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 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:

```
1.  class Vehicle {
2.      private int wheels;
3.      private int cylinders;
4.      public Vehicle (int w) {this.wheels = w; }
5.      public void run() {
6.          System.out.println("Vehicle running");
7.      }
8.  }
9.  class Car extends Vehicle {
10.     private int color;
11.     public Car(int c) {this.color = c; }
12. }
13. class Hatchback extends Car {
14.     private int door;
15.     public Hatchback(int d) {this.door = d; }
16. }
```

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

I. Vehicle constructor should accept two parameters: wheels and cylinders.
  - **i. +0.25 marks**

II. Car constructor should accept three parameters: color, wheels and cylinders.
  - **i. +0.25 marks**

III. Car constructor should call super(wheels, cylinders)
  - **i. +0.5 marks**

IV. Hatchback constructor should accept four parameters: door, color, cylinders and wheels
  - **i. +0.5 marks**

V. Hatchback constructor should call super(color, wheels,cylinders)
  - **i. +0.5 marks**

b) Demonstrate partial overriding of run() in Car class and full overriding of run() in hatchback 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/Hatchback: +0.4
public void run() {          // Same declaration in both Car/Hatchback: +0.4
   super.run();              // +0.4
   do_something_more();      // +0.4
}
```

II. Fully overriding:

```
@Override
public void run() {
   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 run() method would be called in each case below:

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

i. Hatchback's run() (+1 marks)

II. C1 is a Hatchback 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, else correct and show the output? **[2 marks]**

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

No, the statement is not legal. (+1 marks)
Corrected code: System.out.println(C1.getClass() == V1.getClass())(+1 marks)
It will return true

III. Create a Car type object and typecast it to a Hatchback and then call run() using that Hatchback object. **[1 marks]**

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

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
  ….
 }
```

```
16. class Vehicle {
17.     private int wheels;
18.     private int weight;
19.     private Engine engine;
20.     …..
21.     protected Object clone() {
22.         // Ignore the missing code for
23.         // CloneNotSupportedException
24.         Vehicle copy = super.clone();
25.         return copy;
26.     }
```