PART A – THEORY QUESTIONS

Q1)Difference between default, parameterized, and copy
constructors.

Ans:-1.Default constructor

-A default constructor is a constructor with no parameters.

-If you don’t define any constructor, Java automatically provides a default constructor.

-Initializes instance variables with default values (0, null, false).

-Used to create an object without passing values.

-Ex:-class Student {

```
   int id;
   String name;
   Student() {
      System.out.println("Default constructor called");
   }
}
```

2.Parameterized constructor

-A parameterized constructor is a constructor that accepts parameters.

-Used to initialize objects with specific values.

-Java does not create a default constructor if a parameterized constructor exists.

-Improves code readability and control over object creation.

-Ex:-class Student {

```
   int id;
   String name;
   Student(int i, String n) {
      id = i;
      name = n;
   }
```

}

3.Copy constructor

-A copy constructor creates a new object by copying another object of the same class.

-Java does not provide a copy constructor by default.

-Must be user-defined.

-Used to create independent copies of objects.

-Useful when cloning objects safely.

Ex:-class Student {

```
   int id;
   String name;
   Student(Student s) {
      id = s.id;
      name = s.name;
   }
}
```

Q2)What is the use of the this keyword?

Ans:-The this keyword is used to refer to the current object of the class and to resolve ambiguity between instance and local variables.

-1.To differentiate instance variables from local variables

EX:-class Student {

```
   int id;
   Student(int id) {
      this.id = id;
   }
}
```

2.To invoke current class methods

Used to call a method of the same class

Ex:-class Demo {

```
   void show() {
      System.out.println("Show method");
   }
   void display() {
      this.show();
   }
}
```

3.To invoke current class constructor (Constructor Chaining)

Used to call one constructor from another.

Ex:-class Test {

```
   Test() {
      this(10);
      System.out.println("Default constructor");
   }
   Test(int x) {
      System.out.println("Parameterized constructor");
   }
}
```

Q3)What is the use of the super keyword?

Ans:-super is a reference keyword used to refer to the immediate parent (superclass) object.

-To access superclass variables

-Used when parent and child have variables with the same name.

Ex:-class Parent {

```
   int x = 10;
}
```

```
class Child extends Parent {
    int x = 20;

    void show() {
        System.out.println(super.x); // accesses Parent's x
    }
}
```

2.To call superclass methods

-Used when a child class overrides a method and wants to call the parent version.

Ex:-

```
class Parent {
    void display() {
        System.out.println("Parent method");
    }
}

class Child extends Parent {
    void display() {
        super.display();
        System.out.println("Child method");
    }
}
```

3.To call superclass constructor

-Used to invoke the parent class constructor.

Ex:-

```
class Parent {
    Parent() {
        System.out.println("Parent constructor");
    }
}
```

```
class Child extends Parent {
    Child() {
        super();
        System.out.println("Child constructor");
    }
}
```

Q4)What is a static keyword used for?

Ans:-Static members belong to the class, not objects.

-static methods cannot use this or super.

-Static variables are initialized before objects are created.

-main() method is static so JVM can call it without creating an object.

1.Static Variable (Class Variable)

-Shared by all objects of the class.

-Only one copy exists in memory.

EX:-

```
class Student {
    static String college = "ABC";
    int id;
}
```

2?.Static Method

-Can be called without creating an object.

-Can access only static data directly.

EX:-

```
class Test {
    static void show() {
        System.out.println("Static method");
    }
    public static void main(String[] args) {
```

```
        Test.show();
    }
}
```

3.Static Block

-Executes once when the class is loaded.

-Used to initialize static variables.

EX:-

```
class Demo {
    static {
        System.out.println("Static block executed");
    }
}
```

4?.Static Class (Nested Class)

-Only nested classes can be static.

-Used to logically group classes.

EX:-

```
class Outer {
    static class Inner {
        void msg() {
            System.out.println("Static inner class");
        }
    }
}
```

Q5)What are static blocks and static methods?

Ans:-1.Static blocks

-A static block is a block of code that executes only once, when the class is loaded into memory.

-Initialize static variables

-Runs before main()

-Executes only once

-Can have multiple static blocks (run top to bottom)

Example:-

```
class Demo {
    static int x;

    static {
        x = 10;
        System.out.println("Static block executed");
    }

    public static void main(String[] args) {
        System.out.println(x);
    }
}
```

2.Static Method

-A static method belongs to the class, not to objects.

-Perform class-level operations

-Create utility methods

-Called using class name

-Cannot access non-static members directly

-Cannot use this or super

Example:-

```
class Test {
    static void show() {
        System.out.println("Static method called");
    }
    public static void main(String[] args) {
        Test.show();
    }
}
```

PART B – PROGRAMMING QUESTIONS

Q1)Write a program to find sum of n natural numbers.

Ans:-

```
import java.util.Scanner;
    public class Sum{

        public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a number:");
        int n=sc.nextInt();
        int sum=0;

         for (int i = 1; i <= n; i++) {
         sum = sum + i;
           }

           System.out.println("Sum of " + n + " natural numbers is: " + sum);
        }
    }
```

Q2)Write a program to reverse a String.

Ans:-

```
import java.util.Scanner;
    public class Reverse{
        public static void main(String[] args){

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter a String :");
          String str = sc.nextLine();
```

```
String rev = "";

        for (int i = str.length() - 1; i >= 0; i--) {
                rev = rev + str.charAt(i);
        }

        System.out.println("Reversed string: " + rev);
    }
}
```

Q3)Write a program to check if a String is palindrome.

Ans:-

```
import java.util.Scanner;
public class Palindrome{

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

        System.out.println("Enetr a String :");
        String name=sc.nextLine();
        String rev = "";

        for (int i = str.length() - 1; i >= 0; i--) {
        rev = rev + str.charAt(i);
        }

        if (str.equals(rev)) {
        System.out.println("The string is a palindrome");
        } else {
```

```
            System.out.println("The string is not a palindrome");
                        }
            }
}
```

Q4)Write a program to count vowels and consonants in a String.

Ans:-

```
import java.util.Scanner;
            public class Count{

            public static void main(String[] args){
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter a string: ");
            String str = sc.nextLine();
            int vowels = 0, consonants = 0;
             str = str.toLowerCase();

       for (int i = 0; i < str.length(); i++) {
          char ch = str.charAt(i);

          if (ch >= 'a' && ch <= 'z') {
             if (ch == 'a' || ch == 'e' || ch == 'i' ||
                ch == 'o' || ch == 'u') {
                vowels++;
             } else {
                consonants++;
             }
          }
       }
```

Q5)Write a program to count words in a sentence.

Ans:-

```
import java.util.Scanner;
        public class WordCount {
                public static void main(String[] args) {

                    Scanner sc = new Scanner(System.in);
                    System.out.print("Enter a sentence: ");
                String str = sc.nextLine();

               String[] words = str.trim().split("\\s+");
                    int count = words.length;
                    System.out.println("Number of words: " + count);
             }
}
```