Lab Exercise 7

Create the following:

a. Create an Interface for 'Stack' operations.
b. A class that implements the Stack interface and create a fixed length stack.
c. A class that implements the Stack interface and create a dynamic length stack.
d. A class that uses the above stacks through interface reference and does the stack operations that demonstrates the runtime binding

Theory: An interface definition is similar to a class definition except that it uses the interface keyword. All methods in an interface are abstract methods, that is, they are declared without the implementation part since they are to be implemented in the subclasses that use them. To declare an interface, use **interface** keyword. It is used to provide total abstraction. That means all the methods in interface are declared with empty body and are public and all fields are public, static and final by default. A class that implement interface must implement all the methods declared in the interface. To implement interface use **implements** keyword.

Program:

```
package stackinterface;
import java.io.*;
interface operations
  {
   void push();
   void pop();
   void display();
  }
//Class Of Fixed_Stack
class Fstack implements operations
    {
     private int top=-1;
     private int[] fstack=new int[5];
     BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
     public void push()
       {
          if(top==fstack.length-1)
             System.out.println("\nStack Overflow");
          else
```

```
            {
              try
                {
             System.out.print("Enter the Item:\t");
                    fstack[++top]=Integer.parseInt(br.readLine());
                }
              catch(Exception e)
                {
                    System.out.println(e);
                }
            }
        }

  public void pop()
        {
           if(top==-1)
          System.out.println("\nStack Underflow...!");
           else
          System.out.println("\nThe Deleted Item Is:\b"+fstack[top--]);
       }
     public void display()
       {
          if(top==-1)
              System.out.println("\nStack Is Empty");
          else
              {
          System.out.println("The Elements Of Stack are:");
          for(int i=top;i>=0;i--)
          System.out.println(fstack[i]);
              }
       }
    }
//Class Of Dynamic_Stack
class Dstack implements operations
   {
     private int top=-1;
     private int[] dstack;
```

```
 Dstack(int size)
    {
    dstack=new int[size];
    }
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

public void push()
  {
     if(top==dstack.length-1)
        System.out.println("\nStack Overflow");
     else
        {
          try
            {
       System.out.print("Enter the Item:\t");
        dstack[++top]=Integer.parseInt(br.readLine());
    }
          catch(Exception e)
     {
         System.out.println(e);
     }
        }
  }
 public void pop()
   {
      if(top==-1)
          System.out.println("\nStack Underflow...!");
      else
   System.out.println("\nThe Deleted Item Is:\b"+dstack[top--]);
    }

public void display()
    {
      if(top==-1)
  System.out.println("\nStack Is Empty");
      else
        {
```

```
        System.out.println("The Elements Of Stack are:");

        for(int i=top;i>=0;i--)
        System.out.println(dstack[i]);
            }
         }
    }
//Main Class
class Stackinterface
   {
      public static void main (String[] args) throws IOException
        {
          int ch;
          BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
          while(true)
            {
          System.out.println("\n1.Fixed_Stack(5)\n2.Dynamic_Stack\n3.Exit");
          System.out.print("\nEnter Your Choice:\t");
          ch=Integer.parseInt(br.readLine());
          switch(ch)
                {
                  case 1: operations obj=new Fstack();
                          while(true)
                            {
                       System.out.println("\n1.PUSH\n2.POP\n");
                       System.out.println("3.DISPLAY\n4.MAIN_MENU");
                       System.out.print("Enter Your Choice:\t");
                       ch=Integer.parseInt(br.readLine());
                       switch(ch)
                            {
                              case 1:obj.push();
                                        break;
                              case 2:obj.pop();
                                   break;
                    case 3:obj.display();
                                   break;
                                  case 4:break;
```

```
                    default:System.out.println("Invalid Choice");
                      }
                     if(ch==4)
                    break;
                   }
                 break;
          case 2: System.out.print("Enter The Size Of Array:\t");
               int size=Integer.parseInt(br.readLine());
               operations obj1=new Dstack(size);
               while(true)
                    {
               System.out.println("\n1.PUSH\n2.POP\n");
               System.out.println("\n3.DISPLAY\n4.MAIN_MENU");
                System.out.print("Enter Your Choice:\t");
               ch=Integer.parseInt(br.readLine());
               switch(ch)
                    {
                      case 1:obj1.push();
                               break;
                      case 2:obj1.pop();
                                    break;
                      case 3:obj1.display();
                                    break;
                           case 4:break;
                           default:System.out.println("Invalid Choice");
                        }
                     if(ch==4)
                        break;
                  }
                    break;
          case 3:   System.exit(0);
                     System.out.println("\n Invalid Choice");
      }
}    } }
```