

1.00Clase9

Arrays (matrices) y vectores

Arrays

- Los *arrays* son una estructura de datos simples.
- Los *arrays* almacenan un conjunto de valores del mismo tipo.
 - Tipos primitivos (*int*, *double*, etc.)
 - Objetos (Estudiantes, Fechas, etc.)
- Los *arrays* son parte del lenguaje de Java.
 - Los *arrays* son objetos, no primitivos como *int* o *double*.
 - Se declaran igual que otros objetos.
 - El objeto *array* posee un miembro de datos entero, *length*, que proporciona el número de elementos del *array*:

```
int aSize= intArray.length; // aSize= 20
```

- Se accede a cada valor a través de un índice:

```
intArray[0]= 4; intArray[1]= 77;
```

Arrays, 2ª parte

- Los índices de los *arrays* comienzan en 0, no en 1.
 - Un *array* con *N* espacios posee un índice que va de 0 a *N*-1.
 - `intArray` posee elementos que van desde `intArray [0]` a `intArray [19]`.
- La longitud de un *array* no puede modificarse una vez que se haya declarado.
- Cuando se declaran, los *arrays* pueden inicializarse.

```
int[] intArray= {5, 77, 4, 9, 28, 0, -9}; // Longitud 7
// Observe que en este caso, "new" está implícito (no hace falta).
```
- Para copiar un *array*, utilice `arraycopy`.

```
int[] newArray= new int[15] // Puede tener un tamaño distinto
// arraycopy(fromArray, fromIndex, toArray, toIndex, count)
System.arraycopy(intArray, 0, newArray, 0, 15);
// intArray y newArray poseen ahora copias de datos separadas.
```
- Si acabásemos de definir `newArray` sin copiar:

```
int[] newArray= intArray;
newArray[2]= -44; // Esto establece intArray[2]= -44
// intArray y newArray serían simplemente dos nombres para el
// mismo array. Recuerde que los nombres en Java son referencias
```

Pruebe sus conocimientos

- 1.Cuál de las siguientes expresiones no declara ni construye un *array*?
 - a. `int[] arr = new int[4];`
 - b. `int[] arr;`
`arr = new int [4];`
 - c. `int[] arr = {1,2,3,4};`
 - d. `int[] arr;`
2. Dado este fragmento de código:

```
int j= ?;
int[] data = new int[10];
System.out.print(data[ j ]);
```

¿Cuál de los siguientes es un valor legal de *j*?
 - a. -1
 - b. 0
 - c. 1.5
 - d. 10

Pruebe sus conocimientos

3. Dado este fragmento de código:

```
int[] arrayA = new int[4];
int[] arrayB;
arrayB = arrayA;
arrayB[2]=4;
arrayA[0]=arrayB[2];
```

¿Cuáles son los valores de los elementos del array A?

- a. desconocido
- b. 0,0,0,0
- c. 4,0,4,0
- d. 4,0,0,0

4. ¿Cuántos objetos hay presentes después de que el siguiente fragmento de código se haya ejecutado?

```
float[] arrayA = new float[10];
float[] arrayB;
arrayB = arrayA;
```

- a. 1
- b. 2
- c. 10
- d. 20

Pruebe sus conocimientos

5. ¿Para cuál de las siguientes aplicaciones NO es apropiado un *array*?

- a. Almacenar los puntos de los 4 cuartos de un partido de baloncesto.
- b. Almacenar el nombre, balance de cuenta y número de cuenta de un individuo.
- c. Almacenar las lecturas de temperatura tomadas cada hora a lo largo de un día.
- d. Almacenar los gastos mensuales de un año.

6. Dado el siguiente fragmento de código:

```
int[] data = {1,3,5,7,11};
for(_____)
System.out.println(data[indice]
);
```

Rellene los espacios para que el programa muestre un listado ordenado de todos los elementos del *array*.

- a. `int indice=4; indice>0; indice--`
- b. `int indice=0; indice<4; indice++`
- c. `int indice=0; indice<data.length() indice++`
- d. `int indice=0; indice<data.length; indice++`

Pruebe sus conocimientos

7. ¿Cuál es la salida del siguiente programa?

```
class Test{
    public static void main ( String[] args ){
        int valor = 10;
        int[] arr = {10,11,12,13};
        System.out.println("valor antes: "+valor);
        alterValue( valor );
        System.out.println("valor después: "+valor);
        System.out.println("arr[0] antes: "+arr[0]);
        alterArray( arr );
        System.out.println("arr[0] después:"+arr[0]);
    }
    public static void alterValor(int x ){
        x = 0; }
    public static void alterArray (int[]a){
        a[0] = 0; }
}
```

- a. valor antes:10
valor después:0
arr[0] antes:10
arr[0] después:0
- b. valor antes :10
valor después:10
arr[0] antes:10
arr[0] después:10
- c. valor antes:10
valor después:10
arr[0] antes:10
arr[0] después:0
- d. valor antes:10
valor después:0
arr[0] antes:10
arr[0] después:10

Ejercicio

- A. Cree una clase *TestArray*.
- B. Comience escribiendo *main()*:
 - Declare y construya un *array* de *doubles*, llamado *dailyTem* que contenga los datos de la temperatura diaria.
 - Utilice una lista de inicialización entre llaves {}.

Lun	Mar	Mie	Jue	Vie	Sab	Dom
70	61	64	71	66	68	62

- Utilice un bucle *for* para imprimir cada elemento del *array* *dailyTemp* en orden invertido (comenzando por el domingo hasta llegar al lunes).

Ejercicio, 2ª parte

- C. Implemente un método en su clase *Temperature* para hallar la temperatura media semanal:

```
public static double average(double[] aDouble) {  
    // escriba aquí su código  
}
```
- D. En su método *main()*, llame al método *average* (promedio) que acaba de escribir:
 - Pase el *array dailyTemp* como argumento.
 - El método *average* devuelve un *double* que usted debería almacenar en la variable *averageTemp* de *main()*.
 - Imprima la temperatura media de *main()* como:
 - Temperatura semanal media: 66

Vectores

- La clase *Vector* es una versión moderna de un *array*:
 - Un vector puede aumentar automáticamente cuando sea necesario:
 - Posee una capacidad, *capacity()* que aumenta cuando es necesario.
 - Posee un tamaño, *size()*, que es el número real de elementos.
 - Vector puede contener elementos de distintos tipos:
 - Siempre que cada cual sea un objeto (referencia).
 - Un vector no puede contener un tipo básico (como *int*, *double*, etc).
 - En un vector se utilizarán conversiones de tipos explícitas o clases envoltorio.
 - Los envoltorios son objetos (ej., *Double*) que contienen a los tipos primitivos.
 - Los vectores no están en el lenguaje central:
 - Se encuentran en el paquete *java.util*, que usted debe importar:

```
import java.util.*;           // Al comienzo del programa
```
 - Los vectores son algo más lentos que los *arrays*:
 - Tiene importancia sólo en métodos numéricos a gran escala.
 - Los vectores poseen muchos métodos que usted puede utilizar.

Métodos de la clase Vector

<code>void addElement(Object o)</code>	Añade un objeto al final del vector, aumenta en uno el tamaño del vector
<code>int capacity()</code>	Devuelve la capacidad del vector
<code>Object elementAt(int i)</code>	Devuelve el objeto que se encuentra en la posición <i>i</i> del índice
<code>int indexOf(Object o)</code>	Encuentra la primera aparición del objeto; utiliza el método <i>equals</i>
<code>void insertElementAt(Object o, int i)</code>	Inserta un objeto en la posición <i>i</i> del índice
<code>boolean isEmpty()</code>	Devuelve verdadero si el vector no posee objetos, si no, devuelve falso
<code>void removeElementAt(int i)</code>	Borra el objeto que se encuentra en la posición <i>i</i> del índice
<code>void setElementAt(Object o, int i)</code>	Fija el elemento del vector en el índice <i>i</i> , para ser el objeto especificado
<code>int size()</code>	Devuelve el tamaño del vector

Constructores de la clase Vector

- Tres formas básicas (sobrecarga del método):
 - `Vector()`
 - Construye un vector vacío de forma que su *array* de datos internos tenga tamaño 10 y su incremento de capacidad estándar sea cero. La capacidad se duplicará cuando se aumente.
 - `Vector(int initialCapacity)`
 - Construye un vector vacío con la capacidad inicial especificada y con un incremento de capacidad igual a cero. La capacidad se duplicará cuando se aumente.
 - `Vector(int initialCapacity, int capacityIncrement)`
 - Construye un vector vacío con la capacidad inicial especificada y con un incremento de capacidad.

Pruebe sus conocimientos

1. ¿Cuál de los siguientes enunciados sobre Vectores NO es verdadero?
- a. Los vectores son algo más rápidos que los *arrays*.
 - b. Los vectores pueden almacenar elementos de distintos tipos.
 - c. Los vectores pueden aumentar su tamaño para almacenar más elementos.
 - d. Los vectores poseen métodos que gestionan su contenido.
2. Teniendo en cuenta que *myVector* es un Vector que ha sido declarado y construido, ¿cuál de las siguientes expresiones es siempre verdadera?
- a. `myVector.size() >= myVector.capacity()`
 - b. `myVector.size() > myVector.capacity()`
 - c. `myVector.capacity() >= myVector.size()`
 - d. `myVector.capacity() > myVector.size()`

Pruebe sus conocimientos

3. ¿Cuál es la salida del siguiente fragmento de código?

```
Vector myVector = new Vector(2);  
myVector.addElement(new Integer(1));  
myVector.addElement(new Integer(2));  
myVector.addElement(new Integer(3));  
System.out.println(myVector.size()+" "+  
myVector.capacity());
```

- a. 2, 3
- b. 3, 2
- c. 3, 4
- d. 3, 3

Pruebe sus conocimientos

4. Dado el siguiente fragmento de código:

```
Vector myVector = new Vector();  
myVector.addElement("One");  
myVector.addElement("Dos");  
myVector.addElement("Tres");  
myVector.addElement("Cuatro");
```

Cuál de las siguientes expresiones modificará *myVector* para que tenga este resultado: Uno; Dos; Cuatro

One; Two; Four

- a. `myVector.removeElementAt(myVector.elementAt(3));`
- b. `myVector.removeElementAt(myVector.indexOf("Tres"));`
- c. `myVector.removeElementAt(3);`
- d. `myVector.removeElementAt(myVector.elementAt(2));`

Pruebe sus conocimientos

5. Dado el fragmento de código de la pregunta 4, ¿cuál de las siguientes expresiones modificará a *myVector* para obtener esto? Uno; Dos; Tres; Cinco.

One; Two; Three; Five

- a. `myVector[3]="Cinco"`
- b. `myVector[4]="Cinco"`
- c. `myVector.setElementAt("Cinco", myVector.indexOf("Cuatro"));`
- d. `myVector.setElementAt("Cuatro", myVector.indexOf("Cinco"));`

6. Dado el fragmento de código de la pregunta 4, ¿cuál de las siguientes expresiones modificará a *myVector* para obtener esto? Uno; Dos; Tres.

One; Two; Three

- a. `myVector.removeElementAt(2);`
- b. `myVector.removeElementAt(myVector.lastElement());`
- c. `myVector.removeElementAt(myVector.capacity());`
- d. `myVector.removeElementAt(myVector.size()-1);`

Pruebe sus conocimientos

7. Dado el siguiente fragmento de código:

```
Vector myVector = new Vector();  
myVector.addElement(new Integer(1));  
myVector.addElement(new Integer(3));  
myVector.addElement(new Integer(7));
```

¿Cuál de las siguientes expresiones modificará a *myVector* para obtener lo siguiente?

1 3 5 7

- a. `myVector.addElementAt(new Integer(5));`
- b. `myVector.insertElementAt(new Integer(5), 2);`
- c. `myVector.insertElementAt[5, 2];`
- d. `myVector.insertElementAt[5, 3];`

Ejercicio

- **A. Descripción del problema:**
 - Queremos almacenar los nombres de los estudiantes por materia. En cada materia de MIT hay entre 1 y 900 estudiantes, así que es difícil anticipar el tamaño de un *array* que contenga los datos. Utilizaremos una clase *vector*, ya que su tamaño puede variar dinámicamente.
- **B. Escriba el código en Java :**
 - 1. Nombre a su clase *MITCourse* (curso Mit).
 - 2. En su método *main()*:
 - Cree una clase *Vector* *students* (estudiantes) de capacidad 5.
 - Añada 4 estudiantes a la clase *Vector*: “Amy”, “Bob”, “Cindy” y “David”. Estos estudiantes serán *Strings* (cadenas).
 - Añádalos directamente a la clase *Vector*; no declare 4 variables *String*.

Ejercicio, 2ª parte

- 3. Escriba un método para imprimir todos los elementos del vector, su tamaño y su capacidad.
 - Añada un método `printOutVector` a la clase `Course100`(curso100):

```
public static void printOutVector(Vector vec) {  
    // Escriba aquí su código  
}
```
- 4. Invoque al método `printOutVector()` desde `main()`.
 - Pase como argumento el Vector `students` (estudiantes).
- 5. Si tiene un portátil, su salida sería así:

```
Amy  
Bob  
Cindy  
David  
Tamaño: 4  
Capacidad: 5
```