Quiz-1 - CSE201

Q1) Select the correct option:

```
public class Car
{
    public void drive() {
        System.out.println("Driving car");
    }

    public static void main(String[] args) {
        Sedan p = new Sedan();
        p.drive();
        Vehicle q = p;
        q.drive();
          }
}

interface Vehicle {
    public void drive();
}

class Sedan extends Car implements Vehicle {
    @Override
    public void drive() {
        super.drive();
        System.out.println("Driving sedan");
    }
}
```

a) Compilation Error
b) Driving car
   Driving sedan
   Driving car
   Driving sedan
c) Runtime Error
d) Driving sedan
   Driving car
   Driving sedan
e) None of the above

Ans: b

Q2) Which of the following are true about @Override annotation?

a) It enforces the compiler to check if such a method is present in the parent class or not.
b) Enforces Encapsulation.

c) Without it, overriding cannot happen
d) None of the above

Ans: a

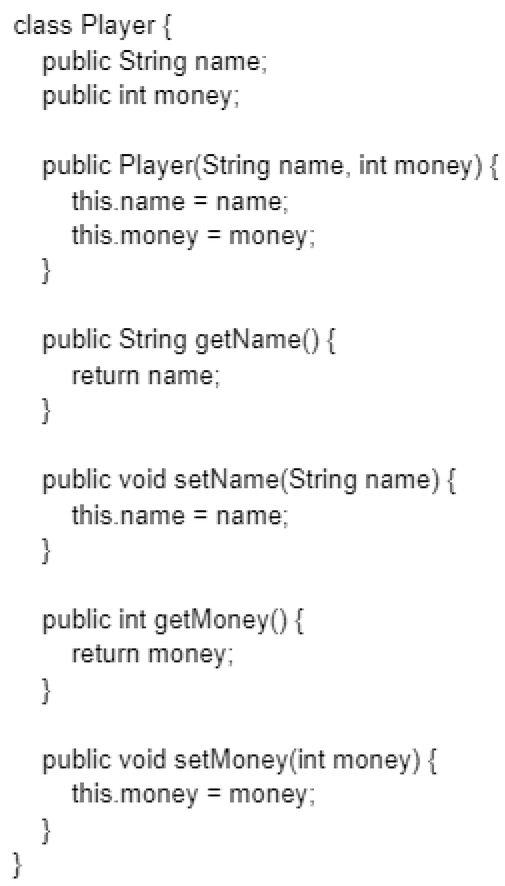
Q3) Find the output.

```
class Lion {
   public void roar() { System.out.println("I roar loud"); }
}

public class Cub extends Lion {
   @Override
   public void roar() {
      this.roar();
      System.out.println("I can roar too");
   }

   public static void main(String[] args) {
      Cub simba = new Cub();
      simba.roar();
   }
}
```

a) Infinite Loop
b) I roar loud
   I can roar too
c) I can roar too
d) Compilation Error
e) Runtime Error

Ans: a

Q4) A Software engineer at a gaming company was almost fired for writing this code because he missed an essential OOP concept. Can you name that OOP concept?



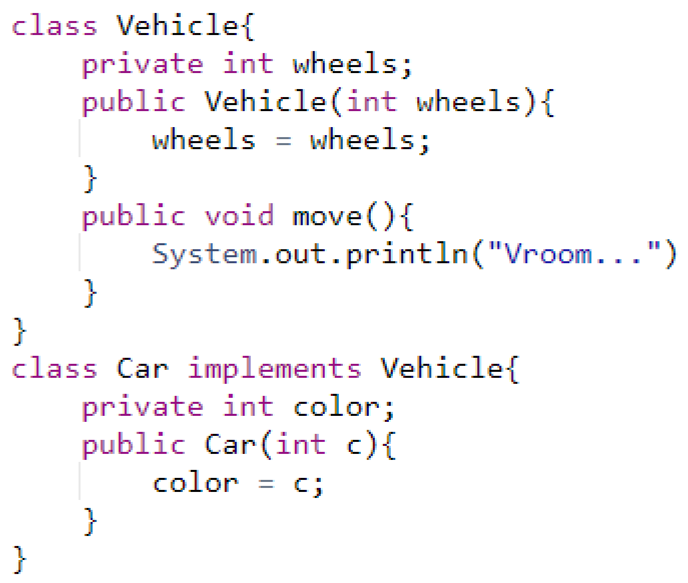
```
class Player {
    public String name;
    public int money;

    public Player(String name, int money) {
        this.name = name;
        this.money = money;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getMoney() {
        return money;
    }

    public void setMoney(int money) {
        this.money = money;
    }
}
```

a) Polymorphism
b) Inheritance
c) Encapsulation
d) Abstraction

Ans: c

Q5) Which of the following is true about Abstract class and Interface?
a) Abstract class can have constructor, interface can have constructor
b) Abstract class cannot have constructor, interface can have constructor
c) Abstract class can have constructor, interface cannot have constructor
d) Abstract class cannot have constructor, interface cannot have constructor
Ans) c

Q6)



```
class Vehicle{
    private int wheels;
    public Vehicle(int wheels){
        wheels = wheels;
    }
    public void move(){
        System.out.println("Vroom...")
    }
}
class Car implements Vehicle{
    private int color;
    public Car(int c){
        color = c;
    }
}
```

Which of the following corrections are needed for the above code?
a)Missing this keyword in Vehicle constructor
b)Should be ‘extends’ instead of ‘implements’
c)Car constructor should invoke constructor of Vehicle class using super()
d)Missing semicolon in the print statement in the move method
e) All of the above

Ans) e

Answer Questions 7 and 8 based on the following code:

```
class AC
{
    public AC() { }

    public void setTemperature(int temperature) { }
}

class Car
{
 AC airConditioner = new AC();
    public Car() { }

    public void move() {

        airConditioner.setTemperature(25);
    }

}

class Driver
{
    public Car taxi;

    public Driver(Car c)
    {
        this.taxi = c;
    }
}
```

Q7) What is the relationship between Car and AC?

a) Car contains AC (Composition)
b) Car knows-about AC (Association)
c) Car depends on AC (Dependency)
d) Car inherits AC (Inheritance)
e) None of the above

Ans: a

Q8) What is the relationship between Car and Driver?

a) Driver contains Car (Composition)
b) Driver knows-about Car (Association)
c) Driver depends on Car (Dependency)
d) Driver inherits Car (Inheritance)
e) None of the above

Ans: b

Q9) What is the output of the following code?

```
class Main{
        public static void main(String args[]){
                Parent[] p={new Child1(),new Parent(),new Child2()};
                func(p);
        }

        static void func(Parent[] p){
                for(int i=p.length-1;i>0;i--){p[i].disp();}
        }
}
class Parent{
        Parent(){}
        void disp(){System.out.println("Parent");}
}

class Child1 extends Parent{
        void disp(){System.out.println("Child1");}
}

class Child2 extends Parent{
        void disp(){System.out.println("Child2");}
}
```

a) Child2
   Parent
   Child1
b) Child1
   Parent
   Child2
c) Child2
   Parent
d) Parent
   Child1

Ans) c

Q10) What’s the output of the following code?

```
public class Main
{
    public static void main(String[] args)
    {
        Person p = new Professor();
        p.wakeUp(10 + "");
    }
}

class Person
{
    public void wakeUp(String persons)
    {
        System.out.println("Send Good Morning message to " + persons + " persons");
    }
}

class Professor extends Person
{
    public void wakeUp(int students)
    {
        System.out.println("Today I will fail " + students + " students");
    }
}
```

a) Compiler Error
b) Send Good Morning message to 10 persons
c) Runtime error
d) Today I will fail 10 students

Ans) b