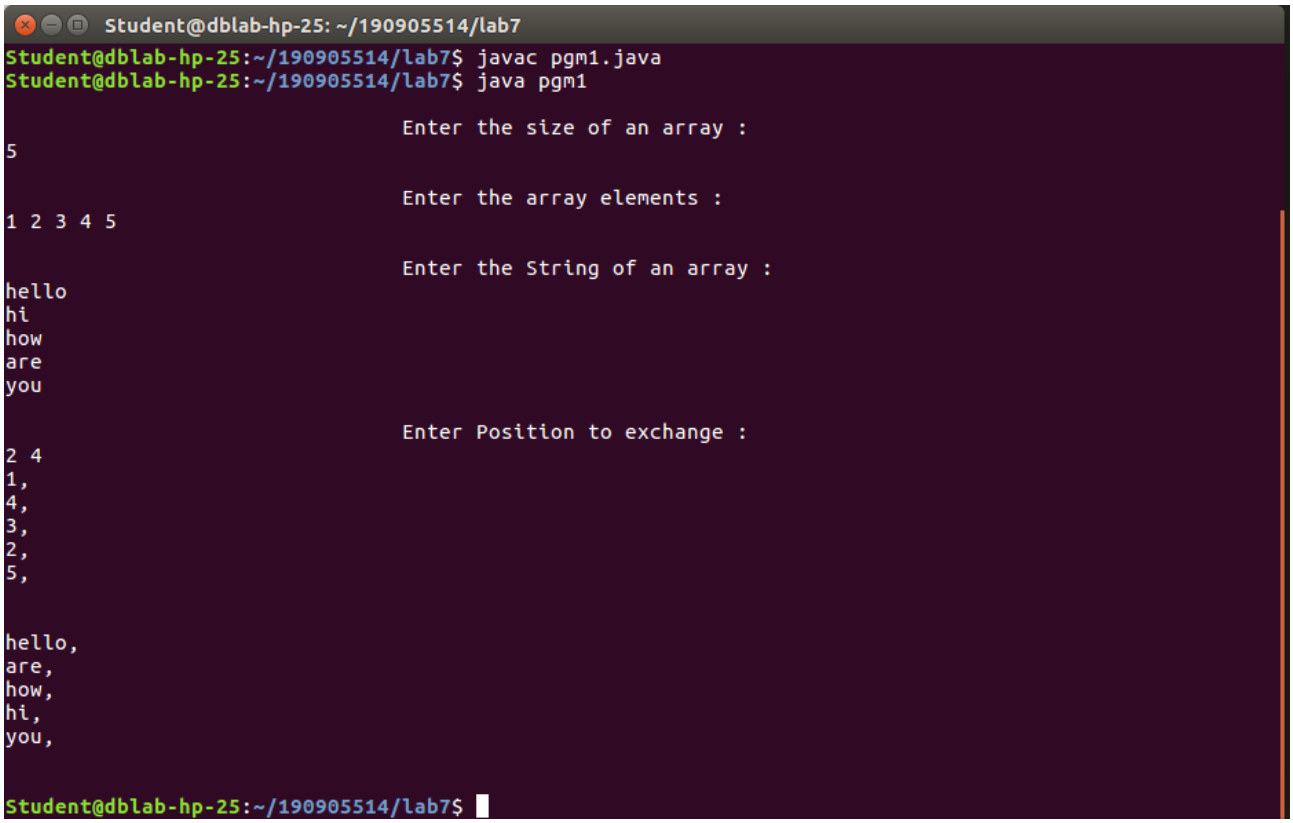
## SESSION 4 LAB 9 :

**1.**Write a generic method to exchange the positions of two different elements in an array.

**pgm1.java**

```
import java.util.*;

class exchange<T> {
   T[] arr;

   public exchange(T[] a) {
      this.arr = a;
   }

   public void exchange(int m, int n) {
      T temp;
      temp = arr[m - 1];
      arr[m - 1] = arr[n - 1];
      arr[n - 1] = temp;


   }

   public void display() {
      for (int i = 0; i < arr.length; i++) {
         System.out.println(arr[i] + ",");
      }
      System.out.println("\n");
   }
}

public class pgm1 {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("\n\t\t\t\tEnter the size of an array :");
      int n = scanner.nextInt();
      Integer arr[] = new Integer[n];
      System.out.println("\n\t\t\t\tEnter the array elements :");
      for (int i = 0; i < n; i++) {
         arr[i] = scanner.nextInt();

      }
      System.out.println("\n\t\t\t\tEnter the String of an array :");
      String[] str = new String[n];
      for (int i = 0; i < n; i++) {
         str[i] = scanner.next();
      }
      System.out.println("\n\t\t\t\tEnter Position to exchange :");
      int m = scanner.nextInt();
      int o = scanner.nextInt();
      exchange<Integer> cob = new exchange<Integer>(arr);
      cob.exchange(m, o);
      cob.display();
      exchange<String> sob = new exchange<String>(str);
      sob.exchange(m, o);
      sob.display();
   }
}
```

**OUTPUT :**




**2.**Define a simple generic stack class and show the use of the generic class for two different class types Student and Employee class objects.

**pgm2.java**

```
import java.util.Scanner;

class Stack<Type> {

   private Type arr[];
   private int tos;

   public Stack (int n) {
      tos = -1;
      arr = (Type []) new Object[n];
   }

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

   public void push (Type item) {
      if (tos == arr.length - 1) {
         System.out.println("\n\tSTACK OVERFLOW!");
         return;
      }
      arr[++tos] = item;
   }

   public Type pop () {
      if (tos == -1) {
         System.out.println("\n\tSTACK UNDERFLOW!");
```

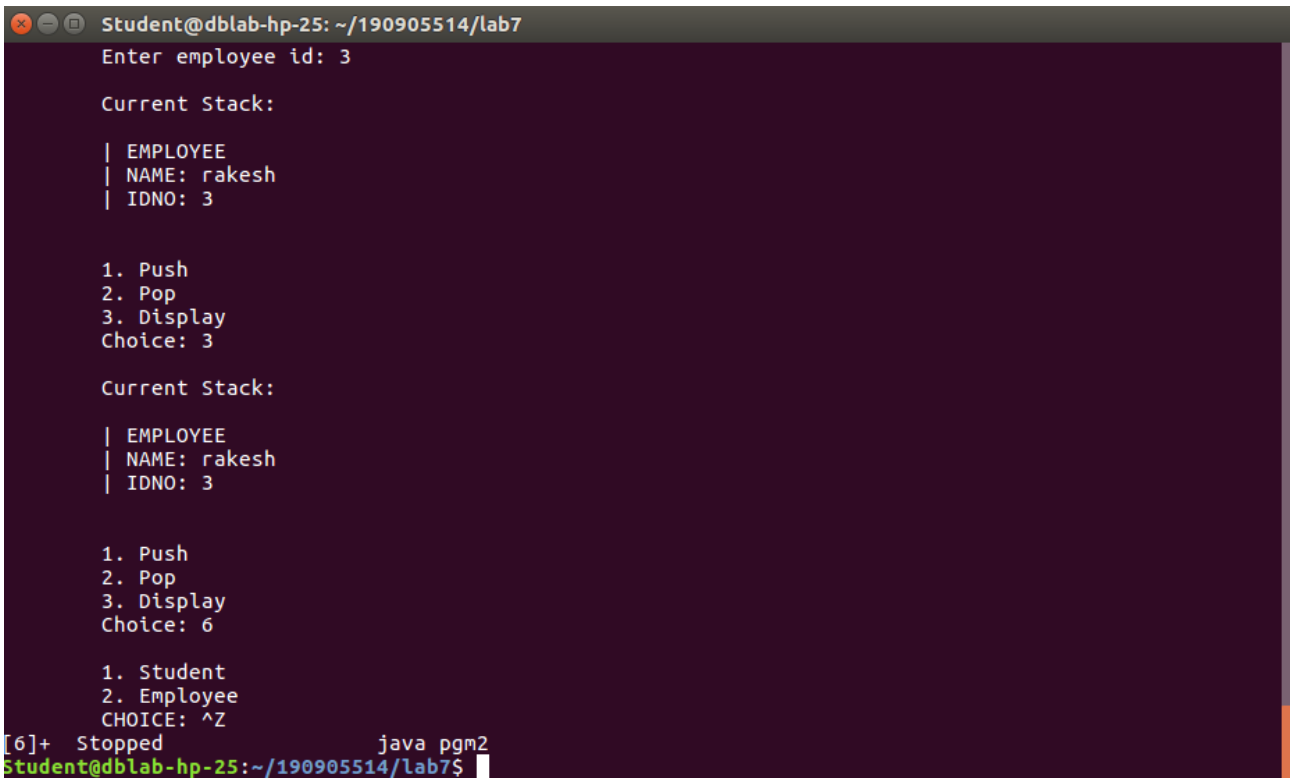
```
            return null;
        }
        return arr[tos--];
    }

    @Override
    public String toString () {
        if (tos == -1)
            return "STACK IS EMPTY!";
        String str = "";
        for (int i = 0; i <= tos; ++i)
            str += "\t" + arr[i];
        return str;
    }

}

class Student {

    private String name;
    private double cgpa;

    public void input () {
        Scanner sc = new Scanner(System.in);
        System.out.print("\n\tEnter student name: ");
        name = sc.nextLine();
        System.out.print("\tEnter student cgpa: ");
        cgpa = sc.nextDouble();
    }

    @Override
    public String toString () {
        return "\n\t| STUDENT\n\t| NAME: " + name + "\n\t| CGPA: " + cgpa + "\n";
    }

}

class Employee {

    private String name;
    private String idno;

    public void input () {
        Scanner sc = new Scanner(System.in);
        System.out.print("\n\tEnter employee name: ");
        name = sc.nextLine();
        System.out.print("\tEnter employee id: ");
        idno = sc.nextLine();
    }

    @Override
    public String toString () {
        return "\n\t| EMPLOYEE\n\t| NAME: " + name + "\n\t| IDNO: " + idno + "\n";
    }

}

    class pgm2{

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

        System.out.print("\n\tEnter the size of the stacks: ");
```

```
        int n = sc.nextInt();
        Stack <Student> sstack = new Stack<>(n);
        Stack <Employee> estack = new Stack<>(n);

        int choice;
        do {
            System.out.print("\n\t1. Student\n\t2. Employee\n\tCHOICE: ");
            choice = sc.nextInt();

            if (choice < 1 || choice > 2) {
                System.out.println("Invalid Choice!");
                System.exit(0);
            }

            int stch;
            do {
                System.out.print("\n\t1. Push\n\t2. Pop\n\t3. Display\n\tChoice: ");
                stch = sc.nextInt();

                if (stch < 1 || stch > 3)
                    break;

                if (stch == 1) {
                    if (choice == 1) {
                        Student stud = new Student();
                        stud.input();
                        sstack.push(stud);
                    }
                    else {
                        Employee empl = new Employee();
                        empl.input();
                        estack.push(empl);
                    }
                }
                else if (stch == 2) {
                    if (choice == 1) {
                        Student stud = sstack.pop();
                        if (stud != null)
                            System.out.print("\nPopped: " + stud);
                    }
                    else {
                        Employee empl = estack.pop();
                        if (empl != null)
                            System.out.print("\nPopped: " + empl);
                    }
                }
                if (choice == 1) {
                    if (!sstack.isEmpty())
                        System.out.println("\n\tCurrent Stack: \n" + sstack);
                }
                else {
                    if (!estack.isEmpty())
                        System.out.println("\n\tCurrent Stack: \n" + estack);
                }

            } while (stch >= 1 && stch <= 3);

        } while (choice == 1 || choice == 2);

    }

}
```

**OUTPUT:**

```
Student@dblab-hp-25: ~/190905514/lab7
exchange.class  pgm1.class        pgm2.class  pgm3.class  stack.class
Student@dblab-hp-25:~/190905514/lab7$ javac pgm2.java
Note: pgm2.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Student@dblab-hp-25:~/190905514/lab7$ java pgm2

        Enter the size of the stacks: 5

        1. Student
        2. Employee
        CHOICE: 1

        1. Push
        2. Pop
        3. Display
        Choice: 1

        Enter student name: tofik
        Enter student cgpa: 5.8

        Current Stack:

        | STUDENT
        | NAME: tofik
        | CGPA: 5.8


        1. Push
        2. Pop
        3. Display
        Choice: 3
```

```
Student@dblab-hp-25: ~/190905514/lab7
        Choice: 3

        Current Stack:

        | STUDENT
        | NAME: tofik
        | CGPA: 5.8


        1. Push
        2. Pop
        3. Display
        Choice: 2

Popped:
        | STUDENT
        | NAME: tofik
        | CGPA: 5.8

        1. Push
        2. Pop
        3. Display
        Choice: 5

        1. Student
        2. Employee
        CHOICE: 2

        1. Push
        2. Pop
        3. Display
```

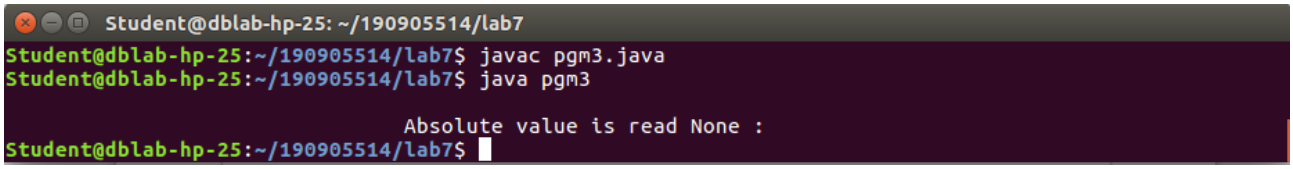



**3.**Write a program to demonstrate the use of wildcard arguments.

**pgm3.java**

```
class genericNum<T extends Number>{
    T num;
    genericNum(T n){
       this.num=n;
    }
    boolean absEqual(genericNum<T> ob){
       if(Math.abs(ob.num.doubleValue())==Math.abs(ob.num.doubleValue())){
          return true;
       }
       return false;
    }
}

public class pgm3 {
   public static void main(String[] args) {
      genericNum<Integer> gob=new genericNum<Integer>(6);
      genericNum<Double> gob1=new genericNum<Double>(9.0);
     if(gob.absEqual(gob)){
        System.out.println("\n\t\t\t\tAbsolute value is read None :");
     }else {
        System.out.println("\n\t\t\t\tAbsolute value different");
        System.out.println("\n\t\t\t\tTesting iob,and obj, and LAB");
        if(gob.absEqual(gob)){
           System.out.println("\n\t\t\t\tAbsolute value is :");
        }else {
           System.out.println("\n\t\t\t\tAbstract value");
        }
     }
   }
}
```

**OUTPUT :**


Student@dblab-hp-25: ~/190905514/lab7
Student@dblab-hp-25:~/190905514/lab7$ javac pgm3.java
Student@dblab-hp-25:~/190905514/lab7$ java pgm3
Absolute value is read None :
Student@dblab-hp-25:~/190905514/lab7$