding standard is working well. Domain experts including lecturers in Ireland colleges, MSc students and the staff from different companies in IT industry were invited to give their views and comments on the new Java coding standard.

1.7 SCOPE AND LIMITATIONS

This project focuses on the different coding standards of the various programming language. In order to make the project be achievable, most of the research will base on the existing Java coding standards. A number of common used Java coding standards will be picked up from the existing Java coding standards. The newly coding standard will be easy to remember and use which will help the programmer to memorize quickly. There are several limitations for this project:

1. There are a wide range of standards in existences that contribute to this project will take time.
2. There are several different versions of Java programming language; those different Java programming languages make the Java coding standards have different copies. This large amount of different coding standards makes it impossible to get all the common used conversions from the existing coding standard.
3. Limitations of resources. The evaluation from programmers from different companies of IT industry is one of the most important resources for this project. The number of people can do the evaluation becomes another limitation.

2. CODING STANDARDS

2.1 INTRODUCTION

Software programming standards are programming rules for the specified programming languages. The programming standards can greatly reduce the probability of introducing errors into applications. Software programming standards can be used regardless of which software development models are used (iterative, eXtreme programming, waterfall); it is just used to create that application. [6]

Coding standards come from the intensive study of industry experts. Those industry experts analyze how bugs or bad code style are generated when source code is written. After analyze stage, they will correlate the bugs to specific coding practices. A set of rules can be formed from the correlations between coding practices and bugs. Those rules are a set of coding standards. Coding standards can provide incredible value to software development organizations as they can avoid a lot of errors when the developers writing code. [7]

Coding standards are classified by different types of programming languages and their usages. Language specified rules and best coding practices are formed by the experts from the IT industry. The usage types and severity levels for a specified programming language are formed by the software developers.

The coding standards for a specified programming language can be followed through two ways: The software developers write the source code by following the specified programming standard. The developers will be enforced to following the rules when they are writing source code and this work should be done before the source code is packaged to the source code repository. The second approach is called "***Automated nightly builds***". [8] The coding standards are enforced for all source code modified during the day by automatically running and testing the code in "batch mode". [9]

2.2 ELEMENTS OF THE CODING STANDARDS

Different programming languages have different coding standards. The table below shows the common parts of the coding standards for all of them. A good coding style is generally to say a good implementation of coding standards elements. [10]

CODING STANDARDS ELEMENTS
Introduction
Control Structure
Program Structure
Input and Output
Common Blunders
Efficiency and Instrumentation
Documentation

TABLE 1: ELEMENTS OF CODING STANDARDS

Based on the existing coding standards, this project is aimed to find a better understandable, more easily picked-up, more reliable coding style that can be generally used and accepted by programmers. Together with the implantation of the develop toolkit hint system, the programmer can achieve and form a good coding habit to improve the efficiency of programming. So that within the elements of a coding standard, all the elements except the "documentation" will be examined in this project. In the following sections, some of the main aspects of the coding standard will be introduced.

2.3 NAMING CONVENTIONS IN CODING STANDARD

"*Name conventions are a set of rules for choosing the character sequence to be used for identifiers which denote variables, types and functions etc in programming*." [11]

Naming conversion is one of the most important concepts in coding standard and it is also the most widely talked elements in programming standard.

These conventions usually cover indentation, comments, declarations, statements, white space, programming practices and etc. software developers are highly recommended to learn the naming conventions before they learn a new programming language. Conventions can be formalized in a documented set of rules that can affect the entire IT companies or organizations. Coding conventions are not enforced by the compilers, however to get a good programming style and a programming habit; it's important to learn coding conventions for a specified programming language. [12]

Name conventions are very useful. It improves the readability of the source code. It also can enhance the appearance of the source code. Good naming conventions can make the code much profession.

The following table contains some common used language-specific conventions:

PROGRAMMING LANGUAGE	SIMPLE DESCRIPTIONS	EXAMPLES
C/C++	C and C++ preserve case sensitivity in their symbol tables while Fortran by default does not, a difference that requires attention. Fortunately, you can use the Fortran directive ATTRIBUTES ALIAS option to resolve discrepancies between names, to preserve mixed-case names, or to override the automatic conversion of names to all lowercase by Fortran. C++ uses the same calling convention and argument-passing techniques as C, but naming conventions differ because of C++ decoration of external symbols. When the C++ code resides in a .cpp file (created when you select C/C++ file from the integrated development environment), C++ name decoration semantics are applied to external names, often resulting in linker errors. The extern "C" syntax makes it possible for a C++ module to share data and routines with other languages by causing C++ to drop name decoration.	

Java	**Classes**
	Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).
	
	Methods
	Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.
	
	Variables
		`class HelloWorld;`
		`Inteface SayHello;`
		
		`void hello();`
		`void sayHello();`
		
		`int index;`
		`char tempChar;`

	Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.	

Microsoft DotNet(c#)	**Namespaces**
	Pascal case no underscores. Use company name, technology name or domain name as root. Any acronyms of three or more letters should be in pascal case (Xml instead of XML) instead of all caps.
	
	Classes and structs
	Pascal Case, no underscores or leading "C" or "cls". Classes may begin with an "I" only if the letter following the "I" is not capitalized;
	
	Interface
	Follow class naming conventions, but start the name with "I" and capitalize the letter following the "I"
	
	Attribute Class
	Follow class naming conventions, but add Attribute to the end of the name
		`System.Windows.Forms;`
		
		`public class WinForm`
		
		`public delegate void HelloWorldCallBack;`

TABLE 2: COMMON USED LANGUAGE-SPECIFIC CONVENTIONS [13, 14 and 15]

2.4 CODING INDENT STYLE IN CODING STYLE

An indent style is a convention governing the indentation of blocks of code to convey the program's structure. It contains some rules which will make the code in a readable fashion. Indent style is not a requirement for a lot of programming languages and nearly all the compiler will skip the indent style. So indent style can't affect the efficiency of the compiled applications. However indent style will affect the size of the compiled applications: the larger of indentation is the bigger file is. The size of the indent is independent of the style. A lot of early program source code used "tab" characters for indentations, for simplicity and to save the source file size.

There are many different common used indent styles. Different programming language may required different indent styles which is much suit for that programming language. The following six indent styles are the most popular used styles. [16]

1. **K&R STYLE**

   The K&R style is from a book called The "*C Programming Language*" (by Brian Kernighan and Denis Ritchie). This indent style is common used in C programming language in this book. It is also popular for C++ and C# programming language. It keeps the first opening brace on the same line as the control statement, indents the statements within the braces, and puts the closing brace on the same indentation level as the control statement.

```
public void helloWorld()
{

    while(true)
    {
        this.sayHello();
    }

    Console.WriteLine("Test...");
    System.Threading.Thread.Sleep(5000);
}
```

FIGURE 4: K&R INDENT STYLE EXAMPLE

2. **VARIANT: 1TBS**

   Variant: 1TBS coding indent style is a set of K&R style, it sometimes refer to "*The one true brace style*". In this style, the constructs that allow insertions of new code lines are on separate lines, and constructs that prohibit insertions are on a single line. The benefits of this style are that the first brace don't need an extra line by itself and the ending braces line up with the statement it conceptually belongs to. This will save a lot of space. It will also make the code have a good

appearance. [17] This coding indent style makes it difficult to scan the source code for the opening brace of a block; however it is easy to find the beginning of the block by locating the first line above the closing brace which is indented to the same level. This coding indent style is for ActionScript, JavaScript, and Java along with the Allman style.

```
public void doArrayListExample() {

      final int MAX = 10;
      int counter = 0;

      System.out.println("test");

      List listA = new ArrayList();
      List listB = new ArrayList();

      for (int i = 0; i < MAX; i++) {
          System.out.println(i);
          listA.add(new Integer(i));
      }

 }
```

FIGURE 5: INDENT STYLE EXAMPLE

3. ALLMAN STYLE

The Allman indent style is named after Eric Allman. It is sometimes referred to as "ANSI style" for its use in the documents describing the ANIS standards. This coding indent style has several advantages. The source code written with this coding indent is clearly set apart from the containing statement by lines that are almost completely whitespace, it also improve the readability of the source code. Besides the source code written with this indent style can be easily commented, removing, editing. [18]

One of the disadvantages of this style is that each of the enclosing braces occupies the entire line. Though this style can improve the readability, however it takes too many space and the length of the code for each component in the source code is very long.

This style is common used in many develop tools: Microsoft VS DotNet 2005 and Apple's Xcode.

```
public void TestMethod()
{
    AsyncCallback acb = new AsyncCallback(this.hello);
    HelloTwoCallBacl htcb = new HelloTwoCallBacl(this.helloTwo);

    // code is easy to be commented
    // htcb.BeginInvoke(acb, null);
}

// code with this style take a lot of spaces
// This following method is very long
// more than 400 lines
private void InitializeComponent()...
```

FIGURE 6: ALLMAN STYLE INDENT STYLE EXAMPLE

4. GNU STYLE

The GNU coding style was first used by Richard Stallman and other GNU Project volunteers in order to make the GNU system clean, consistent, and easy to install. GNU style puts braces on a line by themselves. The braces take up two spaces, and the contained code is indented by another two spaces. This coding style both get the advantages from Allman style by removing the possible Whitesmiths disadvantage of braces not standing out from the block. This coding indent style also refers to the GNU Coding Standards which cover the minimum of what is important when writing GUN code.

```
public void test()
  {
    final int MAX = 10;
    int counter = 0;

    System.out.println("test");

    List listA = new ArrayList();
    List listB = new ArrayList();
  }
otherMethod();
```

FIGURE 7: GNU STYLE INDENT STYLE EXAMPLE

5. BANNER STYLE

The banner style is common used in mark-up language like HTML, XML. The developers can insert closing brackets in the last line of a block. It makes indentation the only way of distinguishing blocks of code. This style makes visual scanning easier. This style is similar to K&R and Allman,

```
<table>
  <tr>
    <td>
    TESTTEST
    </td>
    <td> TESTTEST========== </td>
    <td> ========TESTTEST</td>
    </tr>
  </table>

<table>
  <tr> ... etc
  </table>
```

FIGURE 8: BANNER STYLE INDENT STYLE EXAMPLE

6. BSD KNF STYLE

BSD KNF indent style is also known as Kernel Normal Form style, this coding indent style is common used in the Berkeley Software Distribution (BSD) operating system. This indent style is used widely both in userland code and kernel code. It is a thoroughly-documented variant of K&R style as used in the Bell Labs UNIX source code (Version 6 and Version 7).

```
if (data != NULL && res > 0) {
        if (JS_DefineProperty(cx, o, "data",
            STRING_TO_JSVAL(JS_NewStringCopyN(cx, data, res)),
            NULL, NULL, JSPROP_ENUMERATE) != 0) {
                QUEUE_EXCEPTION("Internal error!");
                goto err;
        }
        PQfreemem(data);
} else {
        if (JS_DefineProperty(cx, o, "data", OBJECT_TO_JSVAL(NULL),
            NULL, NULL, JSPROP_ENUMERATE) != 0) {
                QUEUE_EXCEPTION("Internal error!");
                goto err;
        }
}
```

FIGURE 9: BSD KNF STYLE INDENT STYLE EXAMPLE

2.5 CONCLUSIONS

This chapter gives an overview of the coding standards, and discussed the concept of a coding standard. This chapter also explains the most used and important elements of a coding standard like naming conversion, indent style. Naming conversion helps make the program more readable and meaningful naming conversions make the program easy to understand. Indent style is the style that affects the structure of the code and a good indent style makes the code easy to read and also show the condition relationship clearly. It is also a powerful ally for writing good code.

3. CODING STANDARDS TOOLS

3.1 INTRODUCTION

Coding standards is a broad topic and people have made a lot of attempts to use it to achieve better source code. Technologies from different domains have been developed and used to improve it. In this chapter, technologies such as code inspection technology, code renew technology and code refactoring technology will be discussed as matters that affect coding standards. Most of these technologies will be used in the implementation section of this project.

3.2 CODE INSPECTION TECHNOLOGY

Code inspection is one of the code review practices for the source code of project. The goal of code inspection is to make all the inspectors achieve consensus on the coding style of source code. That means code inspection can help the developers to approve the coding style of the project. A common code inspection typically includes source code requirement specifications and code review plans. In a code inspection process, a sample of source code will be selected and a team will work on the source code to identify defects and improve the coding style of the project. [19]

Benefits of coding inspection:

1. To improve quality of the software design
2. To reduce software decay / software aging, software complexity, software maintenance costs
3. To increase software understanding, software productivity
4. To facilitate future changes

3.2.1 CODE INSPECTION TOOLS

Currently a lot of tools that can do code inspection automatically tend to focus on improving code quality from a technical perspective. These tools will do the job from two perspectives. [20]

1. By checking for the existing source coding.

The fewer bugs that are detected in the source code, the higher quality of that code is. There are also two levels from this perspective: low-level-bug-detect and high-level-bug-detect. The Low-level-bug-detect focuses on the source code itself, the High-level-bug-detect focuses on run-time bugs. When the applications encounters errors, the related source code will be detected and shown to the developer.

2. Improve the coding style of the source code.

The code inspected tools also work on the programming style of the source code. Different code inspection tools contain different versions of programming standards. A good source code inspecting tool can convert the source code to a good style and have good appearance.

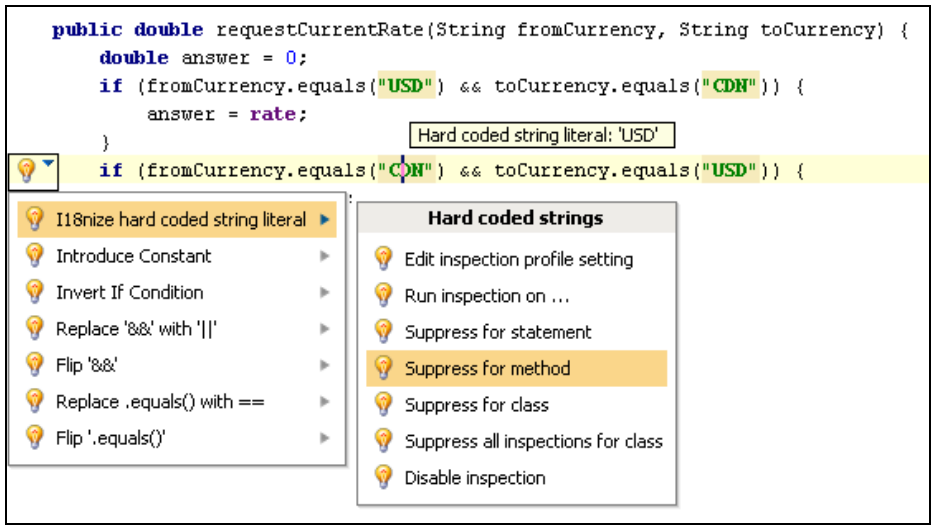


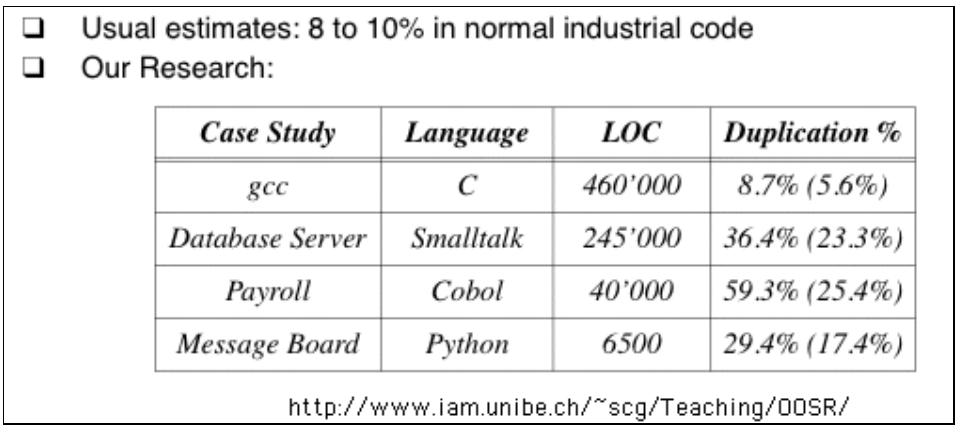
FIGURE 10: CODE INSPECTION TOOLS

3.2.2 INTRODUCTION TO SOFTWARE REFACTORING

Software refactoring is a specific implementation of code inspection. It is a process of changing software source code. It does not alter the external behaviour of the source code but it can improve internal non-functional properties of the software. Software refactoring improves the code from the following perspectives [21]:

- Improve source code readability
- Change code style to a given programming paradigm
- Simplify code structure, to improve maintainability of the source code
- Improve performance or improve code extensibility.

Software code refactoring is one of the best practices for programmers who would like to improve their source code quality.



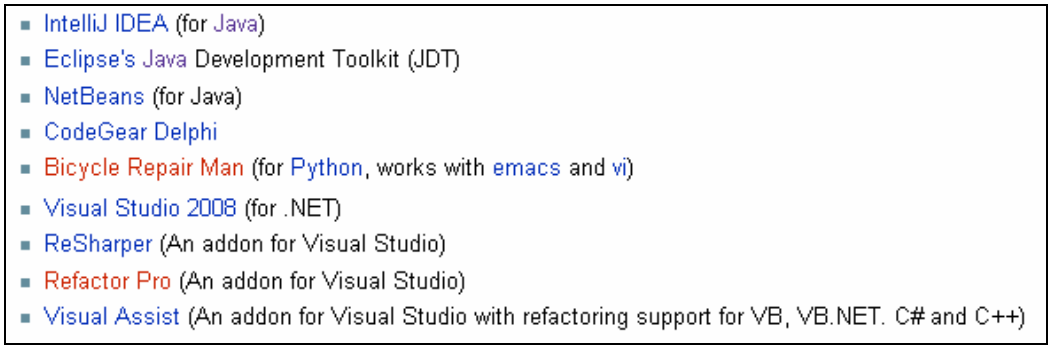
- Usual estimates: 8 to 10% in normal industrial code
- Our Research:

Case Study	*Language*	*LOC*	*Duplication %*
gcc	*C*	*460'000*	*8.7% (5.6%)*
Database Server	*Smalltalk*	*245'000*	*36.4% (23.3%)*
Payroll	*Cobol*	*40'000*	*59.3% (25.4%)*
Message Board	*Python*	*6500*	*29.4% (17.4%)*

http://www.iam.unibe.ch/~scg/Teaching/OOSR/

FIGURE 11: IT'S NECESSARY TO DO CODE REFACTORING

As software refactoring has many benefits, software refactoring becomes one of the most important steps in software development cycle such as extreme programming and agile methodologies.



- IntelliJ IDEA (for Java)
- Eclipse's Java Development Toolkit (JDT)
- NetBeans (for Java)
- CodeGear Delphi
- Bicycle Repair Man (for Python, works with emacs and vi)
- Visual Studio 2008 (for .NET)
- ReSharper (An addon for Visual Studio)
- Refactor Pro (An addon for Visual Studio)
- Visual Assist (An addon for Visual Studio with refactoring support for VB, VB.NET. C# and C++)

FIGURE 12: SOME CODING REFLECTING TOOLS

3.2.3 SOFTWARE REFACTORING PROCESS

Software refactoring contains several transformation steps: [22]

1. Try to set up a goal-reasoning model.
2. According to the software metrics, decide which alternative soft-goal should be applied first.
3. Picking an effective refactoring among the source code to achieve soft-goal;
4. Applying the selected refactoring technique.

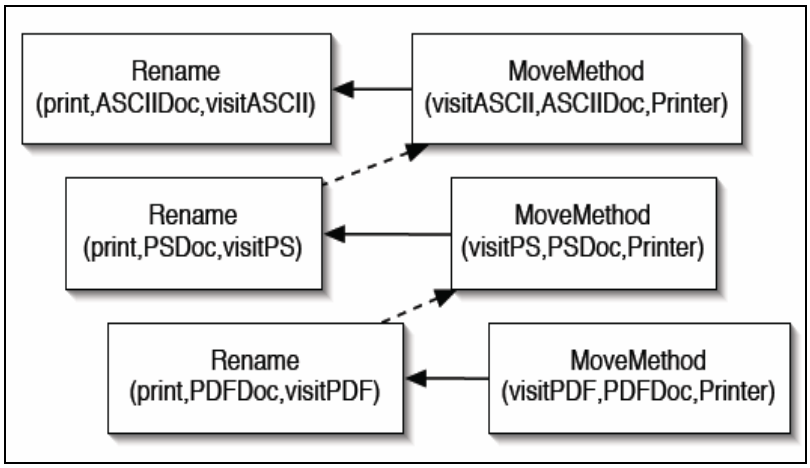


FIGURE 13: PROCESS OF SOFTWARE REFACTORING [24]

When software developers attempt to do code refactoring for source code, they may have a problem: "*When and how to do code refactoring*". In most cases, code refactoring can be done after each component is released. The maintainability of the source code is also arguably a problem of subjectivity. Many developers may find it is much easier to maintain a software system which is developed by themselves than maintaining other programmers' code. Sometimes the developers may feel maintaining other developers' source code is just as hard as re-write that part of program. This means that whether the source code should be refactored is often subjective. The changes that made from the code refactoring are called "transformations" by Martin Fowler who has written a famous book about refactoring. Individually, simple transformations may make small improvements while a lot of transformations may make a far improvement to the source code. Code refactoring commonly includes several different aspects: reducing scope (Do changes in every piece of code), replacing complex instructions (make the code looks simple and clearly) and combining multiple statements into one statement. [21]

BAD SMELL	PROPOSED REFACTORING
Duplicated code	Extract method pull up variable form template method substitute algorithm
Long method	Extract method replace temp with query introduce parameter object preserve whole object replace method with method object
large class	Extract class extract subclass
Lazy class	Move method
Feature envy	Collapse hierarchy inline class

TABLE 3: HOW DO WE KNOW WHAT TO REFACTOR

3.2.4 CODING REFACTORING PATTERNS

Many refactoring patterns have been designed since Martin Fowler published his book which is about coding refactoring. Coding refactoring patterns is a design

pattern that can improve the design of existing code. Using design patterns to improve an existing design is better than using patterns early in a new design which will be improved. So Joshua Kerievsky said: "*The developers can improve designs with patterns by applying sequences of low-level design transformations.*" [25]

Combining the techniques of design patterns and coding refactoring can make great improvements in the project's source code. With pattern-based refactoring applied, developers can get an improved source code structure. The new structure then allows the existing capabilities to be modified and improved. It also benefit from adding new features by localizing the efforts. Pattern-based refactoring also can reduce the cost and the most important contribution of pattern-base refactoring is that it can generate high quality source code which has a good coding style, and the time spent on refactoring the source code has laid the groundwork for future business. [26]

3.3 CODE REVIEW

"Code review is systematic examination of computer source code intended to find and fix mistakes overlooked in the initial development phase." [27]

Find and fix mistakes is one of its benefits. Code review is very useful for improving the style of source code which will make the source code look more professional. Code review can be divided into two main categories: *formal code review* and *lightweight code review.* [28]

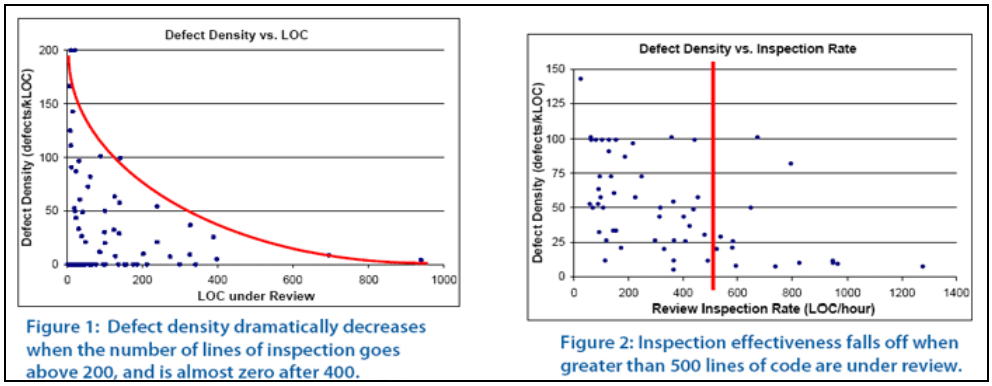


FIGURE 14: CODE REVIEW BRINGS BENEFITS

1 *Formal code review*

Formal code review involves a careful and detailed process with multiple participants and multiple phases.

2 *Lightweight code review*

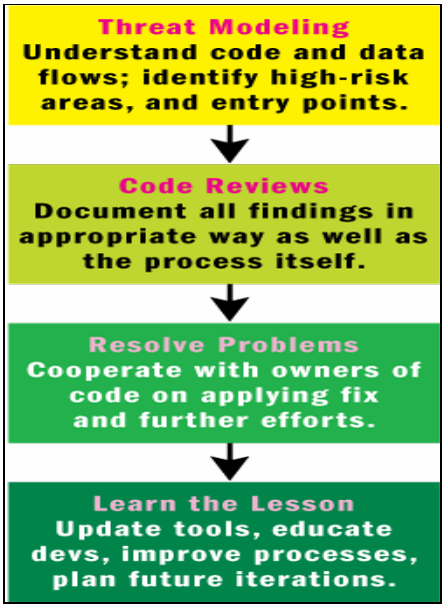
Lightweight code review requires less overhead than formal code inspections, though it can be equally effective when done properly.

3.3.1 APPROACHES TO CODE REVIEW

There are many different approaches to code reviewing which are based on different code review modules, however most code review modules have the same components. Those components are;

1. Structure
2. Techniques
3. Inputs
4. Environment
5. Technology

Most code review modules are designed based on these components. One of the famous examples is security code review model. In a security-related model, the approach to code review depending on model components. With the help of this model, code reviews can show additional values by improving other security tasks such as testing and design. [29]



SECURITY CODE REVIEW MODEL [30]

There are four steps to achieve a security code review

- **Step 1:** Establish the goals and code review object objectives for the code review.
- **Step 2:** Perform a full code scan, to find bugs and improve the style of the code, the most important step for code styles.
- **Step 3:** Use the results of Step 2, performing another analysis.
- **Step 4:** Review for coding style according to the software source code architecture.

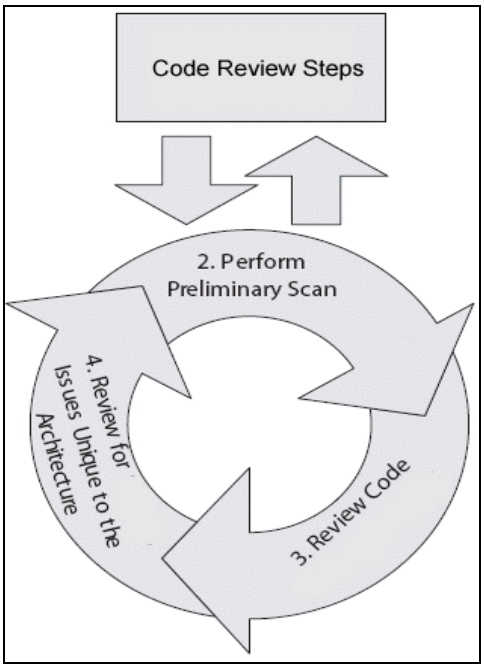


FIGURE 15: PROCESS OF CODE REVIEW [31]

Other models have the similar approaches, besides, there are also some particular tips for achieve good code review.

1. Review fewer than 200-400 lines of code at a time.
2. Aim for an inspection rate of less than 300-500 LOC/hour.
3. Take enough time for a proper, slow review, but not more than 60-90 minutes.
4. Authors should annotate source code before the review begins.
6. Checklists substantially improve results for both authors and reviewers.
7. Verify that defects are actually fixed!
8. Managers must foster a good code review culture in which finding defects is viewed positively.
9. Beware the "Big Brother" effect.
10. Lightweight-style code reviews are efficient, practical, and effective at finding bugs.

Good code review can help software developers to identify bugs, encourage collaboration, make code more maintainable and improve the appearance of source code; this can make the source code to reach high quality.

3.3.2 CODE REVIEWS ACROSS THE TEAM

There are many teams in the same company or organizations. People who are in the same team can follow programming standards easily, however different teams may have different skills. One of resolutions is to have a team leader meeting and develop

the same coding standard document for the company or the organization. All the staff in the company can use this document as a reference which will be used in every team, and then distributed to everybody at the coding standard meeting. All of the members of the same team should go to the meeting and be prepared to discuss the various points in the document. After discussing all points in the programming standards, people may have different opinions about each point, at the end of the meeting, all members must agree upon the coding standards define the uncertain parts. If there are a lot of staffs who don't agree on some points, a new team leader meeting should be held. In the team leader meeting, those points will be discussed again. So a new version of coding standard document with appropriate changes based on the suggestions from all of the team members can then be generated. [32]

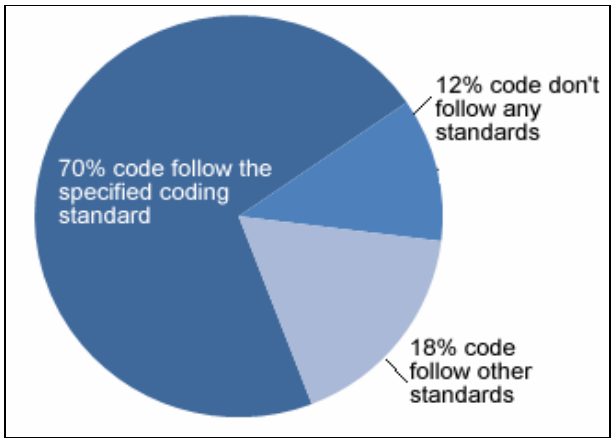


FIGURE 16: CODE WITHOUT CODING REVIEW

It's not enough for the team leader to do the above jobs. According to research regarding this problem, the researcher found about 70% of the code follows the specified coding standard, 18% of the source code follows other standards (Developer's familiar coding standards) and 12% of the source code is in bad coding style. In order to resolve this problem, it's necessary to hold code renew meetings. [14]

When the development of the project starts, code review meetings should be held, to ensure that everybody follows the rules. There are three types of code reviews are recommended when developing the software: [33]

- **Peer-Peer review**
  The members from other developing teams review the code to ensure that the code follows the coding standards and meets requirements which are indicated in the coding standard. This type of coding review can also contain some unit testing. Every file in the developing project should go through this process.

- **Architect review**

The architect of the team should review the core components of the project to ensure that the coding follows the coding standard and meets requirements, and the application also reaches the requirements of the design. Make sure that there are no "big" mistakes that can affect the project in the long run.

- **Group review**
  The team leader or the architect of the team should select one or more files from the core components of the product and ask the team members to review the source code. After the members review the source code, a team meeting should be held in order to let members show problems. In the review meeting, every section of the source code should be gone through and every member should give their own suggestions on how to improve the code quality. Group review can be held 1-2 times every week. [34]

3.4 CONCLUSIONS

This chapter introduces several different types of coding technologies which can affect the quality of the code and the attempt that people have made to achieve better source code. Because of the coding standards can't be strictly followed and used in real life, it cost people more to get the better code and along with the technology developed, people had found several ways combined with coding standards to get programmers writing good ,readable, consistent code. With the helps of those coding technologies, it will greatly help programmers to do code review or code refactoring.

4. COMMONLY USED STANDARDS

4.1 INTRODUCTION

It is less important which programming standard is chosen during the development process, than that the selected process is consistently applied throughout the process. The members in the same team should also follow the same specified coding standards, otherwise, reading code that follows different coding standards can become very difficult. In the follow sections, some commonly used programming standards will be discussed.

4.2 C++ PROGRAMMING STANDARDS

C++ programming was developed by Bjarne Stroustrup at Bell Labs; the reason it was developed was to enhance to the C programming language. It was originally named "*C with Classes*" and was renamed to C++ in 1989. [35]

C++ is a middle-level programming language as it has both high-level and low-level language features. It is widely used in the IT industry. There are a lot of software developed in C++, including application software, systems software, low-level device drivers, embedded software, hardware software, entertainment software and server and client software. The ISO/IEC JTC1/SC22/WG21 working group standardized C++ and published the international standard ISO/IEC 14882:1998 (known as C++98) in 1998. After several years, the working group processed defect reports, and published a new version of the C++ standard, ISO/IEC 14882:2003, in 2003. That standard is still current, but it was re-mended in 2005 (Known as Library Technical Report), however while it is not an official part of the standard, this version of the C++ programming standards specifies a number of extensions to the standard library, which were expected to be included in the next version of C++. The next version of the C++ programming standard is still under development. [36]

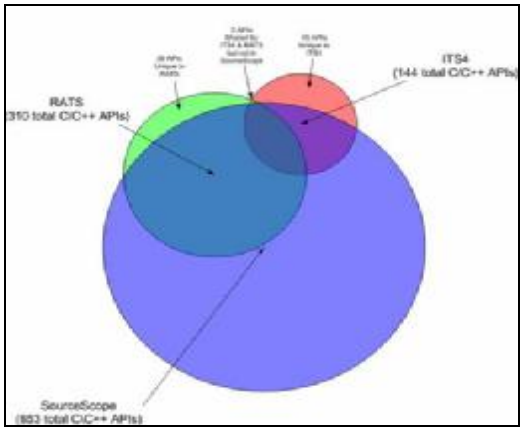


FIGURE 17: A LOT OF SOFTWARE WAS DEVELOPED IN C++/C

After C++ was developed, a lot of different C++ programming standards were created, most of those programming standards are a subset of the C++ programming standards which was formed in 1998 (ISO/IEC 14882:1998).

NAME	AUTHORS (ORGANIZATIONS)
C++ Coding Standard	Geotechnical Software Services
C++ Coding Standards	Alan Bridger&Jim Pisano
High Integrity C++ Coding Standard Manual	www.Programmingresearch.com
C++ Coding Standards	Herb Sutter&Andrei Alexandrescu
C++ Coding Standards	www.arcticlabs.com
C++ Programming Standards and Guidelines	Science Infusion and Software Engineering Process Group (SISEPG)
Google C++ Style Guide	www.google.com
Apache C++ Coding Standard	www.apache.com
C++ Coding Style	Albert Sandberg

TABLE 4: DIFFERENT TYPES OF C++ CODING STANDARDS

The most popular programming style for the C++ (and C) language is the Kernighan and Ritchie (K&R) style. K&R programming style provide a solution which can solve the following problems.

1. Reusability
    a) Self documenting code
    b) Internationalization
    c) Maintainability
    d) Portability
2. Optimization
3. Build process
4. Error avoidance
5. Security

The K&R programming style is far too general in its scope, so more specific rules needed to be established. This programming standard also has an annotated form that makes it far easier to use during project code reviews than many other existing programming styles. In addition, programming recommendations generally tend to mix style issues with language technical issues in some confusing manners. [37]

COMPONENTS	SHORT DESCRIPTIONS	
Layout of the Recommendations	1. Guide-lines should contains motivation, background and additional information	
The Importance of Recommendation	1. Most of the violations to the recommendations are allowed if it enhances readability	
Naming Conventions in C++	1. Names of types should be in mixed case starting with upper case. 2. The names of the namespaces should be all in lower-case. 3. Private class variables should have underscore suffix. 4. Generic variables should have the same name as their type. 5. Variable names should be in mixed case starting with lower case. 6. The name of the constants should be all in upper-case with underscore to separate words.	
Specific Naming Conventions	1. The key words (***get & set)*** should be use used where an attribute is accessed directly. 2. If there is an object or a concept is established, the key words (***initialize)*** can be used	
C++ Source Files	1. C++ header files should have the extension *.h* or *.hpp*. 2. File content must be kept within 80 columns. 3. Special characters like page break can be avoided. 4. All definitions should be in source files.	
The Statements in C++ source files	Types	1. Types that are local to one file should only be declared in that file. 2. Type conversions must always be done explicitly.
	Variables	1. Variables should be initialized as soon as they are declared. 2. It's good habit to use as few global variables as possible 3. Variables should be declared in the smallest scope possible. 4. The meaning of the variables should be clearly.

		5. Class variables can't be declared public.
	Loops	1. The variables in the loop statement should be initialized immediately before the loop. 2. The use of break and continue in loops should be avoided 3. do-while loops can be avoided in the source code file of C++
Layout and Comments	1. Method definitions should have the specified forms, all the methods in the source file or in the same project should be I the same forms. 2. Class and method document or head comments should follow the JavaDoc comment convention.	

TABLE 5: GENERAL TIPS FOR C++ PROGRAMMING STANDARDS [38]

4.3 C# PROGRAMMING STANDARDS

C# is an object oriented, simple, and type-safe programming language which was developed by Microsoft in 2000 and is known as Microsoft's primary developing language. Its principal designer and lead architecture is Anders Hejlsberg who works in Microsoft. He was previously involved with the design of several different programming languages like TurboPascal, Borland Delphi, and Visual J++. C# first appeared in Microsoft Visual Studios 2003. [39]

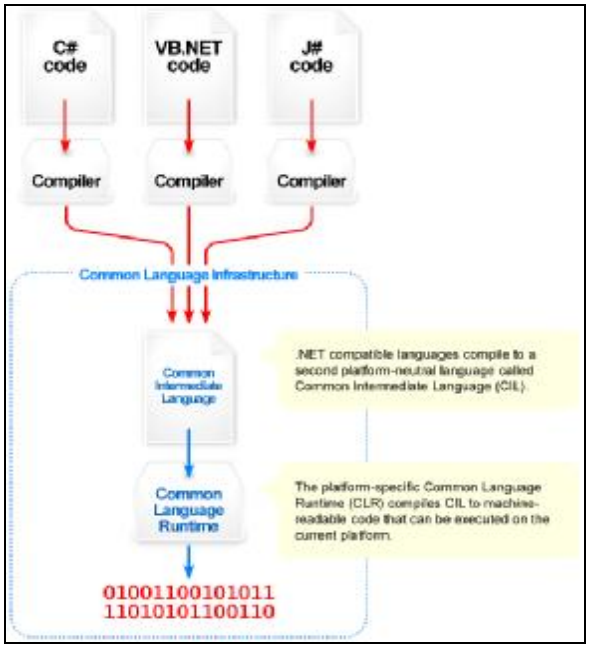


FIGURE 18: C# PROGRAMMING LANGUAGE

C# reflects the underlying Common Language Infrastructure (CLI). CLI is an open specification which is developed by Microsoft. It describes the executable code and runtime environment which forms the core for a number of runtimes (the Microsoft .NET Framework, Mono, and Portable.NET). CLI implements most of the C# runtime intrinsic types which correspond to value-types. However Microsoft does not state that a C# compiler must target a Common Language Runtime or generate a CLI. That means C# compiler could generate machine code like traditional compilers of C++ or C. [40]
Some features of C#: [41]

1. Memory address pointers must be used within specifically marked which indicated that the code is unsafe and programs with unsafe code need some appropriate permissions to run
2. The local variables in C# programming language can't attack variables of the enclosing block

3. C# doesn't define global variables or method so all the methods and members must be declared within classes.
4. C# programming language provides full type reflection and discovery.
5. Same to Java programming, multiple inheritances is not supported in c#, however the class can implement many interfaces.
6. As there is some permission for using the memory address pointers so C# is more type-safe than C++ and C programming language.

Like C++, after C# has been developed, there are many different types of C# coding standards appearing. Most of the C# coding standards are designed based on the same models (Microsoft generated the model). The organizations or companies define their own rules in each section. This model contains: [42]

- Relationship to the Common Type System and the Common Language Specification
- Naming Guidelines
- Class Member Usage Guidelines
- Type Usage Guidelines
- Guidelines for Exposing Functionality to COM
- Error Raising and Handling Guidelines
- Array Usage Guidelines
- Operator Overloading Usage Guidelines
- Guidelines for Implementing Equals and the Equality Operator
- Guidelines for Casting Types
- Common Design Patterns
- Security in Class Libraries
- Threading Design Guidelines
- Guidelines for Asynchronous Programming

NAME	AUTHORS (ORGANIZATIONS)
C# programming standards	Microsoft
C# Naming Conventions	Kingsoft.com
C# Coding Style Guide	Mike Krüger
Naming Conventions for C#	Paulo Morgado
IDesign C# Coding Standard	www.idesign.net
C# coding standards	Jeff Key
Coding Standard: C#	Vic Hartog and Dennis Doomen

TABLE 6: DIFFERENT TYPES OF C# CODING STANDARDS

Although there are many types of C# coding standards, most of them are designed based on C# programming standards (Microsoft Version), so the Microsoft version C# coding standard has become the most commonly used C# coding standard in IT industry. This version is also based on the design of the C# DotNet model. A set of important rules in this standard are listed below.

TYPE	STANDARDS (STANDARD BASED UPON MICROSOFT .NET LIBRARY STANDARDS)
Classes and Structs	Pascal Case, no underscores or leading "C" or "cls". Classes may begin with an "I" only if the letter following the "I" is not capitalized; otherwise it looks like an Interface. Classes should not have the same name as the namespace in which they reside. Any acronyms of three or more letters should be pascal-case, not all caps. Try to avoid abbreviations, and try to always use nouns.
Collection Classes	Follow class naming conventions, but add Collection to the end of the name
Parameters	Camel Case. Try to avoid abbreviations. Parameters must differ by more than case to be usable from case-insensitive languages like Visual Basic .NET.
Delegate Classes	Follow class naming conventions, but add Delegate to the end of the name
Exception Classes	Follow class naming conventions, but add Exception to the end of the name
Class-Level Private and Protected Variables	Camel Case with Leading Underscore. In VB.NET, always indicate "Protected" or "Private", do not use "Dim". Use of "m_" is discouraged, as is use of a variable name that differs from the property by only case, especially with protected variables as that violates compliance, and will make your life a pain if you program in VB.NET, as you would have to name your members something different from the accessor/mutator properties. Of all the items here, the leading underscore is really the only controversial one. I personally prefer it over Straight underscore-less camel case for my private variables so that I don't have to qualify variable names with "this." to distinguish from parameters in constructors or elsewhere where I likely will have a naming collision. With VB.NET's case insensitivity, this is even more important as your accessory properties will usually have the same name as your private member variables except for the underscore.
Interfaces	Follow class naming conventions, but start the name with "I" and capitalize the letter following the "I"
Functions and Subs	Pascal-Case no underscores except in the event handlers. Try to avoid abbreviations. Many programmers have a nasty habit of overly abbreviating everything. This should be discouraged. We're not designing license plates or being charged by the letter

TABLE 7: A SET OF C# PROGRAMMING LANGUAGE STANDARDS [43]

4.4 VISUAL BASIC PROGRAMMING STANDARDS

The latest version of Visual Basic (VB), it is also called Visual Basic .NET (VB.NET). Visual Basic (VB) is an object-oriented computer programming language programming language which is developed by Microsoft. Visual Basic also contains an integrated development environment (IDE) for its COM programming model. The first version of Visual Basic was introduced in 1991. Up to now there have been 10 versions of Visual Basic programming language. Version 6 and the earlier versions are event-derived programming language while the later versions are object-oriented. There are also many programming standards for Visual Basic. The first version of the Visual Basic programming standards were published by Microsoft in 1991, as it is widely used in the early 1990s; many companies or organization also published their own VB coding standards. [44]

NAME	AUTHORS (ORGANIZATIONS)
Microsoft Consulting Services Naming Conventions for Visual Basic	Microsoft
VB6 Naming Conventions	www.vb6.us
Visual Basic/Coding Standards	www.kingsoft.com
Visual Basic Coding Standards	www.opfro.org

TABLE 8: DIFFERENT TYPES OF VISUAL BASIC CODING STANDARDS

Like C#.Net, Visual Basic coding standards also has its own coding model. The coding standard model contains seven sections: [45]

- Naming Procedures
- Modifier Tables
- Data Type Modifiers
- Scope Modifiers
- Miscellaneous Modifiers
- Control Modifiers
- Data Access Objects

All Visual Basic coding standards should based on this coding standard model, the rules defined in the coding standards also should be include in this seven sections.

4.5 JAVA CODING STANDARDS

4.5.1 INTRODUCTION TO JAVA

Java is an object oriented, simple, and type-safe programming language originally developed by James Gosling from Sun Microsystems. The first version of the Java

programming language was released in 1995 as a core component of Sun Microsystems' Java platform. The Java programming Language evolved from another programming language Oak which was also developed by Sun Microsystems. The Java programming Language is a platform-independent language which is aimed at allowing entertainment appliances to communicate [46]. The first version of Java promised "Write once, run anywhere" (WORA). This means it can prove no-cost run-times on the popular platforms. Java is a secure programming language as it allows network and file access restrictions. The new version of Java (Java 2) was released in December of 1998, the new version of Java provides multiple configurations to be built for different platforms. For the marketing purposes, Sun Microsystems renamed Java 2 to Java 2 EE, Java 2 ME and Java 2 SE. [47]

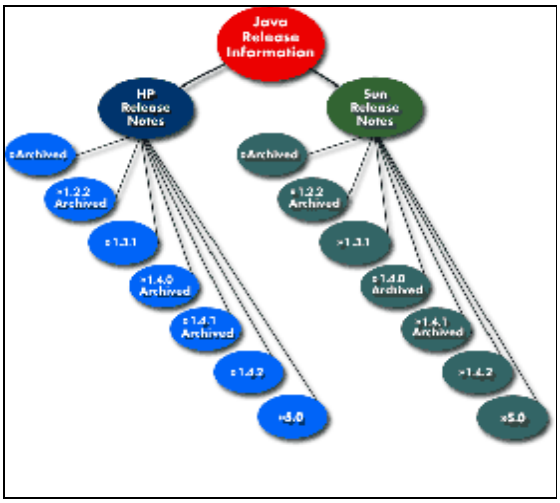


FIGURE 19: JAVA PROGRAMMING LANGUAGE DEVELOP HISTORY

4.5.2 JAVA CODING STANDARDS OVERVIEW

Java is widely used since 1995, and up to now there are more than 4000 different versions of Java programming standards. The most important and famous one is Code Conventions for the Java Programming Language which is published by Sun Microsystems. This version of the Java programming standards contains the standard conventions which are used in Sun Microsystems Inc. It covers filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices and includes a code example [48]. Most other versions of Java programming standards are based on this version. In this research, the new generated Java programming coding standard is also based on this coding standard. [49]

NAME	AUTHORS (ORGANIZATIONS)
Code Conventions for the Java Programming Language	Sun Microsystems
Java Programming Language coding guidelines and standards	IBM
Coding Standards for Java	AmbySoft Inc.
Java Programming Style Guidelines	Geotechnical Software Services
JavaCoding Standards	ESA Board for SoftwareStandardisation and Control(BSSC)
Java Coding Standards	Ray Ontko
Java Coding Standard	Arnaud Blandin

TABLE 9: DIFFERENT TYPES OF C# CODING STANDARDS

4.5.3 JAVA CODING STANDARDS MODEL

The Java coding standards model which is designed by Sun Microsystems contains the following sections:

- filenames
- file organization
- indentation
- comments
- declarations
- statements
- white space
- naming conventions

Filenames mean the name of the Java source code file. Here it also contains File Suffixes and Common File Names.

File organization also can be called package organization; it introduces the source code file saved path and source code file information.

Indentation gives the advice on Line Length and Wrapping Lines.

The section which introduces the *comment* in Java is one of the most important parts in the coding standard. In this section, the engineers introduce two types of comments in Java: Implementation Comment Formats and Documentation Comments.

Declarations introduce the variable, class, interface definition in Java.
Statements tell the programmer how to write source code. The statements section also includes the following aspects.

1. Simple Statements
2. Compound Statements
3. return Statements
4. if, if-else, if else-if else Statements
5. for Statements
6. while Statements
7. do-while Statements
8. switch Statements
9. try-catch Statements

Naming conventions is the most important section in Java. They make source code more understandable by making it easier to read. They can also give information about the method of the identifier [50].

IDENTIFIER TYPE	EXAMPLES
Packages	com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese
Classes	class Raster; class ImageSprite;
Interfaces	interface RasterDelegate; interface Storing;
Methods	runFast(); getBackground();
Variables	char c; float myWidth;
Constants	static final int MAX_WIDTH = 999;

TABLE 10: NAMING CONVENTIONS IN JAVA PROGRAMMING LANGUAGE

The Sun Microsystems' version of coding model is the most famous and important coding model for Java, however it is also one of the most complex coding models. There are also some simple models. For example:

Watts Humphrey's model which is mentioned in his book "*A Discipline for Software Engineering*" [51]

- Structure and Documentation
- Naming conventions
- Recommendations
- Related Documents

Scott Ambler's model [52]

- General Recommendations
- Naming Conventions
- Files
- Statements

- Layout and Comments

In this project, the model from "*The Elements of Java Style*" will be used.

- Formatting Conventions
- Naming Conventions
- Documentation Conventions
- Programming Conventions
- Package Conventions

4.6 CONCLUSIONS

This chapter discussed the coding standard from different programming languages. C++, C# and Java are commonly used and thus there are a lot of coding standards. For different programming languages, the coding standard is different and this is determined by the programming language itself. Based on the different features and advantages, this project takes Java as the programming language to study and improve its coding standard. This is because Java is one of the most widely used programming languages with many different coding standards.

5. TOWARDS A NEW STANDARD

5.1 INTRODUCTION

Some reasons why Java coding standards are important for programming [53]

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
- Facilitates sharing of source coding among different developers, especially teams of programmers working on the same project.
- If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.

The results above show that a good Java coding standard is important. However, as there are several different versions of Java coding standards, some of them are very difficult to learn. According to the survey undertaken in this research of Java programming standards, more than 20% people don't know which version of the Java programming standard they are using, and more than 35% of developers agree that more than 30% of their code doesn't following any programming standard. This survey shows that a large number of developers can't deal with programming standards well. Many of these developers think that it is easy to get confused with the existing coding standards; they are difficult to learn and to remember. A survey for some simple and short Java programming standards which are based on the existing Java convention standard shows that more than 40% developers think it is necessary to get such version of the Java programming standard and 30% of them are willing to learn. Most of them are students or new programmers. [54]

This new version of the Java programming standard comes from the existing Java programming standards. The tips or rules are generated from the most commonly used rules and tips. More than 60% of the developers are using these tips and rules. They are correct, good and easy to learn. A survey shows that more than 50% of developers agree that code following this version of the Java programming standard is good and the coding standard is easy to learn.

5.2 NAMING CONVENTIONS

5.2.1 INTRODUCTION TO NAMING CONVENTIONS

Naming conventions are one of the most important aspects of writing java source code. Good naming conventions can help the programmer to read and understand the

code. It can also help the developer to maintain the source code. For the naming conventions of different components we can set the stage with a few basics: [55]

1. Use meaningful English descriptors which can describe the variables, methods, classes, interfaces, packages and other components clearly. This can make the code easy to understand, maintain and enhance.
2. Use mixed infix-caps case to make the names readable. Different components have different naming conversions; however most of them should follow the mixed infix-caps case.
3. Use meaningful abbreviations sparingly, make sure that people who read the source code can understand them easily.
4. Avoid long names and strange characters, the length of the name should be less than 15. Avoid using some string characters like underscores (_).

Most naming conventions in Java should follow the ways above. A good naming convention can make the source code professional.

IDENTIFIER TYPE	RULES FOR NAMING	EXAMPLES
Packages	The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981. Subsequent components of the package name vary according to an organization's own internal naming conventions. Such conventions might specify that certain directory name components be division, department, project, machine, or login names.	`com.apple.quicktime.v2` `edu.cmu.cs.bovik.cheese`
Classes	Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).	`class Raster;` `class ImageSprite;`
Interfaces	Interface names should be capitalized like class names.	`interface RasterDelegate;` `interface Storing;`

Methods	Methods should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.	`run();`
		`runFast();`
		`getBackground();`

Variables	Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters.	`Int i;` `Char c;`

	Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.	`Float myWidth;`

Constants	The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_"). (ANSI constants should be avoided, for ease of debugging.)	`static final int MIN_WIDTH = 4;`
		`static final int MAX_WIDTH = 999;`
		`static final int GET_THE_CPU = 1;`

TABLE 11: DETAILS FOR INTERFACE/CLASS DEFINITIONS [56]

5.2.2 PACKAGE NAMES

In most object-oriented programming languages, the package names which can used to distribute to other organizations should include the lowercase domain. It is the same in Java coding standards. Commonly used Java coding standards require the programmer to use the reversed, lower-case from of the organization's internet domain names as the root qualifier for the package names. [57]

Examples (Java version):

```
com.java.helloworld.v1
com.java.helloworld.v2
```

Sun Microsystems has also reserved the use of the package names: `java` and `javax`. All the package names starting with java and javax are published by the Sun Microsystems.

The qualified portion of a package name should use a single, lowercase, meaningful English word as the root name. However the name of the package may consist of several meaningful words. A common solution is to use one of the meaningful words which may present the meaning of the package. [58]

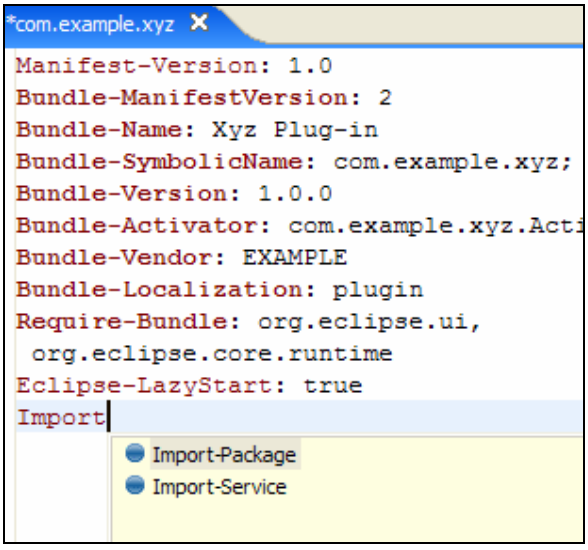


FIGURE 20: ECLIPSE PACKAGE NAME TOOLS

If there is a newly released version package, the new package should have the same name as the previous one. A new package with a lot of contents changed should have a new name.

5.2.3 TYPE NAMES

The type name is also a very important part for the Java coding style. Many C++/C programmers may use C++/C programming style to name Java type names. There are several different ways of naming types in Java, however the key thing is to focus on "*meaningful naming*". When the developer defines a class, variable, method or interface, meaningful names will help the programmers to understand meaning of the code. [59]

File Name	Use
GNUmakefile	The preferred name for makefiles. We use gnumake to build our software.
README	The preferred name for the file that summarizes the contents of a particular directory.

FIGURE 21: FREQUENTLY USED JAVA NAMING WAYS

The name of a Java source file should be same as the public class that contains its source file. All type names should use the infix-caps style. The name should start with an upper-case letter and capitalize the first letter of any subsequent word in the name. All the words in the name should be meaningful. Source file / Class name should be nouns or noun phrases and never use underscores to separate words. The name of the interface can depend on the purpose of the interface. [60]

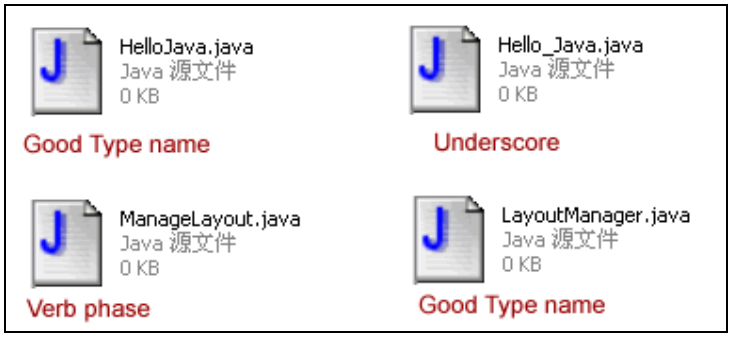


FIGURE 22: EXAMPLES OF JAVA FILE NAMES

5.2.4 CLASS NAMES/ INTERFACE NAMES

The class or interface declarations contain at least two parts: the Java keywords (Class/Interface) and the name of the Class/Interface to be created.
An interface definition can have two more parts: the access specification (public) and a list of super-interfaces. So the full interface declaration can be like this:

```
[Public] interface Name-Of-The-Interface [extends
SuperInterface1, SuperInterface2]
```

The access specification (public) indicates that the interface can be accessed by any other classes (both in the same package or different packages). The commonly used approach to naming the classes and interfaces is to capitalize the first letter of each word that appears in a class or interface. [61]

```
public class PrintStream extends FilterOutputStream {
    //
    //
    //
}


public interface ActionListener extends EventListener {
    //
    //
    //
}
```

FIGURE 23: NAMING THE CLASSES/INTERFACE

According to the commonly used naming convention, the class/interface name also should use nouns or adjectives. The name of the classes/interfaces should relate to the attributes, static service or constants. Give classes/interface that related attributes, static services or constants a name that corresponds to the plural form of the attribute, service, or constant type defined by the class. [62]

```
// all the name should be group related. it's easy for
// people to read and understand the source code
public class LineMetrics {
    public LineMetrics();
    public abstract int getNumChars();
    public abstract float getAscent();
    public abstract float getDescent();
    public abstract float getLeading();
    public abstract float getHeight();
}
```

FIGURE 24: THE NAME OF THE CLASS/INTERFACE SHOULD GROUP RELATED

Alternatives: [63]

1. *Interface name can start with letter "I".*
   Coad and Mayfield (1997) suggest appending the letter 'I' for the first letter of interface names, so the name of interface can be `IInterfaceName` or `IIterfaceNewName`. This approach helps to distinguish interface names from class and package names. This potential naming convention makes the class diagrams easier to read. This interface

naming convention is also popular for Microsoft's COM/DCOM architecture.

2. *Interface name can end with letter "Ifc" as the last part of the name.*
   Lea (1996) suggests appending 'Ifc' to the end of an interface name, so interface name following this naming conversion will be like `NewInterfaceIfc` or `ThreadInterfaceIfc`. This approach is not common used in the really IT industry.

When define the abstract class, public and protected interface. It is important to minimize the size of the types. There are several reasons for this:

1. Learn ability.
2. Reduced coupling.
3. Greater flexibility.

Minimizing the size of the types can help the programmers to learn how to use a class; the only thing they need to do is just to understand it's a public interface. The smaller the public interface, the easier it is to learn a class. The smaller the public interface is, the greater the encapsulation and the flexibility are.

5.2.5 METHOD NAMES

Method names should be also in infix-caps style, and start with a lower-case letter, and capitalize the first letter of any subsequent word in the name. All other characters except the first letter of any subsequent word in the name are lower-case. The Sun Java coding conventions recommend not using underscores to separate words. This is identical to the naming convention for non-constant fields; however, it should always be easy to distinguish the two from context [64].

The method name also should be verbs which can define the actions or operations for the methods. The JavaBeans specification establishes standard naming conventions for methods that give access to the properties of a JavaBeans implementation. These conventions should be applied when naming methods in any class, regardless of whether it implements a Bean [65].

```
// Method names with good style
drawTringle();
addLayout();
helloWorld();


// Method names with bad style
DrawTringle(); // the first character should be in low-case
keyboard(); // not meaningful
fileClosing(); // verb phase it's better to be closeFile();

// Method to get the value of property
// get + Property name
getHeight();
getPhoneNumber();

// Method to get the value of bool property values
isVisible();
isResizeble();
```

FIGURE 25: DIFFERENT TYPES OF COMMENT

5.2.6 VARIABLE/FIELD/PARAMETER NAMES

According to the Sun Java coding standards, variable/field/parameter names should use lowercase for the first word and capitalize only the first letter of each subsequent word that appears in a variable name. The name of those components also should be nouns. Give collections of objects a name that corresponds to the plural form of the object type. This enables programmers who read the source code to distinguish between variables representing multiple values from those representing single values. [66]

Qualify field variables with "this" key words when using them; this will help the developer who will read the code to distinguish the field variables from local variables which make the code much easier to understand and the code become more clearly.

```
class TestClass {

    private Address oldAddress;
    private int value;

    Address[] newAddress = new Address[10];

    public int fetchValue(int oldValue) {
        this.value++;
        return value;
    }
}
```

FIGURE 26: VARIABLES/FIELD/PARAMETER NAMING CONVENTION EXAMPLE

5.2.7 CONSTANT NAMES

Same as C++/C, according to several different versions of Java coding standards, the names of constant variables should use uppercase letters for each word and separate each pair of words with an underscore when naming constants.

5.3 DOCUMENTATION CONVENTIONS

Java source files contain all the code for the applications. A Java source file should contain only one public class or interface definition and it may contain several non-public classes. According to the Sun Java standards, there is no limit to the source file size; however it is better to keep it less than 2000 lines. The size of the method should also be kept to less than 200 lines. For the most trivial Java projects, it is better to keep the source files in a version management system. [67]

5.3.1 PROGRAMMING STANDARDS FOR COMMENTS

Comments can give an overview of the code and provide additional information that is not obviously shown in the source code. The comments should contain information that is relevant to reading and understanding the program. It is a good habit to put Java comments in the source code to make the code easier to enhance and improve readability. Comments in Java code are only used for readability. The Java compiler doesn't deal with the contents of comments, so the size and efficiency of the complied application won't be affected by the volume of comments in the source code. Java programs can have two kinds of comments: *Implementation comments* and *Documentation comments*. [68]

Implementation comments are the comments which are used for commenting out source code or the comments about the particular implementation. It has two different formats: *Line comments* and *Block comments*. *Documentation comments* are used for commenting out the information (usage, functionality) for a particular piece of source code. [69]

5.3.2 LINE COMMENTS

Line comments are always very short. The line comments can appear on a single line which is indented to the level of the code that follows. They shouldn't be used on consecutive lines for text comments; however, they can be used in consecutive multiple lines for commenting out a block of code. [70]

```
// this is some informations about comment
/* some comments here */

// this.button1.Location = new System.Drawing.Point(101, 22);
// this.button1.Name = "button1";
// this.button1.Size = new System.Drawing.Size(75, 23);
this.button1.TabIndex = 0;
this.button1.Text = "button1";
this.button1.UseVisualStyleBackColor = true;
```

FIGURE 27: SINGLE LINE COMMENT

5.3.3 BLOCK COMMENTS

Block comments start with a block comment type "/*". Everything after the forward "/*" will be treated as comment until the characters "*/" end the comment. Block comments are used to provide descriptions of files, methods, data structures and algorithms. Block comments can be used at the beginning of each Class and before each method. They can also be used in other places, such as within methods. Block comments inside a method should be indented to the same level as the code they describe. Block comments can't be nested as they will cause compile errors. [71]

```
/* several different comment
 * comment 2
 * comment 3
 */
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
```

FIGURE 28: BLOCK COMMENTS

The length of each line of the comment can't be very long. There is no need to describe every line in the source code. It is a good habit to indent the comments and keep comments relevant. It is necessary to update the comments after modifying code. If a comment is no longer relevant, the user should either modify or remove it. [72]

5.3.4 DOCUMENTATION COMMENTS

Java documentation comments are used to describe Java classes, interfaces, constructors, methods and fields. Each fragment of Java documentation comments are placed inside the comment delimiters /*……*/, and all the Java documentation comments should appear just before the declaration of methods, classes and interfaces. Java documentation comments can't be put into a method or constructor definition block. It is very important and useful to identify and describe any outstanding problems which associate with a class or method. Indicate any replacements or

workarounds that exist would bring a lot of benefit to the user or client. It is greatly helpful to solve issues and maintain programs if indicate the information. [73]
Java documentation comments should contains public, protected, package, and private members. A basic supply of documentation comments for all members in the source file should include packages name and access details. The comments will help the user or client to gain better control of the project and also improve the lower level documentation. [74]

SUMMARY	DETAILS
Summary description for class, field, interface, and method	Provide a summary description for each class, interface, field, and method. Every class, interface, field, and method should be preceded by a documentation comment that contains at least one sentence that acts as a summary description of that entity.
Describe the signature of each method	Fully describe the signature of each method. The documentation for each method should always include a description for each parameter; each checked exception, any relevant unchecked exceptions, and any return value.
Coding examples	One of the easiest ways to explain and understand how to use software is by giving specific examples. Try to include a simple example in each nontrivial class and method description. Use the HTML <pre>…</pre> tags to maintain the formatting of each example:
Document public, protected, package, and private members	Supply documentation comments for all members, including those with package, protected, and private access. This allows for the generation of detailed, implementation level documentation. The developer who must learn and understand your code before implementing an enhancement or bug fix will appreciate your foresight in providing quality documentation for all class members, not just for the public ones.
Provide a summary description and overview for each package.	The JavaDoc utility provides a mechanism for including package descriptions in the documentation it generates. Use this capability to provide a summary description and overview for each package you create.

TABLE 11: COMMENTS CONTENT DESCRIPTIONS [75]

"*A good description of a type and its methods and fields should be as apparent as better.*" [76] The description should include the purpose, usage and the role it plays in the project. For the same type, same method or fields, a different developer may have a different way of implementation and with the clear definition of the types, methods and fields it will help greatly and reduce the work to do. [77]

ATTRIBUTE	DESCRIPTION	AVAILABILITY	REQUIRED
sourcepath	Specify where to find source files	All	At least one of the three or nested <sourcepath>, <fileset> or <packageset>
sourcepathref	Specify where to find source files by reference to a PATH defined elsewhere.	All	
sourcefiles	Comma separated list of source files -- see also the nested source element.	All	
destdir	Destination directory for output files	All	Yes, unless a doclet has been specified.
maxmemory	Max amount of memory to allocate to the javadoc VM	All	No
packagenames	Comma separated list of package files (with terminating wildcard) -- see also the nested package element.	All	No
packageList	The name of a file containing the packages to process	All	No
classpath	Specify where to find user class files	All	No
Bootclasspath	Override location of class files loaded by the bootstrap class loader	All	No
Classpathref	Specify where to find user class files by reference to a PATH defined elsewhere.	All	No

TABLE 12: A PART OF JAVADOC PARAMETERS [78]

It is good habit to describe the program before writing the code. Simply said, design before working. It is much better to get a reference document well prepared before the development. Documenting everything will save lots of time and effort. With clearly documented purpose, usage and behavior definitions of each class, method and interface, the development will be much easier and more standardized.

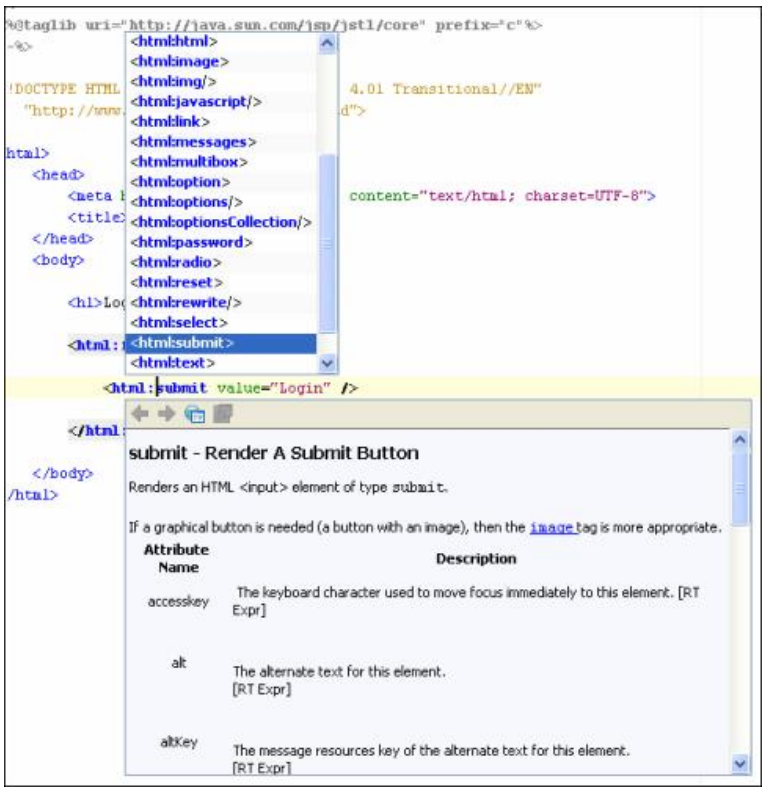


FIGURE 29: JAVADOC TOOLS

It is good to provide a summary description and overview for each package. Java documentation comments with an overview of the project structure and package in it is also useful. With this capability, the developer can easily provide a summary description and overview for each project crafted.

5.3.5 BLANK LINES AND SINGLE BLANK SPACE

Blank lines are an important part of the source code. Blank lines can improve readability by grouping sections of the code that are logically related.

```
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // button1
    //
    this.button1.Location = new System.Drawing.Point(101, 22);
    this.button1.Name = "button1";
    this.button1.Size = new System.Drawing.Size(75, 23);
    this.button1.TabIndex = 0;
    this.button1.Text = "button1";
    this.button1.UseVisualStyleBackColor = true;
    this.button1.Click += new System.EventHandler(this.button1_Click);
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(292, 270);
    this.Controls.Add(this.button1);
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);

}
```

FIGURE 30: FIGURE: BLANK LINE USED IN .NET (C#) EXAMPLE

The related coding can be grouped together by using the blank lines. It is easier for the developer to understand the coding and the programmer can relate the coding by checking the blank lines positions.

A blank line should also be used in the following places: [79]

1. After the copyright block comment, package declaration, and import section.
2. Using the blank lines between class declarations.
3. Using the blank lines between method declarations.
4. Using the blank lines between the last field declaration and the first method declaration in a class.
5. Using the blank lines between before a block or single-line comment, unless it is the first line in a block.

A single blank space also can improve the readability of the source code. A single blank can be used in the following cases.

1. Between two adjacent keywords.
2. Between a keyword or closing parenthesis, and an opening brace" {".
3. Before and after binary operators.
4. After a comma in a list.
5. After the semicolons in a for statement

6. A single blank space can't be used in several different cases. If a single blank is used in these cases, some coding errors will occur.
7. A single blank space can't be used between a method name and its opening parenthesis.

It should not be used between a unary operator and its operand. Using white space correctly and properly will make it easier to understand the source code.

5.4 PROGRAMMING CONVENTIONS

5.4.1 DESIGN OF JAVA COMPONENTS (CLASS/METHOD/INTERFACE)

It is a good habit and necessary to implement classes and methods as small as possible. Small classes and methods are easy to design, understand, test, document, implement and use. Small classes generally have fewer methods and represent simpler functions. Interfaces or a high-level abstract class of those classes tend to exhibit better flexibility. [80]

A Java subclass is a class which inherits a method or methods from a Java super-class. Subclasses are very important, the principle of designing a good subclasses is "*Subclass may be used anywhere their super-classes may be used*" [81]. A subclass can change or restrict its behaviors by changing inherited state and behavior from its super-class. It can only use the component inherited from its super-class but the subclass can modify or override it. As the developer drops down in the hierarchy, the classes become more and more specialized. [82]

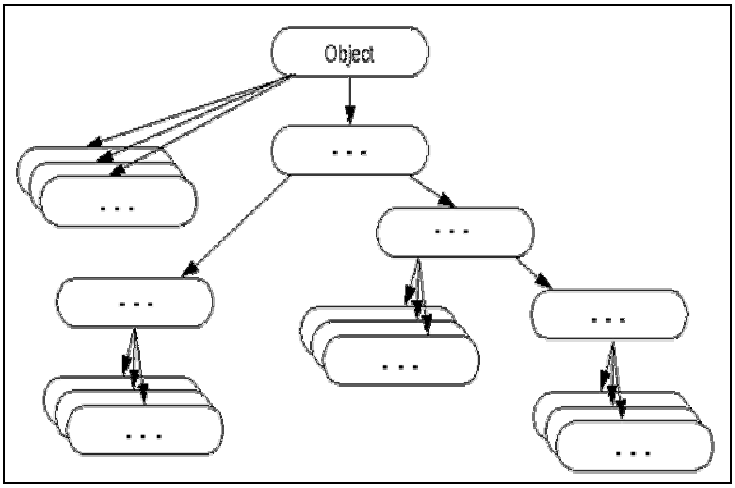


FIGURE 31: SUBCLASS DESIGN

It is important to make all fields private to protect the consistency of Java components. The `Private` keyword makes only the class itself who can make changes to it. The

developers can make some methods for each private field; those methods are called field-access-method. [83]

```
private int i;

public int getI(){
   return this.i;
}

public void setI(int i){
   this.i = i;
}
```

All the member data in the class can be accessed through methods which are in the object. This minimizes coupling between Java objects, which enhances maintainability of the applications.

5.4.2 PROGRAMMING STANDARDS IN MULTI-THREAD

"In computer science, concurrency is a property of systems in which several computations are executing simultaneously, and potentially interacting with each other. The computations may be executing on multiple cores in the same chip, preemptively time-shared threads on the same processor, or executed on physically separated processors." [84]

Java also supports multi-thread processing. In Java, concurrency is implemented as threads and a lot of software can benefit from the use of multi-threads in their implementations. In a concurrent model of execution, the concurrency component of the software are divided into two or more processes or threads by the Java Virtual Machine (JVM) and each thread or process is in its own sequence of statements. [85]

The software which includes concurrency functionality may consist of one or more processes and a process may consist of one or more threads. Each thread will run on two or more Java Virtual Machines on two or more processors in a single machine, or interleaved on a single processor. Multiple-thread has a lot of advantages: [86]

1. Better resource utilization.
2. Simpler program design in some situations.
3. More responsive programs.

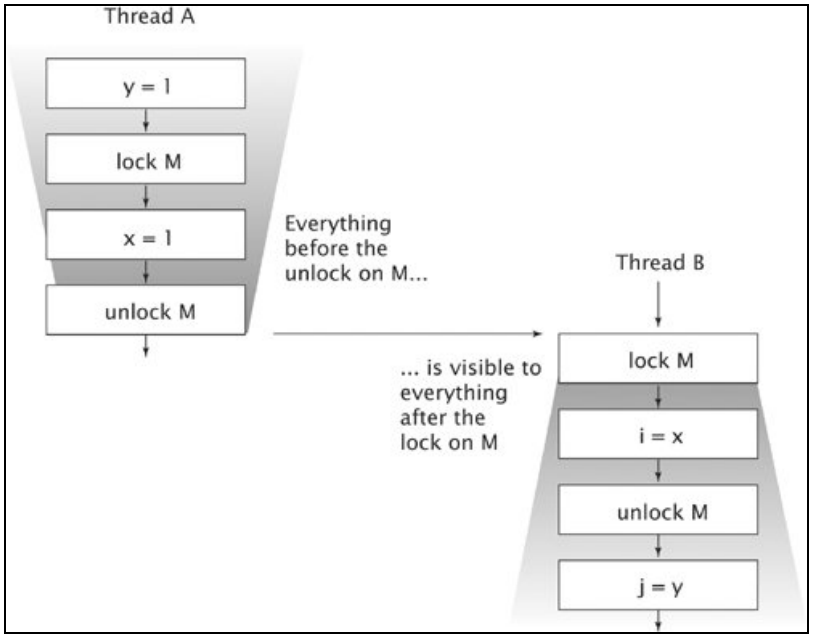


FIGURE 32: CODING STANDARD FOR JAVA THREADS

5.4.3 CODING STANDARD FOR SYNCHRONIZED

Good programming standards can also improve the performance of a multiple-thread for Java source code. "`Synchronized`" is one of the most important key words for Java multiple-thread programming. How to use it (when, where) is important for improving the performance and coding appearance. Use the key word "`Synchronized`" wrappers to provide synchronized classes. A class with "`Synchronized`" provides the same interface as the original one, but its methods are synchronized, which means it is thread-safe. A static method of the synchronized-wrapped class also provides access to the synchronized by default. [87]

```
protected void processRequest () {
    Request request = getNextRequest();
    RequestId id = request.getId();

    synchronize(this) {
        RequestHandler handler = this.handlerMap.get(id);
    }
    handler.handle(request);
}
```

FIGURE 33: SYNCHRONIZED IS USED WITHIN THE METHOD

Deciding when to use the key word "`Synchronized`" for a method is a problem for developers. The developers should try to avoid synchronizing an entire method if the

method contains significant operations that do not need synchronization, a better way is just to synchronize a part of the code in the method. A method with the synchronized keyword acquires a lock on the related components at the beginning of the method and holds that lock until the end of the method. The "`Synchronized`" keyword may lock the code where synchronized is not required. In most of the methods in Java source code, only a few operations within a method may require synchronization. In these situations, the method-level synchronization should not be used as it will greatly reduce the performance of the source code. [88]

5.5 PACKAGE CONVENTIONS

"*A Java package is a used for organizing Java classes into namespaces similar to the modules of Modula*." [89]

Typically source code files or compiled Java files in one directory would have different functionality from those in other directories. In a GUI application, it is very common to find a directory with the name "UI" (user interface); it means that this directory contains files related to the presentation part of the application. It is same in a Java package, the related source files or files from the same organization can be packaged together. A package should provide a unique namespace for the types which it contains, this keeps the package distinct from other packages. There are three aspects to consider when working with packages [90]

1. *Publishing interfaces*
   It is necessary to keep the size of the published interfaces small. This can make sure that the published interfaces will actually be from third-parties.

2. *Granularity*
   Granularity is one of the most important points for defining package behaviour. Both too fine-grained and too coarse-grained behaviour are not good for defining a package. Those defining package behaviour make the package hard to use even unusable. So determining the property granularity is very important for defining a package.

3. *Units in large size*
   If the size of the unit is very large, a package is necessary which can reduce the size of the unit.

5.5.1 PACKAGE ACCESS PROTECTION

Classes can access all the classes and members within the same package if the visited class and members are declared with default access level or higher level. Default

access is enforced when neither the public, protected nor private access level is specified in the declaration. Classes can not visit the members declared with default access if these two members are not in the same package. When choosing the accessing level, there are points that should be taken into consideration: It's important to make sure that using the most restrictive access level which makes sense for a specified Java component. In most cases the private level should be used as the access modifier unless there is a good reason not to. Avoid public fields except for constants. This is necessary as the public access level tend to link the implementation of the class which will limit the flexibility in changing the source code. [91]

ACCESS LEVEL	CLASS	PACKAGE	SUBCLASS	WORLD
Public	Y	Y	Y	Y
Protected	Y	Y	Y	N
default	Y	Y	N	N
private	Y	N	N	N

TABLE 13: ACCESS LEVEL DESCRIPTIONS

Well-designed Java software consist of a lot of Java files, Java APIs and other Java elements and all of these items should be clearly defined individual tasks in the system's overall purpose. Such software is more extensible and flexible than a system with bad design. In order to develop the software with such good extensible and flexible, package the property Java file, Java API and other Java elements are necessary. [92]

5.5.2 REUSABILITY AND MAINTAINABILITY

Reusability is one of the most important purposes of object-oriented software design. When designing the software, it is important to place classes and interfaces what will be usually used together in the same package. In most cases such classes are so closely coupled that the developer can't use one class without usually using the other and placing tightly coupled classes in the same package results in a more cohesive package.

Here are some examples whose classes are so closely coupled. [93]

- Containers and iterations.
- Database tables, rows, and columns.
- Calendars, dates, and times.
- Points, lines, and polygons.

Those classes are closely coupled, so most developers will package the related classes to the same package which is reusable in the future.

5.5.3 HOW TO GET WELL-DESIGNED JAVA PACKAGE

Sometimes, placing these classes in the same package when some classes are not closely coupled, however they are still jointly affected by a required change to system behaviours. This is because the non-closely coupled classes may need some changes and they may work to provide a coarse-grained service, in this case, these classes may not be closely coupled. So if the class needs some changes, they should be as closely located to each other as possible which requires them to be in the same package.

By packaging classes appropriately not only can this make the software component reusable, it also can make the system easy to maintain. As the package can help to reduce the changes made to the system, it results in more timely and reliable software modifications. [94]

Classes which change together should be in the same package and classes that are closely coupled to each other are likely affected by some changes or their functionality. If any changes are made to the interface, abstract class or high level abstracted class this will lead to a lot of changes to the related classes. Placing the closely coupled classes, files and other Java components to the same package can help mitigate the change-management risk. [95]

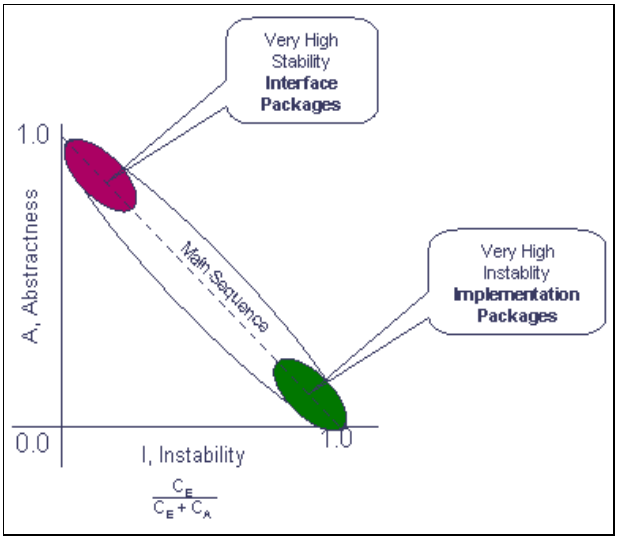


FIGURE 34: JAVA PACKAGE DESIGN

A class which can't be reused should be placed in a separate package. In most cases those Java components don't change frequently, packaging these classes separately means multiple packages use the individual classes. It makes the Java component more stable.

In order to get well-designed software, the top-level design must stabilize as quickly as possible. No software development manager can achieve a good design plan, estimate, schedule, and allocate resources if the architecture of the software is always changing.

Once the design of the high-level architecture of the software is complete, the software development manager can use packages to separate the stable parts of the design from the volatile implementation. Placing the property files, APIs and Java components into Java packages and capturing the high-level abstractions of the design. And then the programmers should create packages by placing the implementation of those abstractions into separate packages which relate to the high-level abstract packages. [96]

When designing a Java package, besides the three aspects discussed above. The following points are also very important.

STATE	ACTIONS
Containing Change	Placing closely coupled classes in a single package can limit all changes to a single package. Containing changes to a single package favours maintainability
Class coupling	When placing closely coupled classes in the same package, the related classes also should be placed to those packages. Ignoring overall system coupling may actually increase the coupling between packages.
Contention	During development of the source code, it's common to morph the package structure accordingly.
Early in the life of the application	The developer may choose to facilitate development and aid maintenance by creating smaller packages. The package structure should be designed to have the goal of reusability.

TABLE 145: TIPS ON HOW TO PACKAGE [97]

5.6 CONCLUSIONS

This chapter explains how different elements of the new Java coding standard have been defined and improved. This coding standard contains four parts which are the naming conversion, the documentation conversion, the programming conversion and package conversion. With the detailed explanation of each concept above, the new coding standard has been articulated.

6. TECHNICAL OVERVIEW

6.1 THE JAVA PROGRAMMING LANGUAGE

"*Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform*" [98] and Java programming language contains 5 primary goals. [99]

1. The Java programming language is simple, object oriented, and familiar.
2. The Java programming language is robust and secure.
3. The Java programming language is architecture neutral and portable.
4. The performance of the Java programming language is high.
5. The Java programming language is interpreted, threaded, and dynamic.

In this project, the implemented CODE-CHECK also attains those goals.

The Java platform allows developers to create and run programs which are written in the Java programming language. As one of the primal goals of the Java programming language is "Write once, run anywhere", the Java platform is not specific to any processor or operating system. A set of standard libraries have been developed for various hardware and operating systems which makes Java programs can run nearly on all operating systems. [100]

VERSION	DESCRIPTIONS
Java Card	Java card refers to a technology that allows small Java-based applications (applets) to be run securely on smart cards and similar small memory footprint devices.
Java 2 ME	J2ME specifies several different sets of libraries (known as profiles) for devices which are sufficiently limited that supplying the full set of Java libraries would take up unacceptably large amounts of storage.
Java 2 SE	J2SE is general for purpose use on desktop PCs, servers and similar devices.
Java 2 EE	A number of libraries are implemented which is useful for multi-tier client-server enterprise applications

TABLE 15: VERSIONS OF JAVA PLATFORM [101]

The Java platform consists of several components; different components in the Java platform can provide different usages. The core components in the platform are the Java language compiler, the Java language libraries (APIs) and the runtime environment. [102]

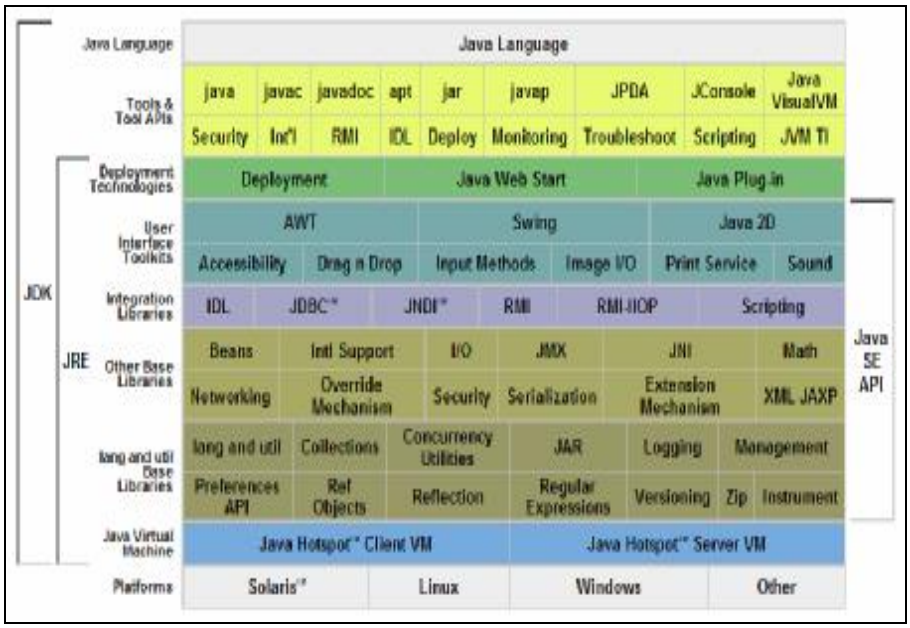


FIGURE 35: JAVA PROGRAMMING LANGUAGE [103]

All these components run on the Java virtual machine. The JVM is an implementation of the Java Virtual Machine Specification (a "*virtual machine*" that executes Java byte code programs). The Java virtual machine interprets compiled Java binary code for a computer's processor or hardware platform. This byte code is the same no matter what operating system the program is running on. The Java programming language is designed to allow the applications which are written in Java to run on any hardware or operating systems without rewritten or recompiled for each separate platform. A Java virtual machine makes this possible because it is aware of the specific instruction lengths and other particularities of the platform. [104]

6.2 ECLIPSE AND ECLIPSE PLUG-IN DEVELOPMENT

Eclipse is a multi-language software development platform that was donated to the open source community by IBM. It provides plug-in architectures which can provide additional functionality for the platform, and most of the plug-ins runs on the top level of the runtime system. However, the platform can also be used to develop software in other programming languages (C/C++, Python, COBOL, Perl and PHP). As it is developed in Java, Eclipse can run on almost any operating system. [105]

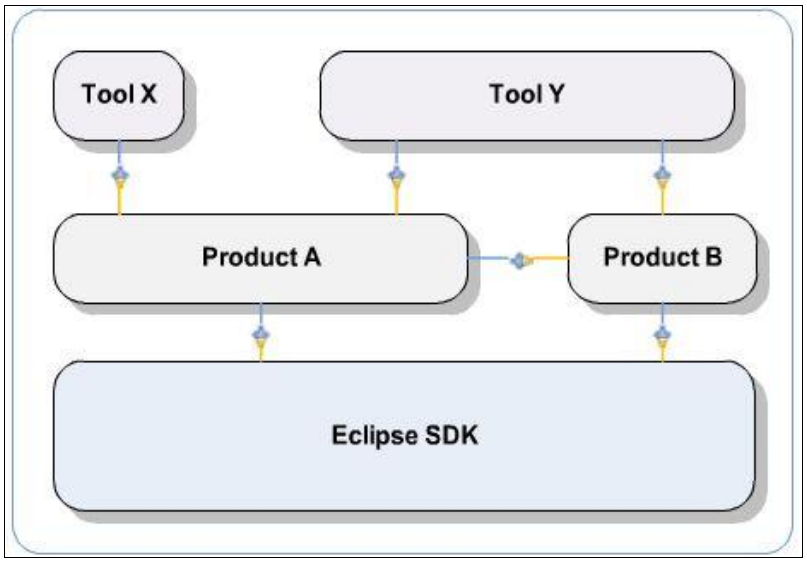


FIGURE 36: ECLIPSE DEVELOPMENT PLATFORM ARCHITECTURE

Eclipse has a lightweight software component framework which allows Eclipse to be extended with other functions. So the plug-in development for Eclipse becomes one of the most important and valuable functions-extended approaches for Eclipse. Eclipse also supports developing any desired extension to the environment with the help provided by the Eclipse SDK. [106]

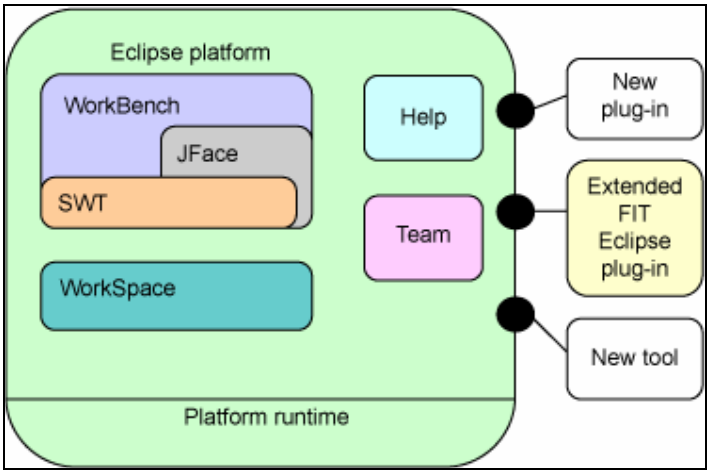


FIGURE 37: ECLIPSE AND ECLIPSE PLUG-IN ARCHITECTURE

The Plug-in development has a specified approach. In most cases, those approaches include:

1. Create Project
2. Create Extensions
3. Create the plug-in function code

4. Restrict the extension
5. Create makers
6. Deploying a plug-in.

In this project, the implementation of the CODE-CHECK will follow these steps. Besides, the architecture of the check-style plug-in is very important, as the development of CODE-CHECK will based it.

6.3 CHECK-STYLE OVERVIEW

6.3.1 INTRODUCTION

Check-style is a source code analysis tool used in software development. It helps developers to write Java source code in a specified coding standard. Check-style checks the source code file in the tool panel and enforces the source code to follow specified coding rules. The Check-style plug-in can do a lot of other jobs as well on the source code. Based on the configuration file, it can affect many aspects of the Java source code. Its main function is to check coding style issues, the latest version of Check-style can find source code design problems, duplicate code, or source code errors. [107]

The Check-style plug-in provides different functions for many different versions of Java coding development tools like Eclipse

IDE / BUILD TOOL	MAIN/INITIAL AUTHOR	AVAILABLE FROM	REMARKS
Eclipse/WSAD	David Schneider	Eclipse-CS Home Page	
Eclipse/WSAD	Marco van Meegen	Checklipse Home Page	
IntelliJ IDEA	James Shiell	Checkstyle-idea Project Page	Provides real-time and on-demand scanning.
IntelliJ IDEA	Mark Lussier	JetStyle Project Page	
NetBeans	Petr Hejl	Checkstyle Beans	Problems with source code are displayed as annotations of the source
NetBeans	Paul Goulbourn	nbCheckStyle	
BlueJ	Rick Giles	bluejcheckstyle home	

		page	
tIDE		Built in	
Emacs JDE	Markus Mohnen	Part of the standard JDEE distribution	
jEdit	Todd Papaioannou	JEdit CheckStylePlugin	
Vim editor	Xandy Johnson	Plugin Homepage	Vim file-type plug-in
Krysalis Centipede	unknown	Checkstyle supported out of the box	
Maven	Vincent Massol	Checkstyle supported out of the box	example report

TABLE 16: PLUS-IN AVAILABLE FOR SEVERAL DIFFERENT IDEs [108]

6.3.2 ADVANTAGES AND LIMITATIONS OF CHECK-STYLE

The specified programming standards adopted by Check-style can help to comply with good programming practices which can greatly improve the code quality, re-usability, readability, and reduce the cost of development and maintenance. The source code which has been checked by the Check-style tool is predictable in its semantics and understood in the same manner across the development team.

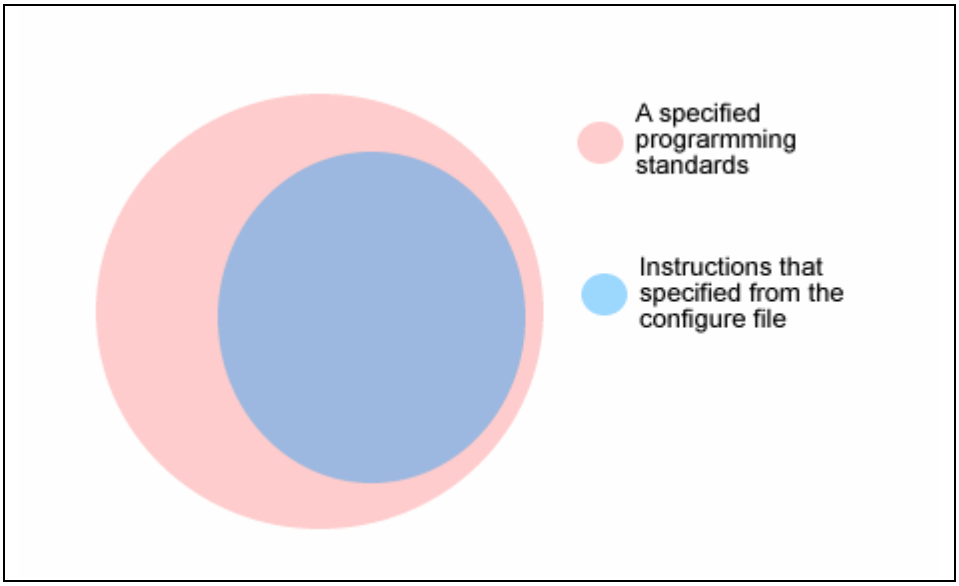


FIGURE 38: INSTRUCTIONS IN THE CONFIGURATION FILE & A SPECIFIED PROGRAMMING STANDARD

The main limitation of the Check-style is the configuration file. The configuration file contains a lot of instructions; however those instructions can't include anything that comes from a specified programming standard. As shown in the figure above, only a part of the specified programming standards can be implemented in the configuration file. However this part of the specified programming standards is the most important set of the standards. So the generated code will be in well-styled-form. The following instance can be edited in the configuration file. [109]

- JavaDoc comments for classes, attributes and methods;
- Naming conventions of attributes and methods;
- Limit of the number of function parameters, line lengths;
- Presence of mandatory headers;
- The use of packets imports, of classes, of scope modifiers and of instructions blocks;
- The spaces between some characters;
- The good practices of class construction;
- The duplicated code sections;
- Multiple complexity measurements, among which expressions.

6.3.3 CONFIGURATION OF CHECK-STYLE

Check-style checks Java source code based on the configuration file. It allows the developers to define a set of instructions which come from a specified coding standard and listed in the XML configuration file. The component of check-style parses the instructions defined in the configuration file and checks the input source code against the specified coding standard. According to the results of the source code checking, the check-style will provide errors, warnings and other messages related to the results. The configuration file uses a DTD XML definition within it. The set of instructions which deal with some particular type of code checking is called a ***module***; the module is always defined like a class. [110]

In the configuration file, there are a lot of attributes, the value of which will be provided to the checking-component. The most common used attributes are listed below:

NAME	DESCRIPTIONS	TYPE	VALUE
`Module`	A module element in the configuration XML document specifies a module identified by the element's name attribute	Null	The value of a module property is a command line.
`Basedir`	base directory name; stripped off in messages about files	String	null
`localeCountry`	locale country for messages	String; either the empty string or an uppercase ISO 3166 2-letter code	default locale country for the Java Virtual Machine
`charset`	name of the file charset	string	System property "file.encoding"
`cacheFile`	caches information about files that have checked ok; used to avoid repeated checks of the same files	string	null
`file`	the name of the suppressions XML document file	String	null
`messageFormat`	message pattern to suppress	regular expression	null
`checkFormat`	check pattern to suppress	regular expression	.* (all checks)
`influenceFormat`	a negative/zero/positive value that defines the number of lines preceding/at/following the suppression comment	regular expression	0 (the line containing the comment)
`tokens`	tokens to check	subset of tokens PARAMETER_DEF, VARIABLE_DEF	VARIABLE_DEF
`ignoreAbstractMethods`	Controls whether to ignore parameters of abstract methods.	Boolean	false
`classes`	Classes that should not be instantiated	String set	{}

TABLE 17: A VERY SMALL SET OF COMMON USED ATTRIBUTES [111]

6.4 CHECKER-STYLE API

The implementation of this project is based on the Style-check plug-in. The Eclipse plug-in is used as the main component. However, the plug-in itself is not efficient, extended features and styles are to be added into this. As the plug-in uses the configuration file, the new coding standard will be extracted and form a properly formed XML file.

6.4.1 CODING PROBLEMS IN THE SOURCE CODE

Coding problems like coding errors are not included in the programming standard; however coding problems will affect the code quality and compiled result. So in the implementation, this problem also will be analysed and resolved. Only one module is provided for resolving this problem. [112]

Equals & Hash-code Problems

In most cases, if the `equals()` method is overridden for an object, the `hashCode()` method should be also overridden for the same object.

6.4.2 CLASS DESIGN CHECKING

Class design is one of the most important things for developing. In this project the most commonly used checking will be implemented. [113]

1. Extension problems check

This module checks whether the class is designed for extension, this attribute will enforce the subclass to implement the abstract method. The checking will be based on the access level of the method and class. It must also ensure that all the non-private, non-static methods of classes that can be extended must have one of abstract, final or having an empty implementation.

```
<module name="DesignForExtension"/>
```

FIGURE 39: EXTENSION PROBLEM CHECKING EXAMPLE

JavaDoc comment checking

This module is used to check for JavaDoc comments for Object source code definitions (Class or interface). Typically, the developer can specify the following properties for the JavaDoc comments

a) Scope

The scope is used to check the visibility scope of JavaDoc comments

b) Author format and version format

By default, it doesn't check for author or version tags. In order to active these, the developer should set the property author format or version format respectively to a regular expression

NAME	DESCRIPTION	TYPE	DEFAULT VALUE
`lineSeparator`	type of line separator	One of "system" (system default), "crlf" (Windows-style), "cr" (Mac-style) and "lf" (Unix-style)	system default
`fileExtensions`	File type extension of the files to check.	String Set	all files

TABLE 18: COMMON USED PROPERTIES FOR JAVADOC COMMENT CHECKING [114]

6.4.3 BLOCK STYLE CHECKING

There are several different kinds of blocks styles for Java. The specified programming standard has a specified block style. In the Style-check tool, several different types of block check can be defined. According to coding styles, two of the defined block styles checked will be implemented in this project. [115]

1. Empty block check

Typically, the empty block is useless in Java. Removing the useless empty block can improve the appearance of the Java source code. Defining the proper empty block instructions in the configuration file can resolve this problem.

```
<module name="EmptyBlock">
    <property name="option" value="text"/>
    <property name="tokens" value="LITERAL_CATCH"/>
</module>
```

FIGURE 40: EXTENSION PROBLEM CHECKING EXAMPLE

2. Nested blocks check

Blocks are a piece of source code which is grouped together. Blocks consist of one or more statements. Blocks within another block will result in nested blocks. Nested blocks can break up programs into several logical work units and proper statements of nested blocks make code more readable and flexible. However sometimes, nested blocks also reduce the performance or readability. So the nested blocks check is necessary.

```
<module name="NestedBlockCheck"/>
```

FIGURE 41: NESTED BLOCK CHECK CONFIGURATION

6.4.4 DUPLICATE CODE DETECTION

"Code duplication is generally considered a mark of poor or lazy programming style." [116]

In judging a good coding standard, the most common aspects are reusability, flexibility and maintenance ability of the code. One big issue that affects the quality of the code is the removal of duplicate code. There are three types of problems related to duplicate code:

1. Code bulk affects comprehension
2. Purpose masking
3. Update anomalies

One problem the developers will face is how to avoid duplicate code. Duplicate code detection helps the developer to find duplicate code that has been generated by cloning (Copy/Paste) or other duplicate actions. Reducing duplicate code through improving the coding style of the source code is one of the best ways.

The check-style tool provides an attribute: `StrictDuplicateCode`, which is used to facilitate checking large amounts of code. It supports multiple languages, but needs very high performance. In order to reduce duplicate code in the source file, the attribute `StrictDuplicateCode` should be used in the configuration file.

```
<module name="StrictDuplicateCode">
   <property name="min" value="15"/>
 </module>
```

FIGURE 42: IMPROVE THE CODING STYLE TO REDUCE THE DUPLICATE CODE

With the help of the attribute `StrictDuplicateCode`, most of the source code issues related to duplicate code is improved, and the style of that code is greatly improved.

6.4.5 NAMING CONVENTION IMPROVEMENTS

The check-style tool provides intelligent functionally that can improve the naming convention of the source code. Naming conventions are one of the most important parts of a specified programming language, and improving the naming conventions can greatly improve the coding style. According to the generated coding standard, a series of definitions can be configured to the XML file. There are several different

modules included in the check-style components. Each of these naming convention modules can validate identifiers for specific Java source code elements and the valid identifiers for a naming convention module are specified by its format. [117]

MODULE	VALIDATES IDENTIFIERS FOR	DEFAULT VALUE OF FORMAT
AbstractClassName	abstract classes	^Abstract.*$\|^.*Factory$
ClassTypeParameterName	class type parameters	^[A-Z]$
ConstantName	constants (static, final fields)	^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$
LocalFinalVariableName	local, final variables, including catch parameters	^[a-z][a-zA-Z0-9]*$
LocalVariableName	local, non-final variables, including catch parameters	^[a-z][a-zA-Z0-9]*$
MemberName	non-static fields	^[a-z][a-zA-Z0-9]*$
MethodName	methods	^[a-z][a-zA-Z0-9]*$
MethodTypeParameterName	method type parameters	^[A-Z]$
PackageName	packages	^[a-z]+(\.[a-zA-Z_][a-zA-Z0-9_]*)*$
ParameterName	parameters	^[a-z][a-zA-Z0-9]*$
StaticVariableName	static, non-final fields	^[a-z][a-zA-Z0-9]*$
TypeName	classes and interfaces	^[A-Z][a-zA-Z0-9]*$

TABLE 19: DESCRIPTIONS OF NAMING CONVENTION MODULES [118]

According to the setting of the XML configuration file, different naming convention will be checked in the source. Invalid or bad naming convention will be notified to the developer by a warning from the check-style plug-in.

6.4.6 VISITOR PATTERN IN CHECKSTYLE

The Visitor design pattern is a pattern for separating an algorithm from an object structure upon which it operates. The result of this separation is the ability to add new operations to existing object structures without modifying those structures. So with the help of the visitor pattern, the source code can follow the open/closed principle. [119]

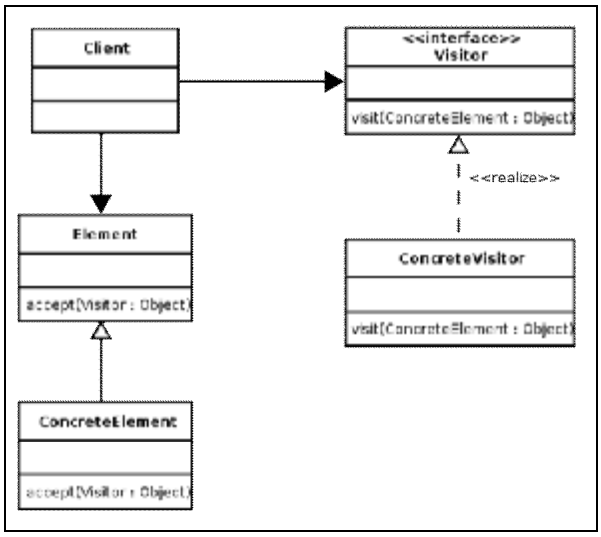


FIGURE 43: SURVEY FOR THE IMPORTANCE OF PROGRAMMING STANDARD [120]

The Check style's kernel does not implement any functionality that can check the coding style of the source code. Instead, the Check style's kernel needs a set of modules that contains several Check interfaces. The interface provides some helper methods. With the help of the visitor design pattern, the Check provides methods that takes an AST as an argument and performs the checking process for that AST. In this project, a number of visitor patterns will be used when implementing the new plug-in called **CODE-CHECK**. As the CODE-CHECK needs to use and extend the function of the check-style APIs, A specified pattern is necessary. [121]

6.5 FACTORY METHOD PATTERN

" The factory method pattern is an object-oriented design pattern. Like other creational patterns, it deals with the problem of creating objects (products) without specifying the exact class of object that will be created." [122]

The factory method pattern separates the creating actions from the major class, this makes the source code more flexible. A Factory method pattern is a simplified version of an abstract factory pattern. A normal Factory method pattern contains three parts: `Creator`, `ConcreteCreator` and `Product`. [123]

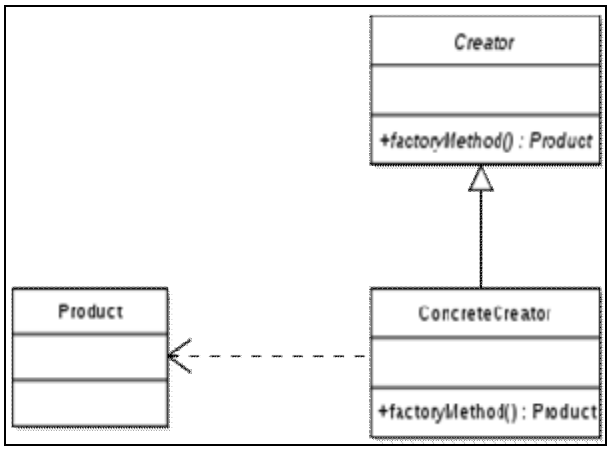


FIGURE 44: SURVEY FOR TFACTORY METHOD PATTERN CLASS DIAGRAM [124]

- **Product:** Product is the object which is created by the factory method in ConcreteCreator.
- **Creator:** Creator is an interface which defines the factory method. The factory method returns an object of type Product.
- **ConcreteCreator:** ConcreteCreator is a class that implements the Creator interface. It overrides the factory method and returns an instance of a Product.

The benefits of this design pattern is [125]

- The Factory method pattern can extract the object creating code from the source, this improves the flexibility of the code.

- It enables the subclasses to get extended functions; this means the extended version of subclasses can be created without change the code in `ConcreteCreator`.

In this project, a number of factory method patterns will be used in the function module development, as this pattern provides a more flexible structure which allows the developer to add new features without changing any other code.

6.6 CONCLUSIONS

This chapter introduces the technical background which will be used in the implementation of this project. The first two sections introduce the Java development environment, the Eclipse development environment and the Eclipse plug-in. The third and fourth sections introduce the check-style APIs and the check-style module development; these two sections give an overview on how to use check-style APIs and how to develop a check-style structured module. This project use a number of design patterns, a frequently used design pattern is described in the last section, which gives a “factory method pattern”. Based on those patterns, the new plug-in is implemented to match the new coding standard.

7. IMPLEMENTATION

7.1 INTRODUCTION

Based on the coding standard from the research in Chapter 5, an Eclipse plug-in which is called **CODE-CHECK** will be implemented. CODE-CHECK is a plug-in which can not only check the code style; it also can do code refactoring and improve the performance of the Java source code. CODE-CHECK will use the APIs from a similar plug-in which is called Check-style. However, CODE-CHECK provides more functions. Code-Check has three components: a coding review component, a code refactoring component and a performance component. The coding review component can perform a code structure check based on the existing coding standard. This component extends to functions of check-style. The other two components provide new functions: code refactoring and performance improvement. Code-check is more powerful and helpful with these functions.

7.2 DESIGN OF CODE-CHECK

7.2.1 CODE-CHECK ARCHITECTURE

The Code-Check implementation is based on an Eclipse plug-in called Check-style. Check-style is an Eclipse plug-in that provides coding review functions, however Check-style provides only the reviewing for the Sun Microsystems version of Java. In this project, extended functions are implemented to fit into the new improved coding standard. The new Code-Check plug-in provides more functions and also many self-defined rules and styles have been added. Below is an architecture that explains the relationship between the check-style plug-in and the Code-Check plug-in and also the main extended functions that the Code-Check provides.

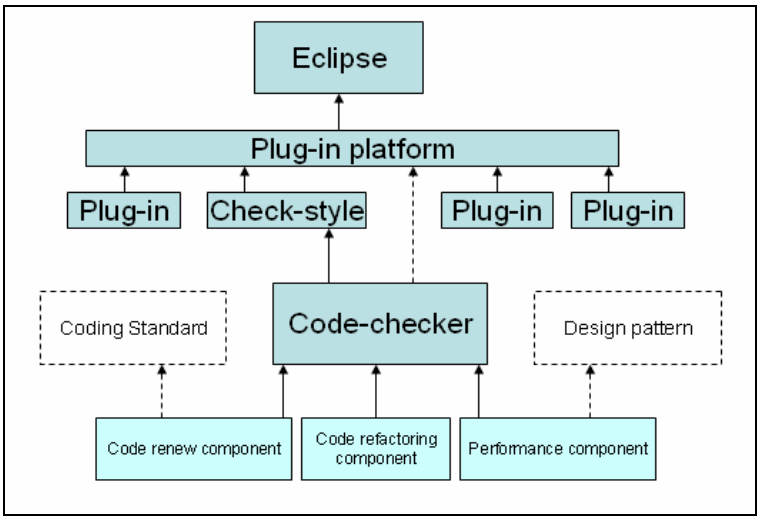


FIGURE 45: CODE-CHECK ARCHITECTURE

Extended functions of Code-Check:

1. Provide a coding review for self-defined rules and style.
2. Provide functions that can improve the performance of source code by using the pre-existing design patterns.
3. Provide a code refactoring function (A good coding refactoring tool is intelligent and complex, in this project a model with three rule-checks will be implemented and the improvement of more intelligent should be implemented in future work).

   Code refactoring components provides:
   1. Warning the developers to rename a method, variable, class or other Java elements if the source code has naming problems.
   2. Warning the developers to do "Code moving" if the source code has structure problems.
   3. Warning the developers to extract methods or classes if there are too many lines in them.

As the Code-Check has three major functions, the system will be designed to have three components: the Coding review component, the code refactoring component and the performance component.

The "*Coding review component*", this is the most important component in this project. The component is used to do coding reviews, and then it provides warnings and simple advice to the Java developers. The coding reviews will be done based on the coding standards as crafted in Chapter 5.

The "*Code refactoring component*", this is a complex component. The challenge of this component comes from its high intelligence and rule indeterminacy. The component in this project does the code refactoring based on the rules from the book "***Refactoring***". [126]

The "*Performance component*" is a component that can give advice on the existing source code. This component can be also very complex as the existing design patterns are complex.

7.2.2 USE CASE DIAGRAMS

There are two user roles involved in this project; one is a new version of the Java programming standards provider, and the other is the software developers who will use the implemented source coding standard. The Java programming standards provider is supposed to find anything that can improve the quality of the programming standard. At the same time, she should also update the Code-Check configuration file which affects the implementation of the programming standards.
With the limitation of the Code-Check plug-in, the programming standards provider should also add new features to the Code-Check configuration file. Another user is the software developer who is willing to use the new version of the standard. When she

writes the source code, the Code-Check plug-in will provide warnings and errors if the source doesn't following the specified coding standard. She can then edit the code as soon as the warning appears or do code refactoring after she is finished writing the source code.

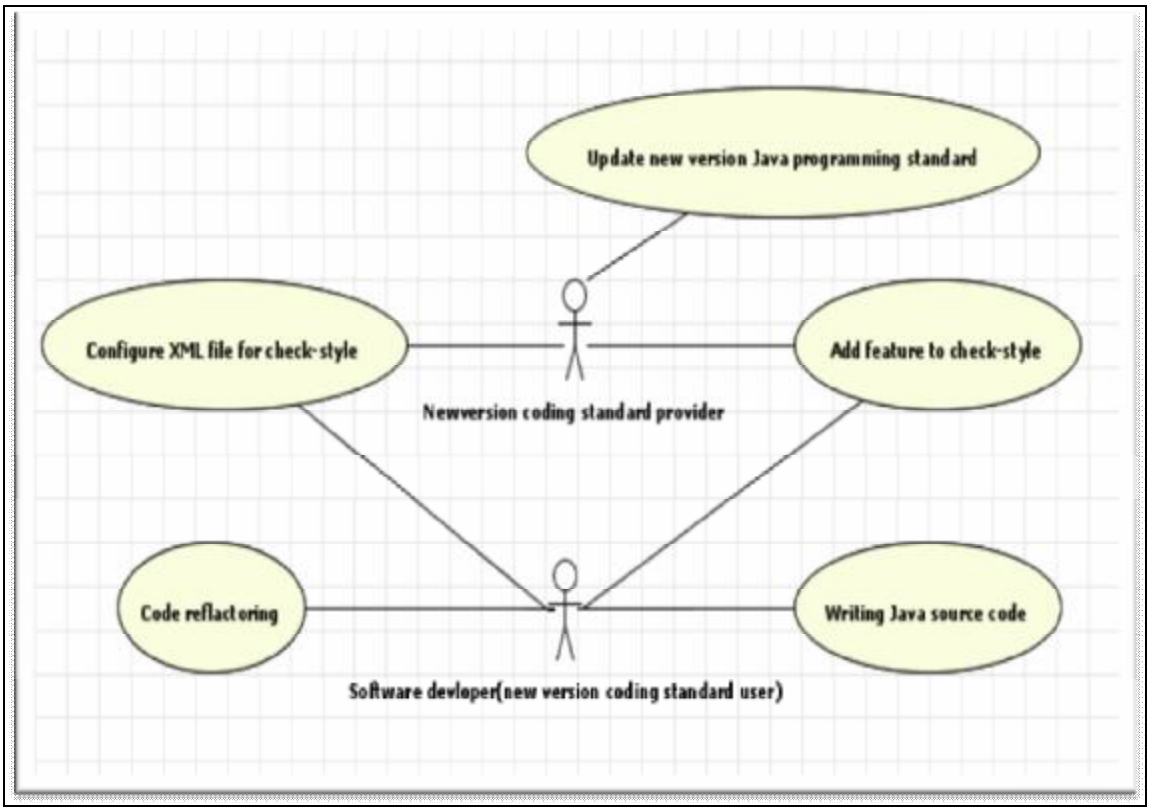


FIGURE 46: USE CASE OF THIS PROJECT

7.2.3 SEQUENCE DIAGRAM

In this project, there are five different objects. The programming standard provider will analyze and sum-up the existing Java programming standard and generate a new version of the Java programming standard. According to the generated Java programming standard, the configuration file of the Code-Check plug-in can be specified. This XML configuration file is used as a set of rules that represent the coding standard. The Code-Check plug-in should then check the source code against the configuration within the XML file. The developer who follows this Java coding standard can get error messages or warnings if her source code doesn't fulfil the requirement of the specified coding standard. However, she can also give any feedback to the coding standard provider for any improvements or changing the code-style plug-in setting. With this feedback, the coding standard provider can improve the coding standards and XML configuration files.

There are many different types of Java coding standards and each standard also have lots of rules. Based on different requirements, the coding standard and the

Code-Check settings should be changed as well. However, one thing that won't change is that the coding standard should be easy to remember, simple to use and widely understand by the software developers.

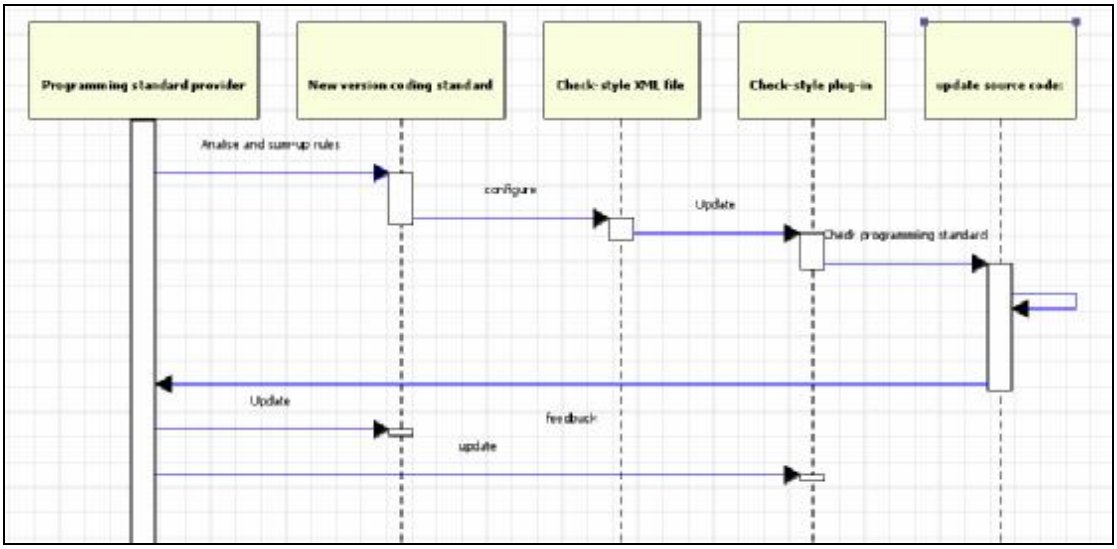


FIGURE 47: SEQUENCE DIAGRAM OF PROJECT

7.3 CODE-CHECK IMPLEMNTATION OVERVIEW

As the Code-Check has three major functions, the system will be designed to have three major components: Coding review component, code refactoring component and performance component. The implementation of Code-Check will be divided to four stages as shown below.

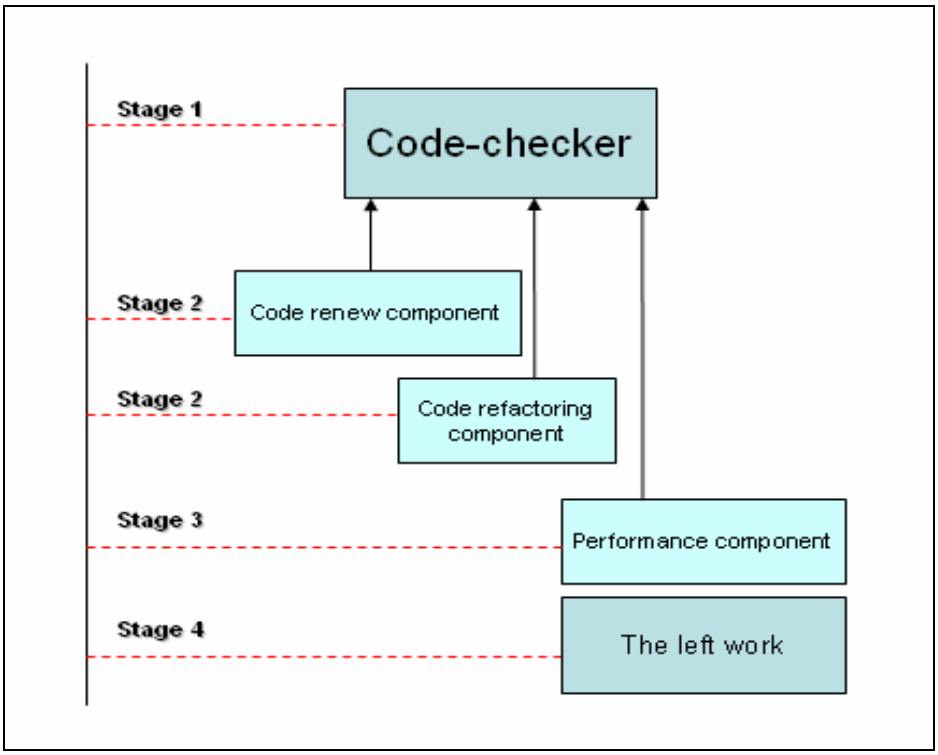


FIGURE 48: DEVELOPMENT OF CODE-CHECK

Stage 1: The Code-Check main application structure design
Stage 2: Coding review component design and develop
Stage 3: Code refactoring component and performance component design and develop
Stage 4: Combined the components and main application platform together

7.4 CODE-CHECK MAIN( )

In this project, the component will be designed as plug-in for the main application of the Code-Check. So the Code-Check not only needs to follow the check-style API structure, it also has to follow the plug-in design pattern. The Code-Check main program contains: a plug-in loader, a class loader, a library finder and a class finder.

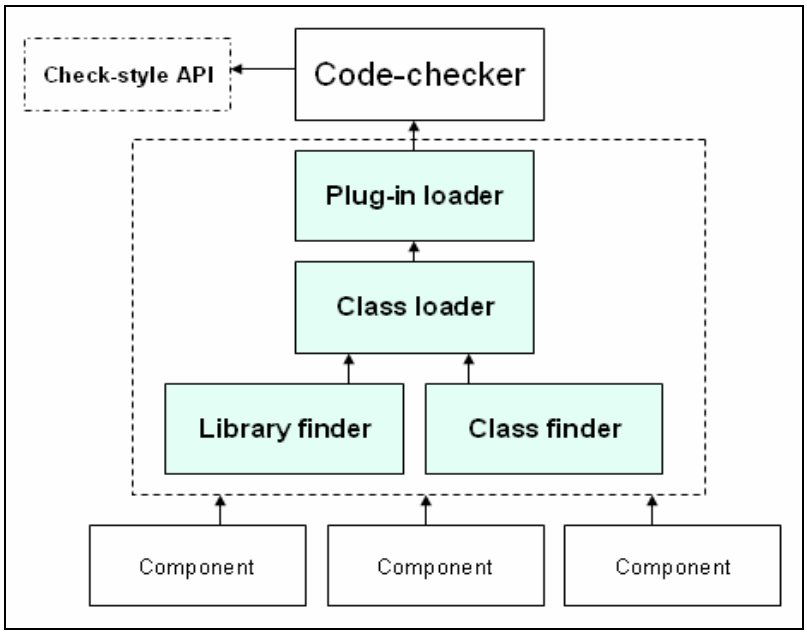


FIGURE 49: PLUG-IN FRAMEWORK ARCHITECTURE

The Plug-in loader is used to load the files of the component. The file contains `component_info.xml` and a component compiled files package. The `component_info.xml` file contains the information of the component, such as name, version, path and id.

The Class loader can find the compiled Java file from the package. Those Java classes must have the same structure which means they should be implementing the same interface. In this project this interface is called "My*Component*".

The Library finder is aimed to find the related resource that will be used in the component. For example the coding review component will use the API from check-style. Then the library finder will find those APIs and tell the class loader about the objects to load the library. The Class finder object will find the component from Java files and pass those messages to the class loader. The Library finder and class finder can be designed together, however as they find the different types of resources. Individual design makes the class smaller and easier to understand.

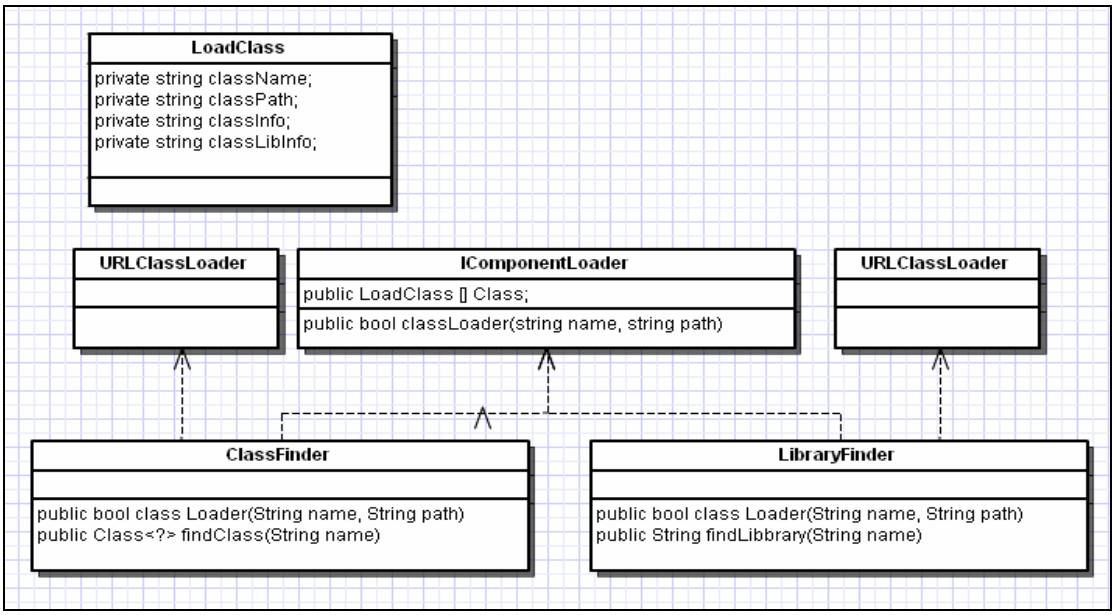


FIGURE 50: CLASS DIAGRAM FOR COMPONENT LOADER.

Code Example:

```
/*******************************/
/* Example of IComponentLoader */
/*******************************/
public interface IComponentLoader {
    public LoadClass [] classArray;
    public bool classLoader(String name, String path);
}

/*******************************/
/* Example of LibraryFinder     */
/*******************************/
Public  class  LibraryFinder  extends  URLClassLoader  implements
IComponentLoader{
   public String findLibrary(String name) {
      Hashtable libs = savedLib.getLibrarys();
          if (libs.containsKey(name)) {
             return libs.get(name);
          } else {
             return "";
          }
   }
                          ... ...
```

```
/*******************************/
/* Example of ClassFinder      */
/*******************************/

Public  class  ClassFinder  extends  URLClassLoader  implements
IComponentLoader{
   public Class<?> findClass(String name){
       Class<?> classAim = null;
      try {
         classAim = super.findClass(name);
      } catch (ClassNotFoundException e) {
         if (ClassArray == null)
            throw new ClassNotFoundException();
         else {
            for (int i=0; i<ClassArray.length; i++){
               try {
                     classAim=ClassArray[i].loadClass(name);
               } catch (ClassNotFoundException ex) {}
            }

            if (ClassArray == null) {
               throw new ClassNotFoundException();
            }
         }
      }

      return classAim;
   }
                           ... ...
```

7.5 CUSTOM DEVELOPED CHECK-STYLE MODULE

The components of this project will use a number of APIs from check-style, it also needs to follow the pre-defined structure from check-style development organization. The component contains modules called custom developed check-style modules. The modules should follow the pre-defined structure from the check-style development organization if they want to be used by the check-style APIs. When the source code of a specified structure is developed, a new function can be added by using the visitor pattern which has been introduced in Chapter 6. When the module is released, it can automate the process of checking the Java code to help developers get rid of this tedious task.

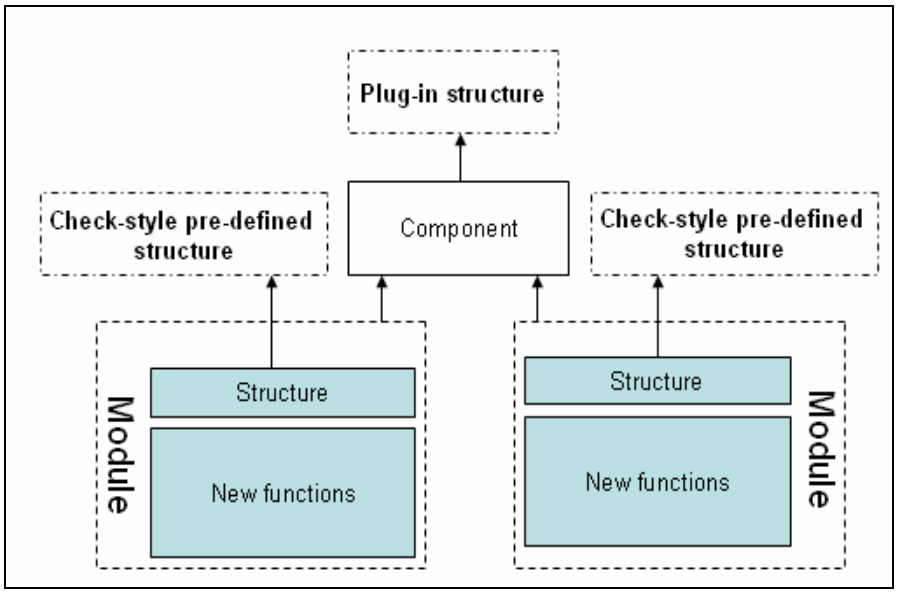


FIGURE 51: MODULE ARCHITRAVE

7.5.1 STRUCTURE OF THE MODULE

A normal module contains three parts: a pom.xml file, a packagenames.xml and some Java code with a specified structure.

POM.XML
The Pom.xml file includes information about the modules, such as module version, module name, module ID and so on.

PACKAGENAMES.XML
The Packagenames.xml includes the implementation of the modules. It describes the functions and the coding structure of the Java source code file.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE checkstyle-packages PUBLIC
  "-//Puppy Crawl//DTD Package Names 1.0//EN"

"http://www.puppycrawl.com/dtds/packages_1_0.
dtd">

<checkstyle-packages>
  <package name="com.mycompany.checks"/>
  <package
name="com.puppycrawl.tools.checkstyle">
    <package name="checks">
      <package name="blocks"/>
      <package name="duplicates"/>
      <package name="header"/>
      <package name="imports"/>
      <package name="javadoc"/>
      <package name="metrics"/>
      <package name="modifier"/>
      <package name="naming"/>
    </package>
    <package name="filters"/>
  </package>
</checkstyle-packages>
```

JAVASOURCEFILE.JAVA

The Java source code file contains the code that does the check actions. According to the check-style document, a check method needs an argument whose type is `DetailAST`. The following code is an example from this project.

```
public class NumberOfMethodCheck extends Check
{
    int methodNumber=0;
    public int[] getDefaultTokens(){
        return new int[]{TokenTypes.CLASS_DEF,
TokenTypes.INTERFACE_DEF};
    }

    public void visitToken(DetailAST ast){
        DetailAST objBlock =
ast.findFirstToken(TokenTypes.OBJBLOCK);
        methodNumber =
objBlock.getChildCount(TokenTypes.METHOD_DEF);

        log(ast.getLineNo(),
                "There are" + methodNumber + "in this source code
file");
    }
}
```

Some specific modules can also be the containers for other modules. The structure of these modules are formed in a tree structure; the top-level module is the kernel of the functions and other modules included in this module are used to implement the sub-functions.

The extended functions in this project are implemented in Java. After the source code is compiled, the generated APIs will be used in the source code which follows a specified structure. The implementation of the extended functions will be described in the component develop sections.

7.5.2 CONFIGURATION OF THE MODULES

A Check-style configuration file specifies which modules to plug in and apply to the Java source files. The modules are structured in a tree style, the root of this tree is the check module and the second level of tree can be `fileSetChecks`, `filters`, `auditListeners`.

There are two steps for creating the configuration file in this project.

Step 1. Create the file `checkstyle.xml` in the root of the project that wants to use the check modules.

```
<?xml version="1.0" ?>
// here is come XML file definitions
<module name="Checker">
  <module name="DissertationProject">
    <module name="NumberOfMethod"/>
  </module>
</module>
```

Step 2. Configure the check-style plug-in to use the check modules that are developed in this project.

This step tells the check-style how to use the custom developed modules. The configurations should be defined in the POM.XML file.

```
<?xml version="1.0" ?>
<project>
…  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-checkstyle-plugin</artifactId>
        <version>2.3</version>
        <dependencies>
          <dependency>
            <groupId>DissertationProject</groupId>
            <artifactId>check-modules</artifactId>
          </dependency>
        </dependencies>
      </plugin>
          …
```

7.5.3 BUILDING A JAR FOR THE CHECK MODULES

In order to use the check modules in other projects, check module packages need a specified structure. The structure is listed below:

```
NumberOfMethodCheck-1.0.jar
|-- POM.XML
|-- META-INF
|   |-- MANIFEST.MF
|   `-- maven
|       -- NumberOfMethodCheck
|          |-- POM.XML
|          `-- POM.PROPERTIES
|-- checks
    |-- packagenames.xml
    `-- MethodLimitCheck.class
```

7.6 CODING REVIEW COMPONENT IMPLEMENTATION

The Coding review component is the most important component in this project. The coding review component has three parts: the configuration file, the component framework and the functions modules.

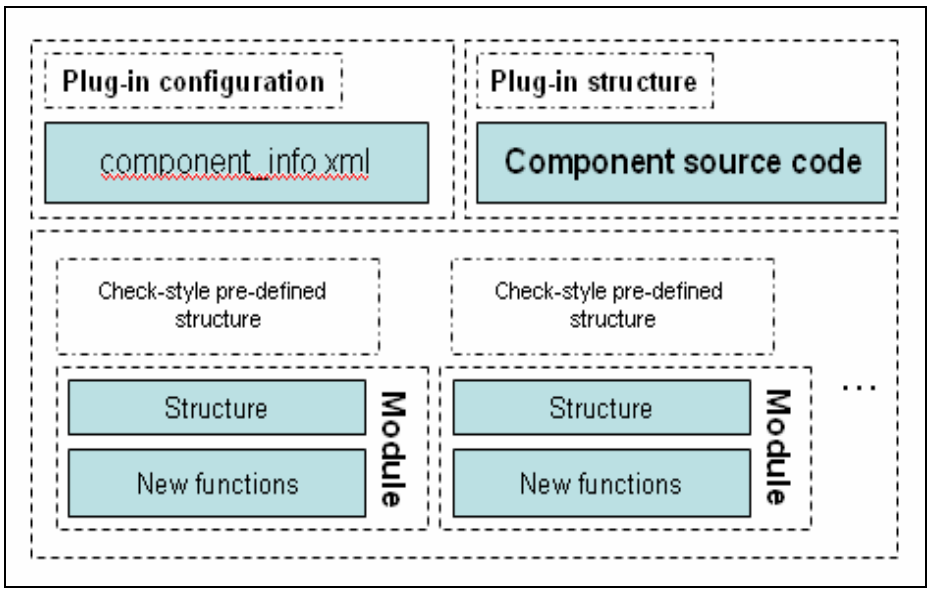


FIGURE 52: PLUG-IN COMPONENT ARCHITECTURE

The first part is the development of the configuration file. A component configuration file describes the information about the component; the information contains component id, name, version and author. It also contains function module information which tells the component framework how to load and use the modules. The configuration file contains more information; however the structure of this file should follow an XML file format. The component framework will deal with the variables in this file. In this project, a configuration file for the coding review component is defined:

```
<?xml version="1.0" encoding="utf-8"?>
<plugin id="CodingViewComponent" name="CVC" version="CVC1"
author="DITStudent">
  <runtime path="CodingViewComponent.jar"/>
</plugin>
```

The Component framework is code which follows a plug-in structure. The code deals with the functions implemented in the modules. Most of the classes in the component framework should implement the interface `IMyComponent`:

```
public interface IMyComponent {
    public void componentInit();
    public void componentStart();
    public void componentDestory();
    public void componentInfo();
    public IExtension getExtension(String componentId);
    public String infor;
    public String componentId;
}
```

This interface provides five methods which describe the behaviours of a component. The following code is an example of one of the component classes. The class is used to check the lines in the source code; the structure follows the module implementation structure and also the component plug-in structure.

```
public class CRLengthCheck extends Check implements
IMycomponent{
   int numberOfLines=0;
   StrToolFactory stf = new StrToolFactory();
    public int getLines(String sourceCode){
        return Integer.valueOf(stf.
createStringTool()).intValue();
    }
    public void componentInit(){}
    public void componentStart(){}
    public void componentDestory(){}
    public void componentInfo(){
       return this.info;
    }
    public IExtension getExtension(String componentId){
        this.componentId = componentId;
    }                ...          ...
```

```
Public interface IMyStringTool{
    public String dealWithStr(String str);
    public String str;
}

public class CRStrLinesCheck implements IMyStringTool{
    int count = 0;
    public String dealWithStr(String str){
        for( int i=0; i<str.length(); i++){
            if(str.charAt(i) == '\n' || str.charAt(i) == '\r'){
               count++;
            }
         }
        return count+"";
    }
}

public interface IAbstractStrToolFactory {
    public IMyStringTool createStringTool();
}

public class StrToolFactory implements
IAbstractStrToolFactory {
    public IMyStringTool createStringTool () {
       return new CRStrLinesCheck ();
    }
}
```

The sample code above is to do the length checking for the source code. The `RTLengthCheck` class is extended from the `Check` class, besides it also implements `IMycomponent`. This means that `CRLengthCheck` can be used by both the check-style module and the component plug-in framework. The example also follows the "*Factory method pattern*" for creating objects.

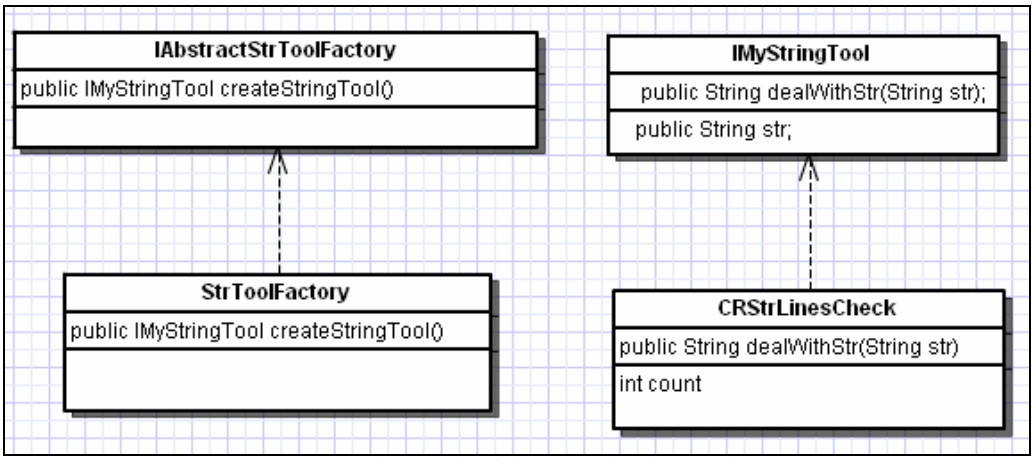


FIGURE 53: STRING FACTORY ARCHITECTURE

`IAbstractStrToolFactory` is an interface which contains the functions of a `ToolFactory`, and `StrToolFactory` should implement the interface. `StrToolFactory` should create a `CRStrLinesCheck` object. As the object creating action is encapsulated in `IAbstracStrToolFactory`, when changes the behaviours of `CRStrlineScheck`. This means that this component has a flexible design, the developers can update the component by modifying the code in the function class.

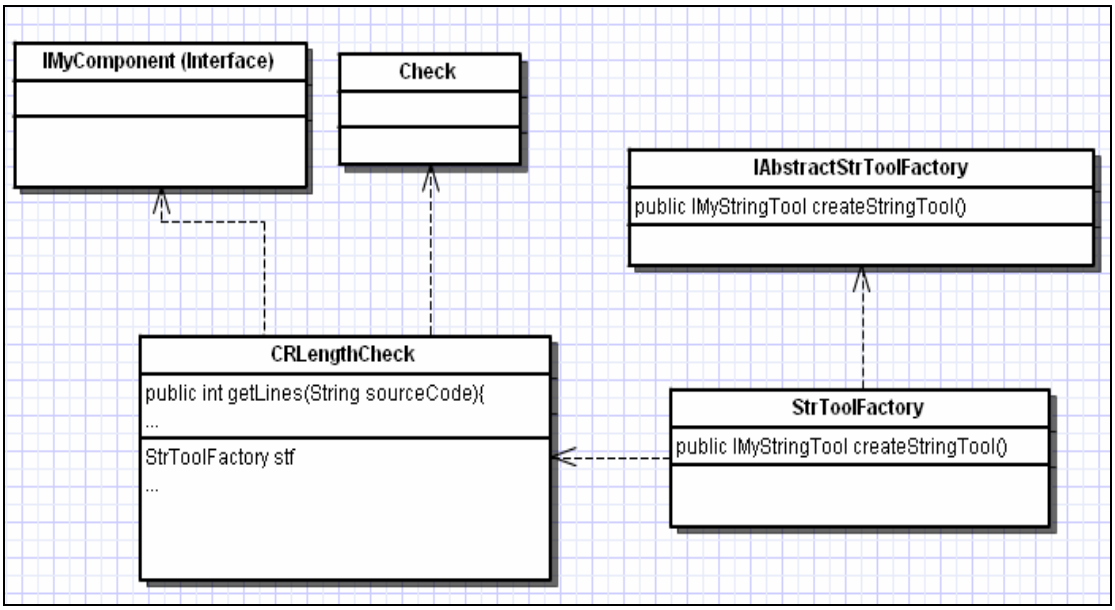


FIGURE 54: LENGTH CHECK MODULE ARCHITECTURE

`CRLengthCheck` gets the function from the `StrToolFactory` class, and all of the classes in the class diagram make a functional unit which is used in a module. All of the modules in this project will follow this design structure. With the help of this design structure, the component not only implements the pre-defined functions, it also has a very flexible structure: "***updating the component by changing only one method***".

The code review component has a number of modules, some of those modules are:

- **CRBlockCheck:**
  Block check; include nested block check and empty block check.

- **CRDumplicateCodeCheck:**
  Duplicate code detection

- **CRClassDesignCheck:**
  Class design checking

These modules build the functions of the component. Eclipse will give warnings to the developer if any problems are detected.

7.7 CODE REFACTORING COMPONENT IMPLEMENTATION

The Code refactoring component has the same architecture as the coding review component. It is also designed to use the "Factory method design pattern", and all of its functions are created from the factory class. There are three modules in this component.

- **CRefactoringNaming**
  Warning the developers to rename a method, variable, class or other Java elements if the source code has naming problems

- **CRefactoringCodeMoving**
  Warning the developers to do "Code moving" if the source code has structure problems

- **CRefactoringCodeExtract**
  Warning the developers to extract methods or classes if there are too many lines in them

Take the **CRefactoringCodeExtract** implementation as an example. This module has the same structure as other modules in this project. However it has more function factories. The development challenge from this module also comes from its functional development. This component development includes functional module development and component module development. Component module has the same architecture as the code review component module, and the functional module will go thought the following processes.

1. **Generate rules for extract code.**
   Code extract is a high intelligent process. The problems are which code need to be extracted and how to extract the code. The first problem will be resolved by checking the general rules for extracting code. Those rules can be:

   - **Check the length of the code** (class, method and other fields). For example if the length of the class is more than 200 lines, the code should be extracted.

   - **Check the number of elements** (methods, field, inner classes or other elements in the source code). A large number of elements will make the code hard to read and to understand, and that will make the functions provided by this class complicated and messy. So extraction of the code is necessary.

   - **Proper use of structure based key words** (e.g. while, do, for, if, try, catch). These key words may make the code messy; code extraction could improve the coding style.

- **Check code structure** (e.g. the position of variable definition, method implementation, inner class). Determine where to define the variables, methods, inner classes and other Java elements.

- There are also many other rules for code extraction. Those rules can be implemented as modules in this component.

2. **Analyse the rules, extract the rules to small unit.**
   In this section, each rule will be divided into several smaller units. Take the check code structure as an example.

   Check code structure (e.g. the position of variable definition, method implementation, inner class).
   The aim of this rule is to check the location of the variable definition, the method implementation and the inner class. This rule can be divided into three parts;
   - Check position of variables
   - Check position of methods
   - Check position of inner class.

   And those small units give a solution to the interface design.

```
public interface ICECodeStructure{
   public int[] variablePosition();
   public int[] methodPosition();
   public int[] innerClassPosition();

   ArrayList sourceCode = new ArrayList();
}
```

3. **Implement the rules**
   Rules can be implemented by creating the subclass of its related interface.
   Take the check code structure module as an example.
   According to the related interface designed, the subclass of this interface can be designed to implement the functions only. With the help of the factory method, the function module is independent of the component module.

```
public class CodeStructure implements ICECodeStructure{
   ArrayList counts = new ArrayList();
   public int[] variablePosition(){
      counts.clear();
      for( int i=0; i<sourceCode.size(); i++){

   if(  StructChecker.isVariable(issourceCode.get(i).toStr
ing())){
            counts.add(i+1);
         }
      }
      return this.arrayListToIntArray(counts);
   }
```

```
public int[] methodPosition(){
      counts.clear();
      for( int i=0; i<sourceCode.size(); i++){

   if(  StructChecker.isMethod(issourceCode.get(i).toStrin
g())){
            counts.add(i+1);
         }
      }

      return this.arrayListToIntArray(counts);
   }

   public int[] innerClassPosition(){
      counts.clear();
      for( int i=0; i<sourceCode.size(); i++){

   if(  StructChecker.isSubClass(issourceCode.get(i).toStr
ing())){
            counts.add(i+1);
         }
      }
      return this.arrayListToIntArray(counts);
   }
}
```

4. **Link the function modules to the component modules.**
   Rules are implemented in the function modules. In order to combine the function modules with the component modules, the factory method pattern are also used in this step.

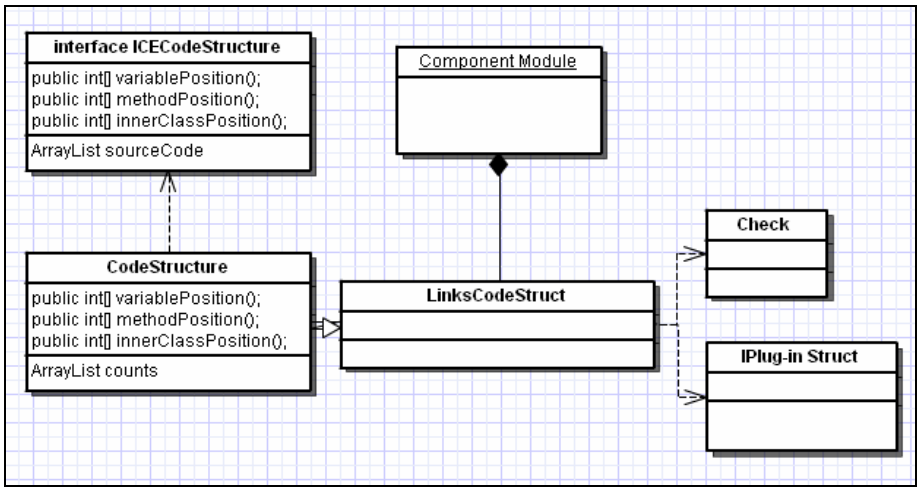


FIGURE 55: MODULE RELATIOION ARCHITECTURE

A related class `FMCodeStructure` is a class that extends the `Check` class, and that means the class follows a check-style API Style and it can be used by both check-style APIs and the component module class. This class also implements the plug-in module structure, besides; there are many other functional factory methods in it. This class links the functional module to the component module.

While all the component modules and functional modules are implemented, the whole component can be used as a plug-in for the code checker.

7.8 PERFORMANCE COMPONENT IMPLEMENTATION

As mentioned above, most components in this project have the same architecture. The performance component also has the same architecture, it has two major parts: one is the component module which links the component to the Code-Check and it will also deal with the functional modules in the other part. The other is the functional module. This module also contains sub-modules which provide related functions to its super-module. The implementation of this component has a lot of challenges. The challenge comes from the design pattern implementation and code improvement advice. There are many design patterns, but in this project only one pattern will be implemented. The name of this design pattern is *"Flyweight pattern"*. As the detail of this pattern has been introduced above, the "***Flyweight pattern***" can minimize memory usage by sharing as much data as possible with other similar objects. This means it can improve the performance of the application.

In order to implement the functional module (*"Flyweight pattern"* based function module), it is important to know when to use this design pattern. One of the solutions is to create a rule library. The rule library contains the situations about when to use the design. There is also a class which will deal with those rules. When the source code meets the requirement of the rules, the Code-Check should offer advice.

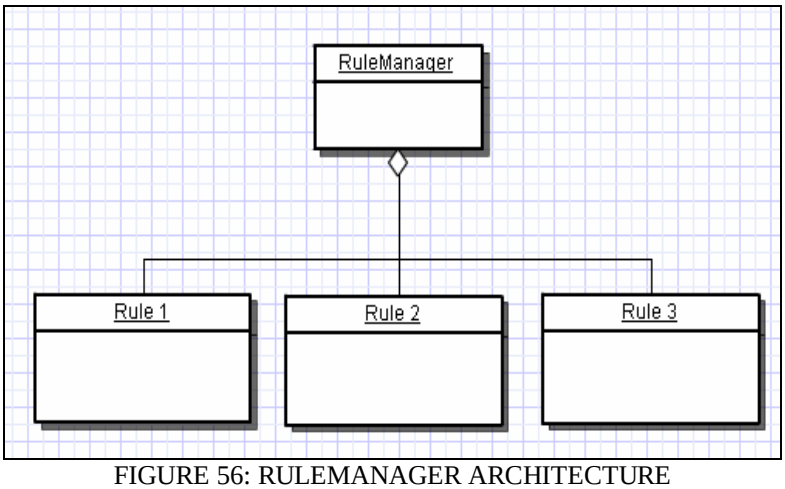


FIGURE 56: RULEMANAGER ARCHITECTURE

One rule in this project is:

When the developer creates an object which has the same type with another object, the developer may use the "*Flyweight pattern*". The following source code creates two objects with the same type "*MyClass*", and they don't change any behaviours of the object. This indicates that the developers can use the flyweight pattern here. The functional module will then create a "*flyweight pattern*" framework with the existing source code, and give the developer a warning message.

```
public class MyClass{
   public void printHello(){
      System.out.print("hello");
   }
}

public class anotherClass{
   public static void Main(String [] args){
      MyClass mc1 = new MyClass();
      mc1.printHello();

      MyClass mc2 = new MyClass();
      mc2.printHello();

   }
}
```

```
/********** Updated Class ************/
public class MyClassFactory {
   private Hashtable myClass = new Hashtable();
   public MyClass getMyClass( Object key ) {
      MyClass newClass = (MyClass) myClass.get(key);
      if( newClass == null ) {
         newClass = new MyClass();
         myClass.put( key, newClass );
      }
      return newClass;
   }
}

public class anotherClass{
   public static void Main(String [] args){
      MyClassFactory mcf = new MyClassFactory ();
      mcf.get MyClass("key1").printHello();

      mcf.get MyClass("key2").printHello();
   }
}
```

As there is a time limit on this project, the rules used are very simple while there should be a lot more complicated situations in real-world programming. As the component has a very complex design, it gives the ability to allow the user to change and extend the features in the future. When the developer wants to add new rules or features, what she should do is to implement a new rule, and put the compiled class into the specified package.

7.9 CODE-CHECK DEPLOYMENT

The Code-Check deployment includes component deployment and module deployment. The component modules are implemented based on the check-style APIs, so the configuration of deployment also follows the check-style configuration format. There are two types of configuration file in the component modules. The first type of configuration file includes the definitions of the specified coding standard; this file is used in the coding review component. This file should include all the definitions that the Java coding standards will use. And this makes the coding review component able to check the source code in the coding panel of Eclipse. Below is part of the configuration file.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <module name="Checker">
    <module name="PackageHtml" />
    <module name="NewlineAtEndOfFile" />
    <module name="Translation" />
  - <module name="TreeWalker">
      <module name="AvoidStarImport" />
      <module name="IllegalImport" />
      <module name="RedundantImport" />
      <module name="UnusedImports" />
      <module name="FileLength" />
      <module name="LineLength" />
      <module name="MethodLength" />
      <module name="ParameterNumber" />
    - <module name="HiddenField">
        <property name="tokens" value="VARIABLE_DEF" />
      </module>
      <module name="ArrayTypeStyle" />
      <module name="FinalParameters" />
    - <module name="GenericIllegalRegexp">
        <property name="format" value="\s+$" />
        <property name="message" value="Line has trailing spaces." />
      </module>
      <module name="TodoComment" />
      <module name="UpperEll" />
    </module>
  </module>
```

FIGURE 57: A PART OF THE CONFIGURATIONS WHICH IS BASED ON THE CODING STANDARD

Another type of configuration file is the plug-in configuration file, this type of configuration tells the Code-Check framework how to handle the components.

```
<?xml version="1.0" encoding="utf-8"?>
<plugin id="CodeRefactoringComponent" name="CRC" version="CRC1"
author="DITStudent">
  <runtime path=" CodeRefactoringComponent.jar"/>
</plugin>
```

When all of the configuration files, Java packages and libraries have been packaged together, the Code-Check can be deployed. The packaged Code-Check should be put to the Eclipse plug-in folder, besides; the check-style package also should be installed as the Code-Check will use a lot of APIs from that package.

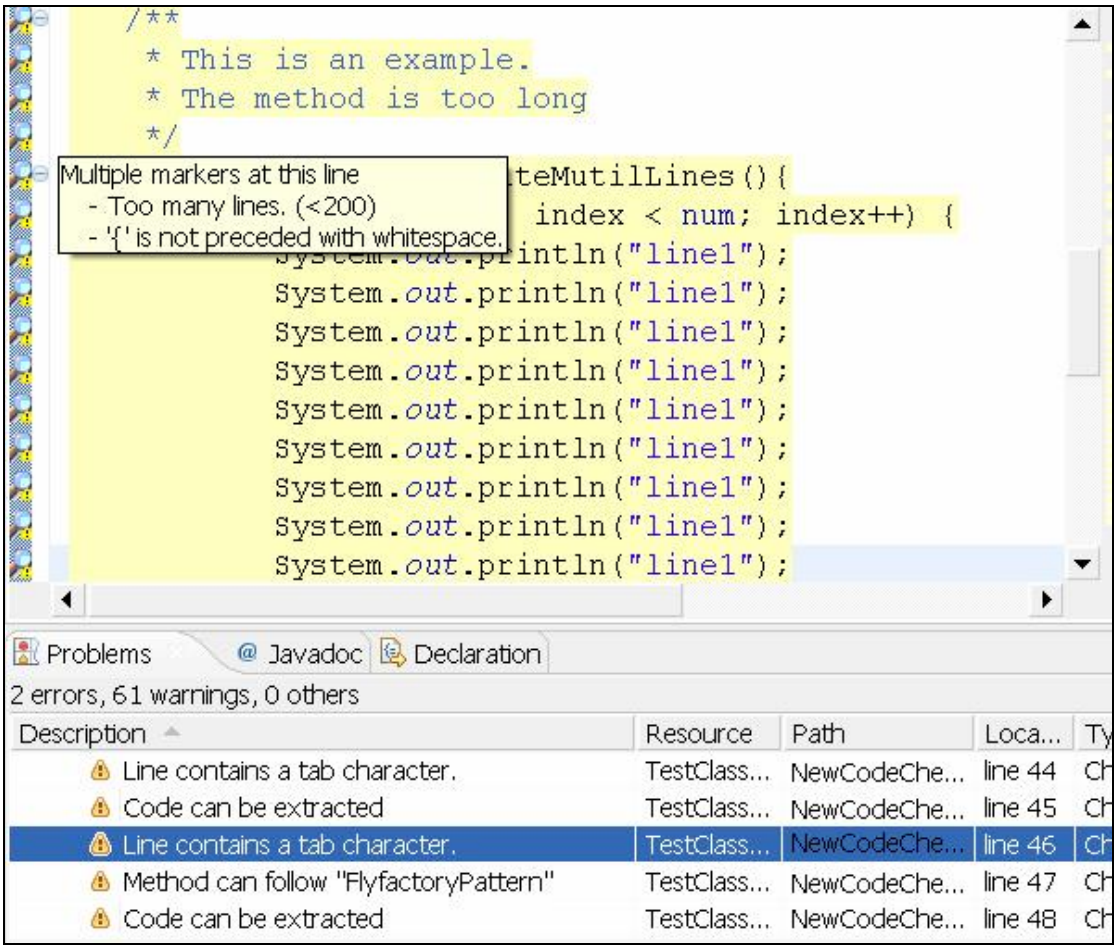


FIGURE 58: CODE CHECK EXAMPLE

7.10 CONCLUSIONS

This chapter explained the use cases for this project and a sequence of actions that the user undertakes. It then discusses the implementation of the Code-Check plug-in for Eclipse. The plug-in is an implementation of the new version of the Java coding standard. The Code-Check plug-in uses an XML file as the reference of rules. Based on the configuration file supplied, Eclipse can perform the code checking, coding review, code refactoring and performance checking for Java. This plug-in is aimed to help the developers to improve their programming habits and source code quality.

8. EVALUATION

8.1 INTRODUCTION

This chapter discusses the evaluation of the project. The evaluation has been carried out by both survey and interview. The survey concerns the coding standards reality and attitude of developers towards new coding standards and the Code-Check. Some developers and students from DIT were asked to do the interviews for evaluation and feedback.

8.2 SURVEY FOR THE PROJECT

The evaluation was carried out by two parts. The first survey was published on a survey website at: **http://www.surveymonkey.com/** with the following link: **http://surveymonkey.com/s.aspx?sm=DUTYQm7kshyrVgEFpIufGQ_3d_3d** and the second one is Chinese version which is published at **http://www.zdiao.com/vtest_show.asp?testid=113773**

In this survey, the questions are supplied as shown in Appendix C about how people feel about their current coding style and if they are willing to engage in a new coding standard. The survey was distributed through e-mail to the people that work in the software industry. There were a total of 678 responses (Chinese 646, Ireland 32), and they came from students from DIT, programmers current working in Ireland, China and Singapore.

8.2.1 DO YOU FEEL THAT CODING STANDARDS ARE IMPORTANT?

Results (Ireland): Most people gave 3, 4 and 5 in the Ireland version.
Results (China): Half of the respondents gave 3, 4, and 5 in the Chinese version.

FIGURE 59: SURVEY RESULTS (CHINA)

The results from the Chinese and Ireland version are very different. From this feedback, it could be verified that the developers in Ireland pay more attention to

software code quality. Potentially, the reason for this difference can also be explained by the following question.

8.2.2 IF YOU ARE A JAVA DEVELOPER, HOW MANY YEARS HAVE YOU BEEN PROGRAMMING IN JAVA?

Results (Ireland): 1/3 of the respondents gave 1-2 years, and 1/3 of the respondents gave 2-5 years
Results (China): More than 1/2 of the respondents gave 0-1 years.

As the above result shows, it appears that the coding standards are more important for those developers with more experience. More experienced developers realize coding standards are important, however, many developers even don't know the name or publisher of the coding standards they are using.

8.2.3 DO YOU KNOW THE NAME OF THE JAVA CODING STANDARD YOU ARE CURRENTLY USING?

Results (Ireland): More than 1/3 of the respondents don't know the name of the coding standards that they are using.
Results (China): More than 1/3 of the respondents don't know the name of the coding standards they are using.

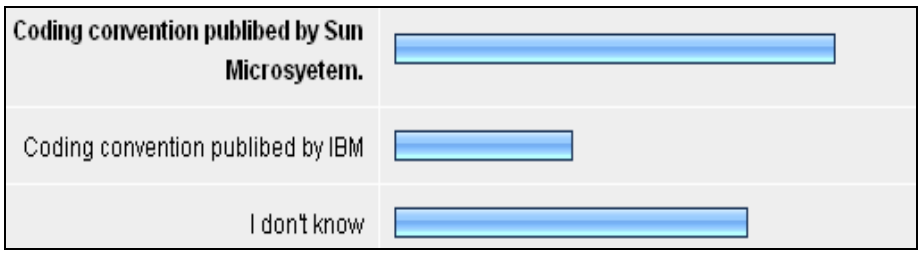


FIGURE 60: SURVEY RESULTS (IRELAND)

According to the results, developers don't tend to know their coding standards' names and also the standard publishers are not that important to them. What really matter is the coding standard itself and if it is easy to use.

8.2.4 WILL YOU FOLLOW THE CODING STANDARD STRICTLY?

Most of the respondents who are experienced developers provide the following answer "*70%-90% of the code follows the coding standard.*", also some developers with 0-1 years coding experience provide "*Less than 30% of the code follows the coding standard.*" The results show that most of the developers can't follow coding standards strictly and it is necessary to provide a new approach to help them to achieve source code which follows coding standards strictly.

8.2.5 HOW HAVE YOU LEARNED A SPECIFIED PROGRAMMING STANDARD?

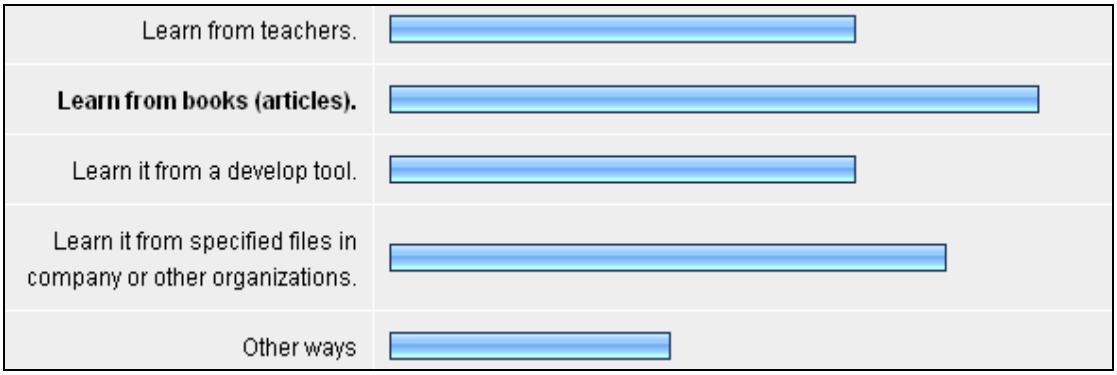


FIGURE 61: SURVEY RESULTS (IRELAND)

Results (Ireland): Nearly all of the developers learn coding standards through four approaches, but "*Learn from book (articles)*" is the most popular approach.

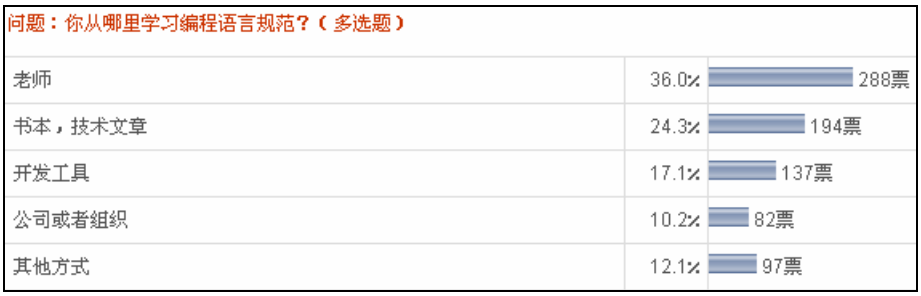


FIGURE 62: SURVEY RESULTS (CHINA)

Results (China): Nearly all the developers learn coding standards through two approaches and "*Learn from teachers*" is the most popular approach to Chinese developers.

The above results show that "*learning from teachers*" and "*learning from books (articles)*" are the most common methods for the developers to learn coding standards. The developers can handle most of the coding standards from those two approaches; however they can't deal with the remaining 20% of the coding standards. In this project, a solution will be provided to resolve this problem. The solution is to provide a coding-tool which will help the developer to improve the quality of the source code.

8.2.6 IF THERE WAS A TOOL TO HELP YOU WITH THE CODING STANDARDS, WOULD YOU USE IT?

Results (Ireland): Nearly all of the respondents are positive about using the tool (27/30)
Results (China): Nearly 500 of the respondents would like to use it while 145 of the respondents will never use it.

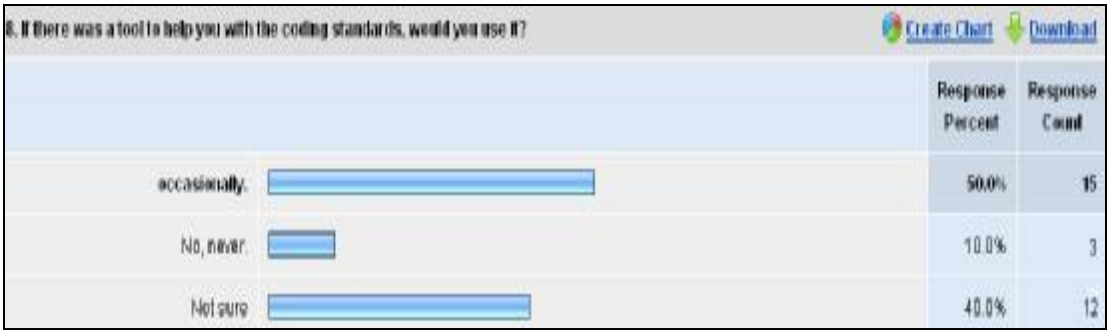


FIGURE 63: SURVEY RESULTS (IRELAND)

The results show that a positive attitude towards the tool and people are happy to try the new tool to help them improve. Although there are still a certain number of people who are uncertain about using the tool, they may be users in the future.

8.3 INTERVIEW FOR THE PROJECT

In undertaking this evaluation five interview questions were asked to the developers (1 from CR2 Ltd., 1 from VirtualAccess Ltd. and 1 from a Chinese software company UFIDA software ltd.). The first three questions focused on the existing coding standards problems in the companies.

- How do you encourage developers to follow coding standards in your company?
- How do you handle poor quality code from your team members?
- Do you have any specified developers who will check the coding standards in your company?

The other two questions focused on coding-tools.

- Which development tool do you use in your company and what do you think of its ability to handle source code?
- If there was a tool to help you with standards, would you use it?

8.3.1 HOW DO YOU ENCOURAGE DEVELOPERS TO FOLLOW CODING STANDARDS IN YOUR COMPANY?

In response to the first question one of the developers had the following response:
"... By reviewing the coding standards with the staffs in team ..."

The second developer gives a similar response:
"... our team leader will give us the coding standards document ..."
The third developer provided the following comments:
"... we use StyleCop analyzes our C# source code, besides, we also have our own coding standards document ..."

The feedback shows that many IT companies have their own coding standards. They encourage developers to follow coding standards by providing a coding standards document and ask them to follow this document. Several companies also provide coding-tools like StyleCop which can improve their source code quality.

8.3.2 HOW DO YOU HANDLE POOR QUALITY CODE FROM YOUR TEAM MEMBERS?

In responding to the second question, the developer from UFIDA Software Ltd gave the following response:
"… *Our team leader will check our code quality, if the quality of the source code is too bad; we have to rewrite the code*…"

The developers from the Irish company gave a similar response:
"… *We need to do code review and our QA also help us* …"

8.3.3 DO YOU HAVE ANY SPECIFIED DEVELOPERS WHO WILL CHECK THE CODING STANDARDS IN YOUR COMPANY?

In response to the third question a developer from UFIDA responded:
"… *no, I need to do code review by myself, however, my team leader may check my source code* …"

The developer from CR2 gave the following comments:
"… *we don't have any specified developers who will check my code, but I will do code review and QA may help us* …"

The developer from VirtualAccess provided the comment:
"… *I check the code by myself* …"

The responses from second and third question show that code reviews are a common way to help the developer to handle the poor quality source code. Code reviews are done by the developers or QA in their organisation. These three developers also indicate that few developers in their organisations use a code review tool. This means, a good code review tool may help developers to improve their source code quality.

8.3.4 WHICH DEVELOPMENT TOOL DO YOU USE IN YOUR COMPANY AND WHAT DO YOU THINK OF ITS ABILITY TO HANDLE SOURCE CODE?

In response to the fourth question:
The developer from UFIDA provided the following response:
"*... My group focuses on ERP development. Most of us use VB and C# as the programming language and my company provide Visual Studios as our develop tool.*

It is a very good developer tool, it can deal with the code structure, however I need to do code review by myself …"

The developer from VirtualAccess gave the following response:
"*… We use Visual Studio 2005 and it has great ability on code check. I know there are some plug-ins for coding check, but never tried …*"

The above feedback shows that many companies use Visual Studio as their development tool, however the Visual Studio plug-in for code checking is not commonly used. In order to make those tools popular, it requires a considerable amount of training using the plug-in.

8.3.5 IF THERE WAS A TOOL TO HELP YOU WITH STANDARDS, WOULD YOU USE IT?

In response to the fifth question:
Three developers gave similar answers. They would like to use the tool; however, the tool should be easy to use.

8.3.6 INTERPRETATION OF RESPONSES

From the feedback in the interviews, it is verified that most developers in IT companies realize that coding standards are important; however, they do not put too much effort in improving their coding style. Visual Studio coding-tools also are not very popular in these companies; most developers do coding check by themselves.

8.4 CODING STANDARDS AND CODE-CHECK EVALUAION

8.4.1 CODE-CHECK EVALUATION

The evaluation was done based on the results from the four Code-Check testers (3 developers and 1 student). The Code-Check testers are from:

- A software developer from CR2 in Ireland
- A software developer from VirtualAccess in Ireland
- A software developer from UFIDA in Beijin
- A student from DIT in Ireland

They used the Code-Check for 2-3 days and gave marks to the Code-Check from four aspects of the Code-Check, besides, an interview about Code-Check are also performed.

Coverage: how the Code-Check works on the rules of the coding standards, e.g. it deals with 60% of the rules defined in the new coding standards (good).

Correctness: how the Code-Check works on the rules, whether it works correctly, e.g. when it checks the length of the method, it gives the wrong result (weak).

Efficiency: how quickly it deals with the source code, e.g. it takes 20 minutes to deal with 100 lines Java code (weak)

Easy-of-use: how easy it is to use, e.g. everybody who can use Eclipse plug-in can use it. (Very good)

The following results are average results:

	1-weak	2-ok	3-good	4-very good	5- excellent
Coverage			3.1		
Correctness			3.3		
Efficiency				3.9	
Easy-of-use		2.25			

TABLE 20: RATEING OF THE CODE-CHECK

The results above show that the Code-Check is a good tool with room to improve. It covers most of the rules of the coding standards and most of those rules can be checked correctly. Especially its efficiency, it can deal with the source code very quickly. However, the Code-Check is not that easy to use, as the configuration is complex and a good understanding of the elements is needed.

8.4.2 CODE-CHECK FEEDBACK

An interview about the Code-Check also has been done and the four testers had given their feedback. There are four questions in this interview:

- Do you think it is easy to configure the Code-Check?
- Did you get any benefit from the Code-Check?
- If you could extend the functionality of the Code-Check, how would you change it?
- Would you be willing to use this Code-Check in future?

DCR2: A software developer from CR2 in Ireland
DVA: A software developer from VirtualAccess in Ireland
DUFIDA: A software developer from UFIDA in Beijin
SDIT: A student from DIT in Ireland

In responding to the first question, the tester gave the follow comment:
DUFIDA: *"… it is very hard to configure the Code-Check, but it is very easy to use …"*
DCR2: *"… I don't know how to configure the plug-in …"*

SDIT: *"… it is not very hard to configure by following your help document …"*
DVA: *"… not easy and you should improve it …"*

Four testers gave the similar results. From the result, it is verified that the Code-Check is too complex to configure. The developer from DUFIDA also gave some advice
DUFIDA: *"…your helps is also hard to follow, I think you should do a video on how to configure the code-check…"*

According to this advice, a video "*Deploy and test CODE-CHECK*" has been crafted and uploaded to http://www.youtube.com/watch?v=BIYC2TvEemU. This video introduces how to configure the CODE-CHECK and also includes some testing on CODE-CHECK. The video got 333 views in one month.

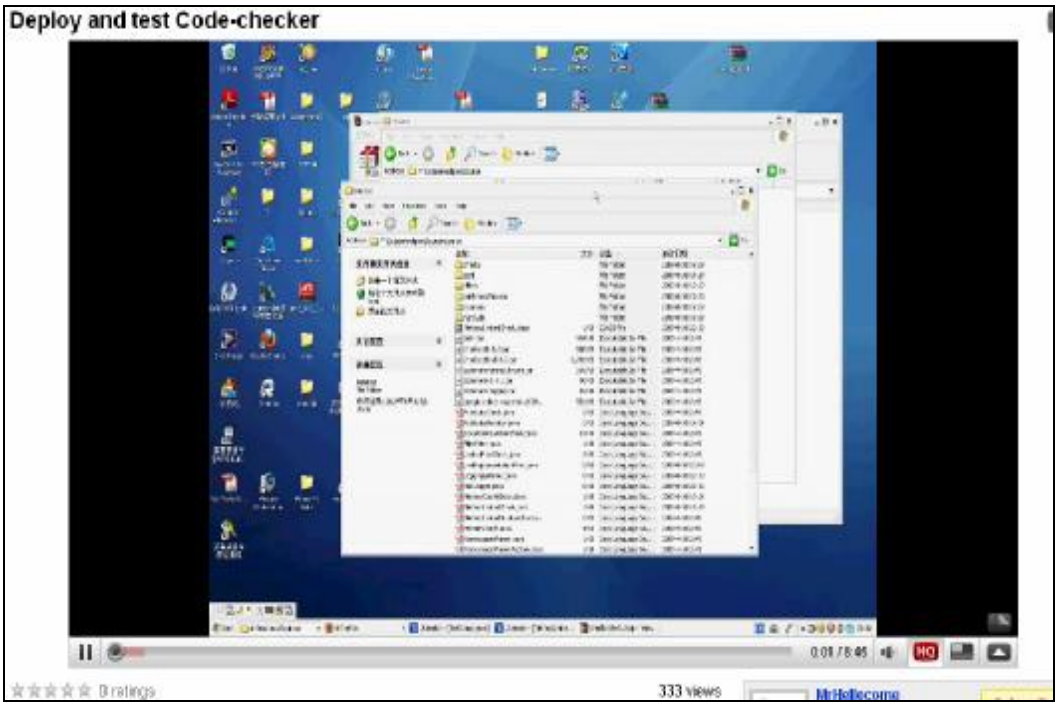


FIGURE 64: YOUTUBE VIDEO

In responding to the second question, the testers gave the follow comments:
DUFIDA: *"… it is helpful and I like the code-review function, I can do less work on code-review with it now…"*
DVA: *"… it is a good idea to develop this tool, but I would like to get a Visual Studios version …"*
DCR2: *"…self code review is good and the code refactoring can be improved …"*
SDIT: *"… it is helpful. I learned some design pattern with it …"*

The results show that the four testers all got benefits from CODE-CHECK. Three of them indicate that the code-review function is very helpful, they can save a lot of time with the CODE-CHECK; however the CODE-CHECK should be improved. The student from DIT also gave advice: *"… Make sure that the advice is correct, it would be a waste of time if the coding standard is not correct …"*

In responding to the third question, the testers gave the follow advice:

- Provide Visual Studios Version (C#)
- Make sure the advice is correct
- Develop a self-install package
- The code refactoring is not clever enough, need to improve the algorithm
- Provide more design patterns in performance component, and the algorithm also should be improved.
- Fix bugs.
- Giving advice is not enough; the CODE-CHECK should automatically change the structure of the code if necessary.

In responding to the fourth question, the testers all would like to use the CODE-CHECK in future if it is improved. They feel that a good CODE-CHECK saves them a lot of time on doing code reviews. And they are glad to have the tool to improve the structure of the source code, and also give good advice on improving the performance and design of the source code.

8.4.3 CODING STANDARDS EVALUATION

Together with CODE-CHECK evaluation, an XML file that contains the standard element has been supplied to the four testers. For those four people who experienced using the new coding standard, they were asked to rate the coding standard from the following areas with 5 marks given for each fields (0 is bad and 5 is excellent):

1. Correction of the coding standard
2. Difficulties in using the coding standard
3. Easy to remember
4. How it works with CODE-CHECK

	CORRECTION	DIFFICULTIES	EASY TO REMEMBER	HOW IT WORKS
Average Points	4.2	2.3	2.6	3.6

TABLE 21: RATEING OF THE NEW CODING STANDARD

As shown in the rating of the new coding standard, the new coding standard got high marks in the "*Correction*" which means the coding standards meet the common requirement of Java coding conventions. The "*Difficulties*" show bad results and the marks for "*How it works*" and "*Easy to Remember*" are average. The testers all agree that this version of the coding standards is good, but it also need improvements. Compared to learning coding standards without the tool, it has made a lot of improvements though it is still not that easy to use. The large number of rules in coding standards determines that there is no coding standard which is very easy to remember, what can be done is to improve the coding standards and find the balancing point between them.

8.5 CONCLUSIONS

This chapter discusses the project evaluation and the future work of the project. The project evaluation is carried out by surveys and interviews. The survey is for gathering people's attitude against the current coding standard and if they are willing to accept new coding standards. And the interview is using software developers that has or will experience the new standard and give their responses. As the surveys show, people would like to accept a new coding standard once it is easy to follow and simple to use. And there are is good feedback for the new standard which is encouraging. However, to make the new coding standard accepted by more people, a lot of work still needs to be done to achieve it.

9. CONCLUSIONS AND FUTURE WORK

9.1 CONCLUSIONS

This project is aimed to help developers gain a better coding standard and so to make them write code with better consistency. There is no doubt that writing good code following the standard bring a lot of benefits to the programmer himself and also the whole team. However, many programmers also complain about the standard that it is so hard to follow the specific rules and also sometime especially when people work in a team; it became very hard for them to work under a unique coding standard.

Take Java for example, so many programming standards existed like standard from Sun Microsystems, IBM which makes it hard to gain a unique and simple coding standard. With this issue existed, it takes much effort to get code reviewed once the code has been written.

In aim to help to improve with the situation, this project has analysed the basic elements of a coding standard. Take Java for example, trying to work out a simple but useful coding standard that can fit in the most requirements in the programming. With the understanding of the existing coding standards, the project is trying to abstract the useful rules and force them to be used in the programming.

With the coding standard that researched from this project, an XML file with a checklist of elements is supplied. This XML file can be imported to a plug-in that developed for ECLIPESE. The plug-in called CODE-CHECK is aimed to force the developers to follow the rules that have been defined in the XML file. Those rules are based on the coding standard that is achieved in this project. And then when the programmers use this plug-in during their coding, he will get the prompt or warnings wherever appropriate. It is designed so as to help the developers get familiar with the coding standard and use it automatically in a near future.

There is a survey for programmers that is used to find out the current usage of the coding standards, and as the survey shows, people is willing to follow the coding standards but it is hard to do so in reality. And also they hold positive opinions in trying the new tool to help them gain better a coding standard. Followed by a more detailed interview, it shows that different companies have different ways of dealing with bad codes that not following the coding standards, and thus the interviewers are invited to test the new plug-in and return with their rating and comments. From their comments, it shows good feedback and interests on this helping tool. Also they have give suggestions for improvements. From the comments collected from the developers, it shows the positive future but there is still things need to be done in the future work.

The future work is made of two parts. First is the improvement of the current version of Java coding standard. It can be extended and become a lot more complicated. As the current algorithm used is simple for code refactoring, it can be made more

intelligence and complex. And the other part is to get coding standard for other widely used languages like C++, VB, etc.

9.2 FUTURE WORK

9.2.1 CODING STANDARDS IMPROVEMENT

Within this project, several existing coding standards are examined and based on all of the analysis on the existing ones; a new coding standard was developed. However, due to time limitations, there is more work to be done. For example, the name checking, it works for checking the length of the naming, however, it is still not possible to check if the naming is meaningful and appropriate.

Also the tool could be extended to other programming languages like C++, C# that are widely used. The feedback from the testers also shows that the coding standards can be improved in "*easy to use and easy to remember*". This work based on a good understanding of specified coding standards. In the future work, the existing rules also will be analyzed again in order to make it easier to follow.

9.2.2 CODE CHECKER IMPROVEMENT

The CODE-CHECK is the implementation of the coding standards defined in this project. It works well now, however there are still many issues. According to the CODE-CHECK testers' comments and advices, the following work can be done in future;

- Do more testing and fix the bugs.
- Provide Visual Studios Version
- Improve the advice given by the CODE-CHECK
- Develop a self-install package, make the installation easy
- Improve the algorithm of the code refactoring component
- Add more rules in code refactoring component
- Provide more design patterns in performance component
- Improve the algorithm of performance component
- Provide "*automatically improve code structure*" function in CODE-CHECK.

REFERENCES

[1] Coding Standard Definitions
Website: http://code.google.com

[2] Steven, D., Programming Standards-wikipedia-parser, 2007
Website: http://code.google.com/

[3] Xiaosong Li, Advance Technical Program, 2007

[4] Robert C. Martin : Design Principles and Design Patterns

[5] Scott W. Ambler, Software process mentor, Writing Robust Java Code
The Ambysoft Inc. Coding standards for Java
Version: January 15.2000

[6] Science Infusion Software Engineering Process Group
General Software Development Standards and Guidelines Version 3.5

[7] Programming styles
http://en.wikipedia.org/wiki/Programming_style

[8] Michael Perry, Automated Nightly Build System

[9] Batch program
http://encyclopedia2.thefreedictionary.com/batch+program

[10] Vermeulen, Ambler, Bumgardner, Metz, Misfeldt, Shur, & Thompson
Cambridge University Press, 2000
The Elements of Java Style

[11] Naming conventions
http://en.wikipedia.org/wiki/Naming_conventions_(programming)

[12] Jonathan Nagler. Coding Style and Good Computing Practices. The Political Methodologist, Vol. 6, No. 2 (Spring 1995).

[13] Herb Sutter & Andrei Alexandrescu, C++ Coding Standards

[14] Doug Lea, Draft Java Coding Standard

[15] .NET Programming Standards and Naming Conventions
http://www.irritatedvowel.com/Programming/Standards.aspx

[16] Indent style
http://en.wikipedia.org/wiki/Indent_style#K.26R_style

[17] Variant: 1TBS
http://en.wikipedia.org/wiki/Indent_style#Variant:_1TBS

[18] Allman style
http://en.wikipedia.org/wiki/Indent_style#Allman_style_.28bsd_in_Emacs.29

[19] G. W. Russell. Experience with inspection in ultralarge-scale developments. IEEE Software.

[20] S. Chiba Campus, Building Coding Inspection Technology
Last version 2002

[21] Tom Mens, Tom Tourw´, A Survey of Software Refactoring

[22] Yijun Yu John Mylopoulos Eric Yu Julio Cesar Leite Linda Lin Liu Erik D'Hollander , Software refactoring guided by multiple soft-goals

[23] Tom Mens, An introduction toSoftware Refactoring

[24] Code refactoring
Website: http://en.wikipedia.org

[25] Joshua Kerievsky , Refactoring To Patterns published by Addison Wesley in August 2004

[26] Mark Tabladillo, Application Refactoring With Design Patterns

[27] Code review
http://en.wikipedia.org/wiki/Code_review

[28] Adam Shostack, Code Review Guidelines

[29] Performing a Security Code Review
http://msdn.microsoft.com

[30] Michael Howard, Microsoft, A Process for Performing Security Code Reviews

[31] J.D. Meier, Alex Mackman, Blaine Wastell, Prashant Ban
How To: Perform a Security Code Review for Managed Code

[32] Development Team/Code Review
http://wiki.sugarlabs.org/go/Development_Team/Code_Review

[33] Deborah A. Trytten University of Oklahoma
A design for team peer code review

[34] Project Review Group (PRG) – Process and Code of Practice
http://www.hm-treasury.gov.uk/d/prg_code_of_practice_v3_2009.pdf

[35] Published by Juan Soulie, C++ Language Tutorial

[36] Rob Miller, David Clark,Bob White
An Introduction to the Imperative Part of C++

[37] The C Programming Language
http://en.wikipedia.org/wiki/The_C_Programming_Language_%28book%29

[38] The C++ standards committee (the ISO/IEC JTC1/SC22/WG21 working group),
Standard for ProgrammingLanguage C++

[39] Anders Hejlsberg, Scott Wiltamuth, Peter Golde
The C# Programming Language

[40] Microsoft developer center, Visual C# Highlights

[41] Scott Guthrie, C# 3.0 Language Features

[41] Anders Hejlsberg, Scott Wiltamuth, Peter Golde C# Language Specification
Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

[42] Microsoft developer center
C#.NET Programming Standards and Naming Conventions

[44] Microsoft developer center
Microsoft Visual Basic for Developers

[45] Microsoft developer center
Visual Basic Coding Conventions

[46] Roy W. Miller IBM, Introduction to Java programming

[47] Nick Parlante, Java Introduction

[48] Sun Macrosystems, Code Conventions for the Java Programming Language
Website: http://java.sun.com

[49] Grand, M. (1997). Java Language Reference. Sebastopol, CA: OReilly &
Associates, Inc.

[50] Ray Ontko, Java Coding Standards

[51] Watts S. Humphrey
A Discipline for Software Engineering

[52] Scott Ambler
AmbySoft Inc. Coding Standards

[53] Sun Microsystems, Inc.
Code Conventions for the Java Programming Language

[54] G Bracha, Sun Microsystems
Software Reality: Coding Style

[55] The coding style of Java programming language
Website: http://en.wikipedia.org

[56] Strunk, W., Jr., E. B. White.
The Elements of Coding Style.

[57] Sun Microsystems, Inc.
Package Naming Conventions

[58] Geotechnical Software Services
Java Programming Style Guidelines

[59] Scott W. Ambler , Practice Leader, Agile Development, Rational Methods Group, IBM，Java naming conventions

[60] Eng.Omar H. Al-Nahal, Java Programming I - Laboratory Course
Faculty of Engineering & IT Software Engineering Department

[61] The coding style of Java programming language
http://en.wikipedia.org

[62] Paul Leahy, Using Java Naming Conventions
http://java.about.com

[63] Raffle, Interface Naming Conventions

[64] Sun Macrosystems, Method Naming Conventions for the Java Programming Language

[65] The coding style of Java programming language
http://en.wikipedia.org

[66] Sun Microsystems, Inc.
Code Conventions for the Java Programming Language Version 2

[67] Tips for Java Doc Comments
Website: http://linux2-cs.johnabbott.qc.ca

[68] Sun Macrosystems, How to Write Doc Comments for the Javadoc Tool

[69] David Flanagan , Java Programming and Documentation Conventions
Java™ in a Nutshell: A Deskop Quick Reference

[70] Sun Macrosystems, Comments Conventions for the Java Programming Language

[71] Aron Roberts , Java Source Files - Beginning Comments Block

[72] Sun Microsystems, Comments Conventions for the Java Programming Language

[73] David Flanagan, Java Documentation Comments

[74] Sun Microsystems, Inc. Developer Resources for Java Programming Language
Documentation Comments

[75] Sun Macrosystems, Comments Conventions for the Java Programming Language

[76] Strunk, W., Jr., and E. B. White. The Elements of Coding Style.
New York: Macmillan, 1979.

[77] Colin Depradine, Javadoc Tutorial, Java Document Comments

[78] David Flanagan, Java Documentation Comments, Java in a nutshell]

[79] Sun Microsystems, Blank line convention for the Java Programming Language

[80] Code Project Group, Software Programming Standards Introductions.

[81] Sun Microsystems, Inc. Developer Resources for Java Programming Language
The Java Platform Class Hierarchy

[82] Y. Daniel Liang, Introduction to Java Programming: Comprehensive Version

[83] Bill Venners, Java Design Issues: A Conversation with Ken Arnold

[84] Concurrency
http://en.wikipedia.org/wiki/Concurrency_%28computer_science%29
[85] McGraw-Hill, Osborne, Multithreading in Java

[86] Incheon Paik,Computer Industry Lab.
Programming Java Multithreaded Programming

[87] Sun Microsystems, Inc. Developer Resources for Java Programming Language
Synchronized Methods

[88] Brian Goetz, Software consultant, Quiotix
Threading lightly, Part 1: Synchronization is not the enemy

[89] Sun Microsystems, Inc. Developer Resources for Java Programming Language
Java package

[90] Sun Microsystems, Inc. Developer Resources for Java Programming Language
Lesson: Packages

[91] Sun Microsystems, Inc. Java Packages Programming, Java Language
Specification

[92] Sun Microsystems, Inc.
Java Packages Access Protection

[93] Java Reuse Tutorials
http://www.tutorialhero.com/tag-14-Reuse.php

[94] Doug Lea, Draft Java Coding Standard

[95] Patrick Bouklee, Java Package Tutorial

[96] Sun Microsystems, Inc. Developer Resources for Java Programming Language
Creating and Using Packages

[97] Mary Smiley , The Java Package Tutorial

[98] Java (programming language)
http://en.wikipedia.org/wiki/Java_%28programming_language%29

[99] Sun Microsystems, Inc.
Code Conventions for the Java Programming Language

[100] Jan Tobochnik, Harvey Gould, Introduction to Java

[101] Sun Microsystems, Inc., Introducing Java

[102] Java (software platform)
http://en.wikipedia.org/wiki/Java_%28software_platform%29

[103] Sun Microsystems, Inc.
Understanding the Java Platform Architecture

[104] Sun Microsystems, Inc. Developer Resources for Java Programming Language
The Java Platform, Standard Edition (Java SE)

[105] Eclipse Plugin Central, Eclipse Plug-in Architecture

[106] Chris Aniszczyk, Software Engineer, IBM, Software Group,
Eclipse Plugin Development TUTORIAL

[107] Oliver Burn, Check-style Overview

[108] Oliver Burn, Check-style plugin for IDEs

[109] Check-Style Group, Features of Check-Style

[110] Check-Style Group, Checkstyle configuration
http://checkstyle.sourceforge.net/config.html

[111] Check-Style Group, Checkstyle Modules configuration

[112] Check-Style Group, Checkstyle APIs Document

[113] Check-Style Group, Checkstyle Class Design introduction

[114] Check-Style Group, Checkstyle Class Design APIs descriptions

[115] Check-Style Group, Checkstyle Block Check introduction

[116] Diomidis Spinellis, The Bad Code Spotter's Guide

[117] Check-Style Group, Checkstyle Modules API descriptions

[118] Check-Style Group, Checkstyle APIs Document

[119] Check-Style Group, Develop check-style self-designed modules.

[120] Visitor pattern introductions
http://en.wikipedia.org/wiki/Visitor_pattern

[121] Lea, D. Concurrent Programming in Java Design Principles and Patterns.
Reading

[122] Factory method pattern
http://en.wikipedia.org/wiki/Factory_method_pattern

[123] Mark Grand, Pattern Summaries: Factory Method

[124] A&P Web Consulting Corp.
GoF Pattterns Introductions, Factory Method Design Pattern, UML diagram

[125] Gopalan Suresh Raj, The Factory Method (Creational) Design Pattern

[126] Martin Fowler, Kent Beck
Refactoring: improving the design of existing code

APPENDIX A

List of refactoring techniques

Here is a very incomplete list of code refactorings. A longer list can be found in Fowler's Refactoring book and in Fowler's Refactoring Website[10].

- Techniques that allow for more abstraction
  - Encapsulate Field - force code to access the field with getter and setter methods
  - Generalize Type - create more general types to allow for more code sharing
  - Replace type-checking code with State/Strategy
  - Replace conditional with polymorphism
- Techniques for breaking code apart into more logical pieces
  - Extract Method, to turn part of a larger method into a new method. By breaking down code in smaller pieces, it is more easily understandable. functions.
  - Extract Class moves part of the code from an existing class into a new class.
- Techniques for improving names and location of code
  - Move Method or Move Field - move to a more appropriate Class or source file
  - Rename Method or Rename Field - changing the name into a new one that better reveals its purpose
  - Pull Up - in OOP, move to a superclass
  - Push Down - in OOP, move to a subclass
  - Shotgun surgery - in OOP, when a single change affects many classes.

APPENDIX A

Survey about Coding standard

1. Default Section

1. How many programming languages do you know?

- 0
- 1-4
- 4-10
- More than 10

2. If you can use more than 2 programming language, do you have difficulty follow the specified coding standard?

- Easy to follow
- Not very hard, but make mistakes sometimes.
- Hard to follow, always make mistakes.

3. If you are a Java developer, how many years have you been programming in Java?

- 0-1
- 1-2 years
- 2-5 years
- More than 5 years

4. Do you feel the coding standard is important?(1=not important, 5=very important)

- 1
- 2
- 3
- 4
- 5

5. Hove you worked in organizations that use specified programming standard?

- Yes, we have our own pre-existing programming standards
- Yes, and we use a specified programming standard(Not pre-existing).
- No, we don't following a coding standard

6. How have you learned a specified programming standard?

- [ ] Learn from teachers.
- [ ] Learn from books (articles).
- [ ] Learn it from a develop tool.
- [ ] Learn it from specified files in company or other organizations.
- [ ] Other ways

7. Will you follow the coding standard strictly?

- ○ 90%-100% of the code follows the coding standard.
- ○ 70%-90% of the code follows the coding standard.
- ○ 50%-70% of the code follows the coding standard.
- ○ 30%-50% of the code follows the coding standard.
- ○ Less than 30% of the code follows the coding standard.

8. If there was a tool to help you with the coding standards, would you use it?

- ○ occasionally.
- ○ No, never.
- ○ Not sure

9. Do you know the importance of the coding standard.

- ○ I can list 1-2 importance of the coding standard.
- ○ I can list 3-5 importance of the coding standard.
- ○ I can list 5-10 importance of the coding standard.
- ○ I can list more than 10 importance of the coding standard.

10. Do you know the name Java coding standard you are currently using?

- ○ Coding convention publibed by Sun Microsyetem.
- ○ Coding convention publibed by IBM
- ○ I don't know

APPENDIX B

Question					
How many programming languages do you know?	Zero	1 to 4	4 to 10	More than 10	
Result	0	15	13	0	
If you can use more than 2 programming language, do you have difficulty follow the specified coding standard	Easy to follow	Not very hard, but make mistakes sometimes.	Hard to follow, always make mistakes.		
	7	17	4		
If you are a Java developer, how many years have you been programming in Java?	0-1 year	1-2 years	2-5 years	More than 5 years	
	2	9	11	6	
Do you feel the coding standard is important?(1=not important, 5=very important)	1	2	3	4	5
				6	22
Hove you worked in organizations that use specified programming standard?	Yes, we have our own pre-existing programming standards	Yes, and we use a specified programming standard(Not pre-existing).	No, we don't following a coding standard		
	7	19		2 people did't choose anything	
How have you learned a specified programming standard?	Learn from teachers.	Learn from books (articles).	Learn it from a develop tool.	Learn it from specified files in company or other organizations.	Other ways
Mutil-choose	21	24	13	11	18

Will you follow the coding standard strictly?	90%-100% of the code follows the coding standard.	70%-90% of the code follows the coding standard.	50%-70% of the code follows the coding standard.	30%-50% of the code follows the coding standard.	Less than 30% of the code follows the coding standard.
	7	13	6	2	0
If there was a tool to help you with the coding standards, would you use it?	occasionally.	No, never.	Not sure		
	19	0	9		
Do you know the importance of the coding standard.	I can list 1-2 importance of the coding standard.	I can list 3-5 importance of the coding standard.	I can list 5-10 importance of the coding standard.	I can list more than 10 importance of the coding standard.	
	3	21	4	0	
Do you know the name Java coding standard you are currently using?	Coding convention publibed by Sun Microsyetem.	Coding convention publibed by IBM	I don't know		
	18	0	5	5 people didn't choose anything	

APPENDIX C

计算机编程语言代码规范调查JAVA

问题：您会使用几种编程语言（单选题）

0	31.1%	201票
1-4	41.0%	265票
4-10	14.8%	96票
大于10种	13.0%	84票

问题：如果你会用2种或者2种以上的计算机编程语言，你在使用每种语言的代码规范上会有问题吗（单选题）

我觉得使用起来很容易	45.3%	293票
不是特别难，但是有时候会混淆	29.2%	189票
挺难的，常常会弄错	25.3%	164票

问题：如果你是一个Java开发人员，你有几年的Java使用经历（单选题）

0-1	53.2%	344票
1-2	25.2%	163票
2-5	12.8%	83票
大于5年	8.6%	56票

问题：你觉得代码规范重要吗（1-5，1不重要，5很重要）（单选题）

1	43.3%	280票
2	18.8%	122票
3	14.7%	95票
4	8.2%	53票
5	14.8%	96票

问题：你所在的公司或者组织有特定的编程规范吗（单选题）

我们有自己公司制定的的规范	42.8%	277票
我们用网上或者其他的编码规范，但不是我们公司自己制定的	29.2%	189票
我们没有编码规范	27.8%	180票

问题：你从哪里学习编程语言规范？（多选题）

老师	36.0%	288票
书本，技术文章	24.3%	194票
开发工具	17.1%	137票
公司或者组织	10.2%	82票
其他方式	12.1%	97票

问题：你严格按照编码规范写程序吗（单选题）

我的代码有 90%-100% 遵守编码规范	39.3%	254票
我的代码有 70%-90% 遵守编码规范	23.9%	155票
我的代码有 5%-70% 遵守编码规范	12.6%	82票
我的代码有 30%-50% 遵守编码规范	7.5%	49票
我的代码有 小于30% 遵守编码规范	16.4%	106票

问题：如果有一种帮助你学习使用编码规范的工具，你会使用吗（单选题）

可能会吧	56.9%	368票
肯定不用	22.4%	145票
我也不清楚	20.5%	133票


问题：你知道你现在使用的Java编码规范的名字吗（单选题）
Sun公司制定的规范
48.6%
314票
IBM公司制定的规范
23.2%
150票
不清楚
28.1%
182票