Name - Rishika
SAP ID - 590016337
Batch - 11

Assignment - 1 (OOPS)

Evolution of Object - Oriented Programming

1950s - Machine Language
↓
1960s - Assembly Language
↓
1970s - ~~Structured~~ Procedural Programming (C)
↓
1980s - Structured Programming
↓
1990s - Object Oriented Programming (C++, Java)

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

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

Need of OOP

1. improves code reusability
2. Reduce development time
3. Provides data security using encapsulation
4. Handles complex & large programs easily.

Q2. Properties of OOPs

1. Encapsulation → It binds data & methods together into a single unit called a class.

Code:

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

}
```

2. Abstraction → It hides internal details & shows only necessary information

eg → using ATM without knowing internal processing.

3. Inheritance → It allows one class to acquire properties of another class.

code :

```
class A { int x; }

class B extends A { }
```

4. Polymorphism → One method can perform multiple actions.

Code :

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

Q3. Features of Java

1. Object - Oriented

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

2. Simple

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

3. Secure

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

4. Platform independent

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

5. Robust

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

Q4. 1. Command-line Arguments

These are values passed to a program at runtime.

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

→ used when input is provided externally during execution.

2. Buffered Reader Class for User Input

· Reads text from input stream efficiency

code:-
```
BufferedReader  br =
  new BufferedReader (new InputStreamReader(System.i
                                                    )
  String name = br.readLine ();
```

→ It's faster & used when large input is required.

3. Scanner class for User Input

Scanner is easy to use & reads different data types

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

→ commonly used in beginner's programs.

5. Length of an array

Program :-

```
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);
  }
}
```

Min, Max & Average of Array

Program :-

```
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));
  }
}
```

Sum of array of elements

Program :-

```
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);
   }
}
```

06. import java.util.Arrays;

```
class Arrayfunctions {
  public static void main(string args[]){
      int a[] = { 5,2, 9, 1,7};
      Array.sort(a);
      System.out.println(a.length);
      system.out.println( Arrays.tostring (a));
      system.out.println( Array.binarysearch (a,7));

      int b[] = Arrays.copyof(a,3);
      Arrays.fill (b,10);
      system.out.println(Arrays.equals (a,b));
   }
}
```

Q7.

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

      System.out.print In( s.length());
      System.out.print In( s.toUpperCase());
      System.out.print In (s.toLowerCase());
      System.out.print In( s.charAt (2));
      System.out.print In ( s.indexOf ('a'));
      System.out.print In ( s.substring (5));
      System.out.print In ( s.replace ("Java", "core Java"));
      System.out.print In ( s.equals ("Java"));
      System.out.print In (s.trim());
      System.out.print In (s.contains ("program"));
     System.out print
   }
}
```