Tutorial 2 – Object Oriented Design

Today we are going to focus one of the key features of Java: object orientation. This tutorial will help you understand how to create and understand the fundamental concepts of classes and how to create and use simple UML diagrams to represent such classes.

1. Remind ourselves of what the following keywords for OO in java mean:

|  |  |
| --- | --- |
| public |  |
| private |  |
| protected |  |
| static |  |

2.Write a basic Java class for a sphere that has the following: a double for radius, a three element  array  for  the  Cartesian  coordinates  of  the  shape  (centroid),  a  count  that  is  shared  between  all instances of sphere (count how many spheres the code is using).

Give it a constructor that lets us set the radius and position (as three doubles) of the sphere (make  sure your class increments the allocation count).  Implement a method to get the count of allocated spheres and the computations for surface area and volume.

NB: the surface area of a circle is 4πr2 and the volume is πr3. Skeleton code is provided below for you to complete.

public class Sphere {

//TODO

public double surfaceArea() {

//TODO

}

public double volume(){

//TODO

}

public int getCount(){

//TODO

}

}

3. What would the output of the following code be?

Sphere sphere1 = new Sphere();

Sphere Sphere2 = new Sphere();

System.out.println(sphere1.getCount());

|  |
| --- |
|  |

4. Create a class to represent a Car object. For the **state**: each car has a year and make, eg 2005, “Mazda”. As well as this each car has a price, current speed, and current gear.

The behaviour that the cars needs to implement include getting the current speed, and current gear and incrementing the speed by a certain amount. Complete the skeleton class below.

|  |
| --- |
| public class Car {  //TODO    /\*  Constructor for Car object that is passed the individual state of the car  \*/  public Car(//TODO//) {  }  public double getCurrentSpeed() {  //TODO  }  public void incrementSpeed(double speed){  //TODO  }    public String getCurrentGear(){  //TODO  }  } |

5. Class Diagrams

Create class diagrams for:

1. The Sphere class you created above

|  |
| --- |
|  |

1. The Car class you created above

|  |
| --- |
|  |

Now create the following class diagrams, drawing the appropriate associations and multiplicities between classes:

1. There is a Lecturer and Course class. Each lecturer has a name and a list of courses they are teaching, which can be zero or more. Each course has a name, course number and room and must be taught by exactly one Lecturer.

|  |
| --- |
|  |

1. There is a Library and Book class. Each Library has a name and location as well as a list of books it currently contain which must be greater than 1000. Each book has a name and genre and the Library it belongs to. Each book can only belong to exactly Library.

|  |
| --- |
|  |