Lab Work – Java refresher

Eclipse Use Eclipse for java development in this week's lab (unless you want to use a different editor on your own laptop). Open up Eclipse on the machine. You may be prompted to set the workspace (this is where code will be stored). If so, just set to **your** u drive (or wherever you want to store your code)

Set up a new java project / New package (reverse URL is the convention)/ New class

PART 1 – Person class

Create a `Person` class that holds the following information about a person: name and gender (either 'M' or 'F').

Add a constructor for the Person class that initialises the two instance variables, name and gender, to values passed as parameters in. The constructor should ensure gender is always stored as an upper case value.

Make sure your attributes are **encapsulated**.

Create a separate `Control` class that contains the java "main" method and instantiate two people objects; include statements to print them out (using System.out.println(obj_name);

Run the control class. **What does the output shown mean** when you run those print commands mean?

Part 2 – Person Class with more details.

Add a `toString()` method to the Person class that prints out all the person's details in a reasonably formatted way. Check out the `toString()` method at the top level Object class (Java API link is available above) that this method is overriding.

Re-run the Control class.

Part 3 - Inheritance

1. Create a Student class to become a a subclass of Person. In addition to name and gender, student should hold a **studentId**.

2. Add a constructor to the Student class that initialises all three instance variables to values passed in as parameters. (Don't forget to make use of the constructor from

the Person superclass).

3. Add a `toString()` method to the Student class that prints out the three the various the student details. Make sure to re-use Person class methods where you can.

Part 4 – Interfaces

Interfaces are java's way of sharing and enforcing behaviour across classes.

1. Write an interface "PublishDetails" that contains two methods: confirmDetails(); getCourseCode().

2. Get the student class to implement the PublishDetails interface. **What happens if** the interface is implemented at the class level (i.e. using "implements" but the methods confirmDetails() getCourseCode() are not included in your student class?

3. For the student class, ConfirmDetails() should print information about the student name; getCourseCode() should print out the course code.

4. Instantiate a student object in your control Main class, and test out the Parts 3 and 4 are working by calling the interface behaviou

Part 5 – Object types

1. In java `instanceof` keyword is a binary operator used to test if an object (instance) is a subtype of a given Type. The "Type" of an object is the class that it is an instance of. E.g. if you have a person object "boy1".. `boy1 instanceof Person` will return true.

   Check out: Is your student object from Part 4 of type student? Should be!
   Now try: Is your student object of type PublishDetails? **Why do you get the answer you get?** Is your Person object of type PublishDetails?

Part 6 – Static

1. Add studentIDCounter as a static field to your student class. This holds the next student number to be allocated. Change your code so that instead of passing in the studentID, the id is allocated in the constructor. **Why does the static variable help**?