Assignment - 12

Problem Statement -

Write a program for ADT implementation of stack in JAVA. Stack is an abstract base 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 -

Understanding implementation of templates and exception handling.

S/W & H/W Requirements -

64 bit processor, Fedora 20, Eclipse IDE for JAVA, Dell Optiflex 3020 MT.

Theory -

Templates are used to solve general program problems. There are total 23 design patterns. template design pattern is a behavioural design pattern which provides base methods for base algorithms called template methods.

Scanned with CamScanner

Exception Handling -

An exception is an error condition that changes normal execution of the program. When something unexpected occurs, program should reflect the error.

There are 3 types of exceptions -

- checked
- Unchecked
- Error

Pseudocode -

Template method :

1. Define abstract data ~~with~~ class with template method.
2. common implementation of individual steps are defined in base class.
3. override and implement specific steps in sub class.
4. Template method should be overridden.

Exception class Pseudocode -

```
class IntStack extends Stack {
    int [ ] stack = new int [50];
    int tos = -1;
① protected int pop();
    {
    RuntimeException re = new RuntimeException
    ("Underflow !.!" );
    try {
        if (tos == -1)
            throw re;
```

Scanned with CamScanner

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

2.
```
protected int tof() {
    RuntimeException re = new RuntimeException
                          ("Underflow!");
    try {
        if (tof == -1)
            throw re;
        else {
            System.out.println(" :" + stack[tof] + " ");
            return 1;
        }
    }
    catch (RuntimeException ret) {
        System.out.println("Exception caught- " + ret)
        return 0;
    }
}
```

Scanned with CamScanner

```
3. protected void fpush() {
    Scanner obj = new Scanner (System.in);
    int x = obj.nextInt();
    try {
    RuntimeException re = new RuntimeException
                          ("Overflow!");
    if (tos == 6)
        throw re;
    else
        stack[++tos] = x
    }
    catch (RuntimeException ret)
        System.out.println("Exception caught-" + ret);
  }
}
```

- Test cases

Case	Expected O/P	Actual O/P
fpush(1)	1	Success
fpush(2)	2 1	Success
fpush(6)	Stack Overflow!	Success
fpop()	5	Success
fpop()	4	Success

Conclusion -

Thus, we have implemented stack operations using template methods and checked for errors using exception handling.

Scanned with CamScanner