Assignment 6

Part 1: Implementation of Fixed and Growable Stack using Interface.

Github Link: https://github.com/manan3044/Assignment_6

Code

1) Main.java

```
package Assignment6P1;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in) ;
        FixedStack fs = new FixedStack(5);
        GrowableStack gs = new GrowableStack(5);
        boolean run= true;
        int stack_choice,sc2, choice,pushele,popele;

        System.out.print("\n\nWhat type of stack do you
want:\n1)Fixed Stack\n2)Growable Stack\n3)Exit\n=");
        sc2 = sc.nextInt();
        do{

            stack_choice = 2;

            System.out.print("\nWhat operation do you want to
perform\n1.Push Element\n2.Pop Element\n3.Display Stack\n=");
            choice = sc.nextInt();
            switch(stack_choice)
            {
                case 1:
                    switch (choice)
                    {
                        case 1:
                            System.out.print("\nEnter element you
want to push: ");
                            pushele = sc.nextInt();
                            fs.push(pushele);
                            break;

                        case 2:
                            popele = fs.pop();
                            if (popele == -1)
                            {
                                continue;
```

```
                            }
                            else {
                                System.out.println(popele+ "
Element Popped Successfully");
                            }
                            break;


                        case 3:
                            fs.printStack();
                            break;
                    }
                    break;

                case 2:
                    switch (choice)
                    {
                        case 1:
                            System.out.print("\nEnter element you
want to push: ");
                            pushele = sc.nextInt();
                            gs.push(pushele);
                            break;

                        case 2:
                            popele = gs.pop();
                            if (popele == -1)
                            {
                                System.out.println("The Stack is
empty. Cannot pop elements.");
                            }
                            else {
                                System.out.println(popele+ "
Element Popped Successfully");
                            }
                            break;

                        case 3:
                            gs.printStack();
                            break;
                    }
                    break;

                case 3:
                    run = false;
                    break;

            }
        } while(run);
    }
}
```

2)Interface

```
package Assignment6P1;

public interface IntStack {
    void push(int num);
    int pop ();
    boolean isUnderflow();
    boolean isOverflow();
    void printStack();

}
```

3)GrowableStack.java

```
package Assignment6P1;
import java.util.ArrayList;

public class GrowableStack implements  IntStack{
    private ArrayList<Integer> stack;

    public GrowableStack(int capacity)
    {
        stack = new ArrayList<>(5);
    }

    @Override
    public void push(int num) {
            stack.add(num);
    }

    @Override
    public int pop() {
        if (isUnderflow())
        {
            return -1;
        }
        int poppedElement = stack.getLast();
        stack.removeLast();
        return poppedElement;
    }

    @Override
    public boolean isUnderflow() {
        return stack.isEmpty();
    }

    @Override
    public boolean isOverflow() {
        return false;
    }
```

```
    @Override
    public void printStack()
    {
        System.out.print("\nStack: ");
        for (Integer integer : stack) {
            System.out.print(integer + " ");
        }

        System.out.print("\nLength of Stack: "+stack.size());
    }
}
```

4)FixedStack.java

```
package Assignment6P1;

public class FixedStack implements IntStack{
    private final int[] stack;
    private int top;

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

    @Override
    public void push(int num) {
        if (isOverflow())
        {
            System.out.println("The Stack is full. Cannot push
more elements.");
        }
        else {
            stack[++top] = num;
        }
    }

    @Override
    public int pop() {
        if (isUnderflow())
        {
            System.out.println("The Stack is empty. Cannot pop
elements.");
            return -1;
        }
        int poppedElement = stack[top];
        stack[top] = 0;
        top--;
        return poppedElement;
    }

    @Override
```

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

    @Override
    public boolean isOverflow() {
        return stack.length == top + 1;
    }

    @Override
    public void printStack()
    {
        System.out.print("\nStack: ");
        for(int ele: stack)
        {
            System.out.print(ele+" ");
        }
        System.out.println("\nLength of Stack: "+stack.length);
    }
}
```

Output

1)Fixed Stack Output

```
What type of stack do you want:
1)Fixed Stack
2)Growable Stack
3)Exit
=1

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 42

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 90
```

```
Enter element you want to push: 90

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 23

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 69
The Stack is full. Cannot push more elements.
```

```
What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=2
90 Element Popped Successfully

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=2
42 Element Popped Successfully

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=2
The Stack is empty. Cannot pop elements.
```

```
What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=3

Stack: 12 34 345
Length of Stack: 3
```

2)Growable Stack Output

```
What type of stack do you want:
1)Fixed Stack
2)Growable Stack
3)Exit
=2

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 12

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 69
```

```
Enter element you want to push: 69

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1

Enter element you want to push: 34

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=1
```

```
What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=3

Stack: 12 69 34 34 34
Length of Stack: 5
```

```
What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=2
69 Element Popped Successfully

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=2
12 Element Popped Successfully

What operation do you want to perform(Stack Size=3)
1.Push Element
2.Pop Element
3.Display Stack
=2
The Stack is empty. Cannot pop elements.
```

Part 2: Program to implement multiple inheritance.

Code:

1)Main.java

```
package Assignment6P2;
import java.util.*;

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

        System.out.print("Enter Student Name:");
        String name = sc.nextLine();

        System.out.print("Enter PRN:");
        int prn = sc.nextInt();

        System.out.print("Enter Marks 1:");
        double mark1 = sc.nextDouble();

        System.out.print("Enter Mark 2:");
        double mark2 = sc.nextDouble();

        System.out.print("Enter Mark 3:");
        double mark3 = sc.nextDouble();

        result result = new result(name, prn, mark1, mark2,
mark3);
        result.displayExamDetails();

    }
}
```

2)examInterface.java

```
package Assignment6P2;

interface examInterface {
    void displayExamDetails();
    double Percent_cal(double mark1, double mark2, double mark3);
}
```

3)result.java

```
package Assignment6P2;

class result extends Student implements examInterface {
    public result(String name, int prn, double mark1, double
mark2, double mark3) {
        super(name, prn, mark1, mark2,mark3);
```

```
    }

    @Override
    public double Percent_cal(double mark1, double mark2, double
mark3) {
        return (mark1 + mark2 + mark3) / 3.0;
    }

    @Override
    public void displayExamDetails() {
        System.out.println("\nPRN: " + prn);
        System.out.println("Name: " + name);
        System.out.println("Mark 1: " + mark1);
        System.out.println("Mark 2: " + mark2);
        System.out.println("Mark 3: " + mark3);
        System.out.println("Percentage: " + Percent_cal(mark1,
mark2, mark3) + "%");
    }
}
```

4)Student.java

```
package Assignment6P2;

class Student {
    String name;
    int prn;
    double mark1, mark2, mark3;

    public Student(String name, int prn, double mark1, double
mark2, double mark3) {
        this.name = name;
        this.prn = prn;
        this.mark1 = mark1;
        this.mark2 = mark2;
        this.mark3 = mark3;
    }
}
```

Output

```
Enter Student Name:Manan
Enter PRN:62
Enter Marks 1:89
Enter Mark 2:67
Enter Mark 3:94

PRN: 62
Name: Manan
Mark 1: 89.0
Mark 2: 67.0
Mark 3: 94.0
Percentage: 83.33333333333333%
```