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	**Can be accessed outside the class (anywhere).**
private	**Can only be accessed inside the class it is declared in.**
protected	**Can be accessed by child classes.**
static	**Is shared by all instances of the declaring class**

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 area of a circle is $4\pi r^2$ and the surface area is $\frac{4}{3}\pi r^3$. Skeleton code is provided below for you to complete.

```
public class Sphere {
private double r;
private double[] loc = new double[3];
private static int count = 0;


public Sphere(double x, double y, double z, double r) {
this.r = r;
loc[0] = x;
loc[1] = y;
loc[2] = z;
count++;
}


public double surfaceArea() {
return 4d*Math.PI*(r*r);
}


public double volume(){
return (4d/3d)*Math.PI*(r*r*r);
}


public int getCount(){
return count;
}
}
```

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

```
Sphere  sphere1 = new Sphere();
Sphere Sphere2 = new Sphere();
System.out.println(sphere1.getCount());
```

```
2 – there is only one count field and it is shared by ALL
spheres
```

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.

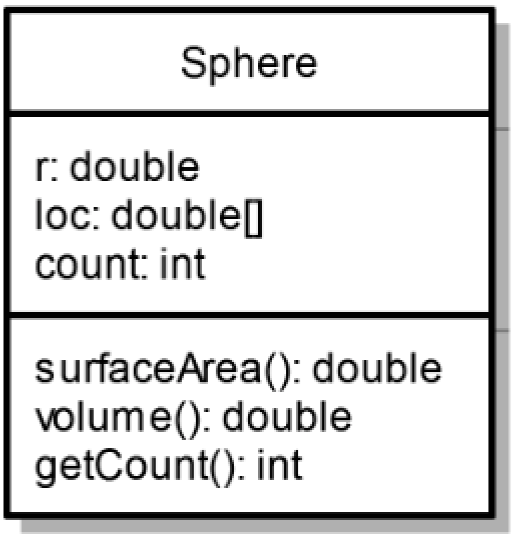
The behaviour that the cars needs to implement include getting the current speed, and current gear and incrementing the speed by a certain amount. Complete the skeleton class below.

```
public class Car {
  private int year;
  private String make;
  private double price;
  private double currentSpeed;
  private String currentGear;

  /*
  Constructor for Car object that is passed the individual state of the
car
  */
  public Car(int year, String make, double price, double currentSpeed,
String currentGear) {
    this.year=year;
    this.make=make;
    this.price=price;
    this.currentSpeed=currentSpeed;
    this.currentGear=currentGear;

  }

  public double getCurrentSpeed() {
    return currentSpeed;
  }


  public void incrementSpeed(double speed){
    currentSpeed+= speed;
  }
  public String getCurrentGear(){
    return currentGear;
  }
}
```

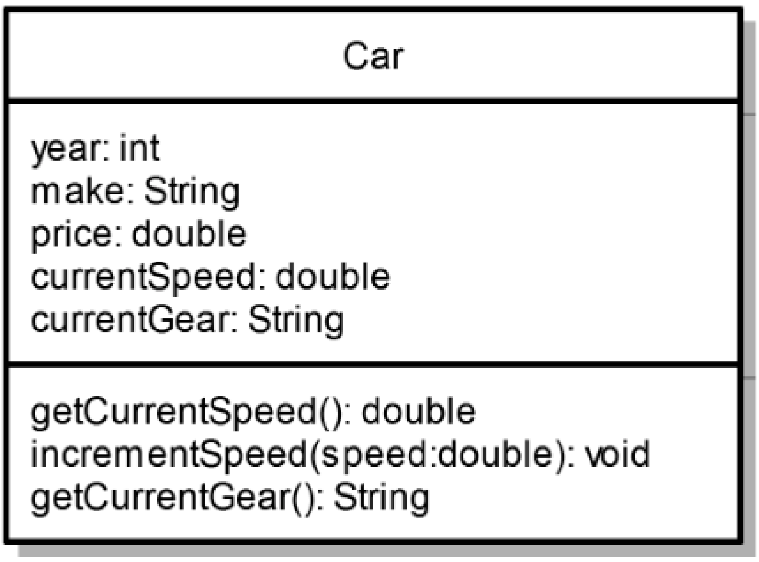
5. Class Diagrams

Create class diagrams for:

a.) The Sphere class you created above

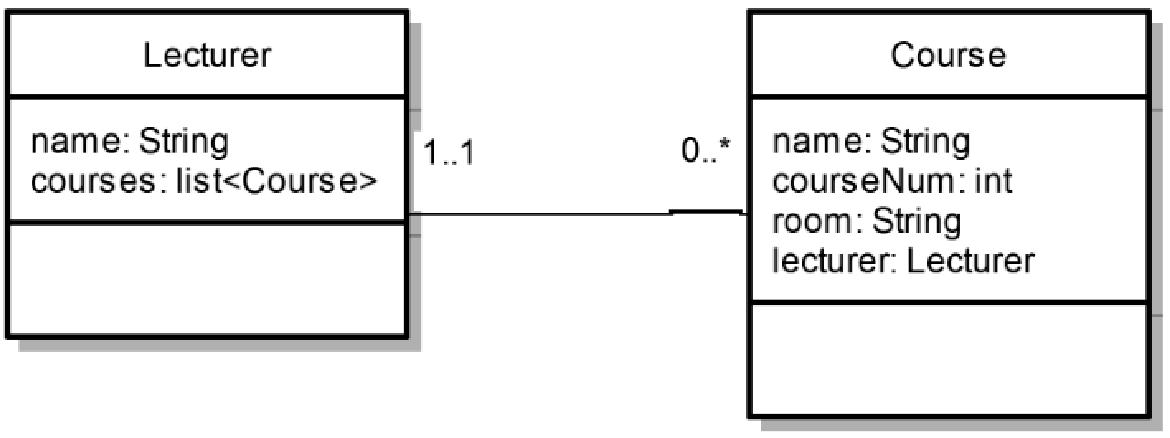


b.) The Car class you created above

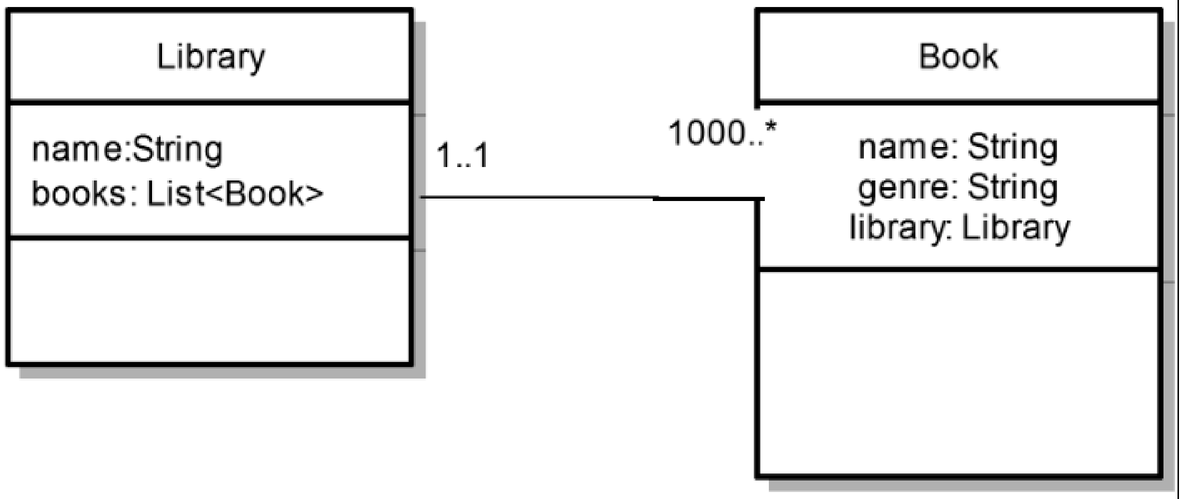


Now create the following class diagrams, drawing the appropriate associations and multiplicities between classes:

c.) 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.



d.) 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.


Library
name:String
books: List<Book>
1..1
1000..*
Book
name: String
genre: String
library: Library