Handling Data in Java

Reading in Data

To get input from the user, Java has the Scanner class, so we need to do the following:

```
import java.util.Scanner;     // import the Scanner class
```

And from there we can read the system input (`System.in`) by creating an object:

```
Scanner myObj = new Scanner(System.in);
String userName;
System.out.println("Enter username: ");
userName = myObj.nextLine();
```

There are other methods as well as `nextLine()` which reads in the next string.

Reading Methods

Method	Description
`nextBoolean()`	Reads a boolean value from the user.
`nextByte()`	Reads a byte value from the user
`nextDouble()`	Reads a double value from the user
`nextFloat()`	Reads a float value from the user
`nextInt()`	Reads a integer value from the user
`nextLong()`	Reads a long value from the user
`nextShort()`	Reads a short value from the user

File Handling

To control files we need to import the `Scanner` class, as well as the `File` class:

```
import java.io.File;        // Import the File class
```

And then to read in a file, we do the following:

```
File myObj = new File("filename.txt");
Scanner myReader = new Scanner(myObj);
while (myReader.hasNextLine()) {
    String data = myReader.nextLine();
    System.out.println(data);
}
myReader.close();
```

File Methods

Method	Description
`getName()`	Returns the name of the file
`getAbsolutePath()`	Returns the absolute pathname to the file
`canWrite()`	Returns whether you can write to the file
`canRead()`	Returns whether you can read from the file
`length()`	Returns the length of the file
`createNewFile()`	Creates a new file.

To write to a file, we import the `FileWriter` class, instead of thr the `File` class:

```
import java.io.FileWriter;      // Import FileWriter class
```

And we can use the method `write(String)` to add to the file.

Handling Data in Java

Arrays in Java

To declare an array in Java, get can do the following:

```
int[] Age;
```

To initialise the array:

```
int[] Age = {44, 23, 42, 33, 16};
```

To access the first element in the array:

```
System.out.println(Age[0]);
```

A program to print out all of the values in an array can be as follows:

```
int[] Age = {44, 23, 42, 33, 16};
for (int i = 0; i < Age.length; i++) {
  System.out.println(Age[i]);
}
```

For a String array it's almost exactly the same:

```
String[] cars = {"Volvo", "BMW", "Ford"};
for (int i = 0; i < cars.length; i++) {
  System.out.println(cars[i]);
}
```

Linked Lists in Java

Java has a linked list class to help in creating and accessing linked lists:

```
import java.util.LinkedList;  // Import LinkedList class
```

And we can create an linked list object as follows:

```
LinkedList<String> cars = new LinkedList<String>();
cars.add("Volvo");
cars.add("BMW");
```

There are also several methods to get, set, add and remove items from the linked list:

Method	Description
`get(`*`index`*`)`	Returns the item at location *index*.
`set(`*`index, value`*`)`	Sets the item at location *index* to the value *value*.
`remove(`*`index`*`)`	Removes the item at location *index*.
`addFirst(`*`value`*`)`	Add an item to the start of the list with the value *value*.
`addLast(`*`value`*`)`	Add an item to the end of the list with the value *value*.
`removeFirst()`	Removes the first item from the list.
`removeLast()`	Removes the last item from the list.
`getFirst()`	Returns the first item of the list.
`getLast()`	Returns the last item of the list.
`clear()`	Clears the list.

Together this gives us a lot of functionality to manpulate linked lists.