Course Name:	**Object Oriented Programming**	**Semester:**	**III**
Date of Performance:	**01 / 09 / 2025**	**Batch No:**	**Batch A1**
Faculty Name:	**Prof. Amrita Naiksatam**	**Roll No:**	**20**
Faculty Sign & Date:		**Grade/Marks:**	**___/15**

Experiment No: 5
Title: Use of Interface

Aim and Objective of the Experiment:

Learn the how to use Interfaces

COs to be achieved:

CO2: Understand concepts of Object Oriented Programming and basic characteristics of Java.

Tools used:

Visual Studio Code

Theory:

(About Interfaces)

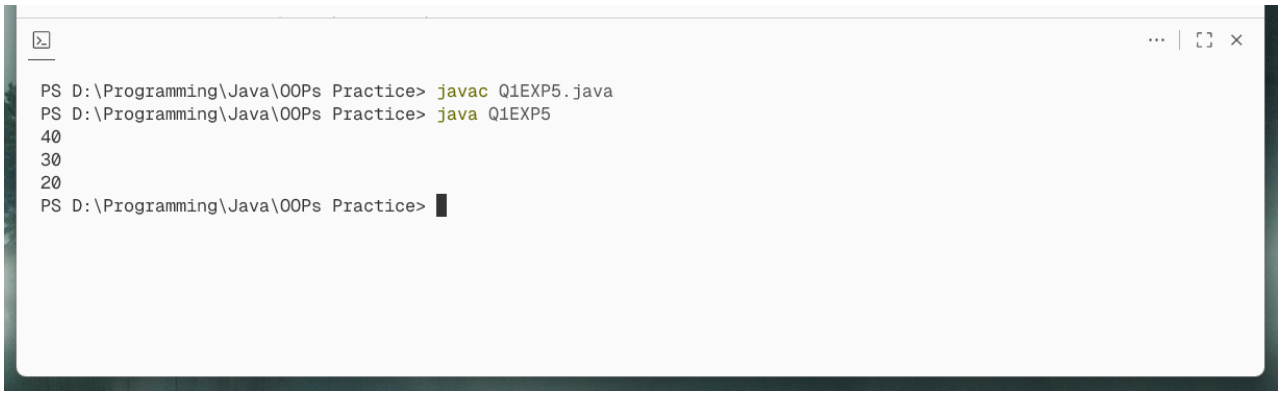
Code:

1. Write a program to design interface named *stack* with the following methods
    a. *push* and *pop* elements from the stack
    b. check for stack *empty* and *full* using method
2. Design class of *college* which uses two interfaces named *student* and *teacher*. Class *college* will use methods declared in the interface to *display* student and teacher personal information.

Q1.
Program:

```
interface Stack
{
    void push(int item);
    int pop();
    boolean isEmpty();
    boolean isFull();
}

class ArrayStack implements Stack
{
    int[] stack;
    int top;
    int capacity;

    public ArrayStack(int size)
    {
        capacity = size;
        stack = new int[capacity];
        top = -1;
    }

    public void push(int item)
    {
        if (!isFull())
        {
            stack[++top] = item;
        }
    }

    public int pop()
    {
        if (!isEmpty())
        {
            return stack[top--];
        }
        return -1;
    }

    public boolean isEmpty()
    {
        return top == -1;
    }

    public boolean isFull()
    {
        return top == capacity - 1;
    }
}

public class Q1EXP5
{
    public static void main(String[] args)
    {
        ArrayStack s = new ArrayStack(5);
        s.push(20);
        s.push(30);
        s.push(40);
        while (!s.isEmpty())
        {
            System.out.println(s.pop());
        }
    }
}
```

Output:



Q2.
Program:

```
interface Student
{
    void studentInfo();
}

interface Teacher
{
    void teacherInfo();
}

class College implements Student, Teacher
{
    public void studentInfo()
    {
        System.out.println("Student: Soham, ID: 101");
    }

    public void teacherInfo()
    {
        System.out.println("Teacher: Prof. Amrita Naiksatam, ID: T01");
    }
}

public class CollegeDemo
{
    public static void main(String[] args)
    {
        College c = new College();
        c.studentInfo();
        c.teacherInfo();
    }
}
```

Output:

Post Lab Subjective/Objective type Questions:

1. What is interface? How is it different from an abstract class?
   Ans:
   Interface: A blueprint of a class with only abstract methods (till Java 7). From Java 8+, it can have default and static methods. Variables are always public static final.
   Abstract Class: Can have both abstract and concrete methods, instance variables, and constructors.
   Key difference: A class can implement multiple interfaces, but can extend only one abstract class.

2. Explain how multiple inheritance can be implemented using interfaces.
   Ans:
   Java does not support multiple inheritance with classes (to avoid ambiguity).
   But a class can implement multiple interfaces, thus achieving multiple inheritance.

```
interface A { void methodA(); }
interface B { void methodB(); }

class C implements A, B {
    public void methodA() { System.out.println("From A"); }
    public void methodB() { System.out.println("From B"); }
}
```

3. What will be the function of the following code ? Will this code work ? If not, why ?

```
class DupValue
{
        public static void main(String[] args)
        {
                Vector v = new Vector();
                v.add("Delhi");
                v.add("Mumbai");
                v.add("Calcutta");
                v.add("Chennai");
                v.add("Delhi");

                Vector tempVector = new Vector();
                String tmpValue;

                for (int j = 0;j<= v.size(); j++)

                {
                        tmpValue = (String)v.elementAt(j);
                        if(tmpValue!=null) {
                                if( tmpVector.isEmpty() )
                                tmpVector.addElement(tmpValue);
                                if(tmpVector.indexOf(tmpValue)==-1){
                                tmpVector.addElement(tmpValue);}}
                }
                for(int j = 0; j < tmpVector.size(); j++) {
                        System.out.print(tmpVector.elementAt(j));
                }
        }
}
```

Ans:

The code is intended to remove duplicate values from the Vector and print only unique cities.
But it won't work correctly because:
Loop uses <= v.size() → causes ArrayIndexOutOfBoundsException.
Duplicate check uses == 1 instead of == -1.

After fixing these, the code will correctly print unique cities:
Delhi Mumbai Calcutta Chennai

Conclusion:

The experiment demonstrated the use of interfaces in Java to achieve abstraction and multiple inheritance. By implementing stack operations and combining student–teacher interfaces in a college class, we understood how interfaces define contracts for classes and promote flexibility in object-oriented design.

Signature of faculty in-charge with Date: