Advanced Programming 2021 – Year 2
Labwork 8: (This lab is worth 60 points out of 500 points for labwork this semester)

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

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. 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 (2 LABS SUBMISSIONS OF FIVE LABS THROUGHOUT THE SEMESTER).**

- **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.**

Part 1 – Polymorphism: use an interface instead of a class (15 points)

Create an Eclipse Project called **Lab8Part1**. Create a **Car** class hierarchy. Create an **interface** to represent a generic Car. The Car **interface** must provide an **abstract** method called **changeGearInstructions()** that will provide instructions at the subclass level to the driver on how to change gears by returning a String of instructions. Create TWO subclasses called AutomaticCar and ManualCar that implement the Car interface and implement the **changeGearInstructions()** method appropriate to the car type (i.e. the automatic car gear change instruction will not include pushing the clutch whereas the manual car will include pushing the clutch). Create a test program called **TestCars** that creates several instances of Cars (some Automatic and some Manual) and places them on a Vector<Car>. Use a loop to output the gear change instructions of all of the cars added to the vector using **dynamic binding**.

Required activities and marking guideline:

- Car interface implemented (MUST use an **interface**) (3 points)
- AutomaticCar class implementing interface (3 points)
- ManualCar class implementing interface (3 points)
- Class to create Car vector and print info using dynamic binding (6 points)

Part 2 – Polymorphism example (20 points)

Create an Eclipse Project called **Lab8Part2**. Create your own hierarchy to represent polymorphism based on some everyday objects of which there are many types (think of something from you own life, e.g., sport or hobbies). Create at least one generic superclass and at least two specific subclasses. Set specific attributes for the subtypes and create a polymorphic method for output of the specific information that can be placed in a generic Vector<Superclass>. Create a test program to output a list of ten of the objects in your hierarchy and use dynamic binding to output the details of the objects from the generic vector.

Required activities and marking guideline:

- Superclass created (3 points)
- First sub-class with specific attributes\behaviours (4 points)
- Second sub-class with specific attributes\behaviours (4 points)
- Create at least 10 objects of the sub-types in test class vector (5 points)
- Demonstrate dynamic binding by outputting objects in vector (4 points)

Part 3 – Applying Polymorphism with images and GUI (25 points)

Create an Eclipse Project called **Lab8Part3**. 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 (4 points)
- Create the 1st subclass (e.g. FavouriteSport) – with constructor (4 points)
- Create the 2nd subclass (e.g. FavouriteAnimal) – with constructor (4 points)
- Create the MyAlbum class with Vector (4 points)
- Create at least 5 objects of each subclass and add to Vector (5 points)
- Use **dynamic binding** to display images and facts (4 points)