SKILL-1

ID: 21000300221

NAME: KAKUMANU NAGA VENKATA SAI MOHANA

1. Expression Evaluation

Implement Interpreter Design pattern to solve postfix expression in Java.

Input:

7 3 - 2 1 + *

Output:

12

PROGRAM:

```
package skilling;
public interface Experssion {
public int interpert();
}
package skilling;
public class Add implements Experssion{
private final Experssion leftExpression;
 private final Experssion rightExpression;
 public Add(Experssion leftExpression,Experssion rightExpression ){
 this.leftExpression = leftExpression;
 this.rightExpression = rightExpression;
 }
 public int interpert() {
 return leftExpression.interpert() + rightExpression.interpert();
 }
package skilling;
public class Product implements Experssion {
 private final Experssion leftExpression;
 private final Experssion rightExpression;
 public Product(Experssion leftExpression,Experssion rightExpression ){
 this.leftExpression = leftExpression;
 this.rightExpression = rightExpression;
 }
 public int interpert() {
 return leftExpression.interpert() * rightExpression.interpert();
```

```
 }
package skilling;
public class Subtract implements Experssion{
 private final Experssion leftExpression;
 private final Experssion rightExpression;
public Subtract(Experssion leftExpression,Experssion
rightExpression ){
 this.leftExpression = leftExpression;
 this.rightExpression = rightExpression;
 }
 public int interpert() {
 return leftExpression.interpert() -
rightExpression.interpert();
 }
}
package skilling;
public class Number implements Experssion{
private final int n;
public Number(int n){
 this.n = n;
}
public int interpert() {
 return n;
}
}
package skilling;
public class ExperssionUtils {
public static boolean isOperator(String s) {
 if (s.equals("+") || s.equals("-") || s.equals("*"))
 return true;
 else
 return false;
 }
 public static Experssion getOperator(String s, Experssion left,
Experssion right) {
 switch (s) {
 case "+":
 return new Add(left, right);
 case "-":
 return new Subtract(left, right);
```

```
 case "*":
 return new Product(left, right);
 }
 return null;
 }
}
package skilling;
import java.util.Stack;
public class TestInterpreterPattern {
public static void main(String[] args) {
 String tokenString = "7 3 - 2 1 + *";
 Stack<Experssion> stack = new Stack<>();
 String[] tokenArray = tokenString.split(" ");
 for (String s : tokenArray) {
 if (ExperssionUtils.isOperator(s)) {
 Experssion rightExpression = stack.pop();
 Experssion leftExpression = stack.pop();
Experssion operator = ExperssionUtils.getOperator(s,
leftExpression,rightExpression);
 int result = operator.interpert();
 stack.push(new Number(result));
 } else {
 Experssion i = new Number(Integer.parseInt(s));
 stack.push(i);
 }
 }
 System.out.println("( "+tokenString+" ):
"+stack.pop().interpert());  }
```

2. Assume you are going to buy some items from the Departmental store. You want to calculate the Total cost of the items which you want to purchase from the store. Following items to be purchased:

- Parker pen
- Pilet pen
- One Notebook with more than 250 pages (Ex: Price is Rs. 50)
- One Notebook with less than 250 pages (If pages < 250, reduce the cost by Rs.5)

Find the Total cost by implementing Visitor pattern

PROGRAM:

```
package cost;
public interface Item {
public int accept(Visitor visitor);
}
package cost;
public interface Visitor {
int visit(Pen pen);
 int visit(Notebook notebook);
}
package cost;
public class Pen implements Item{
private int price;
 private String model;

 public Pen(int price, String model) {
 this.price = price;
 this.model = model;
 }

 public int getPrice() {
 return price;
 }

 public String getModel() {
 return model;
 }
 @Override
 public int accept(Visitor visitor) {
 return visitor.visit(this);
}
}
package cost;
public class VisitorImpl implements Visitor{
public int visit(Pen pen) {
 int price = pen.getPrice();
 System.out.println(pen.getModel() + " costs " + price);
 return price;
 }
public int visit(Notebook notebook) {
```

```
 int price = 0;
 if(notebook.getNumberOfPages() > 250) {
 price = notebook.getPrice()-5;
 } else {
 price = notebook.getPrice();
 }
 System.out.println("Notebook costs " + price);
 return price;
 }
}
package cost;
public class VisitorImpl implements Visitor{
public int visit(Pen pen) {
 int price = pen.getPrice();
 System.out.println(pen.getModel() + " costs " + price);
 return price;
 }
public int visit(Notebook notebook) {
 int price = 0;
 if(notebook.getNumberOfPages() > 250) {
 price = notebook.getPrice()-5;
 } else {
 price = notebook.getPrice();
 }
 System.out.println("Notebook costs " + price);
 return price;
 }
}
package cost;
public class StackAbuseJavaDesignPatterns {
public static void main(String[] args) {
 Item[] items = new Item[]{new Pen(10, "Parker"), new Pen(5,
"Pilot"), new Notebook(50, 150), new Notebook(75, 300)};
 int total = getTotalPrice(items);
 System.out.println("Total price of items: " + total);
 }
 private static int getTotalPrice(Item[] items) {
 Visitor visitor = new VisitorImpl();
int result = 0;
 for(Item item : items) {
```

```
 result = result + item.accept(visitor);
 }
 return result;
 }
```

3. Implement BubbleSort sorting algorithm using Strategy pattern

Input: 34,12,67,10,7

Output: 7,10,12,34,67

PROGRAM:

```
        package user;
import java.util.Scanner;
public class EndUser implements SortingStrategy
{
 public int[] sort( int[] inputArray )
 {
 int n = inputArray.length;
 for( int i = 0; i < n; i++ )
 {
 for( int j = 1; j < (n - i); j++ )
 {
 if( inputArray[j - 1] > inputArray[j] )
 {
 swap(j - 1, j, inputArray);
 }
 }
 }
 System.out.println("Array is sorted using Bubble Sort
Algorithm");
 return inputArray;
 }
 private void swap( int k, int l, int[] inputArray )
 {
 int temp = inputArray[k];
 inputArray[k] = inputArray[l];
 inputArray[l] = temp;
 }
}
```