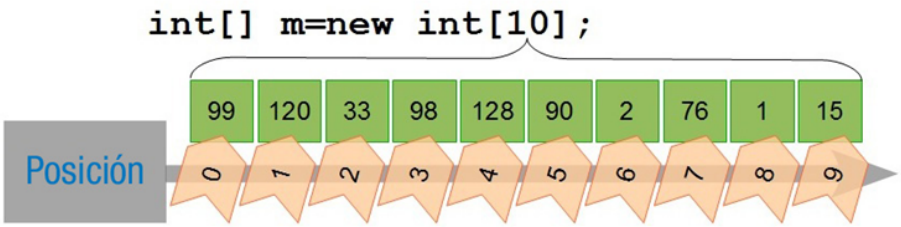
6.D. Arrays unidimensionales.

Sitio: Formación Profesional a Distancia
Curso: Programación
Libro: 6.D. Arrays unidimensionales.
Imprimido por: Iván Jiménez Utiel
Día: martes, 7 de enero de 2020, 22:34

Tabla de contenidos

1. Arrays unidimensionales.

Todos los lenguajes de programación permiten el uso de arrays, veamos como son en Java.



Los arrays permiten almacenar una colección de objetos o datos del mismo tipo. Son muy útiles y su utilización es muy simple:

- **Declaración del array**. La declaración de un array consiste en decir "esto es un array" y sigue la siguiente estructura: "tipo[] nombre;". El tipo será un tipo de variable o una clase ya existente, de la cual se quieran almacenar varias unidades.

- **Creación del array**. La creación de un array consiste en decir el tamaño que tendrá el array, es decir, el número de elementos que contendrá, y se pone de la siguiente forma: "nombre=new tipo[dimension]", donde dimensión es un número entero positivo que indicará el tamaño del array. Una vez creado el array este no podrá cambiar de tamaño.

Veamos un ejemplo de su uso:

```
int[] n; // Declaración del array.
```

```
n = new int[10]; //Creación del array reservando para el un espacio en memoria.
```

```
int[] m=new int[10]; // Declaración y creación en un mismo lugar.
```

Una vez hecho esto, ya podemos almacenar valores en cada una de las posiciones del array, usando corchetes e indicando en su interior la posición en la que queremos leer o escribir, teniendo en cuenta que la primera posición es la cero y la última el tamaño del array menos uno. En el ejemplo anterior, la primera posición sería la 0 y la última sería la 9.

Autoevaluación

¿Cuáles de las siguientes opciones permitiría almacenar más de 50 números decimales?

- [ ] int[] numeros; numeros=new int[51];
- [ ] int[] numeros; numeros=new float[52];
- [ ] double[] numeros; numeros=new double[53];
- [ ] float[] numeros=new float[54];

Resolver

1.1. Uso de arrays unidimensionales.

Ya sabes declarar y crear de arrays, pero, ¿cómo y cuando se usan? Pues existen tres ámbitos principalmente donde se pueden usar, y los tres son muy importantes: modificación de una posición del array, acceso a una posición del array y paso de parámetros.

La modificación de una posición del array se realiza con una simple asignación. Simplemente se especifica entre corchetes la posición a modificar después del nombre del array. Veamoslo con un simple ejemplo:

```
int[] numeros=new int[3]; // Array de 3 números (posiciones del 0 al 2).
```

```
numeros[0]=99; // Primera posición del array.
```

```
numeros[1]=120; // Segunda posición del array.
```

```
numeros[2]=33; // Tercera y última posición del array.
```

El acceso a un valor ya existente dentro de una posición del array se consigue de forma similar, simplemente poniendo el nombre del array y la posición a la cual se quiere acceder entre corchetes:

```
int suma=numeros[0] + numeros[1] + numeros[2];
```

Para nuestra comodidad, los arrays, como objetos que son en Java, disponen de una propiedad pública muy útil. La propiedad length nos permite saber el tamaño de cualquier array, lo cual es especialmente útil en métodos que tienen como argumento un array.

```
System.out.println("Longitud del array: "+numeros.length);
```

El tercer uso principal de los arrays, como se dijo antes, es en el paso de parámetros. Para pasar como argumento un array a una función o método, esta debe tener en su definición un parámetro declarado como array. Esto es simplemente que uno de los parámetros de la función sea un array. Veamos un ejemplo:

```
int sumaarray (int[] j) {
```

```
        int suma=0;
```

```
        for (int i=0; i<j.length;i++)
```

```
            suma=suma+j[i];
```

```
        return suma;
```

```
}
```

En el método anterior se pasa como argumento un array numérico, sobre el cual se calcula la suma de todos los números que contiene. Es un uso típico de los arrays, fijate que especificar que un argumento es un array es igual que declarar un array, sin la creación del mismo. Para pasar como argumento un array a una función, simplemente se pone el nombre del array:

```
int suma=sumaarray (numeros);
```

En Java las variables se pasan por copia a los métodos, esto quiere decir que cuando se pasa una variable a un

método, y se realiza una modificación de su valor en dicho método, el valor de la variable en el método desde el que se ha realizado la invocación no se modifica. Pero cuidado, eso no pasa con los arrays. Cuando dicha modificación se realiza en un array, es decir, se cambia el valor de uno de los elementos del array, si que cambia su valor de forma definitiva. Veamos un ejemplo que ilustra ambas cosas:

```
public static void main(String[] args) {
```

```
    int j=0; int[] i=new int(1);  i[0]=0;
```

```
    modificaArray(j,i);
```

```
    System.out.println(j+"-"+i[0]);  /* Mostrará por pantalla "0-1", puesto que el contenido del array es
```

```
         modificado en la función, y aunque la variable j también se modifica, se modifica una copia de la misma,
```

```
         dejando el original intacto */
```

```
}
```

```

```

```
int modificaArray(int j, int[] i); {
```

```
    j++; int[0]++;  /* Modificación de los valores de la variable, solo afectará al array, no a j */
```

```
}
```

1.2. Inicialización.

Rellenar un array, para su primera utilización, es una tarea muy habitual que puede ser rápidamente simplificada. Vamos a explorar dos sistemas que nos van a permitir inicializar un array de forma cómoda y rápida.

En primer lugar, una forma habitual de crear un array y rellenarlo es simplemente a través de un método que lleve a cabo la creación y el rellenado del array. Esto es sencillo desde que podemos hacer que un método retorne un array simplemente indicando en la declaración que el valor retornado es tipo[], donde tipo de dato primitivo (`int, short, float`,...) o una clase existente (`String` por ejemplo). Veamos un ejemplo:

```
static int[] ArrayConNumerosConsecutivos (int totalNumeros) {
```

```
        int[] r=new int[totalNumeros];
```

```
        for (int i=0;i<totalNumeros;i++) r[i]=i;
```

```
        return r;
```

```
}
```

En el ejemplo anterior se crea un array con una serie de números consecutivos, empezando por el cero, ¿sencillo no? Este uso suele ahorrar bastantes líneas de código en tareas repetitivas. Otra forma de inicializar los arrays, cuando el número de elementos es fijo y sabido a priori, es indicando entre llaves el listado de valores que tiene el array. En el siguiente ejemplo puedes ver la inicialización de un array de tres números, y la inicialización de un array con tres cadenas de texto:

```
int[] array = {10, 20, 30};
```

```
String[] diassemana= {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado",
"Domingo"};
```

Pero cuidado, la inicialización solo se puede usar en ciertos casos. La inicialización anterior funciona cuando se trata de un tipo de dato primitivo (`int, short, float, double`, etc.) o un `String`, y algunos pocos casos más, pero no funcionará para cualquier objeto.

Cuando se trata de un array de objetos, la inicialización del mismo es un poco más liosa, dado que el valor inicial de los elementos del array de objetos será `null`, o lo que es lo mismo, crear un array de objetos no significa que se han creado las instancias de los objetos. Hay que crear, para cada posición del array, el objeto del tipo correspondiente con el operador `new`. Veamos un ejemplo con la clase `StringBuilder`. En el siguiente ejemplo solo aparecerá `null` por pantalla:

```
StringBuilder[] j=new StringBuilderStringBuilder[10];
```

```
for (int i=0; i<j.length;i++) System.out.println("Valor" +i + "="+j[i]); //
Imprimirá null para los 10 valores.
```

Para solucionar este problema podemos optar por lo siguiente, crear para cada posición del array una instancia del objeto:

```
StringBuilder[] j=new StringBuilder[10];
```

```
for (int i=0; i<j.length;i++) j[i]=new StringBuilder("cadena "+i);
```

Reflexiona

Para acceder a una propiedad o a un método cuando los elementos del array son objetos, puedes usar la notación de punto detrás de los corchetes, por ejemplo: `diassemana[0].length`. Fijate bien en el array `diassemana` anterior y piensa en lo que se mostraría por pantalla con el siguiente código:

```
System.out.println(diassemana[0].substring(0,2));
```

Autoevaluación

¿Cuál es el valor de la posición 3 del siguiente array: `String[] m=new String[10]`**?**

- null.
- Una cadena vacía.
- Daría error y no se podría compilar.