Departamento de Ciencias de la Computacion
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
CC3001-2 Algoritmos y Estructuras de Datos

NOTACIÓN POLACA INVERSA

Informe Tarea 3

Integrante: Martin Araya
Profesor: Patricio Poblete
Auxiliares: Daniela Campos
Sven Reisenegger
Ayudantes: Bruno Rodríguez
Cristian Palma
Gabriel Chandía
Tomas Vallejos
Fecha de realización: 12 de noviembre de 2018
Fecha de entrega: 12 de noviembre de 2018
Santiago, Chile

1. Descripción del Problema

El problema a resolver es la cremación de un programa que sea capaz de leer lineas del input escritas en notación polaca inversa y mostrar el resultado de las mismas en la salida estándar.

La notación polaca inversa, o PRN por sus siglas en ingles debe su nombre a la analogía con la notación polaca, o notación de prefijo introducida en 1920 por el matemático polaco Jan Lukasiewicz, donde cada operador estaba antes de sus operandos. sin embargo, en la notación polaca inversa, esto se invierte, estando primero los operandos y posteriormente los operadores

2. Descripción de la solución

Para solucionar el problema planteado se propuso una solución ocupando la estructura de datos conocida como stack o pila, por su lógica de last in last out, que nos permite agregar los operandos a la estructura libremente, mientras que, en caso de que se intente agregar un operador, se sacan los elementos necesarios de la pila, se operan, y se vuelve a agregar a esta. El algoritmo se puede resumir en los siguientes pasos, luego de haber entregado los elementos de la entrada estándar como elementos de una lista de string:

- Si no se ha recorrido la lista
  - Si el elemento correspondiente NO es un operando, se agrega al stack
  - Si el elemento es un operando:
    - Si es un operando binario
      - Se desapilan los dos últimos elementos del stack, se operan, y se regresan al stack
    - Si es un operando unario
      - Se desapila el ultimo elemento del stack, se opera, y se regresa al stack
    - Si es el operando '='
      - Si es el ultimo valor en la lista
      - Se imprime el elemento que esta mas bajo en la pila usando 'println' y se acaba el programa
      - Se imprime el elemento que esta mas abajo en la pila, usando 'print'

Además, se incluyen consideraciones en los casos comunes de error

3. Resultados

Para los resultados, se dividieron estos en 3 partes, siendo estas:

1. Comprobación operaciones individuales
2. Múltiples operaciones
3. Errores

Para cada uno de estos se ideo una(s) secuencia(s) de string que nos permite comprobar su funcionalidad

3.1. Comprobación operaciones individuales

Para cada una de las operaciones que planteadas en el enunciado se ocupo un string que únicamente operara con la operación indicada, ocupando el numero mínimo requerido de operandos y haciendo una operación a la vez, estando los resultados en el código 1

Código 1: Resultados operaciones

```
=
0
1 =
1
1 1 + =
2
1 1 - =
0
1 2 * =
2
4 2 / =
2
1 _ =
-1
5 ! =
120
```

Como se muestra, cada uno de los códigos cumple con las operaciones realizadas, además, si no se hace ninguna operación y se pide el resultado, se devuelve un 0

3.2. Múltiples operaciones

En este caso se ocuparon las operaciones mostradas en el enunciado de la presenta tarea, debido a que estas ocupan a cabalidad las operaciones codificadas

Código 2: Resultados Múltiples operaciones

```
2 3 + 4 + = 7 _ + 2 * =
9 4
2 2 2 2 + + + =
8
2 = 2 _ 2 + =
2 0
6 ! 5 ! / =
6
0 ! =
1
```

3.3. Errores

En este caso se ocuparon los errores mas frecuentes que se podrían realizar en el uso del programa

Código 3: Errores

```
1 + = //Error al utilizar un operador binario con 1 operando
Exception in thread "main" java.lang.NumberFormatException: null
    at java.lang.Integer.parseInt(Integer.java:542)
    at java.lang.Integer.parseInt(Integer.java:615)
    at Main.desapilar(Main.java:27)
    at Main.calculo(Main.java:61)
    at Main.main(Main.java:103)
! = //Error al intentar utilizar un operador unario sin operandos
Exception in thread "main" java.lang.NumberFormatException: null
   at java.lang.Integer.parseInt(Integer.java:542)
   at java.lang.Integer.parseInt(Integer.java:615)
   at Main.desapilar(Main.java:27)
   at Main.calculo(Main.java:84)
   at Main.main(Main.java:103)
```

4. Código Fuente

Código 4: Código final

```
import java.util.Scanner;

public class Main {
    // Variables para usar en la implementación de un stack, dentro de la misma clase para el uso de afeed//
    private int[] arreglo;
    private int tope;
    private int MAX_ELEM = 100;
    // Constructor de un stack
    private Main(){
        arreglo = new int[MAX_ELEM];
        tope = -1;
    }
    // Metodo para apilar elementos en el stack, chequeando que este no este completo
    private void apilar(int x) {
        if (tope + 1 < MAX_ELEM) {
            this.tope++;
            arreglo[tope] = x;
        }
    }
    // Método para retornar el ultimo elemento agregado al stack, eliminandolo de este
    private int desapilar() {
        if (estaVacia()) {
```

```
23              int x = arreglo[tope];
24              this.tope--;
25              return x;
26          }
27          return Integer.parseInt(null); //Esta linea retorna error, sin embargo, solo se correra si se requiere un numero mayor de operandos que los que se han ingresado al stack, por ende es provocar el error a proposito//
28      }
29      // Método para determinar si el stack esta vacio
30      private boolean estaVacia() {
31          return tope != -1;
32      }
33      //Método que retorna el ultimo elemento del stack, sin eleminarlo del mismo
34      private int tope() {
35          if (estaVacia()){
36              return arreglo[tope];
37          }
38          return 0;
39      }
40      //Método para modificar el tope del stack a su posición inicial, reiniciando el stack
41      private void reiniciar (){
42          this.tope = -1;
43      }
44      //Método para calcular el factorial de un numero
45      private static int factorial (int x){
46          int out = 1;
47          for(int i=x; i>0 ; i--){
48              out*=i;
49          }
50          return out;
51      }
52      //Método para calcular las operaciones según la notación polaca inversa, reiniciando el stack creado cada vez//
53      private static void calculo(String[] op, Main pila){
54          pila.reiniciar();
55          for (int i = 0; i<op.length; i++) {
56              if (!op[i].equals("+") && !op[i].equals("-") && !op[i].equals("*") && !op[i].equals("/") && !op[i].equals("_") && !op[i].equals("!") && !op[i].equals("=")){
57                  pila.apilar(Integer.parseInt(op[i]));
58              }
59              else {
60                  if(op[i].equals("+")){
61                      int a = pila.desapilar();
62                      int b = pila.desapilar();
63                      pila.apilar(a+b);
64                  }
65                  if(op[i].equals("-")){
66                      int a = pila.desapilar();
67                      int b = pila.desapilar();
68                      pila.apilar(b-a);
69                  }
70                  if(op[i].equals("*")){
```

```
71                      int a = pila.desapilar();
72                      int b = pila.desapilar();
73                      pila.apilar(a*b);
74                  }
75                  if(op[i].equals("/")){
76                      int a = pila.desapilar();
77                      int b = pila.desapilar();
78                      pila.apilar(b/a);
79                  }
80                  if(op[i].equals("_")){
81                      int a = pila.desapilar();
82                      pila.apilar((-1)*a);
83                  }
84                  if(op[i].equals("!")){
85                      int a = pila.desapilar();
86                      pila.apilar(factorial(a));
87                  }
88                  if(op[i].equals("=")){
89                      if(i<op.length-1) {
90                          System.out.print(pila.tope() + " ");
91                      }
92                      else {
93                          System.out.println(pila.tope());
94                      }
95                  }
96              }
97          }
98      }
99      public static void main(String[] args){
100         Main pila = new Main();
101         Scanner sc = new Scanner(System.in);
102         while (sc.hasNextLine()){
103             String[] input = sc.nextLine().split(" ");
104             calculo(input,pila);
105         }
106     }
107 }
```