JAVA PROGRAMMING ASSIGNMENT – 3

PART A – THEORY QUESTIONS

1) Explain single, multilevel, and hierarchical inheritance.

Ans: Using Inheritance we can **Reuse** all methods and data member in next class.

Types of Inheritance

1) **Single Inheritance**

One parent → one child

A child class inherits from only one parent class.

Example:
```
class Parent {
                  void show() {
                                 System.out.println("Parent class");
                                                    }
                       }
               class Child extends Parent {
      void display() {
            System.out.println("Child class");
      }
}
public class Test {
      public static void main(String[] args) {
```

```
                    Child c = new Child();

                    c.show();           // from Parent

                    c.display();    // from Child

                }
```

2)Multilevel Inheritance

Ans: Grandparent → Parent → Child

A class inherits from a class that is already inherited from another class.

Example:

```
class A {

        void a() {

          System.out.println("Class A");

        }

}

class B extends A {

        void b() {

                System.out.println("Class B");

        }

}

class C extends B {

        void c() {

                System.out.println("Class C");

        }

}

public class Test {

        public static void main(String[] args) {
```

```
C obj = new C();

obj.a(); // from A

obj.b(); // from B

obj.c(); // from C

}

}
```

3)Hierarchical Inheritance

Ans: One parent → multiple children

Multiple child classes inherit from the same parent class.

Example:
```
class Parent {

void show() {

System.out.println("Parent class");

}

}

class Child1 extends Parent {

    void c1() {

        System.out.println("Child1 class");

    }

}

    void c2() {

        System.out.println("Child2 class");

    }

}

public class Test {

    public static void main(String[] args) {
```

```
                    Child1 obj1 = new Child1();

                    obj1.show();

                    obj1.c1();

                    Child2 obj2 = new Child2();

                    obj2.show();

                    obj2.c2();

                }

            }
```

3)What is polymorphism?

Ans:Polymorphism is a many form of method and constuctor.

Polymorphism is the ability of an object to take many forms, where the same method behaves differently based on the object type.

Polymorphism in Java is one of the core concepts in object-oriented programming (OOP) that allows objects to behave differently based on their specific class type. The word polymorphism means having many forms, and it comes from the Greek words poly (many) and morph (forms). This means one entity can take many forms.

Example: // Base class Person

```
                class Person {

           void role() {

                System.out.println("I am a person.");

                    }

           }

        class Father extends Person

           @Override

             void role() {

                System.out.println("I am a father.");
```

```
          }
     }
     public class Main {
          public static void main(String[] args)
               Person p = new Father();
                    p.role();
          }
     }
```

3)Difference between compile time and runtime polymorphism.

Ans: 1)Compile-time polymorphism is a type of polymorphism in which the method call during compilation.

-It is achieved through method overloading, where multiple methods have the same name but differ in the number, type, or order of parameters.

-The decision is made at compile time, it is also known as early binding.

-It does not require inheritance and is faster in execution.

2.Runtime polymorphism is a type of polymorphism in which the method call during program execution.

- It is achieved through method overriding, where a child class provides a specific implementation of a method already defined in its parent class.

-The decision is made by the JVM at runtime based on the object type, so it is also called late binding.

- Inheritance is required, and it provides greater flexibility.

4)What is method overloading?

Ans: Overloading (in Java) means using the same method name but with
different parameters in the same class.

Method Overloading means:

1)Same method name

2)Different parameter list (number, type, or order)

3)Happens at compile time

Example:

```
class Calculator {
    void add(int a, int b) {
    }

    void add(int a, int b, int c) {
        System.out.println(a + b + c);
    }

    void add(double a, double b) {
        System.out.println(a + b);
    }
}
```

Valid overloading

```
sum(int a)
sum(int a, int b)
sum(double a)
```

5)What is method overriding?

Ans: Child class provides its own implementation of a method that is already defined in the parent class.

In simple words:

same method name + same parameters, but new behavior in child class.

}

Key points (easy to remember)

Happens in inheritance

Method name must be same

Parameters must be same

Return type must be same (or covariant)

Occurs at runtime

Called runtime polymorphis

```
Example:class Parent {
    void show() {
        System.out.println("Parent show");
    }
}
class Child extends Parent {
    @Override
    void show() {
        System.out.println("Child show");
    }
}
public class Test {
    public static void main(String[] args) {
        Parent p = new Child();
        p.show();
    }
}
```

PART B – PROGRAMMING QUESTIONS

Q1)Write a program to find the sum of digits of a number.

```
Ans:-import jav.util.Scanner;
        public class Digits{
                public static void main(String[] args){


                System.out.println("Enter a Number :");
                Scanner sc=new Scanner(System.in);
                int a=sc.nextInt();
                int sum=0;


        while(a !=0){
                sum=sum + a%10;
                a=a/10;
        }
System.out.println("Sum of digits ="+sum);
        }
}
}
```

2)Write a program to check if a number is prime.

```
Ans:-import java.util.Scanner;
        public class Prime{
                public static void main(String[] args){
                    System.out.println("Enetr a number :");

                    Scanner sc=new Scanner(System.in);

                    int n=sc.nextInt();

                    int count=0;


                    for(int i=0;i<=n;i++){

                              if(n % i==0)

                                        count++;

                    }

          if(count==2)

                    System.out.println("Prime number");


          else

                    System.out.println("Not prime number");

          }

}
```

Q.3)Write a program to print all prime numbers between 1 and100.

Ans:-

```
  import java.util.Scanner;


class PrimeNumbers {

       public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter a number:");
        int n = sc.nextInt();

        for (int i = 2; i <= n; i++) {
            boolean isPrime = true;

            for (int j = 2; j <= i / 2; j++) {
                if (i % j == 0) {
                    isPrime = false;
                    break;
                }
            }

            if (isPrime) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Q4)Write a program to print multiplication table of a number.

```
Ans:-import java.util.Scanner;

        public class Multi{

                public static void main(String[] args){


                System.out.println("Enter a number :");

                Scanner sc=new Scanner(System.in);

                int n=sc.nextInt();


                for(int i=1;i<=10;i++){

                int num=n * i;

                        System.out.println("Multiplication is:"+num);

                }

        }

}
```

Q5)Write a program to count the number of digits in a number.

```
Ans:-import java.util.scanner;

        public class NumDigit{

                public static void main(string[] args){


                System.out.println("Enter a number :");

                Scanner sc=new Scanner(system.in);

                int num=sc.nextInt();

                int count=0;
```

```
                    int count = 0;


                          if (num == 0) {
                               count = 1;
                          } else {
                                   while (num != 0) {
                                        count++;
                                      num = num / 10;
                                    }
                   }


                 System.out.println("Number of digits = " + count);
          }
}
```