Advanced Programming 2025 – Year 2
Labwork 6: (5% - or 50 points out of 300 points for labwork this semester)

NOTE: ALL LABS TO BE COMPLETED IN PROJECTS USING ECLIPSE OR EQUIVALENT IDE (NO MORE TEXTPAD, EVER EVER EVER!!!)

TOPICS INCLUDE: Lambda expression for listeners, Utility class, Inner classes, class reflection, polymorphism

IMPORTANT NOTES:

- **NO COPYING PERMITTED AND ZERO MARKS WILL APPLY TO COPIED WORK. FURTHER ACTION MAY BE TAKEN AGAINST STUDENTS THAT HAVE BEEN FOUND TO COPY WORK.**

- **ASSESSMENT WILL INVOLVE ONE-TO-ONE QUESTIONS ABOUT YOUR SUBMITTED WORK. A COMPLETED SELF-ASSESSMENT SHEET WILL BE USED TO GUIDE THE ASSESSMENT. USE COMMENTS IN YOUR CODE TO ENSURE YOU DON'T FORGET WHY YOU WROTE CODE YOU MAY LATER BE ASKED ABOUT.**

- **ALL WORK MUST BE SUBMITTED TO MOODLE BY DATES SPECIFIED (SUBMISSION DEADLINES WILL BE POSTED ON MOODLE).**

- **MANY OF THE TASKS ASSIGNED BELOW CAN BE COMPLEX AND\OR THE DESCRIPTIONS MAY REQUIRE FURTHER CLARIFICATIONS. PLEASE USE THE AVAILABLE LAB TIMES TO ASK FOR CLARIFICATIONS AND ADVICE\HINTS ON THE TASKS BELOW.**

- **YOU MUST USE AN IDE TO COMPLETE TASKS (e.g., Eclipse or IntelliJ or NetBeans or similar). Support and help provided for Eclipse primarily.**

- **CHATGPT and other similar AI tools that can code simple solutions are NOT PERMITTED. THEY DO NOT TEACH YOU HOW TO BECOME A GOOD PROGRAMMER EITHER!**

Part 1 – Utility class and Lambda listener (10 points)

Create a project called **Lab6Part1**. Create a small **Utility** class with one static method to calculate the circumference of a circle. The method will return the result as a double and use a **final static** variable defined in the class for the value of **PI** (e.g. 3.14159).

Create a small GUI that will input the radius of any circle and call the Utility class method to calculate the circumference of a circle based on the value of the radius input. You will need to supply a button to enact the calculation and label(s) and field(s) to input and output. YOU MUST IMPLEMENT THE BUTTON HANDLER AND LISTENER AS A LAMBDA EXPRESSION USING '->'.

Required activities and marking guideline:

- Create the small **utility class** with static method and final static PI (3 points)
- Create the simple GUI to input radius and output circumference (2 points)
- Implement the handler and listener of button as lambda use '->' (4 points)
- Javadoc the classes and methods (1 point)

Part 2 – Inner\Outer class with anonymous inner class listener (10 points)

Create an Eclipse Project called **Lab6Part2**. Create a main outer JFrame class that has a JTextArea, JButton and a String field. The text area and button will be added to the frame using a reasonable layout.
Create an INNER CLASS in the same Java file as the outer class frame and call it **InnerInfoClass**. This class will perform the following tasks:

1. It will add a button listener and handler to the outer class button using the **ANONYMOUS INNER CLASS** approach.
2. Using the button, the inner class will add the name of the inner class AND the name of the outer class to the text area. The inner class will use class reflection to gather the information and an outer class reference to set the text area. You can use the outer class String field to build up the string to add to the text area.
3. Also using the button of the outer class, the inner class will add the names of the fields found in the outer class to the text area of the outer class The inner class will use class reflection to gather the information and an outer class reference to set the text area. Again you can use the outer class String field to build up the string to add to the text area.

Required activities and marking guideline:

- Create outer class with GUI (will instantiate inner class instance) (2 points)
- Create inner class in the same Java file (2 points)
- Add handler and listener to button in INNER class using **anonymous inner class** approach (2 points)
- Achieve the output of the inner and outer class name to text area (2 points)
- Achieve the output of the outer class field names to the text area using anonymous inner class handler method (2 points)

Part 3 – Class reflection – investigate a class (15 points)

Create a Java JFrame class called **InvestigatorClass**. Add a text area and a button to the class. Import the **MysteryJar** jarfile contained in the code listing for Lab 6 (file called MysteryJar.jar). Investigate the **MysteryJarClass** (contained in the jarfile with NO SOURCE) using class reflection (in package mystery). Find out the names of all of the methods and the number and type of all parameters to the each method and lay them out for display nicely in the JTextArea. One of the methods in the **MysteryJarClass** has no parameters, when you find this method call it using **invoke**. (Note: for potential private methods you may have to call *setAccessible(true)* to read them and call them, otherwise you might get a "private" access error…also this lab will be easy if you do not include a **module** in your project)

Required activities and marking guideline:

- Create the frame with button and text area for display (1 point)
- Add the listeners and handlers to the button to update text area (1 point)
- Import the MysteryJar and MysteryJarClass (1 point)
- Investigate the names of all of the methods and add to text area (4 points)
- Add parameter type for each method parameter found to text area (4 points)
- Call the blank parameter method when you find it using invoke (4 points)

Part 4 – Applying Polymorphism with images and GUI (15 points)

Create an Eclipse Project called **Lab6Part4**. Create a class called **MyAlbum** which will display your favourite images and some information about what is shown in the image in a GUI with a JLabel (e.g., Animals and Sports) by applying polymorphism using a polymorphic methods called **getImage()** and **getFacts()**. The facts listed for each sub-type can be displayed in a JTextArea added to the GUI. Set specific attributes for each sub-type. Sports can have the attributes originOfSport and mainScoreType (e.g. "goal" for soccer). Your program will provide a superclass called **AlbumItem** and the two subclasses (containing concrete implementations of the polymorphic getImage() and getFacts() methods e.g. **FavouriteSport** and **FavouriteAnimal** classes). The MyAlbum class will store a Vector of **AlbumItem** objects used for displaying in the JLabel by calling the polymorphic **getImage()** method. The GUI will provide a forward button to page through your favourite images and facts relating to each of the images currently being displayed. (Note: Do not use large images for this project it will just needlessly make the project submission too large.)

Required activities and marking guideline:

- Create the abstract AlbumItem class (2 points)
- Create the 1st subclass (e.g. FavouriteSport) – with constructor (2 points)
- Create the 2nd subclass (e.g. FavouriteAnimal) – with constructor (2 points)
- Create the MyAlbum class with Vector (3 points)
- Create at least 5 objects of each subclass and add to Vector (3 points)
- Use **dynamic binding** to display images and facts (3 points)