**Lab Work**

---------------------------------------------------------------------------------

Part 1 The concept of static in java

Part 2 Abstract classes

Part 3 More on interfaces

----------------------------------------------------------------------------------

Note: You’ll need the following formulae in case you can’t remember them…

Pi= 3:14

Area of a circle = pi \*radius\*radius

Volume of a sphere = 4/3\*pi\*radius\*radius\*radius

Surface area of a sphere = 4\*pi\*radius\*radius

**Part 1**

Last week, you created a person class – and a student class that inherited from the person class. The code is on webcourses. Please make sure that you understand what each class is doing.

* Add another constructor to the Student class that initialises the name and gender to values passed as parameters but automatically assigns a unique student id number . The id should increment by 1 for each new student.
* Test your code by instantiating several students from your main method in your control class. Print their details (using toString() in the Student class) and make sure the unique ID is working.

**Part 2 Abstract classes**

From our lecture, an abstract class represents some generic concept. Subclasses then provide their own specific implementations of any abstract methods of the abstract class.

* Implement an **abstract** class called Shape –Include an abstract method double area() that returns the area of the shape.
* Implement two concrete **subclasses** of Shape: A Square class and a Circle class. Implement the area() abstract method in each of these two subclasses. Include any accessor methods (i.e. “get” methods) needed and a toString() method in each subclass. The toString() method should print out the type of shape it is, and it’s area.
* Make sure “pi” is a constant that can’t be edited!

3. AS before, your control class will contain a main method. Edit the control class that tests these classes by setting up an *array* of shapes and initialising each element of the array to a different shape, either a circle or a square. Then print out the details of *each* shape, including its *class* (see getClass() method at Object level) and the *area* of the shape.

**Part 3 More Interfaces**

As before, an interface is a set of unimplemented methods. When a class implements an interface, the methods in the interface must be implemented in the class.

* Set up an *interface* called ThreeDShape that has represents a 3D shape. Include an abstract method that allows the calculation of the *volume* of the 3D shape.
* Amend the class hierarchy for shape to include two new subclasses representing the 3-d shapes **cube** and **sphere**. Get these classes to implement your new interface ThreeDShape.
* Implement the area() method for the two new classes (cube and sphere) to return the surface area of the shape.
* Amend the test class to test the new classes and methods you have added by including the new shapes (cube and sphere) in the array you have setup.