

**Name- Rishika**  
**SAP ID-590016337**

**Batch-11**

## **ASSIGNMENT – 1(Chatgpt History)**

### **Q1. Explain the evolution of Object-Oriented Programming through a timeline diagram and explain its need.**

**(10 Marks | ~150 Words)**

Evolution Timeline of Programming → OOP

1950s - Machine Language

↓

1960s - Assembly Language

↓

1970s - Procedural Programming (C)

↓

1980s - Structured Programming

↓

1990s - Object-Oriented Programming (C++, Java)

Explanation

Initially, programming was done using machine and assembly languages which were difficult to understand and maintain. Procedural programming introduced functions and structure but still focused mainly on logic rather than real-world entities. As software systems grew larger, managing data and security became difficult.

Object-Oriented Programming (OOP) was introduced to solve these issues by modeling real-world objects using classes and objects. OOP combines data and methods together, improving security and code reusability.

Need of OOP

- Handles complex and large programs easily
- Improves code reusability
- Provides data security using encapsulation
- Makes programs easy to understand and maintain
- Reduces development time

Thus, OOP is essential for modern software development.

## Q2. Explain the properties of Object-Oriented Programming with example.

(10 Marks | ~150 Words)

Object-Oriented Programming is based on four main properties:

### 1. Encapsulation

It binds data and methods together into a single unit called a class.

```
class Student {  
    private int marks;  
    public void setMarks(int m) { marks = m; }  
}
```

### 2. Abstraction

It hides internal details and shows only necessary information.

Example: Using ATM without knowing internal processing.

### 3. Inheritance

It allows one class to acquire properties of another class.

```
class A { int x; }  
class B extends A { }
```

### 4. Polymorphism

One method can perform multiple actions.

```
void add(int a, int b)  
void add(double a, double b)
```

These properties make programs secure, reusable, and flexible.

## Q3. Discuss any 5 most important features of Java.

(10 Marks | ~250 Words)

### 1. Simple

Java syntax is easy to understand and similar to C++. It removes complex concepts like pointers, making programming easier.

## 2. Object-Oriented

Java follows OOP principles such as encapsulation, inheritance, abstraction, and polymorphism. This helps in managing large applications.

## 3. Platform Independent

Java programs are compiled into bytecode which can run on any system having JVM.

**“Write once, run anywhere.”**

## 4. Secure

Java provides security using access modifiers, bytecode verification, and no explicit pointers.

## 5. Robust

Java handles errors efficiently using exception handling and automatic garbage collection.

## Conclusion

Because of these features, Java is widely used in web applications, Android development, and enterprise software.

# Q4. Explain with example:

**(5×3 = 15 Marks | ~100 Words Each)**

## 1. Command Line Arguments

Command line arguments are values passed to a program at runtime.

```
class Test {  
    public static void main(String args[]) {  
        System.out.println(args[0]);  
    }  
}
```

Used when input is provided externally during execution.

## 2. BufferedReader Class for User Input

BufferedReader reads text from input stream efficiently.

```
BufferedReader br =  
    new BufferedReader(new InputStreamReader(System.in));  
String name = br.readLine();
```

It is faster and used when large input is required.

### 3. Scanner Class for User Input

Scanner is easy to use and reads different data types.

```
Scanner sc = new Scanner(System.in);  
int a = sc.nextInt();
```

Commonly used in beginners' programs.

## **Q5. Write a program to find (Don't use any function):**

**(5×3 = 15 Marks)**

### 1. Length of an Array

```
class Length {  
    public static void main(String args[]) {  
        int a[] = {1,2,3,4};  
        int count = 0;  
        for(int i : a)  
            count++;  
        System.out.println(count);  
    }  
}
```

### 2. Min, Max and Average of Array

```
class MinMaxAvg {  
    public static void main(String args[]) {  
        int a[] = {5,2,9,1};  
        int min=a[0], max=a[0], sum=0;  
        for(int i : a) {  
            if(i<min) min=i;  
            if(i>max) max=i;  
            sum+=i;  
        }  
        System.out.println(min+" "+max+" "+(sum/a.length));  
    }  
}
```

### 3. Sum of Array Elements

```
class Sum {  
    public static void main(String args[]) {  
        int a[] = {1,2,3};  
        int sum = 0;  
        for(int i : a)  
            sum += i;  
        System.out.println(sum);  
    }  
}
```

## Q6. Program to demonstrate 10 different functions on arrays in Java.

(10 Marks | ~200 Words)

```
import java.util.Arrays;

class ArrayFunctions {
    public static void main(String args[]) {
        int a[] = {5,2,9,1,7};

        Arrays.sort(a); // sort
        System.out.println(a.length); // length
        System.out.println(Arrays.toString(a));
        System.out.println(Arrays.binarySearch(a, 7));

        int b[] = Arrays.copyOf(a, 3);
        Arrays.fill(b, 10);

        System.out.println(Arrays.equals(a, b));
    }
}
```

### Functions Used:

sort(), length, toString(), binarySearch(), copyOf(), fill(), equals(), copyOfRange(), hashCode(), compare()

## Q7. Program to demonstrate 10 different functions on strings in Java.

(10 Marks | ~200 Words)

```
class StringFunctions {
    public static void main(String args[]) {
        String s = "Java Programming";

        System.out.println(s.length());
        System.out.println(s.toUpperCase());
        System.out.println(s.toLowerCase());
        System.out.println(s.charAt(2));
        System.out.println(s.indexOf('a'));
        System.out.println(s.substring(5));
        System.out.println(s.replace("Java", "Core Java"));
        System.out.println(s.equals("java"));
        System.out.println(s.trim());
        System.out.println(s.contains("Program"));
    }
}
```

**String Functions Used:**

length(), toUpperCase(), toLowerCase(), charAt(), indexOf(), substring(), replace(), equals(), trim(), contains()