Antonio José de Sucre
CORPORACIÓN UNIVERSITARIA

PILAS – ESTRUCTURAS DE DATOS

Ing. Jaider de la Rosa Bertel

Jaider.delarosa@cecar.edu.co

DESARROLLO

CONCEPTO DE PILAS:

Las pilas son estructuras de datos que almacenan un conjunto de valores y donde las operaciones de agregar y eliminar elementos se efectúan siempre por el tope o cima de la pila; por esta razón los primeros elementos en ser ingresados en la pila son los últimos en ser eliminados, y los últimos elementos en ser añadidos serán los primeros en ser eliminados. Debido a este comportamiento las pilas también se les conoce como listas LIFO (Last In, First Out), el último elemento en entrar, será primero en salir; o FILO (Fisrt In, Last Out), el primero en entrar, será el último en salir (Zahonero y Joyanes Aguilar, 1999).

Se pueden tomar como ejemplo una pila de libros, en la que un nuevo libro se pone encima de los otros y si usted va a quitar un libro de la pila procura retirar el que está más arriba, que corresponde al último que puso en la pila de libros.

La estructura de datos pila no tiene un medio propio para almacenar los datos o elementos que la componen, por lo que dependen de otras estructuras para guardar sus datos. El mecanismo utilizado para almacenar la información contenida en una pila se conoce como proceso de implantación de la pila. Para implementar la pila se utilizan arreglos o listas enlazadas simples como estructura auxiliar.

La pila le da un tratamiento especial a las operaciones de inserción y eliminación de datos en el arreglo o la lista según sea el caso, de manera que dichas operaciones se efectúan por un mismo extremo de la estructura (Tope) con la que se ha implantado la pila.

La característica principal de una pila se le llama Tope o Cima y hace referencia al extremo por el cual se efectúan las operaciones de inserción y eliminación de elementos. Además, el valor o elemento del tope representa al elemento que se ha de eliminar en la siguiente operación de eliminación. La segunda característica de una pila es el tamaño, que determina la cantidad de elementos máximo que la estructura puede contener en un momento dado.

De acuerdo a la cantidad actual de elementos contenidos en una pila, esta puede estar vacía si no contiene elementos, llena si el número de elementos almacenados es igual al tamaño máximo y un estado intermedio que no es ni vacía ni llena y que no es de mayor importancia. En cuanto a las operaciones comunes que una pila debe realizar, se encuentran:

- Agregar nuevos elementos (encima del tope).
- Eliminar elementos (quitar elemento del tope).
- Consultar el elemento del tope.

El estado de la pila (llena o vacía) determina la posibilidad de efectuar alguna de estas operaciones: agregar o quitar elementos. Así para agregar un nuevo elemento, la pila no debe estar llena y para eliminar o consultar el valor del tope la pila no puede estar vacía. Sin importar el mecanismo de

implantación de la pila, solo se da acceso al elemento del tope, de modo que por fuera de la estructura no hay acceso a los demás elementos de la pila.

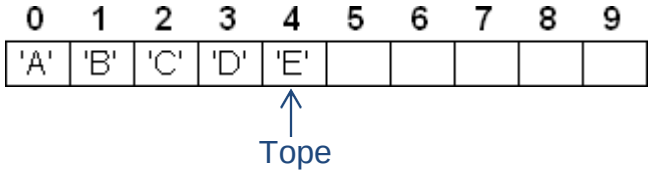
La implementación de las operaciones de una pila varía en función de la estructura escogida, ya sean arreglos o listas, pues el modo de representar el tope y el tamaño de la estructura es diferente en ambos casos, siendo distinta también la codificación de las operaciones de agregar y eliminar elementos.

IMPLEMENTACIÓN CON ARREGLOS:

Para la implementación con arreglos el tope se representa con un número entero que indica la posición en el arreglo del elemento que está en la parte superior de la pila, siendo esta posición el extremo por el cual se hacen las operaciones de agregación y eliminación (Allen Weiss, 2004).

El tipo de datos representado por una pila se define por el tipo de datos con el que se declara el arreglo que se usa como estructura de datos auxiliar, siendo el tamaño de la pila el mismo establecido para el arreglo. Para una pila se espera que el tamaño de la estructura no varié en tiempo de ejecución, aun cuando el arreglo sea dinámico y permita cambiar su tamaño.

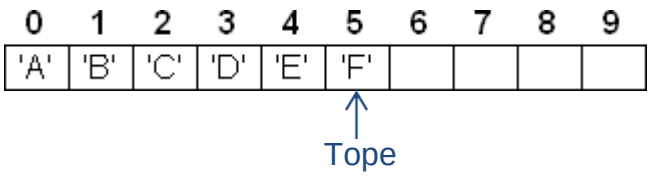
El tope para la pila puede ser el extremo izquierdo o derecho del arreglo. A continuación ilustramos de forma gráfica las operaciones con una pila, considerando que tamaño del arreglo es 10 y que el tope estará en el extremo derecho de la estructura.



Para agregar un nuevo elemento a la pila por ejemplo la letra ’F’ se han de efectuar los siguientes pasos:

- Verificar que la pila no esté llena (Tope<Tamaño).
- Aumentar Tope en uno, es decir, hacer Tope = Tope + 1.
- Guardar en el arreglo en la posición del Tope el dato, Arreglo[Tope] = ‘F’.

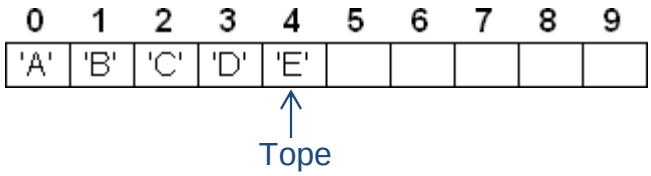
Con lo cual el arreglo de la pila queda de la siguiente manera:



Cuando se hace una eliminación siempre se elimina el elemento del tope, por lo que bastará con disminuir en uno el valor del tope y los pasos para realizar esta operación serán los siguientes:

- Verificar que la pila no esté vacía (Tope>=0).
- Disminuir el valor de tope en uno, es decir, hacer Tope = Tope – 1.

Después de realizar estos pasos el arreglo para la pila queda de la siguiente manera:



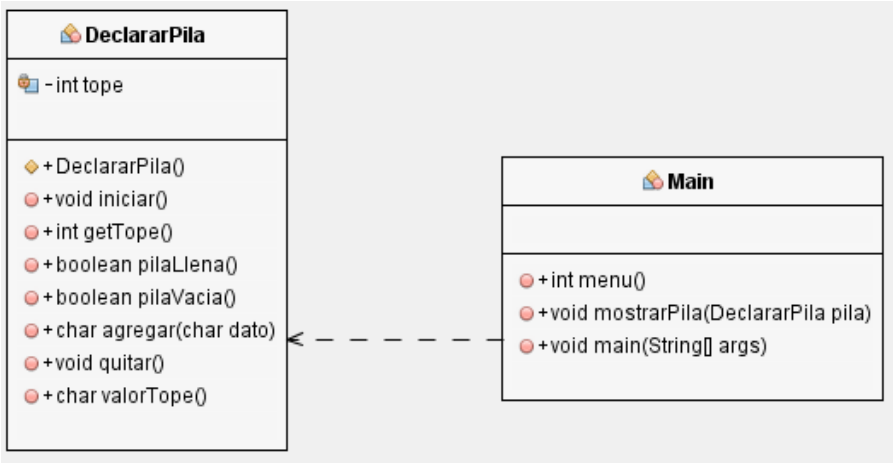
EJERCICIO PROPUESTO CON ARREGLOS:

A continuación se implementa un ejercicio que permite almacenar en una pila datos de tipo carácter, el ejercicio en su implementación permite al usuario las siguientes opciones:

- Agregar datos a la pila.
- Eliminar datos de la pila.
- Consultar la información del tope de la pila.
- Procesar la información almacenada en la pila.

Para la solución del ejercicio, se implementara la pila utilizando como estructura auxiliar un vector, posteriormente se realizara la implementación con listas. En la implementación se utilizaran dos ficheros, en donde se implementarán las operaciones de agregar y quitar elementos de la pila; así como procesar la información de la pila.

- **Diseño de clases UML de la solución**



- **Implementación de la clase DeclararPila en el fichero DeclararPila.java**

En el fichero **DeclararPila.java** se declara la clase DeclararPila, que tendrá los métodos que permiten realizar las operaciones de agregar, quitar y consultar el valor del tope de la pila.

```
public class DeclararPila {
//Constante para determinar el número de elementos de la estructura auxilia (el vector).
   static final int N = 5;
//Entre los atributos privados para implementar el ejercicio se encuentra el Tope y un arreglo para
//guardar los datos. Se define el atributo tope para determinar la posición del elemento del tope
//(arriba de la pila). El tope es el índice o posición de la cima dentro de la pila. Este atributo es de
//solo lectura (sin método set).
   private int tope;
//Se define el atributo o vector datosPila, este es de solo lectura (sin método set). datosPila es el
//arreglo para guardar los datos de la pila. Para efectos del ejemplo se declara de tipo char, pero
//puede ser de otro tipo dependiendo el ejercicio planteado, el vector datosPila será la estructura
//auxiliar para alojar los datos de la pila.
   private char[] datosPila = new char[N];
//Declaración del método constructor de la clase DeclararPila
   public  DeclararPila(){
      iniciar(); //También puede ser: tope = -1;
   }
//Para vaciar la pila basta con poner el tope en -1, pues los datos activos de la pila van de 0 hasta
//el Tope.
   public void iniciar(){
      tope = -1;
   }
//Declaración del método que retorna la posición del tope, a qué posición del vector apunta el tope.
   public int getTope(){
      return tope;
   }
//Métodos para determinar el estado de la pila (pila llena o pila vacía). Para esto se declaran dos
//funciones de tipo booleano. Si la pila está Llena se retorna el valor de true, en caso contrario se
//retorna el valor false. Lo mismo si la pila está vacía se retorna el valor de true, en caso contrario
//se retorna el valor false.
   public boolean pilaLlena(){
//La pila estará llena si el tope alcanza la posición máxima del arreglo (N-1).
      if (getTope() == N-1){
         return true;
      }else{
         return false;
      }
   }
   public boolean pilaVacia(){
//La pila está vacía si el tope es -1, entonces no hay elementos en la pila.
      if (getTope() == -1){
         return true;
      }else{
         return false;
      }
   }
//Implementación del método que agrega elementos a la pila, para esto se verificar que la pila no
//está llena, se aumenta el tope en uno y se guarda en el arreglo el dato, en la posición del tope.
   public void agregar(char dato){
```

```
        if (! pilaLlena()){ //Si la pila no está llena, entonces hay espacio para el nuevo elemento.
            tope = getTope() + 1; //Se reserva espacio para un nuevo elemento.
            datosPila[getTope()] = dato; //Se asigna el elemento al vector, donde está el tope.
        }
    }
//Implementación del método que permite eliminar el elemento del tope, para esto se verificar que
//la pila no está vacía y se disminuye el valor de tope en uno.
    public void quitar(){
        if(! pilaVacia()){
            tope = getTope() - 1;
        }
    }
//Implementación del método que permite obtener el valor del dato almacenado en el tope.
   public char valorTope(){
        return datosPila[getTope()]; //Se obtiene el valor almacenado en la cima de la pila.
    }
}
```

- **Implementación de la clase Main en el fichero Main.java**

En el fichero **Main.java** se declara la clase Main, que tendrá el método que permite visualizar los elementos de la pila y el menú de opciones para realizar las operaciones de agregar y quitar datos de la pila.

```
public class Main {

    public static int menu(){
        int opcion = 0;
        do{
            opcion = Integer.parseInt(JOptionPane.showInputDialog("===== IMPLEMENTACIÓN DE PILAS -
OPCIÓN DEL MENU ===== \n"
            +"1. Agregar Datos a la Pila \n"+"2. Eliminar Datos de la Pila \n"+"3. Mostrar Datos de la Pila \n"+
                    "4. Vaciar la Pila \n"+"5. Salir"+"\n \n Seleccione una opción del 1 al 5"));
        }while(opcion <= 0 || opcion > 5);
        return opcion;
    }

    public static void mostrarPila(DeclararPila pila){
//Se crea un objeto de la clase DeclararPila (temp), para crear una pila temporal en donde
//almacenar los datos y que estos no se pierdan mientras se visualizan.
        DeclararPila temp = new DeclararPila();
        String verDatosPila = "";
        while (! pila.pilaVacia()){ //Mientras la pila no este vacía se muestran sus datos.
            verDatosPila = verDatosPila+String.valueOf("---- "+pila.valorTope()+"\n");
            temp.agregar(pila.valorTope()); //Se agrega a una pila temporal el valor del tope de la pila.
            pila.quitar(); //Se elimina el tope de la pila actual, para pasar al siguiente elemento.
        }
//Se van visualizando en pantalla los datos almacenados en la pila.
        JOptionPane.showMessageDialog(null, "========= ELEMENTOS DE LA DE PILA
==========="+"\n"+verDatosPila+"\n");

//Posteriormente se copia nuevamente desde la pila temporal hasta la pila original.
        while (! temp.pilaVacia()){
//Se agrega a la pila original el valor del tope de la pila temporal.
```

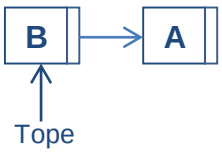
```
            pila.agregar(temp.valorTope());
            temp.quitar(); //Se elimina el tope de la pila temporal, para pasar al siguiente elemento.
        }
    }

    public static void main(String[] args) {
        DeclararPila pila = new DeclararPila();
        char dato;
        int opcion;
        do{
            opcion = menu();
            switch(opcion) {
            case 1:
                dato = JOptionPane.showInputDialog(null,"Digite el dato que quiere agregar a la
Pila").charAt(0);
                if (! pila.pilaLlena()){
                    pila.agregar(dato);
                    mostrarPila(pila);
                }else{
                    JOptionPane.showMessageDialog(null, "============ IMPLEMENTACIÓN DE PILAS
============"+"\n\n"+
                            "La pila está llena, No puede agregar más elementos \n\n");
                }
                break;
            case 2:
                if (! pila.pilaVacia()){
                    pila.quitar();
                    mostrarPila(pila);
                }else{
                    JOptionPane.showMessageDialog(null, "============ IMPLEMENTACIÓN DE PILAS
============"+"\n\n"+
                            "La pila está vacía, No pueden quitar datos \n\n");
                }
                break;
            case 3:
                mostrarPila(pila);
                break;
            case 4:
                pila.iniciar();
                mostrarPila(pila);
                JOptionPane.showMessageDialog(null, "============ IMPLEMENTACIÓN DE PILAS
============"+"\n\n"+
                            "La pila está vacía, No pueden quitar datos \n\n");
                break;
            case 5:
                break;
            }
        }while(opcion != 5);

    }
}
```

IMPLEMENTACIÓN CON LISTAS ENLAZADAS:

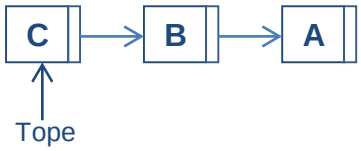
Siguiendo con el concepto de pilas, se mencionó que estas no contaban con una estructura de datos propia para almacenar la información; en este sentido su implementación se debe realizar utilizando una estructura de datos auxiliar, que puede ser un vector o una lista enlazada. En el ejercicio anterior se realizó la implementación de la pila utilizando un vector como estructura auxiliar, en esta guía se realizara la implementación usando listas enlazadas.

Para la implementación con listas el tope de la pila será la cabeza de la lista, esto teniendo en cuenta que el primer nodo de una lista es la referencia que conocemos para agregar información. Independientemente de la forma como se implemente la pila el concepto será el mismo; entonces tendremos que agregar y quitar la información siempre por el nodo de tope. También se deben tener en cuenta los estados de la pila (llena o vacía) para realizar las operaciones de agregar o quitar elementos.

Cuando se agregue el primer nodo de la pila este será el nodo del tope de la pila, para agregar un nuevo nodo se debe verificar el la pila no esté llena, en este caso se procede a agregar el nuevo nodo.



Si se quiere agregar un nuevo elemento a la pila, se verifica que la pila no esté llena como se mencionó anteriormente, luego se crea un nuevo nodo con su respectiva información y se le dice a su apuntador que apunte a donde está el tope; finalmente se pone a que tope apunte al nuevo nodo que se agregó.



EJERCICIO PROPUESTO CON LISTAS ENLAZADAS:

A continuación se implementa un ejercicio que permite almacenar en una pila datos de tipo cadena, el ejercicio en su implementación permite al usuario las siguientes opciones:

- Agregar datos a la pila.
- Eliminar datos de la pila.
- Consultar la información del tope de la pila.

- Procesar la información almacenada en la pila.

Para la solución del ejercicio, se implementara la pila utilizando como estructura auxiliar una lista enlazada. En la implementación se utilizaran tres ficheros, en donde se implementarán las operaciones de agregar y quitar elementos de la pila; así como procesar la información de la pila.

- **Diseño de clases UML de la solución**

- **Implementación de la clase Nodo en el fichero Nodo.java**

En el fichero **Nodo.java** se declara la clase Nodo, que tendrá en nodo de la lista con su respectiva información y enlace (atributos).

```
public class Nodo {
    private String dato;
    private Nodo siguiente;
    public Nodo(){
        dato = "";
        siguiente = null;
    }
    public void setDato(String inf){
        dato = inf;
```

```
    }
    public void setSiguiente(Nodo sig){
        siguiente = sig;
    }
    public String getDato(){
        return dato;
    }
    public Nodo getSiguiente(){
        return siguiente;
    }
}
```

- **Implementación de la clase DeclararPila en el fichero DeclararPila.java**

En el fichero **DeclararPila.java** se declara la clase DeclararPila, que tendrá los métodos que permiten realizar las operaciones de agregar, quitar y consultar el valor del tope de la pila, haciendo la implementación con una lista enlazada.

```
public class DeclararPila {
//Constante para determinar la cantidad de nodos que soporta la pila.
    static final int cantidadNodos = 10;
//El atributo totalNodos, es un contador de nodos que devuelve en todo momento la cantidad de
//nodos almacenados en la pila.
    private int totalNodos;
    private Nodo tope;
    public DeclararPila(){
        totalNodos = 0;
        tope = null;
    }
//Método que devuelve la cantidad de nodos que tiene la pila.
    public int getTotalNodos(){
        return totalNodos;
    }
//Método que devuelve el nodo del tope de la pila.
    public Nodo getTope(){
        return tope;
    }
//Métodos para determinar el estado de la pila (pila llena o pila vacía). Para esto se declaran dos
//funciones de tipo booleano. Si la pila está Llena se retorna el valor de true, en caso contrario se
//retorna el valor false. Lo mismo si la pila está vacía se retorna el valor de true, en caso contrario
//se retorna el valor false.
    public boolean pilaLlena(){
        if (getTotalNodos() == cantidadNodos){
            return true;
        }else{
            return false;
        }
    }
    public boolean pilaVacia(){
        if (getTotalNodos() ==0){
            return true;
        }else{
            return false;
```

```
        }
    }
//Implementación del método que agrega nodos a la pila por el tope, que en este caso será la
//cabeza.
    public void agregar(String dato){
        if (! pilaLlena()){ //Mientras que la pila no esté llena se pueden agregar nodos.
            Nodo nuevo = new Nodo(); //Se crea una nueva instancia de laclase nodo.
            nuevo.setDato(dato); //Al nodo nuevo en el campo de información se le agrega el dato.
            nuevo.setSiguiente(getTope()); //El apuntador del nuevo nodo, apuntara al que esta de tope
            tope = nuevo; //Ahora, se asigna que el tope será el nuevo nodo insertado.
            totalNodos = totalNodos+1; //Se incrementa el contador de nodos en uno.
        }
    }
//Implementación del método que elimina o quita nodos de la pila.
    public void quitar(){
//Apuntador de la clase nodo para almacenar temporalmente el nodo a eliminar (tope).
        Nodo temp;
        if (! pilaVacia()){ //Mientras que la pila no este vacía se pueden eliminar nodos.
            temp = getTope(); //Se asigna temporalmente el tope de la pila a el apuntador temp.
            tope = tope.getSiguiente(); //Ahora el tope será el nodo que le sigue al que estaba de tope.
            temp = null;
            totalNodos = totalNodos-1; //El contador de nodos se decremento en uno.
        }
    }
//Implementación del método consulta la información del tope, aquí también se puede retornar todo
//el nodo, en caso de que se tengan varios datos en el nodo.
    public String valorTope(){
        return tope.getDato(); //Retorna la información que tiene el nodo del tope en el campo dato.
    }

//Implementación del método que elimina todos los nodos de la pila. Mientas que la pila no este
//vacía se va quitando el elemento del tope.
    public void limpiarPila(){
        while (! pilaVacia()){
            quitar();
        }
    }
}
```

- **Implementación de la clase Main en el fichero Main.java**

En el fichero Main.java se declara la clase Main, que tendrá el método que permite visualizar los elementos de la pila y el menú de opciones para realizar las operaciones de agregar y quitar datos de la pila.

```
public class Main {
    public static int menu(){
        int opcion = 0;
        do{
            opcion = Integer.parseInt(JOptionPane.showInputDialog("===== IMPLEMENTACIÓN DE PILAS -
OPCIÓN DEL MENU ===== \n"
            +"1. Agregar Datos a la Pila \n"+"2. Eliminar Datos de la Pila \n"+"3. Mostrar Datos de la Pila \n"+
                    "4. Vaciar la Pila \n"+"5. Salir"+"\n \n Seleccione una opción del 1 al 5"));
        }while(opcion <= 0 || opcion > 5);
```

```
        return opcion;
    }

        public static void mostrarPila(DeclararPila pila){
//Se crea un objeto de la clase DeclararPila (temp), para crear una pila temporal en donde
//almacenar los datos.
        DeclararPila temp = new DeclararPila();
        String verDatosPila = "";
        while (! pila.pilaVacia()){ //Mientras la pila no esté vacía se muestran sus datos.
            verDatosPila = verDatosPila+String.valueOf("---- "+pila.valorTope()+"\n");
//Se agrega a una pila temporal el valor del tope de la pila actual.
            temp.agregar(pila.valorTope());
            pila.quitar(); //Se elimina el tope de la pila actual, para pasar al siguiente elemento.
        }
        JOptionPane.showMessageDialog(null, "========= ELEMENTOS DE LA DE PILA
==========="+"\n"+verDatosPila+"\n");

//Posteriormente se copia nuevamente la pila temporal hasta la pila original.
        while (! temp.pilaVacia()){
//Se agrega a la pila original el valor del tope de la pila temporal.
            pila.agregar(temp.valorTope());
            temp.quitar(); //Se elimina el tope de la pila temporal, para pasar al siguiente elemento.
        }
    }
    public static void main(String[] args) {
        DeclararPila pila = new DeclararPila();
        String dato;
        int opcion;
        do{
            opcion = menu();
            switch(opcion) {
            case 1:
                dato = JOptionPane.showInputDialog(null,"Digite el dato que quiere agregar a la Pila");
                if (! pila.pilaLlena()){
                    pila.agregar(dato);
                    mostrarPila(pila);
                }else{
                    JOptionPane.showMessageDialog(null, "============ IMPLEMENTACIÓN DE PILAS
============"+"\n\n"+
                            "La pila está llena, No puede agregar más elementos \n\n");
                }
                break;
            case 2:
                if (! pila.pilaVacia()){
                    pila.quitar();
                    mostrarPila(pila);
                }else{
                    JOptionPane.showMessageDialog(null, "============ IMPLEMENTACIÓN DE PILAS
============"+"\n\n"+
                            "La pila está vacía, No pueden quitar datos \n\n");
                }
                break;
            case 3:
                mostrarPila(pila);
                break;
            case 4:
```

```
            pila.limpiarPila();
            mostrarPila(pila);
            JOptionPane.showMessageDialog(null, "============ IMPLEMENTACIÓN DE PILAS
============"+"\n\n"+
                    "La pila está vacía, No pueden quitar datos \n\n");
            break;
         case 5:
            break;
         }
      }while(opcion != 5);
   }
}
```

BIBLIOGRAFÍA

- Zahonero, I., y Joyanes Aguilar, L. (1999). Estructura de Datos - Algoritmos, Abstracción y Objetos. España: McGraw-Hill.

- Allen Weiss, M. (2004). Estructuras de datos en Java. España: Addison Wesley - Pearson. 776 pp.

- Cairó, O. & Guardati, S. (2002). Estructuras de datos (2nd ed.). México: McGraw-Hill.

- Joyanes, L. (2000). Programación en C++: Algoritmos, estructuras de datos y objetos (1st ed.). Madrid: McGraw-Hill.

- Kernighan, B., & Ritchie, D. (1991). El Lenguaje de Programación C (2nd ed.). México: Prentice Hall.

- Stroustrup, B. (2002). El Lenguaje de Programación C++ (1th ed.). Madrid: Pearson Educación.