Assignment 12

Title - Template design pattern and exception handling.

Problem statement –

Write a program for ADT implementation of stack in JAVA Stack is an abstract class with template for push and pop. Handle stack full and empty conditions using exceptions.

Objective –

- To understand use of templates.
- To understand exception handling.
- Learning to use multiple templates.

Outcome -

To understand implementation of templates and exception handling.

Theory -

Templates are used to solve general program problems. There are total 23 design patterns. Template design pattern is a behavioral design pattern which

provides base methods for base algorithm called template method.

Exception Handling:

An exception is an error condition that changes normal exception execution of the program when something unexpected occurs program should defect the error.

Three types of exceptions:

- Checked.
- Unchecked.
- Error.

Algorithm –

- Template method:
    1. Define abstract class with template method.
    2. Common implementation of individual st4eps are defined in based class.
    3. Override and implement specific steps in subclass.
    4. Template method should be overridden.

- Exception class pseudocode:

```
protected int pop()
```

```
    {
        RuntimeException re= new
RuntimeException("Underflow!!");
        try
        {
            if(top==-1)
            {
                throw re;
            }
            else
            {
                System.out.print(" '" +stack[top--
] +"'");
                return 1;
            }
        }
        catch(RuntimeException ret)
        {
```

```
                System.out.println("Exception
Caught-: "+ret);
                return 0;
            }
        }


        protected int top()
        {

            RuntimeException re= new
RuntimeException("Underflow!!");
            try
            {
                if(top==-1)
                {
                    throw re;
                }
                else
```

```
                {
                        System.out.println("""
+stack[top] +""");
                        return 1;
                }
        }
        catch(RuntimeException ret)
        {
                System.out.println("Exception
Caught-: "+ret);
                return 0;
        }
    }


    protected void push()
    {
        Scanner obj= new Scanner(System.in);
        int x=obj.nextInt();
        try
```

```
            {
                RuntimeException re= new
RuntimeException("Overflow!!");
                if(top==6)
                {
                    throw re;
                }
                else
                {
                    stack[++top]=x;
                }
            }
            catch(RuntimeException ret)
            {
                System.out.println("Exception
Caught-: "+ret);
            }
        }
```

```
}



class Charstack extends stack
{
        int top=-1;
         char[] stck=new char[50];
        protected int pop()
        {
                RuntimeException re= new
RuntimeException("Underflow!!");
                try
                {
                        if(top==-1)
                        {
                                throw re;
                        }
                        else
```

```
                {
                        System.out.print("'"+ stck[top--]
+"'");
                        return 1;
                }
        }
        catch(RuntimeException ret)
        {
                System.out.println("Exception
Caught-: "+ret);
                return 0;
        }
    }


    protected int top()
    {
```

```
        RuntimeException re= new
RuntimeException("Underflow!!");
        try
        {
            if(top==-1)
            {
                throw re;
            }
            else
            {
                System.out.print("'"+ stck[top]
+"'");
                return 1;
            }
        }
        catch(RuntimeException ret)
        {
            System.out.println("Exception
Caught-: "+ret);
```

```
            return 0;
        }
    }

    protected void push()
    {
        Scanner obj= new Scanner(System.in);
        char x=obj.next().charAt(0);
        try
        {
            RuntimeException re= new
RuntimeException("Overflow!!");
            if(top==49)
            {
                throw re;
            }
            else
            {
```

```
                stck[++top]=x;
            }
        }
        catch(RuntimeException ret)
        {
            System.out.println("Exception
Caught-: "+ret);
        }
    }
```

Test cases-

1. Input:

```
        push(5)
        push(10)
        push(20)
        push(30)
        push(40)
```

   Output:

```
        5
        10   5
        20   10   5
```

30 20 10 5

Overflow

Expected O/P:

5

10 5

20 10 5

30 20 10 5

Overflow

Result:

Pass

Pass

Pass

Pass

Pass

2. Input:

pop()

pop()

pop()

pop()

pop()

Output:

30

20

10

5

Empty

Result:

Pass

Pass

Pass

Pass

Pass

Conclusion:

We have successfully implemented stack operations using template methods and checked for errors using exception handling.