

# VIT<sup>®</sup>

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## **Computer Programming Java**

### **MOODLE LAB TASK 5**

**Faculty: Sathya Raj R**

**Slot: L3+L4+L29+L30**

**Venue: SJT 515**

**NAME: Soumyojyoti Saha**

**Reg ID: 21BCE4007**



## Inheritance Questions

**Q1. Write a Java program to display the salary and bonus of a Programmer class, where Programmer extends an Employee class. The Employee class has a salary attribute, and the Programmer class has an additional bonus attribute. The program should print the salary and bonus of a Programmer instance.**


### CODE:

```
class Employee {  
    float salary = 40000;  
}  
  
class Programmer extends Employee {  
    int bonus = 10000;  
  
    public static void main(String[] args) {  
        Programmer p = new Programmer();  
        System.out.println("Programmer salary is: " + p.salary);  
        System.out.println("Bonus of Programmer is: " + p.bonus);  
    }  
}
```

## OUTPUT:

```
J Programmer.java >  Programmer >  main(String[])
1  class Employee {
2      float salary = 40000;
3  }
4
5  class Programmer extends Employee {
6      int bonus = 10000;
7
8      Run | Debug
9      public static void main(String[] args) {
10         Programmer p = new Programmer();
11         System.out.println("Programmer salary is: " + p.salary);
12         System.out.println("Bonus of Programmer is: " + p.bonus);
13     }
14 }
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> cd "c:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7" & if (\$?) { java Programmer }
- Programmer salary is: 40000.0
- Bonus of Programmer is: 10000
- PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> 

**Q2. Write a Java program to demonstrate inheritance where:**

- 1. An Animal class has a method eat() that prints "eating...".**
- 2. A Dog class extends the Animal class and adds a method bark() that prints "barking...".**
- 3. In the TestInheritance class, create an instance of Dog and call both bark() and eat() methods on it. The program should print the outputs of both methods.**

**CODE:**

```
class Animal {  
    void eat() {  
        System.out.println("eating...");  
    }  
}  
  
class Dog extends Animal {  
    void bark() {  
        System.out.println("barking...");  
    }  
}  
  
class TestInheritance {  
    public static void main(String[] args) {  
        Dog d = new Dog();  
        d.bark();  
        d.eat();  
    }  
}
```

## OUTPUT:

```
J TestInheritance.java > ...
1  class Animal {
2      void eat() {
3          System.out.println(x:"eating...");
4      }
5  }
6
7  class Dog extends Animal {
8      void bark() {
9          System.out.println(x:"barking...");
10     }
11 }
12
13 class TestInheritance {
14     Run | Debug
15     public static void main(String[] args) {
16         Dog d = new Dog();
17         d.bark();
18         d.eat();
19     }
20 }

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> cd "c:\Users\Soumyojyoti
.java } ; if ($?) { java TestInheritance }
barking...
eating...
PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> 
```

**Q3. Write a Java program to demonstrate multilevel inheritance where:**

- 1. An Animal class has a method eat() that prints "eating...".**
- 2. A Dog class extends the Animal class and adds a method bark() that prints "barking...".**
- 3. A BabyDog class extends the Dog class and adds a method weep() that prints "weeping...".**
- 4. In the TestInheritance2 class, create an instance of BabyDog and call the weep(), bark(), and eat() methods on it. The program should print the outputs of all three methods.**

**CODE:**

```
class Animal {  
    void eat() {  
        System.out.println("eating...");  
    }  
}  
  
class Dog extends Animal {  
    void bark() {  
        System.out.println("barking...");  
    }  
}  
  
class BabyDog extends Dog {  
    void weep() {  
        System.out.println("weeping...");  
    }  
}
```

```

class TestInheritance2 {
    public static void main(String[] args) {
        BabyDog d = new BabyDog();
        d.weep();
        d.bark();
        d.eat();
    }
}

```

## OUTPUT:



The screenshot shows an IDE with a Java file named `TestInheritance2.java`. The code defines three classes: `Animal`, `Dog` (which extends `Animal`), and `BabyDog` (which extends `Dog`). The `TestInheritance2` class has a `main` method that creates a `BabyDog` object and calls its `weep`, `bark`, and `eat` methods. The output window shows the results of these calls: `weeping...`, `barking...`, and `eating...`.

```

J TestInheritance2.java > TestInheritance2
1  class Animal {
2      void eat() {
3          System.out.println(x:"eating...");
4      }
5  }
6
7  class Dog extends Animal {
8      void bark() {
9          System.out.println(x:"barking...");
10     }
11 }
12
13 class BabyDog extends Dog {
14     void weep() {
15         System.out.println(x:"weeping...");
16     }
17 }
18
19 class TestInheritance2 {
20     public static void main(String[] args) {
21         BabyDog d = new BabyDog();
22         d.weep();
23         d.bark();
24         d.eat();
25     }
26 }
27

```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> cd "c:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7" & if ($?) { java TestInheritance2 }
weeping...
barking...
eating...
PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7>

```

**Q4. Write a Java program to demonstrate multiple inheritance where:**

- 1. An Animal class has a method eat() that prints "eating...".**
- 2. A Dog class extends the Animal class and adds a method bark() that prints "barking...".**
- 3. A Cat class extends the Animal class and adds a method meow() that prints "meowing...".**
- 4. In the TestInheritance3 class, create an instance of Cat and call the meow() and eat() methods on it. The program should print the outputs of both methods. Uncommenting the line c.bark(); should result in a compile-time error.**

**CODE:**

```
class Animal {  
    void eat() {  
        System.out.println("eating...");  
    }  
}  
  
class Dog extends Animal {  
    void bark() {  
        System.out.println("barking...");  
    }  
}  
  
class Cat extends Animal {  
    void meow() {  
        System.out.println("meowing...");  
    }  
}
```



```

class TestInheritance3 {
    public static void main(String[] args) {
        Cat c = new Cat();
        c.meow();
        c.eat();
        // c.bark(); // Uncommenting this line will cause a compile-time error
    }
}

```

## OUTPUT:



The screenshot shows an IDE with a Java file named `TestInheritance3.java`. The code defines three classes: `Animal`, `Dog` (which extends `Animal`), and `Cat` (which also extends `Animal`). The `Animal` class has an `eat()` method that prints "eating...". The `Dog` class has a `bark()` method that prints "barking...". The `Cat` class has a `meow()` method that prints "meowing...". The `TestInheritance3` class has a `main` method that creates a `Cat` object, calls `meow()`, and `eat()`. A commented-out line `// c.bark();` is present, with a note that uncommenting it would cause a compile-time error.

The output window shows the results of running the program: "meowing..." followed by "eating...". The terminal window shows the command prompt where the program was executed.

```

J TestInheritance3.java > Cat > meow()
1 class Animal {
2     void eat() {
3         System.out.println(x:"eating...");
4     }
5 }
6
7 class Dog extends Animal {
8     void bark() {
9         System.out.println(x:"barking...");
10    }
11 }
12
13 class Cat extends Animal {
14     void meow() {
15         System.out.println(x:"meowing...");
16     }
17 }
18
19 class TestInheritance3 {
20     public static void main(String[] args) {
21         Cat c = new Cat();
22         c.meow();
23         c.eat();
24         // c.bark(); // Uncommenting this line will cause a compile-time error
25     }
26 }
27
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> cd "c:\Users\Soumyojyoti
3.java } ; if ($?) { java TestInheritance3 }
meowing...
eating...
PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7>

```

**Q5. Write a Java program to demonstrate inheritance and method overriding where:**

- 1. A Bicycle class has two fields: gear and speed, and a constructor to initialize these fields. It includes methods to applyBrake(int decrement), speedUp(int increment), and an overridden toString() method to print information about the bicycle.**
- 2. A MountainBike class extends Bicycle and adds an additional field seatHeight. It has a constructor to initialize the seatHeight along with gear and speed, a method setHeight(int newValue) to modify the seatHeight, and an overridden toString() method to include seatHeight information along with the inherited information.**
- 3. In the Test class, create an instance of MountainBike, initialize it with appropriate values, and print its information using the toString() method.**

**CODE:**

```
class Bicycle {  
    // the Bicycle class has two fields  
    public int gear;  
    public int speed;  
  
    // the Bicycle class has one constructor  
    public Bicycle(int gear, int speed) {  
        this.gear = gear;  
        this.speed = speed;  
    }  
  
    // the Bicycle class has three methods  
    public void applyBrake(int decrement) {  
        speed -= decrement;  
    }  
}
```

```
public void speedUp(int increment) {  
    speed += increment;  
}
```

// toString() method to print info of Bicycle

@Override

```
public String toString() {  
    return ("No of gears are " + gear + "\n"  
        + "speed of bicycle is " + speed);  
}  
}
```

// derived class

```
class MountainBike extends Bicycle {
```

// the MountainBike subclass adds one more field

```
public int seatHeight;
```

// the MountainBike subclass has one constructor

```
public MountainBike(int gear, int speed, int startHeight) {  
    // invoking base-class(Bicycle) constructor  
    super(gear, speed);  
    seatHeight = startHeight;  
}
```

// the MountainBike subclass adds one more method

```
public void setHeight(int newValue) {
```

```
        seatHeight = newValue;
    }

    // overriding toString() method of Bicycle to print more info
    @Override
    public String toString() {
        return (super.toString() + "\nseat height is " + seatHeight);
    }
}

// driver class
public class Test {
    public static void main(String[] args) {
        MountainBike mb = new MountainBike(3, 100, 25);
        System.out.println(mb.toString());
    }
}
```

## OUTPUT:

```
J Test.java > Bicycle > Bicycle(int, int)
1  class Bicycle {
2      // the Bicycle class has two fields
3      public int gear;
4      public int speed;
5
6      // the Bicycle class has one constructor
7      public Bicycle(int gear, int speed) {
8          this.gear = gear;
9          this.speed = speed;
10     }
11
12     // the Bicycle class has three methods
13     public void applyBrake(int decrement) {
14         speed -= decrement;
15     }
16
17     public void speedUp(int increment) {
18         speed += increment;
19     }
20
21     // toString() method to print info of Bicycle
22     @Override
23     public String toString() {
24         return ("No of gears are " + gear + "\n"
25             + "speed of bicycle is " + speed);
26     }
27 }
28
29 // derived class
30 class MountainBike extends Bicycle {
31     // the MountainBike subclass adds one more field
32     public int seatHeight;
33 }
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> cd "c:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7" & java Test

```
f ($?) { java Test }
No of gears are 3
speed of bicycle is 100
seat height is 25
```

- PS C:\Users\Soumyojyoti Saha\OneDrive - vit.ac.in\Desktop\java sem 7> █

END

