SCHOOL OF COMPUTER SCIENCE

PRIFYSGOL
BANGOR
UNIVERSITY

Object Oriented Programming in Java

Laboratory 6
Interface Types

Dave Perkins

Introduction

This laboratory session provides an introduction to the concept of an **interface.** From reading Horstmann's Chapter 9 in *Big Java:Late Objects* you should know that an interface lists a standardized set of methods which can be invoked on objects of an appropriate class (i.e. a class that implements the interface). In other words the standard methods are the set of methods that the class *implements*.

In Java the interface definition states the names of the methods and their return types and argument signatures. You should note that there is no executable body for any method - that is left to each class that implements the interface. Once a class implements an interface, the Java compiler knows that an instance of the class will contain the specified set of methods, therefore, it will allow you to call those methods for an object referenced by a variable whose type is the interface.

Implementing an interface enables a class to be "plugged in" in any situation that requires a specific behaviour (manifested through the set of methods).

There is no assessed work associated with this laboratory.

Exercise 1: Creating a DataSet Class for Type Double

Create the version of the **`DataSet`** class which provides methods for computing

- the mean(average) of a set of double values;
- the maximum of a set of double values;
- the minimum of a set of double values;

For a definition of the **`DataSet`** class see *Lecture 10 – Interfaces*.

Exercise 2: Develop a DataSetTester

Develop a class called **`DataSetTester`** to perform a *unit test* on the **`DataSet`** class. When you run the test supply five double values as inputs via the keyboard. See *Lecture 10* for details.

Exercise 3: Creating an Interface Type

Following Horstmann's example, write the interface **`Measurable`**. Compile the interface and check that you obtain a file called **`Measureable.class`**.

Exercise 4: Develop a Generalised DataSet Class

Re-write the class **`DataSet`** so that the **`add`** method has the following header:

```
public void add(Measureable x)
```

This class now has greater generality because it can work with instances of any class that implements the **`Measureable`** interface.

Exercise 5: Implementing the Measureable Interface

Modify Horstmann's **`BankAccount`** class so that it implements the **`Measureable`** interface. Test that you have done this correctly by running an integration test involving the classes:

- **`BankAccount;`**
- **`DataSet;`**
- **`DataSetTester.`**

To perform the test create a data set of five different bank accounts.

Exercise 6: More Implementations

Repeat Exercise 5 but this time do it using the classes **`Die`** and **`Counter`**. In both cases perform appropriate tests.

Note that Exercises 7-10 are *Challenge Exercises* and will prove more difficult.

Exercise 7: Developing a Callback Method(Challenge)

Read Horstmann *Big Java* (4th edition) Section 8.4 on the use of *callback* methods (a mechanism for bundling up a block of code so that it can be used at a later time). Note that this material makes reference to the **`Rectangle`** class. Alternatively see:

http://www.iro.umontreal.ca/~pift1025/bigjava/Ch11/ch11.html

http://ece.uprm.edu/~ahchinaei/courses/2014jan/icom4015/Slide09.pdf

Declare the **`Measurer`** interface given below:

```
public interface Measurer
{
      double measure(Object obj);
}
```

Now create a class **RectangleMeasurer** to implement the **Measurer** interface. The code for this class is provided below:

```
public class RectangleMeasurer implements Measurer
{
   public double measure(Object obj)
   {
      Rectangle rect = (Rectangle) obj;
      double area = rect.getWidth() * rect.getHeight();
      return area;
   }
}
```

Test that this class works. Modify **DataSet** so that it works with a **Measurer** object. Test the modification.

For information about the **Rectangle** class see:

https://docs.oracle.com/javase/7/docs/api/java/awt/Rectangle.html

https://docs.oracle.com/javase/tutorial/2d/geometry/primitives.html

Exercise 8: Measuring Rectangles (Challenge)

Create a **Measurer** object to process a set of **Rectangle** objects and so find the object with the largest *perimeter*. To do this you will need to *modify* the class **RectangleMeasurer.**

Exercise 9: Developing a Filter Interface (Challenge)

Declare an interface **Filter** as follows:

```
public interface Filter
{
      boolean accept(Object x);
}
```

Modify the implementation of the DataSet class to use both a Measurer and a Filter object. Only objects the filter accepts should be added to the data set. Test your modification by having a data set process a collection of bank accounts, filtering out all accounts less than £1,000.

Exercise 10: Using the Comparable Interface (Challenge)

The stanadrd Java library provides a **`Comparble`** interface which is specified below:

```
public interface Comparable
{
   /**
      Compares this object with another.
      @param other the object to be compared
      @return a negative integer, zero or a positive integer if this object
      is less than or equal to, or greater than other.
   */
   public int co
}
```

Modify the **`DataSet`** class to accept **`Comparable`** objects. With this interface , it is no longer meaningful to compute the average; instead, the data set should record the maximum and minimum data values.

Test your modified version of **`DataSet`** by adding five strings to the set. The class **`String`** implements the **`Comparable`** interface.