Tarea 3: Calculadora en notación polaca inversa

Autor: Vanessa Gaete
Correo: naan.u.285@gmail.com

Profesor: Jeremy Barbay
Auxiliares: Cristóbal Muñoz
Daniela Campos
Sven Reisenegger
Bernardo Subercaseaux
Curso: CC3001

Fecha de entrega: 11 de Noviembre de 2018
Santiago, Chile

Índice de Contenidos

1. Descripción del problema

El problema a resolver en esta tarea consiste en crear un programa en Java que calcule el resultado de un input ingresado en notación polacka inversa. El resultado debe ser entregado como output cada vez que haya un signo " = " en el input, además el programa debe poder recibir más de un input.

Las operaciones que debe reconocer y realizar el programa son las siguientes:
*Suma (" + ")
*Resta(" − ")
*Multiplicación(" ∗ ")
*División("/")
*Factorial("!")
*Negación("-"). Multiplica el entero por -1.
*Igual(" = ").Retorna el resultado de calculo.

Para realizar este código se supuso que el usuario ingresaría siempre un input válido, es decir, que cada carácter estuviese separado por un espacio del siguiente, que esté compuesto solo por números enteros y que nunca se realice una división por cero.

En este código no se utilizaron casos bases.

2. Descripción de la solución

El programa utiliza el TDA pila o "FILO". Así, el código consta de dos secciones, una clase donde se implementa este TDA llamada "$Pila2$" y otra donde se implementan los métodos que calcularán el input.

En cuanto a la primera sección, se implementa el método "$push$" para agregar elemntos a la pila, "$pull$" para extraer el último elemento ingresado y "$duplicarEspacio$" para agrandar la pila en caso de que esté llena y aún se deban agregar elementos.

Con respecto a la segunda sección, el método "$parsear$" recibe un arreglo de strings que corresponde al arreglo con los carácteres del input ingresado. Primero se crea una pila de enteros inicialmente vacía y una variable de string vacía "r" que almacenará los resultados. "$parsear$" funciona en base a condicionales anidados dentro de un "for" que recorre el arreglo desde principio a fin:

*Si el elemento del arreglo que se está evaluando es un operador binario, se sacan los últimos dos elementos de la pila con "$pull$" y se les aplica el operador, el primero que se saca será el segundo en la operación. Luego de aplicarla el resultado se guarda en la pila con "$push$". Un ejemplo de este caso es la resta, que fue implementada de la siguiente forma:

```
1            }else if ("-".contains(npi[i])) {
2
3                // se sacan los 2 últimos números.
4                int x = p.pull();
5                int y = p.pull();
6                // se reinserta el resultado.
7                p.push(y - x);
```

*Si el operador es unario se saca el último, se le aplica la operación y se guarda el resultado en la pila. Por ejemplo el código de la negación es el siguiente:

```
1            } else if ("_".contains(npi[i])) {
2                // se saca el último número.
3                int x = p.pull();
4                // se reinserta el resultado.
5                p.push(x * -1);
```

*Si el operador es un igual se saca el último elemento de la pila (pues es éste el que contiene el resultado de las operaciones que se han aplicado antes del símbolo " = ") y se guarda en el string "r" donde se irán sumando todos los resultados para printearlo cuando se termine de revisar el arreglo completo. Para que los resultados queden separados por espacios a los números guardados en el string se les suma un espacio al final. El código es el que sigue:

```
1            } else if ("=".contains(npi[i])) {
2                // se saca el último número.
3                int x = p.pull();
4                // se guarda en la pila.
5                p.push(x);
6                // se guarda tambien en el string vacío y se suma a los otros resultados
7                r= r  +x+ " ";
```

*Si el elemento del arreglo no es ninguno de los anteriores es porque es un número, así, se le aplica la función "$Integer.parseInt()$" para guardarlo en la pila como un número y no como string.

Después de haber recorrido todo el arreglo se le resta el último carácter a la variable "r" (que corresponde a un espacio en blanco) con "$substring$" y luego se printea.

Por otra parte, cabe notar que en el método "$main$" de la clase se utiliza un while para que el programa no se detenga al printear el primer resultado, y esté habilitado para recibir más operaciones.

3. Resultados

Se intentaron probar los casos más representativos y que abarcaran varios tipos de casos posibles:

CASO 1	RESULTADO
1 2 +	

CASO 2	RESULTADO
0 ! =	1

CASO 3	RESULTADO
4 3 ! =	6

CASO 4	RESULTADO
3 2 + = 1 + =	5 6

CASO 5	RESULTADO
6 _ 6 =	6
6 _ 6 + =	0

CASO 6	RESULTADO
3 1 / _ 2 - = 8 + = =	-5 3 3
8 _ 8 + 7 + 4 =	4

CASO 7	RESULTADO
1 3 8 * = _ 3 + 8 + =	24 -13
6 8 + = 1 * = 3 * 5 / 8 6 - 7 + 8 - 7 / 4 * = 7 + 7 * 2 ! 1 ! ! 4 2 8 8 - = _ 7 * = 7 + = =	14 14 0 0 0 7 7

4. Código Fuente

El siguiente es el código completo que se implementó para resolver el problema de la tarea.

```
import java.util.Scanner;



public class Main {
    public static void parsear(String[] npi) {
        Pila2 p = new Pila2();
    /*
    Cada vez que vemos un número lo apilamos, si vemos una operación se sacan los 2 últimos números si es binaria o
    solo uno si es unaria, y se reinserta en la pila un nuevo elemento correspondiente a los números operados.
    Se asume que la expresión ingresada es válida por lo que no se chequean errores.
    */

        String r="";
        for (int i = 0; i < npi.length; i++) {

            // chequeamos si es suma.
            if ("+".contains(npi[i])) {

                // se sacan los 2 últimos números.
                int x = p.pull();
                int y = p.pull();
                // se reinserta el resultado en la pila. notar que el segundo número que sale es el
                // primero en la operación.
                p.push(y + x);

                // chequeamos si es resta.
            }else if ("-".contains(npi[i])) {

                // se sacan los 2 últimos números.
                int x = p.pull();
                int y = p.pull();
                // se reinserta el resultado.
                p.push(y - x);

                // chequeamos si es multiplicacion.
            }else if ("*".contains(npi[i])) {

                // se sacan los 2 últimos números.
                int x = p.pull();
                int y = p.pull();
                // se reinserta el resultado.
                p.push(y * x);

            }else if ("/".contains(npi[i])) {

                // se sacan los 2 últimos números.
                int x = p.pull();
                int y = p.pull();
                // se reinserta el resultado.
                p.push(y / x);

            } else if ("!".contains(npi[i])) {
                // se saca el último número.
                int x = p.pull();
                // se reinserta el resultado.
                p.push(factorial(x));

            } else if ("_".contains(npi[i])) {
```

```
60                  // se saca el último número.
61                  int x = p.pull();
62                  // se reinserta el resultado.
63                  p.push(x * -1);
64
65              } else if ("=".contains(npi[i])) {
66                  // se saca el último número.
67                  int x = p.pull();
68                  // se guarda en la pila.
69                  p.push(x);
70                  // se guarda tambien en el string vacío y se suma a los otros resultados
71                  r= r  +x+ " ";
72              } else {
73                  // si no es operador es número y se guarda como entero en la pila.
74                  p.push(Integer.parseInt(npi[i]));
75              }
76          } System.out.println(r.substring(0, r.length() - 1));
77          /*
78      Cuando se termina de recorrer el arreglo se printea la variable r que almacena todos los
         resultados
79      */
80
81      }
82
83      public static int factorial(int a){
84          int m=1;
85          for (int j=1;j<=a;j++){
86              m=m*j;
87          }
88          return m;
89      }
90
91
92      public static void main(String[] args) {
93          // creamos el objeto scanner que nos permite leer el input del usuario.
94          Scanner in= new Scanner(System.in);
95          while (in.hasNextLine()) {
96              String linea = in.nextLine();
97              // creamos un arreglo de strings que guarde cada elemento entregado en el input
        separado por los espacios.
98              String[] arreglo = linea.split(" ");
99              parsear(arreglo);
100         }
101     }
102 }
103
104 // Estructura de Pila que sigue la regla FILO - first in last out
105 class Pila2 {
106
107     int[] pila;
108     int fin;
109
110     public Pila2() {
111         this.pila = new int[10];
112         this.fin = 0;
113     }
114
115     public void push(int x) {// agrega un elemento a la pila.
116         this.pila[this.fin] = x;
117         this.fin += 1;
118
119         // si la pila se llena debemos crear una nueva con más espacio.
120         if (this.fin == this.pila.length) {
121             this.duplicarEspacio();
122         }
123     }
```

```
124
125     private void duplicarEspacio() {
126         int[] nuevo = new int[this.pila.length * 2];
127
128         for (int i = 0; i < this.pila.length; i++) {
129             nuevo[i] = this.pila[i];
130         }
131
132         this.pila = nuevo;
133     }
134
135     public int pull() {// saca un elemento de la pila.
136         if (this.fin == 0) {
137             return 1;
138         }
139
140         this.fin -= 1;
141         return this.pila[this.fin];
142     }
143 }
```