Programación

Unidad 5: Tipos enumerados y Arrays

Unidad 5

- Tipos enumerados
- Arrays
- Arrays multidimensionales
- Número variable de parámetros

Tipos enumerados

Unidad 5: Tipos enumerados y Arrays

Tipos enumerados

Estos tipos se incluyeron en la versión 5 de Java.

Los tipos enumerados nos permiten definir tipos de datos con valores limitados.

Se definen de la siguiente forma:

[public] enum Nombre_tipo {VALOR1,...VALORN}

Ejemplo:

```
public enum Direcciones {
    NORTE, SUR, ESTE, OESTE
}
```

Para declarar variables de tipo Direcciones usaremos:

```
private Direcciones orientacion;
```

La variable orientación sólo admitirá que se le asignen valores definidos en la enumeración.

Unidad 5: Tipos enumerados y Arrays

Tipos enumerados

Los tipos enumerados son un tipo especial de clase que hereda de ***java.lang.Enum*** . **NO!** Permiten el uso del operador new para crear objetos.

Enum dispone de las operaciones heredadas y el método estático ***values()***.

```
public void muestraValores() {
    for (Direcciones dir: Direcciones.values()){
        System.out.println(dir);
    }
}
```

```
<termina
NORTE
SUR
ESTE
OESTE
```

Unidad 5: Tipos enumerados y Arrays

Tipos enumerados: Constructores

Los tipos enumerados pueden tener constructores. Estos constructores serán usados en la definición de cada uno de los valores de la enumeración. Los tipos enumerados **NO** permiten el uso de new.

```
public enum Direcciones {
    NORTE(4), SUR(6), ESTE(2), OESTE(2);

    int distancia;

    Direcciones(int d){
        distancia=d;
    }
}
```

Unidad 5: Tipos enumerados y Arrays

Tipos enumerados: Métodos

Dentro de los tipos enumerados también se pueden definir métodos

```
public enum Direcciones {
    NORTE(4), SUR(6), ESTE(2), OESTE(2);

    int distancia;

    Direcciones(int d){
        distancia=d;
    }

    int getDistancia() {
        return distancia;
    }
}
```

Unidad 5: Tipos enumerados y Arrays

Tipos enumerados

```java
public class Barco {
    private Direcciones orientacion;

    public Barco(Direcciones orientacion) {
        super();
        this.setOrientacion(orientacion);
    }

    public void muestraValores() {
        for (Direcciones dir: Direcciones.values()){
            System.out.println(dir);
        }
    }

    public Direcciones getOrientacion() {
        return orientacion;
    }

    public void setOrientacion(Direcciones orientacion) {
        this.orientacion = orientacion;
    }
}
```

```java
public class Test {

    public static void main(String[] args) {
        Barco b1 = new Barco(Direcciones.SUR);
        Direcciones d;

        b1.muestraValores();
        d = b1.getOrientacion();

        System.out.println("Distancia recorrida "+d.getDistancia());

    }

}
```

```
<terminated> Test (2) [Java A
NORTE
SUR
ESTE
OESTE
Distancia recorrida 6
```

Arrays

Unidad 5: Tipos enumerados y Arrays

Arrays

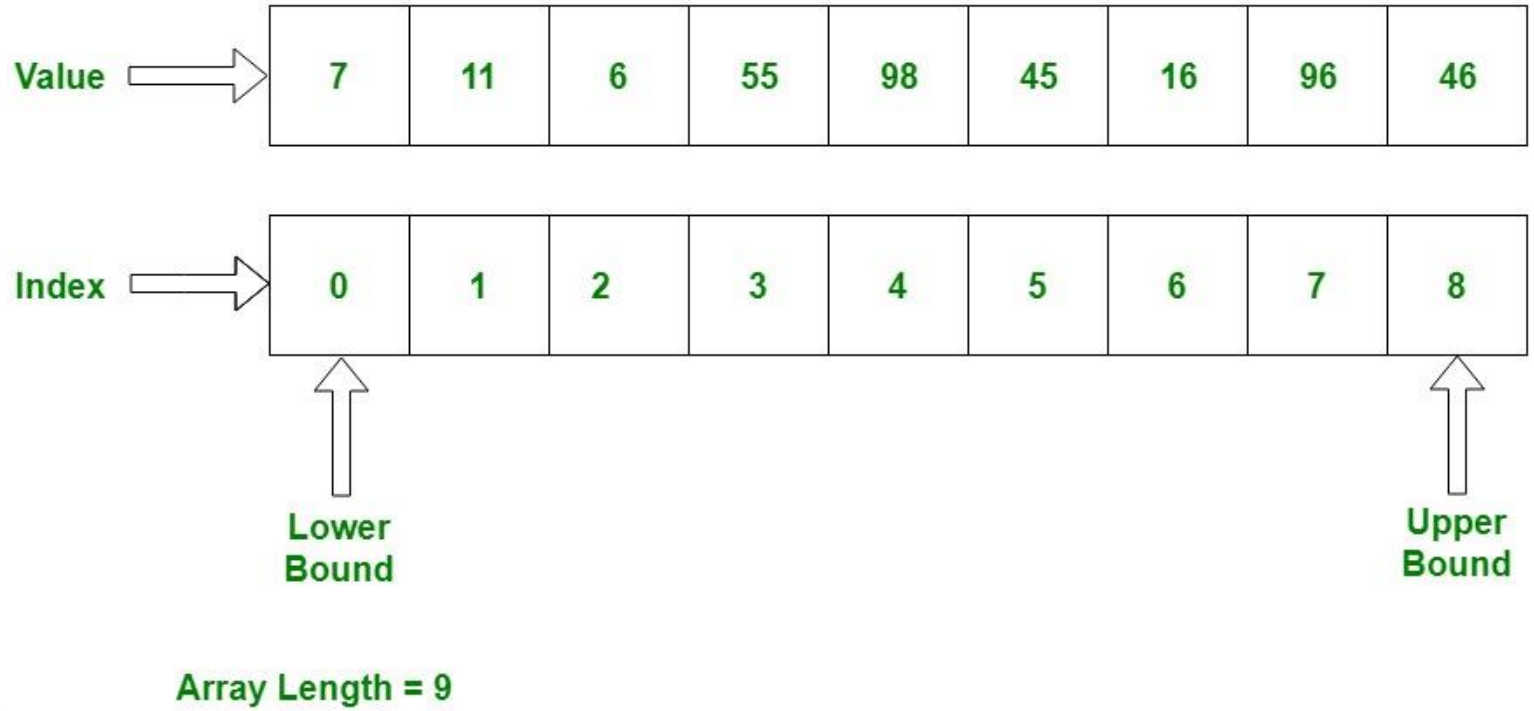
Un array es una estructura de datos que permite albergar **varios elementos del mismo tipo**.

La **longitud** de un array (el tamaño) se establece durante su creación. Una vez establecida la longitud de un array, ya **no se puede modificar**.

Un elemento de un array, es el valor de una de sus posiciones, y se identifica mediante un índice.

Unidad 5: Tipos enumerados y Arrays

Arrays



Unidad 5: Tipos enumerados y Arrays

Arrays

Un array en Java, es un tipo de clase especial que hereda implícitamente de ***java.lang.Object***.

La declaración de un array se realiza mediante el tipo de datos que va a albergar y los corchetes [ ].

Declaración de un array:

```
modificador_acceso tipo[] nombre [= valor_inicial];
```

Ejemplo:

```
private int[] numeros;
private String[] cadenas;
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Los arrays pueden albergar **tanto tipos primitivos como tipos complejos**.

Un array sin inicializar, por defecto vale **null**.

La creación de un array se realiza mediante la keyword ***new***, como con cualquier otra clase.

Creación de un array:

- `modificador_acceso tipo[] nombre = new tipo[longitud];`

Ejemplo:

```
private int[] numeros = new int[10];
private String[] cadenas = new String[5];
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Una vez hayamos creado un array, **todas sus posiciones son inicializadas al valor por defecto** del tipo de variable que albergue.

Es decir, **0** ó 0.0 si se trataba de un número, **false** si se trataba de un boolean y **null** si se trataba de un tipo complejo.

Existe una forma de crear un array inicializando todas sus posiciones a un valor determinado, igualándolo a un listado de elementos separados por comas entre { }. El tamaño del array será el número de elementos del listado.

Unidad 5: Tipos enumerados y Arrays

Arrays

Ejemplos:

```
private int[] numeros = {1,2,3,4,5};
private String[] cadenas = {"hola","adios"};
private Integer[] ints = {12, 98};
```

Para el acceso al elemento de un array se utiliza el nombre del array seguido de unos [ ] con la posición a la que queremos acceder. La primera posición de un array es la 0.

Ejemplo:

```
private int[] numeros = {1,2,3,4,5};
private String[] cadenas = {"hola","adios"};
private Integer[] ints = {12, 98};
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Para conocer la longitud de un array, podremos acceder a su atributo público: ***length***

El **índice** de un array es de tipo **int**

Al no ser dinámico, no podemos:

- Ni eliminar posiciones.
- Ni insertar posiciones.

El borrado será algo lógico, como igualar las posiciones a null, a -1, etc..... dependerá del desarrollador.

Unidad 5: Tipos enumerados y Arrays

Arrays

Es imposible acceder a una posición fuera del array. Llegado el caso se lanzará una excepción: ***ArrayIndexOutOfBoundException***

Ya veremos el tratamiento de excepciones en otro tema.

Unidad 5: Tipos enumerados y Arrays

Arrays

```
public static void main(String[] args) {
    String[] saludos = new String[4];

    saludos[0]="Hola";
    saludos[1]="Adios";
    saludos[2]="Hello";
    saludos[3]="Bye";

    String temp = saludos[2];
    saludos[2]=null;
    saludos[3]=null;

    System.out.println(temp);

    for (int i = 0; i < saludos.length; i++) {
        System.out.println(saludos[i]);
    }

    boolean encontrado=false;
    int i=0;
    while ((!encontrado)&&(i<saludos.length)) {
        if ((saludos[i]!=null)&&(saludos[i].equals("Adios"))) {
            System.out.println("La cadena Adios se ha encontrado en la posición "+i);
            encontrado=true;
        }
        i++;
    }
}
```

```
<terminated> Colecciones [Java Application] /Library/Java/JavaVirtu
Hello
Hola
Adios
null
null
La cadena Adios se ha encontrado en la posición 1
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Envoltorio Arrays. Es una clase que está en el paquete **`java.utils.Arrays`**. Tiene algunos métodos interesantes:

- toString(*array*)

```java
public static void main(String[] args) {
    // Declaro un vector inicial
    int[] miVector = {11, 80, 66, 8, 9};
    // Imprimir un vector
    System.out.println(Arrays.toString(miVector));
}
```

```
/Library/Java/JavaVirtualMachines/
[11, 80, 66, 8, 9]

Process finished with exit code 0
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Tiene algunos métodos interesantes:

- sort(*array*)

```
public static void main(String[] args) {
    // Declaro un vector inicial
    int[] miVector = {11, 80, 66, 8, 9};
    // Ordenación del vector
    Arrays.sort(miVector);
    // Imprimir un vector
    System.out.println(Arrays.toString(miVector));
}
```

```
/Library/Java/JavaVirtualMachines/
[8, 9, 11, 66, 80]

Process finished with exit code 0
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Tiene algunos métodos interesantes:

- binarySearch(*array*)

```java
public static void main(String[] args) {
    // Declaro un vector inicial
    int[] miVector = {11, 80, 66, 8, 9};
    // Imprimir un vector
    System.out.println(Arrays.toString(miVector));
    // Ordenación del vector
    Arrays.sort(miVector);
    // Busco un elemento en concreto en un vector ordenado
    System.out.println(Arrays.binarySearch(miVector, key: 11));
    // Imprimir un vector
    System.out.println(Arrays.toString(miVector));
}
```

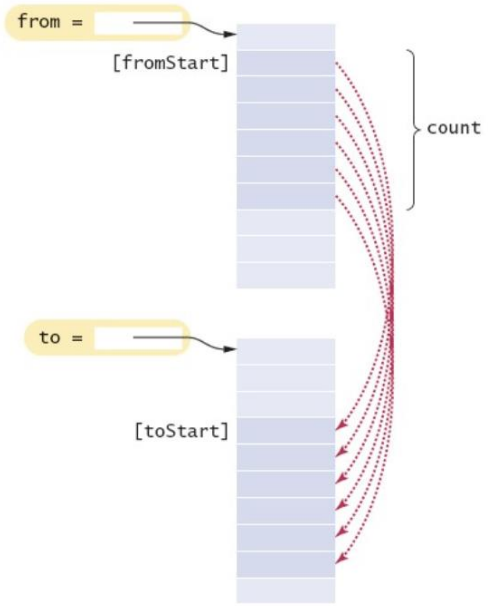
```
/Library/Java/JavaVirtualMachines/
[11, 80, 66, 8, 9]
2
[8, 9, 11, 66, 80]

Process finished with exit code 0
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Copiar un array significa pasar los datos de un array a otro. Existe un método en la clase System que permite hacerlo:

```
public static void main(String[] args) {
    //Declaración e inicialiazación
    int[] primos = {1, 2, 3, 5, 7, 11, 13, 17, 19, 23};
    int[] copia = new int[primos.length];
    //Copia
    System.arraycopy(primos, srcPos: 1, copia, destPos: 3, length: 6);
    //Salida
    System.out.println(Arrays.toString(primos));
    System.out.println(Arrays.toString(copia));
}
```




```
/Library/Java/JavaVirtualMachines/jd
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23]
[0, 0, 0, 2, 3, 5, 7, 11, 13, 0]

Process finished with exit code 0
```

Unidad 5: Tipos enumerados y Arrays

Arrays

Clonado significa hacer una copia exacta. Todos los objetos en Java tienen el método clone que realiza una copia exacta de su contenido en otro espacio de memoria.

Esta operación puede ser muy útil para realizar una copia de un array antes de modificarlo, para no perder su contenido original. El segundo array será totalmente independiente del anterior.

Unidad 5: Tipos enumerados y Arrays

Arrays

```java
public static void main(String[] args) {
    //Declaración e inicialiazación
    int[] primos = {1, 2, 3, 5, 7, 11, 13, 17, 19, 23};
    int[] copiaClonado;
    int[] copiaReferencia;
    //Clonación
    copiaClonado = primos.clone();
    //Cambio el clon, sin afectar al original
    copiaClonado[0] = 0;
    //Copia de referencia
    copiaReferencia = primos;
    //Cambio la copia y afecta al original
    copiaReferencia[1] = 0;
    //Salida
    System.out.println(Arrays.toString(primos));
    System.out.println(Arrays.toString(copiaClonado));
    System.out.println(Arrays.toString(copiaReferencia));
}
```

```
/Library/Java/JavaVirtualMachines/j
[1, 0, 3, 5, 7, 11, 13, 17, 19, 23]
[0, 2, 3, 5, 7, 11, 13, 17, 19, 23]
[1, 0, 3, 5, 7, 11, 13, 17, 19, 23]

Process finished with exit code 0
```

Arrays multidimensionales

Unidad 5: Tipos enumerados y Arrays

Arrays multidimensionales

Un array multidimensional es un **array de arrays**. Es decir, una matriz de 4x2 en realidad en Java está formada por 5 arrays: 1 array con 4 arrays.

Y si estuviéramos hablando de tres dimensiones entonces tendríamos **un array de arrays de arrays**, pero mejor lo vamos a dejar en dos dimensiones.

Creación de un array bidimensional:

- `modificador_acceso tipo[][] nombre = new tipo[long][long];`

Ejemplo:

```
private int[][] numeros = new int[4][2];
```

Unidad 5: Tipos enumerados y Arrays

Arrays multidimensionales

Podemos tener arrays bidimensionales **no cuadrados**. Es decir, que, por ejemplo, la segunda dimensión tenga longitud diferente dependiendo de la primera dimensión.

Creación de un array bidimensional:

- `modificador_acceso tipo[][] nombre = new tipo[long][];`

Ejemplo:

```
private int[][] numeros = new int[4][];
numeros[0] = new int[2];
numeros[1] = new int[10];
numeros[3] = new int[1];
```

Unidad 5: Tipos enumerados y Arrays

Arrays multidimensionales

Al igual que ocurriera en los arrays de una dimensión, también se pueden inicializar en la creación con un listado de valores.

Ejemplos:

```
private int[][] numeros = { {1,2,3} , {1,2,3} };
private String[][] dias = { {"Lunes","Martes"} , {"Miércoles"} };
```

Unidad 5: Tipos enumerados y Arrays

Arrays multidimensionales

```
public static void main(String[] args) {
    int[][] matriz = new int[4][];

    for (int i = 0; i < matriz.length; i++) {
        matriz[i] = new int[5];
        for (int j = 0; j < matriz[i].length; j++) {
            matriz[i][j]=i+j;
        }
    }

    for (int i = 0; i < matriz.length; i++) {
        for (int j = 0; j < matriz[i].length; j++) {
            System.out.print(matriz[i][j]+" ");
        }
        System.out.println();
    }
}
```

```
<terminated>
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
```

Número variable de parámetros

Unidad 5: Tipos enumerados y Arrays

Número variable de parámetros

Se pueden definir métodos para recibir un número variable de argumentos. (Disponible a partir de la versión 5 de Java)

Definición:

- `tipo... nombre_parámetro`

Ejemplo:

- `String... Cadenas`

Este parámetro realmente es un array en el que se almacenan los valores recibidos.

Unidad 5: Tipos enumerados y Arrays

Número variable de parámetros

```
public int sumador(int... valores) {
    int suma=0;
    for (int i:valores) {
        suma+=i;
    }
    return suma;
}
```

Unidad 5: Tipos enumerados y Arrays

Número variable de parámetros

Se pueden declarar tanto parámetros estándares como parámetros variables. Lo único que se debe cumplir es que los parámetros variables se coloquen **siempre en última posición**.

Ejemplo:

- `public void metodo1(int k, String s, int...nums) {}` // Correcto
- `public void metodo2(int p, String... s, long f) {}` // Incorrecto

Unidad 5: Tipos enumerados y Arrays

Número variable de parámetros

```
public class Calcular {
    public int sumador(int... valores) {
        int suma=0;
        for (int i:valores) {
            suma+=i;
        }
        return suma;
    }
}
```

```
public static void main(String[] args) {
    Calcular c = new Calcular();
    System.out.println(c.sumador(3,7,9,11));
    System.out.println(c.sumador(25,43,67));
    System.out.println(c.sumador(19,456));
    System.out.println(c.sumador(23,77,12,11,34,56,99));
}
```

```
<termin
30
135
475
312
```