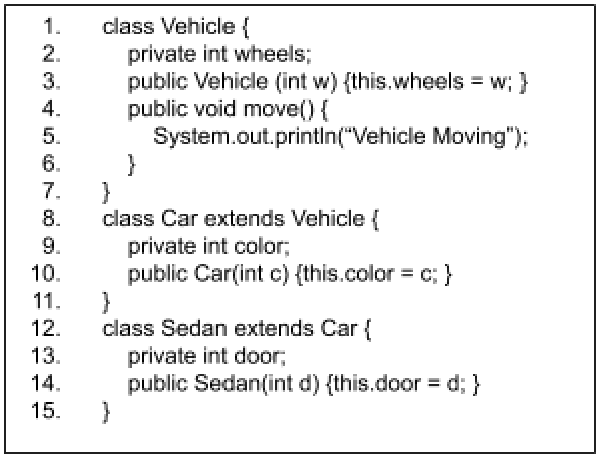
Advanced Programming (CSE201), Quiz -2

Time allocated: 04:15pm – 4:35pm (20 minutes)

Instructions:

- This is a closed book quiz.
- Only reasonable and clearly mentioned assumptions (if any) would be accepted.
- For justifications, please be as concise as possible (2-3 sentences only)
- **IIIT plagiarism policy is applicable if any such cases found**
- **Write your answers on a plain sheet that you can upload by taking a picture of the same (ensure low resolution so that the upload size is smaller)**
- **You can either upload your solution on Google Form link or email the quiz solutions to "ap-m2020-submission@iiitd.ac.in". Only one mode of submission!**
- **Subject of the mail should be Quiz-2.**
- **We will not consider any submission that is emailed beyond 4.45pm. It is your responsibility to ensure you email it or submit through google form on time. Please ensure you have proper internet connectivity as we are giving you sufficient extra time to send the email.**

Question-1) Refer to Figure-1 and answer the below questions:



```
class Vehicle {
   private int wheels;
   public Vehicle (int w) {this.wheels = w; }
   public void move() {
      System.out.println("Vehicle Moving");
   }
}
class Car extends Vehicle {
   private int color;
   public Car(int c) {this.color = c; }
}
class Sedan extends Car {
   private int door;
   public Sedan(int d) {this.door = d; }
}
```

a) Is the code shown correct? What fix is/are required? **[2 marks]**

   I. **Car constructor should accept two parameters: color and wheels.**
      i. **+0.5 marks**
   II. **Car constructor should call super(wheels)**
      i. **+0.5 marks**
   III. **Sedan constructor should accept three parameters: door, color, and wheels**
      i. **+0.5 marks**
   IV. **Sedan constructor should call super(color, wheels)**
      i. **+0.5 marks**

b) Demonstrate partial overriding of move() in Car class and full overriding of move() in Sedan class. Your code should accurately follow best programming practices for OOP. You are **not** allowed to do any changes in Vehicle class. **(No pseudocode). [2 marks]**

   I. **Partial overriding:**

```
@Override                        // Used in both Car/Sedan: +0.4
public void move() {             // Same declaration in both Car/Sedan: +0.4
   super.move();                 // +0.4
   do_something_more();          // +0.4
}
```

   II. **Fully overriding:**

```
@Override
public void move() {
   do_something_more();          // +0.4
}
```

c) Consider the updated version of this program you have after answering above two questions. There is another class Main with the following set of statements. Which move() method would be called in each case below:

   I. Create a Sedan type object and typecast it to a Car and then call move() using that Car object. **[1 marks]**

i. **Sedan's move()** **(+1 marks)**

II. Create a Car type object and typecast it to a Sedan and then call move() using that Sedan object. **[1 marks]**

i. **Run-time error. We can't cast it to a child class** **(+1 marks)**

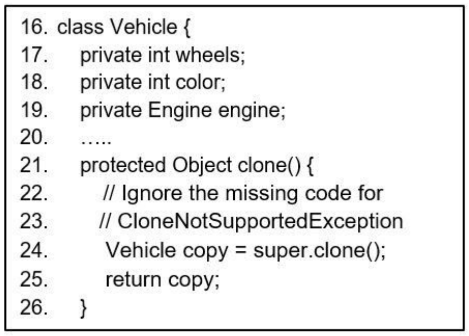
III. C1 is a Sedan type object and V1 is a Vehicle type object obtained after typecasting C1 to a Vehicle. Is the below statement legal, if yes, then what is the output? **[2 marks]**

i. **Yes and it will return true** **(1+1 marks)**

```
System.out.println(C1.getClass() == V1.getClass());
```

Question-2) Correct the code shown for the Vehicle class in Figure-2. **[2 marks]**

```
class Vehicle implements Cloneable { // +0.5
 ……
 public Vehicle clone() {     //+0.5+0.5
   ….
   Vehicle copy = (Vehicle)super.clone(); //+0.5
   ….
 }
```



```
class Vehicle {
    private int wheels;
    private int color;
    private Engine engine;
    .....
    protected Object clone() {
       // Ignore the missing code for
       // CloneNotSupportedException
       Vehicle copy = super.clone();
       return copy;
    }
```