Pila

Concepto de Pila: Una pila (stack) es una colección ordenada de elementos a los cuales sólo se puede acceder por un único lugar o extremo de la pila. Los elementos se añaden o se quitan (borran) de la pila sólo por su parte superior (cima). Este es el caso de una pila de platos, una pila de libros, etc.

A recordar Una pila es una estructura de datos de entradas ordenadas que sólo se pueden introducir y eliminar por un extremo, llamado cima.

Cuando se dice que la pila está ordenada, lo que se quiere decir es que hay un elemento al que se puede acceder primero (el que está encima de la pila), otro elemento al que se puede acceder en segundo lugar (justo el elemento que está debajo de la cima), un tercero, etc. No se requiere que las entradas se puedan comparar utilizando el operador “menor que” (<) y pueden ser de cualquier tipo. Las entradas de la pila deben ser eliminadas en el orden inverso al que se situaron en la misma. Por ejemplo, se puede crear una pila de libros, situando primero un diccionario, encima de él una enciclopedia y encima de ambos una novela, de modo que la pila tendrá la novela en la parte superior.

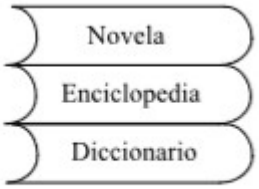


Figura 9.1 Pila de libros

Cuando se quitan los libros de la pila, primero debe quitarse la novela, luego la enciclopedia y por último el diccionario. Debido a su propiedad específica último en entrar, primero en salir se conoce a las pilas como estructuras de datos LIFO (last-in, first-out) Las operaciones usuales en la pila son Insertar y Quitar. La operación Insertar (push) añade un elemento en la cima de la pila, y la operación Quitar (pop) elimina o saca un elemento de la pila. La Figura 9.2 muestra una secuencia de operaciones Insertar y Quitar. El último elemento añadido a la pila es el primero que se quita de ella.

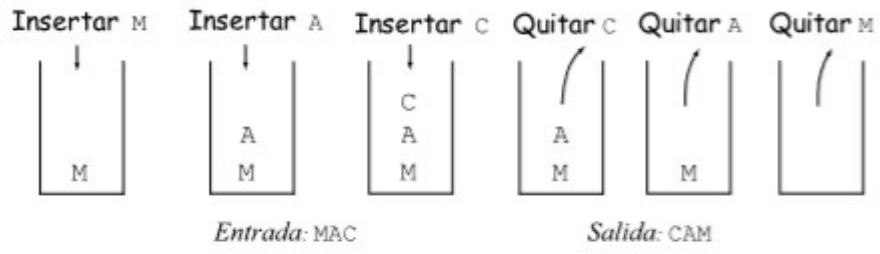


Figura 9.2 Poner y quitar elementos de la pila

La operación Insertar (push) sitúa un elemento dato en la cima de la pila, y Quitar (pop) elimina o quita el elemento de la pila.

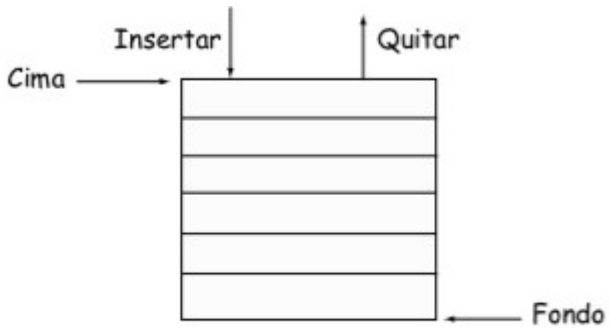


Figura 9.3 Operaciones básicas de una pila

La pila se puede implementar guardando los elementos en un array, en cuyo caso su dimensión o longitud es fija. También se puede utilizar un Vector para almacenar los elementos. Otra forma de implementación consiste en construir una lista enlazada, de modo que cada elemento de la pila forma un nodo de la lista. La lista crece o decrece según se añaden o se extraen, respectivamente, elementos de la pila; ésta es una representación dinámica, y no existe limitación en su tamaño excepto la memoria del computador. Una pila puede estar vacía (no tiene elementos) o llena (en la representación con un array —arreglo—, si se ha llegado al último elemento). Si un programa intenta sacar un elemento de una pila vacía, se producirá un error, una excepción, debido a que esa operación es imposible; esta situación se denomina desbordamiento negativo (underflow). Por el contrario, si un programa intenta poner un elemento en una pila llena, se produce un error, una excepción, de desbordamiento (overflow) o rebosamiento. Para evitar estas situaciones se diseñan métodos que comprueban si la pila está llena o vacía.

Especificaciones de una pila Las operaciones que sirven para definir una pila y poder manipular su contenido son las siguientes (no todas ellas se implementan al definir una pila):

Tipo de dato	Dato que se almacena en la pila.
Operaciones	
Crear Pila	Inicia.
Insertar (push)	Pone un dato en la pila.
Quitar (pop)	Retira (saca) un dato de la pila.
Pila vacía	Comprueba si la pila no tiene elementos.
Pila llena	Comprueba si la pila está llena de elementos.
Limpiar pila	Quita todos sus elementos y deja la pila vacía.
Cima Pila	Obtiene el elemento cima de la pila.
Tamaño de la pila	Número de elementos máximo que puede contener la pila.

Tipo de dato pila implementado con arrays: Los elementos que forman la pila se guardan en arrays (arreglos), en el contenedor Vector o bien formando una lista enlazada. La implementación con un array (arreglo) es estática ya que el arrray es de tamaño fijo. La clase PilaLineal, con esta representación, necesita un array y una variable numérica, cima, que apunte al último elemento colocado en la pila. Al utilizar un array es necesario tener en cuenta que el tamaño de la pila no puede exceder el número de elementos del array, y la condición pila llena será significativa para el diseño. El método usual de introducir elementos en una pila es definir el fondo de la pila en la posición 0 del array y sin ningún elemento en su interior, es decir, definir una pila vacía; a continuación, se van introduciendo elementos en la pila (en el array), de modo que el primer elemento añadido se introduce en una pila vacía y en la posición 0, el segundo elemento en la posición 1, el siguiente en la posición 2 y así sucesivamente. Con estas operaciones, el índice que apunta a la cima de la pila se va incrementando en 1 cada vez que se añade un nuevo elemento. Los algoritmos de introducir, “insertar” (push) y “quitar”, sacar, (pop) datos de la pila son:

Insertar (*push*)

1. Verificar si la pila no está llena.
2. Incrementar en 1 el puntero índice de la pila.
3. Almacenar elemento en la posición del puntero de la pila.

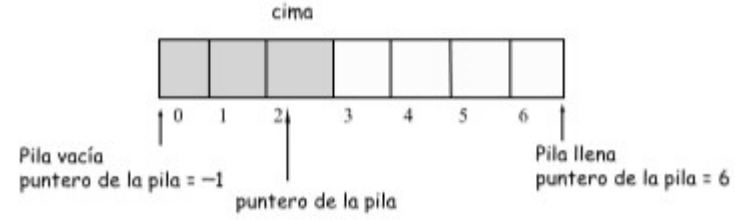
Quitar (*pop*)

1. Verificar si la pila no está vacía.
2. Leer el elemento de la posición del puntero de la pila.
3. Decrementar en 1 el puntero de la pila.

El rango de elementos que puede tener una pila varía de 0 a TAMPILA-1 en el supuesto de que el array se defina de tamaño TAMPILA elementos. De modo que, en una pila llena, el puntero (índice del array) de la pila tiene el valor TAMPILA-1, y en una pila vacía el puntero tendrá el valor -1 (el valor 0, teóricamente, es el índice del primer elemento).

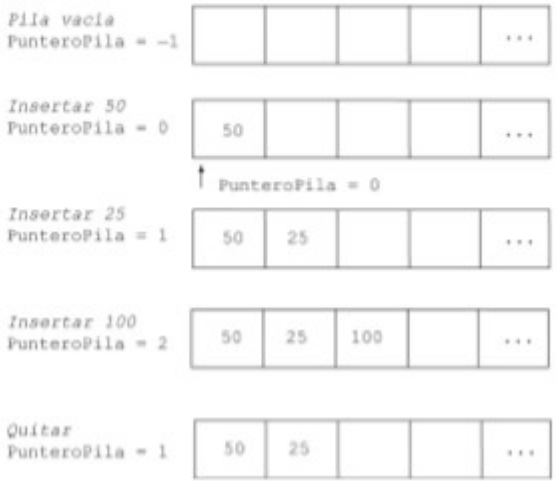
Ejemplo 9.1

Una pila de 7 elementos se puede representar gráficamente así:



Si se almacenan los datos A, B, C, ... en la pila se puede representar gráficamente de alguna de estas formas:

A continuación se muestra la imagen de una pila según diferentes operaciones realizadas en un posible programa.



Clase PilaLineal

La declaración de un tipo abstracto incluye la representación de los datos y la definición de las operaciones. En el TAD Pila los datos pueden ser de cualquier tipo, y las operaciones, las citadas anteriormente en el apartado

1. Datos de la pila (TipoDato es cualquier tipo de dato primitivo o tipo clase).

2.CrearPila inicializa una pila. Es como crear una pila sin elementos, por tanto, vacía.

3. Verificar que la pila no está llena antes de insertar o poner (“push”) un elemento en la pila; verificar que una pila no está vacía antes de quitar o sacar (“pop”) un elemento de la pila. Si estas precondiciones no se cumplen, se debe visualizar un mensaje de error y el programa debe terminar.

4. Pila Vacia devuelve verdadero si la pila está vacía y falso en caso contrario.

5.Pila Llena devuelve verdadero si la pila está llena y falso en caso contrario. Esta-s dos últimas operaciones se utilizan para verificar las precondiciones de insertar y quitar.

6.Limpiar Pila vacía la pila, dejándola sin elementos y disponible para otras tareas.
7.Cima Pila devuelve el valor situado en la cima de la pila, pero no se decrementa su puntero, ya que la pila queda intacta.

Definición

```
package TipoPila;

public class PilaLineal
{
   private static final int TAMPILA = 49;
   private int cima;
   private TipoDeDato []listaPila;

   public PilaLineal()
   {
      cima = -1; // condición de pila vacía
      listaPila = new TipoDeDato[TAMPILA];
   }
   // operaciones que modifican la pila
   public void insertar(TipoDeDato elemento){...}
   public TipoDeDato quitar(){...}
   public void limpiarPila(){...}
   // operación de acceso a la pila
   public  TipoDeDato cimaPila(){...}
   // verificación estado de la pila
   public boolean pilaVacia(){...}
   public boolean pilaLlena(){...}
}
```

Ejemplo 9.2 Escribir un programa que cree una pila de enteros y realice operaciones de añadir datos a la pila, quitar...

Se supone implementada la pila con el tipo primitivo int. El programa crea una pila de números enteros, inserta en la pila elementos leídos del teclado (hasta leer la clave -1) y a continuación extrae los elementos de la pila hasta que se vacía. El bloque de sentencias se encierra en un bloque try para tratar errores de desbordamiento de la pila.

```
import TipoPila.PilaLineal;
import java.io.*;

class EjemploPila
{
   public static void main(String [] a)
   {
      PilaLineal pila;
      int x;
      final int CLAVE = -1;
      BufferedReader entrada = new BufferedReader(
                           new InputStreamReader(System.in));

      System.out.println("Teclea los elemento (termina con -1).");
      try {
         pila = new PilaLineal(); // crea pila vacía
         do {
            x = Integer.parseInt(entrada.readLine());
            pila.insertar(x);
         }while (x != CLAVE);

         System.out.println("Elementos de la Pila: ");
         while (!pila.pilaVacia())
         {
            x = pila.quitar();
            System.out.print(x + " ");
         }
      }
      catch (Exception er)
      {
         System.err.println("Excepcion: " + er);
      }
   }
}
```

La declaración realizada está ligada al tipo de los elementos de la pila. Para alcanzar la máxima abstracción, se puede declarar la clase `PilaLineal` de tal forma que `TipoDeDato` sea el tipo referencia `Object`. Ésta es la clase base de todas las clases de Java y, por tanto, hay una conversión automática de cualquier referencia a `Object`. Como contrapartida, se necesita, en las operaciones que extraen elementos, convertir el tipo `Object` al tipo concreto de elemento. Además, cuando se necesite una pila de elementos de tipo primitivo, por ejemplo `int`, se tiene que crear una referencia que *envuelva* al elemento (`Integer, Double, ...`).

Ramírez Herbert Francisco No.Control: 1721217

Referencias Bibliográficas

Libro: Estructuras de datos en Java

Autores: Luis Joyanes Aguilar Ignacio Zahonero Martínez

Edificio Valrealty, 1. ª Planta

Basauri, 17 28023 Aravaca (Madrid)

Editor: José Luis García Técnico Editorial: Blanca Pecharromán

Compuesto en: Gesbiblo, S. L.

Diseño de cubierta: Gesbiblo, S. L.