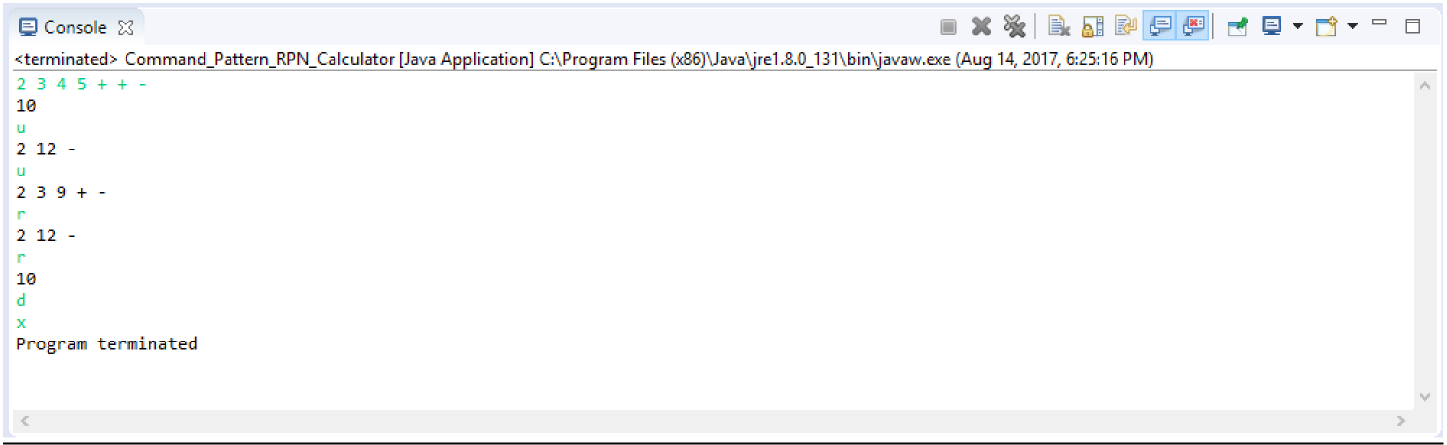
Niyati Shah
Id: 016128513
OOAD_Assignment 4 (Command Pattern)

Output:



```
<terminated> Command_Pattern_RPN_Calculator [Java Application] C:\Program Files (x86)\Java\jre1.8.0_131\bin\javaw.exe (Aug 14, 2017, 6:25:16 PM)
2 3 4 5 + + -
10
u
2 12 -
u
2 3 9 + -
r
2 12 -
r
10
d
x
Program terminated
```

Code:

// 1. Command_Pattern_RPN_Calculator.java

```java
import java.util.Scanner;

public class Command_Pattern_RPN_Calculator {

	static String command;
	private static Scanner sc;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Invoker user = new Invoker();
		sc = new Scanner(System.in);
		while(sc.hasNext()){
			command=sc.nextLine();
			if(command.equalsIgnoreCase("u"))
				user.undo();
			else if(command.equalsIgnoreCase("r"))
				user.redo();
			else if(command.equalsIgnoreCase("d"))
				user.clear();
			else if(command.equalsIgnoreCase("x"))
				break;
			else
				user.compute(command);
		}
		System.out.println("Program terminated");
		}
}
```

// 2. Invoker.java

```java
import java.util.ArrayList;

import java.util.Stack;
```

Niyati Shah
Id: 016128513
OOAD_Assignment 4 (Command Pattern)

```
public class Invoker {
	private Calculator _calculator = new Calculator();//Receiver
	int test =0;
	int current = 0;
	private ArrayList<Command> _commands = new ArrayList<Command>();
	private Stack<Integer> stack = new Stack<Integer>();
	String s="";

	public void undo() {
		Command	c = null;
		if(current > 0){
			c = _commands.get(--current);
		}
		System.out.println(c.UnExecute());
	}

	public void redo() {
		Command	c = null;
		// TODO Auto-generated method stub
		if (current < _commands.size() - 1){
		c = _commands.get(current++);
		s=c.ReExecute();
		}
		else
			compute(s);
	}

	public void clear() {
		// TODO Auto-generated method stub
		stack.clear();
	}
	public void compute(String command) {
		// TODO Auto-generated method stub
		createStack(command);
		System.out.println(test);
	}

	private static String Regex(String expr){
		return expr.replaceAll("[^\\^\\*\\+\\-\\d/\\s]", "");
	}

	public void createStack(String command){
		String input_string = Regex(command);
		Command c;
		for(String token: input_string.split("\\s")){
			Double t = null;
			try{
				t = Double.parseDouble(token);
			}
			catch(NumberFormatException e){}
				if(t!=null){
					stack.push(Integer.parseInt(token+""));
				}
				else if(token.equals("+")){
					int firstOperand = stack.pop();
					int SecondOperand = stack.pop();
					c=new CalculatorCommand(_calculator,'+',firstOperand,
SecondOperand);
					_commands.add(c);
					current++;
					test = c.Execute();
					stack.push(test);
				}
				else if(token.equals("-")){
```

```
                int firstOperand = stack.pop();
                int SecondOperand = stack.pop();
                c=new CalculatorCommand(_calculator,'-',firstOperand,
SecondOperand);
                _commands.add(c);
                current++;
                test = c.Execute();
                stack.push(test);
            }
            else if(token.equals("*")){
                int firstOperand = stack.pop();
                int SecondOperand = stack.pop();
                c=new CalculatorCommand(_calculator,'*',firstOperand,
SecondOperand);
                _commands.add(c);
                current++;
                test = c.Execute();
                stack.push(test);
            }
            else if(token.equals("/")){
                int firstOperand = stack.pop();
                int SecondOperand = stack.pop();
                c=new CalculatorCommand(_calculator,'/',firstOperand,
SecondOperand);
                _commands.add(c);
                current++;
                test = c.Execute();
                stack.push(test);
            }
            else if(token.equals("^")){
                int firstOperand = stack.pop();
                int SecondOperand = stack.pop();
                c=new CalculatorCommand(_calculator,'^',firstOperand,
SecondOperand);
```

```
                                    _commands.add(c);

                                    current++;

                                    test = c.Execute();

                                    stack.push(test);

                              }

                        }

            }


}
```

// 3. Command.java

```
public abstract class Command {
      public abstract int Execute();
    public abstract String UnExecute();
    public abstract String ReExecute();
}
```

// 4. CalculatorCommand.java

```
import java.util.Stack;

public class CalculatorCommand extends Command {

      private char _operator;
    private int _operand1,_operand2;
    private Calculator _calculator;
    private static Stack<Integer> undo_operand = new Stack<Integer>();
    private static Stack<Character> undo_operator = new Stack<Character>();
    private  String s="";
    private String s1="";
    public CalculatorCommand(Calculator calculator,  char operator, int operand1, int operand2){
        this._calculator = calculator;
        this._operator = operator;
        this._operand1 = operand1;
        this._operand2 = operand2;
    }

      @Override
      public int Execute() {
            // TODO Auto-generated method stub
        return _calculator.Operation(_operator, _operand1, _operand2);

      }
```

```
	@Override
	public String ReExecute() {
		String temp = _operand2 + " " + _operand1 + " " + _operator;
		s1 = s.replace(temp, _calculator.Operation(_operator, _operand1, _operand2)+"");
		System.out.println(s1);
		return s1;
		}

	@Override
	public String UnExecute() {
		// TODO Auto-generated method stub
		if(!undo_operand.isEmpty())
		undo_operand.pop();
		undo_operand.push(_operand2);
		undo_operand.push(_operand1);
		undo_operator.push(_operator);
		for(int i =0 ;i<undo_operand.size();i++)
			s=s+undo_operand.get(i)+" ";
		for(int i = undo_operator.size()-1;i>=0;i--)
			s=s+undo_operator.get(i)+" ";
		return s;
	}

}
```

// 5. Calculator.java

```
public class Calculator {
	 private int curr = 0;

	    public int Operation(char operator, int operand1, int operand2){

	      switch (operator){
	        case '+': curr= (operand1 + operand2); break;
	        case '-': curr= (operand1 - operand2); break;
	        case '*': curr= (operand1 * operand2); break;
	        case '/': curr= (operand1 / operand2); break;
	        case '^': curr= (int) Math.pow(operand1, operand2); break;
	      }
	      return curr;

	    }



}
```