Programación

Unidad 5: Aplicación de las estructuras de almacenamiento

Aplicación de las estructuras de almacenamiento

- Estructuras.
  - Listas enlazadas
  - Colas.
  - Pilas
- Creación de arrays.
- Arrays multidimensionales.
- Arrays de objetos en Java.
- Vectores dinámicos en java: La clase Vector

Estructuras

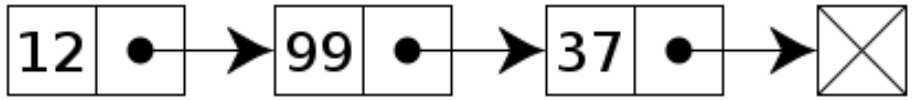
- ¿Que es una estructura de datos?
  - *Es una clase contenedora que proporciona almacenamiento para ítems de datos, y capacidades para almacenar y recuperar estos datos*.
- Algunas estructuras de datos son:
  - Arrays
  - Listas enlazadas
  - Pilas
  - Colas

Listas enlazadas

- Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias (punteros) al nodo anterior o posterior.
- Las listas enlazadas permiten inserciones y eliminación de nodos en cualquier punto de la lista pero no permiten un acceso aleatorio.
- En Java disponemos de la Clase ArrayList
- Existen diferentes tipos de listas enlazadas:
  - Lista Enlazadas Simples
  - Listas Doblemente Enlazadas
  - Listas Enlazadas Circulares
  - Listas Enlazadas Doblemente Circulares.

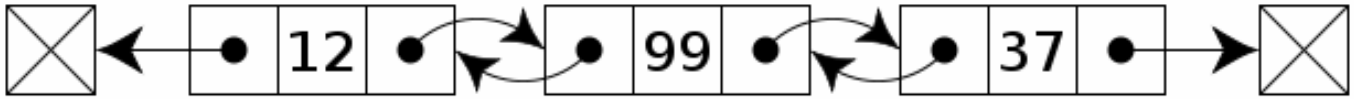
Listas enlazadas

- Existen diferentes tipos de listas enlazadas:
  - Lista Enlazadas Simples



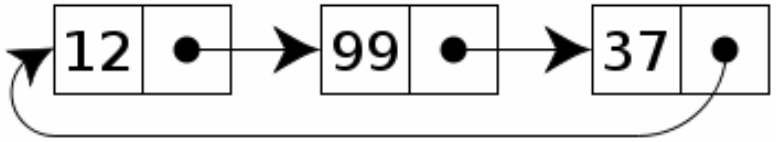
Una lista enlazada simple contiene dos valores: el valor actual del nodo y un enlace al siguiente nodo

  - Listas Doblemente Enlazadas



Una lista doblemente enlazada contiene tres valores: el valor, el link al nodo siguiente, y el link al anterior

  - Listas Enlazadas Circulares



Una lista enlazada circular que contiene tres valores enteros

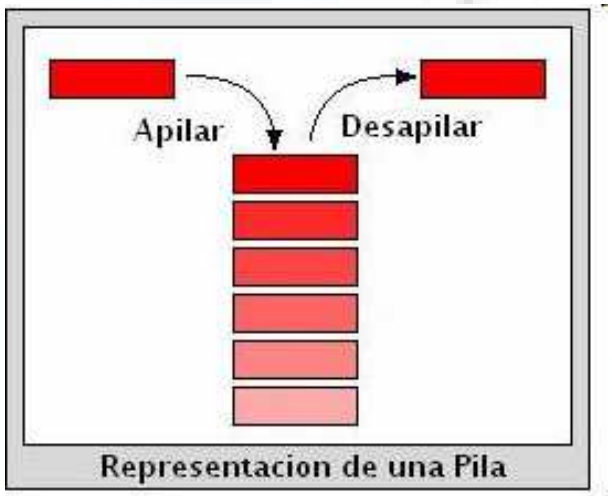
  - Listas Enlazadas Doblemente Circulares.

Colas

- Se caracterizada por ser una secuencia de elementos en la que la operación de inserción *push* se realiza por un extremo y la operación de extracción *pop* por el otro.
- También se le llama estructura FIFO (*First In First Out, es decir,* primero en entrar primero en salir).
  - Es una particularidad de las listas enlazadas
  - Las operaciones básicas son:
    - **Crear**: se crea la cola vacía.
    - **Encolar** (añadir, entrar, insertar): se añade un elemento a la cola. Se añade al final de esta.
    - **Desencolar** (sacar, salir, eliminar): se elimina el elemento frontal de la cola, es decir, el primer elemento que entró.
    - **Frente** (consultar, front): se devuelve el elemento frontal de la cola, es decir, el primer elemento que entró.

Pilas

- Una Pila tiene como filosofía de entrada y salida de datos la estructura LIFO (Last In First Out, es decir, ultimo en entrar, primero en salir).
- En Java disponemos de la Clase Stack (Pila en ingles) en el paquete java.util



Creación de Arrays

- ***Array Unidimensionales***
  - Para declarar un vector se escribe:
    - el tipo de las variables que van a formar el vector
    - a continuación se escriben unos corchetes []
    - después de un espacio
    - Y por último, el nombre que se desea dar al vector.
    - Por ejemplo:
      - int[] a;
      - declara un vector de enteros cuyo nombre es a. Con tal declaración no se ha dicho todavía cuántos elementos tendrá el vector a ni se han dicho cuáles son tales elementos.

Creación de Arrays

- ***Array Unidimensionales***
  - Para inicializar un vector se hacen dos cosas:
    - primero se le da un tamaño
    - luego se crea cada uno de los elementos.
    - Por ejemplo:
      - int[] a=new int[4];
      - determina que a es un vector de 4 enteros.
  - En el caso de vectores new no crea un objeto de una clase sino un vector cuyos elementos serán de la clase indicada, pero esos objetos aún no se han creado.
  - Es decir, la expresión int[] a=new int[4] solamente aparta espacio en memoria para contener los "apuntadores", "punteros" o "manejadores" (pointers) de los objetos, pero los objetos mismos aún no están determinados.
  - Los límites de los arrays se comprueban en tiempo de ejecución para evitar desbordamientos y la corrupción de memoria

Creación de Arrays

- ***Array Unidimensionales***
  - En Java un array es realmente un objeto, porque tiene redefinido el operador *[]*.
  - Tiene una función miembro (método): *length* que se utiliza para conocer la longitud de cualquier array.

```
int a[][] = new int[10][3];
a.length;        /* 10 */
a[0].length;     /* 3 */
```

  - Para crear un array en Java hay dos métodos básicos:
    - Crear un array vacío:

```
int lista[] = new int[50];
```

    - Crear el array con sus valores iniciales:

```
String nombres[] = { "Juan","Pepe","Pedro","Maria" };
```

      - Esto que es equivalente a:

```
String nombres[];
nombres = new String[4];
nombres[0] = new String( "Juan" );
nombres[1] = new String( "Pepe" );
nombres[2] = new String( "Pedro" );
nombres[3] = new String( "Maria" );
```

Creación de Arrays

- ***Array Unidimensionales***
  - No se pueden crear arrays estáticos en tiempo de compilación:

```
int lista[50]; // generará un error en tiempo de compilación
```

  - Tampoco se puede rellenar un array sin declarar el tamaño con el operador *new*:

```
int lista[];
for( int i=0; i < 9; i++ )
    lista[i] = i;
```

  - Todos los arrays en Java son estáticos.
  - Para convertir un array en el equivalente a un array dinámico en C/C++, se usa la clase *vector*, que permite operaciones de inserción, borrado, etc. en el array.

Arrays multidimencionales

- ***Arrays Multidimensionales o Matrices***
  - Se pueden crear matrices multidimensionales de manera muy parecida a como se crean los vectores y sus elementos.
  - Por ejemplo

    ```
    double[][] M;
    M=new double[2][4];
    ```

    - crea una matriz de 2x4 números reales.
    - Para asignar valores a tal matriz se asignan valores a cada uno de los elementos M[i][j] con i entre 0 y 1 y j entre 0 y 3.
    - Una manera de crear dicha matriz y asignarle valores al mismo tiempo sería:

      ```
      double[][] R={ {0.1,0.2,0.3,0.4},{0.2,0.4,0.8,1.6}};
      ```

Arrays de Objetos en Java

- Un array es una colección ordenada de elementos del mismo tipo, que son accesibles a través de un índice.
- Un array puede contener datos primitivos o referencias a objetos.
- Los arrays se declaran:

  [*modificadores*] *tipo_variable [ ] nombre;*

  - Por ejemplo:

    Punto [ ] p;

    - p un array de objetos de tipo Punto.
- Un array se crea como si se tratara de un objeto
  - p = **new** Punto[3];
- En el momento de la creación del array se dimensiona el mismo y se reserva la memoria necesaria.
- Un Ejemplo particular de arrays de Objetos es la clase Vector

Vectores Dinámicos

- **Clase Vector**
  - Permite crear Arrays dinámicos que cambian de dimensión en tiempo de ejecución
  - Métodos:
    - **add** (int index, Object element)
      - Inserta el elemento especificado en la posición especificada en este vector.
    - **capacity** ()
      - Devuelve la capacidad actual de este vector.
    - **clear** ()
      - Elimina todos los elementos de este vector.
    - **clone** ()
      - Devuelve un clon de este vector.
    - **elementAt** (int index)
      - Devuelve el componente en el índice especificado.
    - **ensureCapacity** (int minCapacity)
      - Aumenta la capacidad de este vector, si es necesario, para asegurarse de que puede contener, como mínimo, el número de componentes especificados por el argumento de capacidad mínima.
    - **equals** (Object o)
      - Compara el objeto especificado con este vector de la igualdad.

Vectores Dinámicos

- **Clase Vector**
  - Métodos:
    - **firstElement** ()
      - Devuelve el primer componente (el tema en el índice 0) de este vector.
    - **get** (int index)
      - Devuelve el elemento en la posición especificada en este vector.
    - **set** (int index, Object element)
      - Sustituye el elemento en la posición especificada en este vector con el elemento especificado.
    - **insertElementAt** (Object obj, int index)
      - Inserta el objeto especificado como un componente de este vector se especifica en el index
    - **isEmpty** ()
      - Prueba si este vector no tiene componentes.
    - **lastElement** ()
      - Devuelve el último componente del vector.
    - **remove** (int index)
      - Quita el elemento en la posición especificada en este vector.

Vectores Dinámicos

- **Clase Vector**
  - Métodos:
    - **setSize** (int newSize)
      - Ajusta el tamaño de este vector.
    - **size** ()
      - Devuelve el número de componentes de este vector.
    - **subList** (int fromIndex, int toIndex)
      - Devuelve una vista de la parte de esta lista entre FromIndex, inclusive, y ToIndex, exclusivo.
    - **toArray** ()
      - Devuelve una matriz que contiene todos los elementos de este vector en el orden correcto.
    - **toString** ()
      - Devuelve una cadena de representación de este vector, que contiene las cadenas de representación de cada elemento.
    - **trimToSize** ()
      - Recorta la capacidad de este vector es el vector del tamaño actual.