UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

Tema: PILAS (STACK)

Instructor:

MC. Gerardo Gálvez Gámez

Abril de 2018

Objetivos

Al termino de la unidad el alumno podrá:

- Especificar el tipo abstracto de datos Pila.
- Conocer aplicaciones de las Pilas en la programación.
- Definir e implementar la clase Pila.
- Conocer las diferentes formas de escribir una expresión.
- Evaluar una expresión algebraica.

Temas

- **Pilas**
  - Definición.
  - Características.
  - Representación.
  - Utilización.
  - Estados de una Pila.
  - Operaciones.
  - Clases para la implementación de una pila con arreglos.
  - Ejercicios Prácticos.

Definición de Pila

- Es una colección de elementos homogéneos dispuestos en orden tal que se recuperan en orden inverso a como se introdujeron.
- Es una lista de elementos a la cual puede insertarse o eliminar elementos únicamente por un extremo.
- Una pila viene parametrizada por el tipo de elemento que alberga.


Pila de libros

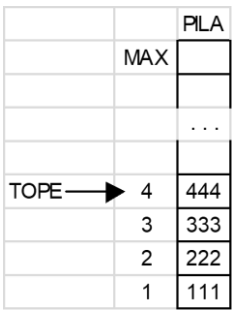

Pila de monedas

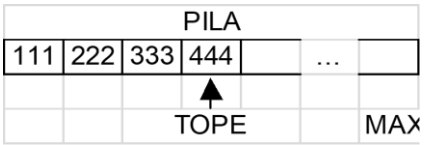
Características de la Pila

- La extracción e inserción de elementos en la Pila se realiza por la parte superior.
- El único elemento accesible es el último.
- Se le conoce como estructura LIFO (Last In, First Out).
- Cuenta con un tamaño máximo de la pila y una variable a la que se denominara TOPE, que será un apuntador al último elemento de la pila.

Representación de Pilas

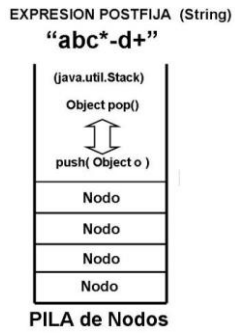
- Mediante el uso de:
  - Arreglos.
  - Listas Enlazadas.
  - Tipo Objeto
    - clase STACK , C#
    - clase Stack, Java





Utilización de Pilas

- Las pilas suelen emplearse en los siguientes contextos:
  - En compiladores, sistemas operativos y programas de aplicaciones, para la organización de la memoria.
  - Evaluación de expresiones aritméticas en notación postfija (notación polaca inversa).
  - Reconocedores sintácticos de lenguajes independientes del contexto.
  - Implementación de recursividad.
  - Ordenación.



Estados de una Pila

Pila Llena

TOPE →	MAX	999
		. . .
	4	444
	3	333
	2	222
	1	111

Con Algunos Elementos

	MAX	
		. . .
	4	
	3	
TOPE →	2	222
	1	111

Pila Vacía

	MAX	
		. . .
	4	
	3	
	2	
	1	
TOPE = 0	→	

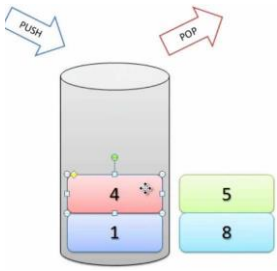
Tipos de Errores

Una vez dado un máximo de capacidad a la pila, no será posible insertar un número de elementos mayor. Si esto sucediera se produciría un error conocido como **desbordamiento (overflow).**

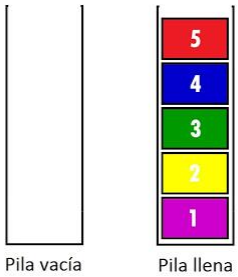
Otro error, es tratar de eliminar un elemento de una pila vacía. Este error se conoce como **Subdesbordamiento (underflow).**

Operaciones

- Insertar un elemento (push).
- Eliminar un elemento (pop).



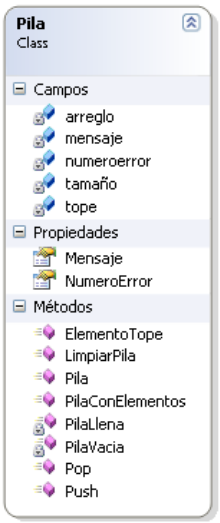
- Auxiliares:
  - Pila Vacía
  - (se presenta al intentar eliminar)
  - Pila Llena
  - (se presenta al tratar de insertar)



Clase para la implementación de pilas

Utilizando arreglos

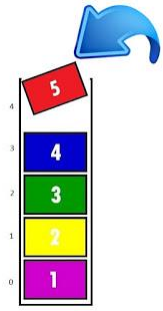
Clase Pila propuesta



Pila
-[] arreglo: string -tope: int -tamaño : int -mensaje: string -numeroerror: int
+Pila() +Pila(Tamaño)
+getTamaño: int +getMensaje: string +getNumeroError:int
+ Push(Dato:string):void + Pop():String + ElementoTope():string + PilaConElementos():bool + LimpiarPila():void - PilaVacia():bool - PilaLlena():bool

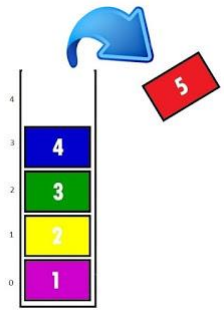
Push(Dato)

- Algoritmo
  - Si la pila esta llena
    - Indicar error de overflow
    - Terminar operación push
  - De lo contrario
    - Incrementar apuntador tope
    - Insertar el elemento en la pila



Pop()

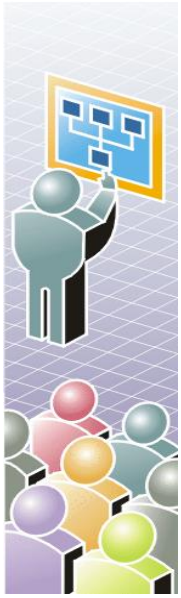
- Algoritmo
  - Si la pila esta Vacía
    - Indicar error de underflow
    - Terminar proceso Pop
  - De lo contrario
    - Quitar elemento en tope de la pila
    - Decrementar el apuntador tope
    - Retornar el elemento eliminado



ElementoTope()

- Algoritmo
  - Si PilaVacia
    - Indicar error de underflow
  - En caso contrario
    - Retornar elemento en tope de la pila

Actividad



- Definir un objeto de la clase Pila y realizar las operaciones de inserción y eliminación, mediante la utilización de un menú.

Ejercicios Prácticos

Resolución de problemas utilizando Pilas

TRATAMIENTO DE EXPRESIONES ARITMETICAS

Una expresión aritmética está formada por operandos y operadores, donde la evaluación de una expresión aritmética da lugar a un valor numérico

Dada una cierta expresión algebraica, existen básicamente tres formas diferentes de escribirla (transformación)

- A + B es notación **infija**
- AB+ es notación **postfija**
- +AB es notación **prefija**

La notación utilizada habitualmente es la **infija**.

Referencias:

Pág. 284 *Estructuras de datos en Java, Joyanes*

Pág. Estructuras de datos, Cairo

Reglas para transformación:

- Solo se permiten los siguientes operadores:
  - ***, /** (multiplicación y división)
  - **+, -** (suma y resta)
- Los operadores de más alta prioridad se ejecutan primero.
- Si hubiera en la expresión dos operadores igual prioridad, se le da preferencia al de la izquierda.
- Las Subexpresiones parentizadas tendrán más prioridad que cualquier operador.

Tabla de Prioridad de los Operadores

Símbolos	Nivel de Prioridad
( )	3 (Cero para evaluar)
* /	2
- +	1

Transformación de Expresión Infija a Postfija

Utilizando PILAS

Ejemplos

Transformar la expresión infija: **X + Z * W**, a Postfija:

PASO	EXPRESION
0	X + Z * W
1	X+ZW*
2	XZW*+

PASO	EI	Simbolo Analizado	PILA	EPOS
0	X + Z * W			
1	+ Z *W	X		X
2	Z * W	+	+	X
3	* W	Z	+	XZ
4	W	*	+ *	XZ
5		W	+ *	XZW
6			+	XZW *
7				XZW * +

Algoritmo (1/3)

```
INICIO
  REPETIR //(Mientras la Expresión sea diferente de la cadena vacía)
      //Tomar el símbolo más a la izquierda de la Expresión, recortando luego la expresión
    SI (símbolo es paréntesis izquierdo) ENTONCES
        poner símbolo en pila
    SI_NO
      SI (símbolo es paréntesis derecho) ENTONCES
        REPETIR
            Elemento= Pop ()//Quitar Símbolo de pila
            ExpresionPosfija = ExpresionPosfija + Elemento
        MIENTRAS (Pila[Tope] <> paréntesis izquierdo)
        Quitar el paréntesis izquierdo de PILA y no agregarlo a ExpresionPosfija
```


Algoritmo Cont. (2/3)

```
      SI_NO
          SI (símbolo es un operando) ENTONCES
              Agregar símbolo a ExpresionPosfija
          SI_NO  // Es un operador
            MIENTRAS (PILA tenga elementos y la prioridad del
                          operador sea menor o igual que la
                            prioridad del operador que esta en
la pila)
                Elemento= Pop (Quitar Símbolo de pila)
                ExpresionPosfija <- ExpresionPosfija + Elemento
             FIN_MIENTRAS
            Poner símbolo (Operador) en pila
         FIN_SI
      FIN_SI
    FIN_SI
MIENTRAS ( Expresión sea diferente de la cadena vacía)
```

Algoritmo Cont. (3/3)

MIENTRAS (PILA tenga elementos)

Elemento= Pop (Quitar Símbolo de pila)

ExpresionPosfija <- ExpresionPosfija + Elemento

FIN_MIENTRAS
IMPRIMIR ExpresionPosfija

FIN

Ejemplo

Expresión Infija: **(X + Z) * W / T * Y – V**, a Posfija

Transformar de Expresión Infija a Prefija

Utilizando PILAS

Expresión Prefija

- Al igual que en la notación postfija, aquí también se siguen las mismas reglas para la conversión, sólo que los operadores se colocan de lado izquierdo.

Convertir la expresión infija: **X + Z * W**, a Prefija:

0	X + Z * W
1	X+*ZW
2	.+X*ZW

X + Z * W

Paso	EI	Símbolo analizado	PILA	EPRE
1	X+Z*	W		W
2	X+Z	*	*	
3	X+	Z		WZ
4	X	+	+	WZ*
5	------	X		WZ*X
6				WZ*X+
7				

Invertir: +X*ZW

Algoritmo

INICIO

REPETIR (mientras Expresión sea diferente de la cadena vacía)

Tomar el símbolo mas a la derecha de la Expresión, recortando luego la expresión

SI (símbolo es paréntesis derecho) **ENTONCES**

poner símbolo en pila

SI_NO

SI (símbolo es paréntesis izquierdo) **ENTONCES**

REPETIR

Elemento= Pop (Quitar Símbolo de pila)

ExpresionPrefija =ExpresionPrefija + Simbolo

MIENTRAS (PILA[TOPE] <> paréntesis derecho)

Quitar el paréntesis derecho de PILA y no agregarlo a ExpresionPrefija

Algoritmo Cont.

SI_NO
SI (símbolo es un operando) **ENTONCES**
Agregar símbolo a ExpresionPrefija
SI_NO // { Es un operador}
MIENTRAS (PILA tenga elementos y la prioridad del operador sea Menor que la prioridad del operador que esta en la pila)

Elemento= Pop (Quitar Símbolo de pila)
ExpresionPrefija = ExpresionPrefija + Elemento

FIN_MIENTRAS
Poner símbolo (Operador) en pila
FIN_SI
FIN_SI
FIN_SI
MIENTRAS (Expresión sea diferente de la cadena vacía)

Algoritmo Cont.

MIENTRAS (PILA tenga elementos)
Elemento= Pop (Quitar Símbolo de pila)
ExpresionPrefija = ExpresionPrefija + Elemento
FIN_MIENTRAS
Invertir la ExpresionPrefija
IMPRIMIR ExpresionPrefija

FIN

Proyectos:

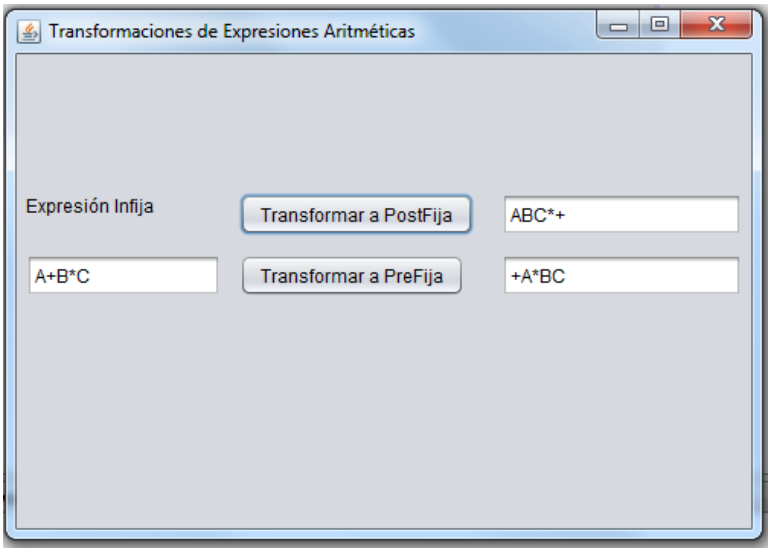
- Realizar **un sólo programa con interfaz grafica,** que pida al usuario una expresión algebraica compuesta por números enteros (de uno o más dígitos) y los operadores '**+**', '**-**', '*****' y '**/**', la pase a notación postfija o Prefija y la evalué ayudándose de una pila, mostrando por pantalla el paso intermedio (expresión en notación postfija) y el resultado final.

- Otros Indicados por el instructor. Pág. 107 Cairo

Propuesta Inicial para el Diseño de la Clase

Transformaciones
-ExpresionInfija:CADENA -Pila:tdaPila
+Transformaciones() +Transformaciones(pExpresion:CADENA)
+setExpresionInfija(pExpresion:CADENA)
-PrioridadOperador(pSimbolo:CARACTER):ENTERO -EvaluarSimboloAritmetico(pSimbolo:CARÁCTER):BOOLEAN +TransformacionPostFija():CADENA +TrasformacionPrefija():CADENA

Propuesta de Interfaz Gráfica



Bibliografía

ESTRUCTURAS DE DATOS
De: OSVALDO CAIRO

Editorial: MCGRAW-HILL INTERAMERICANA
ISBN: 9701059085
Edición: 3ª
Año: 2006
No. de páginas: 467
Idioma: ESPAÑOL
País: MEXICO

Preguntas?

FIN de Unidad