**Advanced Programming 2017 – Year 2**

**Labwork 7: (6% - or 60 points out of 500 points for labwork this semester)**

**NOTE: ALL LABS TO BE COMPLETED IN PROJECTS USING ECLIPSE**

**NOTE: YOU MUST USE YOUR OWN EXAMPLES FOR THESE EXERCISES, I.E., YOU CANNOT RE-USE LECTURE EXAMPLE(S) AS SUBMISSIONS (THESE WILL BE GIVEN ZERO MARKS\POINTS)**

**REVISION EXERCISE 3 – (10 points) – THERE WILL BE 5 x 10 MARKS GOING TOWARD REVISION ACTIVITIES IN THE NEXT 5 LABS**

Create a project called **Revision3**. Create a class called **GenericClassInfoProvider** that defines a **generic attribute** of type <T>. Write a method called called **getSimpleClassName(T type)** that will output the simple name of the class, i.e., use the method **getSimpleName()** of the class Class in the method. Create a second application class called **GenericTestClass** to create pass a generic argument to an object of class **GenericClassInfoProvider** (using the < > angle brackets) and call the **getSimpleClassName(T type)** you have written above (choose any class type or several if you wish to experiment.

Required activities and marking guideline:

* Write the generic class to accept the generic attribute <T> (4 points)
* Write the method to output the getSimpleClassName(T type) (3 points)
* Write test application to test the generic class (3 points)

**Part 1 – Using class *Class* (10 points)**

Create an Eclipse Project called **Lab7Part1**. Create a class called **ReflectionClass** and place it in a package called reflections. Note: When you put a class in a package the full class name is the package name first then the class name, e.g., “*mypackage.MyClass*”. Tests the following **reflection** related methods using this ReflectionClass:

* **getClass()** – use this to get the class to print its own name
* Get the name of the class and print out using the **.class** reference
* Use the **forName()** method to create an instance of the ReflectionClass class and then print its name using **getName()**

Required activities and marking guideline:

* Create the test class (1 point)
* Use getClass() reflection method to output name of class (3 points)
* Use the shorthand .class reference to output the name of the class (3 points)
* Use forName() method to create an instance of the class (3 points)

**Part 2 – Inner classes (10 points)**

Create an Eclipse Project called **Lab7Part2**. Create a class called **GUIWithInnerClass** that simply contains a button and a label (inside panels to make them look neat). Add the ActionListener to the button as normal EXCEPT DO THIS USING AN INNER CLASS CALLED **ButtonResponder** (Note: THIS MUST NOT BE IMPLEMENTED AN ANONYMOUS INNER CLASS IT MUST BE AN INNER CLASS WITH THE CORRECT NAME). Once the button is pushed set the text of the outer class label to say “Outer class label set by inner class ” AND add the name of the inner class to the output label using the getName() method from the class Class (examine the name of the output class!!!...it should be interesting).

Required activities and marking guideline:

* Build GUI in outer class (3 points)
* Add listener to button using inner class (ButtonResponder) (3 points)
* Implement the inner class to change label in outer class (3 points)
* Output name of the inner class in the label using getName() (1 point)

**Part 3 – Anonymous inner classes (10 points)**

Create an Eclipse Project called **Lab7Part3**. Create a class called **GUIWithAnonymousInnerClass** that does exactly the same as Part 2 above EXCEPT USE AN ANONYMOUS INNER CLASS AS THE ACTIONLISTENER. As before once the button is pushed set the text of the outer class label to say “Outer class label set by inner class ” AND add the name of the inner class to the output label using the getName() method from the class Class (again examine the name of the output class!!!...it should be interesting).

Required activities and marking guideline:

* Build GUI in outer class (3 points)
* Add listener to button using anonymous inner class (3 points)
* Implement anonymous inner class to change label in outer class (3 points)
* Output name of the anonymous inner class in the label (1 point)

**Part 4 – Investigating class features using reflection (20 points)**

Create an Eclipse Project called **Lab7Part4**. Create a class called **InvestigatorClass** that will investigate the contents of a class called MysteryClass. You have been supplied with a jarfile called **MysteryClassJar.jar** that contains the compiled code of a mystery class called MysteryClass (note: you will not be supplied with the source code!). Import this jar file into your project using the library import Project->Properties->Libraries (tab)->Add External Jar (button) – point it to your local copy of the MysteryClassJar.jar file. Create an instance of the MysteryClass class in the Investigator class and output the following information using appropriate class reflection methods:

* The fields in the class and their types (there are TWO fields in the class)
* The names of all of the methods in the class (there are THREE methods)
* The return type of each of the methods (each has a non-void return type)
* The parameter type to each of the methods (use toString() to print type of each of the parameters returned via the getParameters() method)

Finally attempt to call ONE of the methods in the Mystery method class using the invoke(Object, Object…) method.

Required activities and marking guideline:

* Get and display all field names in Mystery class (3 points)
* Get and display all field types in Mystery class (3 points)
* Get and display all method names in Mystery class (3 points)
* Get and display all method return types in Mystery class (3 points)
* Get and display all the parameter types of each method (4 points)
* Call any ONE method from Mystery class using **invoke()** (4 points)