Semana 09

Constructores en Java

Definición

Constructores de Java o constructores en Java es una terminología utilizada para construir algo en nuestros programas. Un constructor en Java es un método especial que se usa para inicializar objetos. Se llama al constructor cuando se crea un objeto de una clase. Se puede utilizar para establecer valores iniciales para atributos de objetos.

En Java, un constructor es un bloque de códigos similar al método. Se llama cuando se crea una instancia de la clase. En el momento de llamar al constructor, se asigna memoria para el objeto en la memoria. Es un tipo especial de método que se utiliza para inicializar el objeto. Cada vez que se crea un objeto usando la palabra clave new(), se llama al menos a un constructor.

Nota: No es necesario escribir un constructor para una clase. Es porque el compilador de Java crea un constructor predeterminado (constructor sin argumentos) si la clase no tiene ninguno.

Diferencias con métodos en Java

- Los constructores deben tener el mismo nombre que la clase dentro de la cual se define, no es necesario para el método en Java.
- Los constructores no devuelven ningún tipo mientras que los métodos tienen el tipo de retorno o se anulan si no devuelven ningún valor.
- Los constructores se llaman solo una vez en el momento de la creación del objeto, mientras que los métodos se pueden llamar cualquier número de veces.

Sintaxis

```
class Prueba{
  Prueba(){
    …
  }
}
```

La primera línea de un constructor es una llamada a super() o this(), (una llamada a un constructor de una superclase o un constructor sobrecargado), si no escribe la llamada a super en su constructor, el El compilador le proporcionará una llamada sin argumentos a super en la primera línea de su código, se debe llamar al superconstructor para crear un objeto:

```
import java.io.*;

class Prueba {
	Prueba() { super(); }
	public static void main(String[] args)
	{
		Prueba p = new Prueba();
	}
}
```

Si cree que su clase no es una subclase, en realidad lo es, cada clase en Java es la subclase de la clase Object, incluso si no dice objeto extendido en la definición de su clase.

Necesidad de un constructor

Pensemos en una caja. Si hablamos de una clase de caja, tendrá algunas variables de clase (por ejemplo, longitud, anchura y altura). Pero cuando se trata de crear su objeto (es decir, caja ahora existirá en la memoria de la computadora), entonces ¿puede estar allí un cuadro sin un valor definido para sus dimensiones? La respuesta es no.
Por lo tanto, los constructores se utilizan para asignar valores a las variables de clase en el momento de la creación del objeto, ya sea explícitamente por el programador o por el propio Java (constructor predeterminado).

¿Cuándo se llama a un constructor?

Cada vez que se crea un objeto usando una palabra clave new(), se invoca al menos un constructor (podría ser el constructor predeterminado) para asignar valores iniciales a los miembros de datos de la misma clase.

Las reglas para escribir constructores son las siguientes:

- Los constructores de una clase deben tener el mismo nombre que el nombre de la clase en la que reside.
- Un constructor en Java no puede ser abstract, final, static, o Synchronized.

- Los modificadores de acceso se pueden usar en la declaración del constructor para controlar su acceso, es decir, qué otra clase puede llamar al constructor.

Hasta ahora, hemos aprendido que los constructores se usan para inicializar el estado del objeto. Al igual que los métodos, un constructor también contiene una colección de declaraciones (es decir, instrucciones) que se ejecutan en el momento de la creación del objeto.

Tipos de Constructor en Java

Principalmente hay tres tipos de constructores en Java:

- Constructor sin argumentos
- Constructor con parámetros
- Constructor por defecto

1. Constructor sin argumentos

Un constructor que no tiene ningún parámetro se conoce como constructor sin argumentos o sin argumentos. Si no definimos un constructor en una clase, entonces el compilador crea un constructor (sin argumentos) para la clase. Y si escribimos un constructor con argumentos o sin argumentos, el compilador no crea un constructor por defecto.

Nota: El constructor por defecto proporciona los valores predeterminados para el objeto, como 0, nulo, etc., según el tipo.

Ejemplo en código:

```
import java.io.*;

class Prueba {
	int num;
	String name;

	Prueba() { System.out.println("Se invoco constructor sin argumentos"); }
}

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

```
		Prueba p = new Prueba();

		System.out.println(p.name);
		System.out.println(p.num);
	}
}
```

2. Constructor con parámetros

Un constructor que tiene parámetros se conoce como constructor parametrizado. Si queremos inicializar campos de la clase con nuestros propios valores, entonces use un constructor parametrizado.

Ejemplo en código:

```
import java.io.*;
class Prueba {
	String name;
	int id;
	Prueba(String name, int id)
	{
		this.name = name;
		this.id = id;
	}
}
class Main {
	public static void main(String[] args)
	{
		Prueba p = new Prueba("Ren", 33);
		System.out.println("Nombre :" + p.name + " con ID :" + p.id);
	}
}
```

Recordatorio: ¿Los constructores tienen un valor de retorno?

No hay declaraciones de "valor de retorno" en el constructor, pero el constructor devuelve la instancia de clase actual. Podemos escribir 'return' dentro de un constructor.

Ahora, el tema más importante que entra en juego es la fuerte incorporación de POO con constructores conocida como sobrecarga de constructores. Al igual que los métodos, podemos sobrecargar los constructores para crear objetos de

diferentes maneras. El compilador diferencia a los constructores en función del número de parámetros, los tipos de parámetros y el orden de los parámetros.

Ejemplo en código:

```
import java.io.*;

class Prueba {
	Prueba(String name)
	{
		System.out.println("Constructor con un "
					+ "argumento - String : " + name);
	}

	Prueba(String name, int age)
	{

		System.out.println(
			"Constructor con dos argumentos : "
			+ " String e Integer : " + name + " " + age);
	}

	Prueba(long id)
	{
		System.out.println(
			"Constructor con un argumento : "
			+ "Long : " + id);
	}
}

class Main {
	public static void main(String[] args)
	{
		Prueba p2 = new Prueba("ICC");

		Prueba p3 = new Prueba("Ciencias", 50);

		Prueba p4 = new Prueba(325614567);
	}
}
```

3. Constructor por defecto

Un constructor que no tiene parámetros se conoce como el constructor predeterminado. Un constructor predeterminado es invisible. Y si escribimos un constructor con argumentos o sin argumentos, el compilador no crea un constructor predeterminado. Se saca. Está siendo sobrecargado y llamado constructor parametrizado. El constructor predeterminado cambió al constructor parametrizado. Pero el constructor parametrizado no puede cambiar el constructor predeterminado.

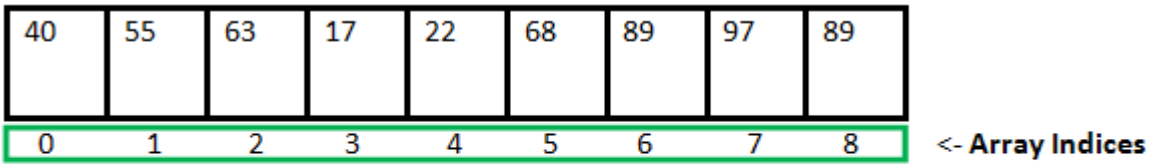
Arreglos en Java

Definición

Un arreglo en Java es un grupo de variables del mismo tipo a las que se hace referencia con un nombre común. Veamos algunos puntos importantes de arreglos en Java:

- En Java, todos los arreglos se asignan dinámicamente.
- Los arreglos se alojan en localidades contiguas de memoria.
- Dado que los arreglos son objetos en Java, podemos encontrar su longitud usando la propiedad length del objeto.
- Una variable de arreglo en Java también se puede declarar como otras variables con [ ] después del tipo de datos.
- Las variables en el arreglo están ordenadas y cada una tiene un índice que comienza con 0.
- El arreglo de Java también se puede usar como un campo estático, una variable local o un parámetro de método.
- El tamaño de un arreglo debe especificarse por valor int o corto y no long.
- La superclase directa de un tipo de arreglo es Object.
- El tamaño de un arreglo no se puede modificar (una vez inicializado). Sin embargo, se puede hacer que una referencia de arreglo apunte a otro arreglo.

Un arreglo puede contener referencias primitivas (int, char, etc.) y de objetos (o no primitivas) de una clase, según la definición del arreglo. En el caso de tipos de datos primitivos, los valores reales se almacenan en ubicaciones de memoria contiguas.



Array Length = 9
First Index = 0
Last Index = 8

Creación, inicialización, y acceso a un arreglo

Arreglos de una dimensión

La forma general de una declaración de un arreglo unidimensional es

```
type var-name[];
```

O

```
type[] var-name;
```

La declaración de un arreglo tiene dos componentes: el tipo y el nombre. type declara el tipo de elemento del arreglo. El tipo de elemento determina el tipo de datos de cada elemento que comprende el arreglo. Al igual que un arreglo de enteros, también podemos crear un arreglo de otros tipos de datos primitivos como char, float, double, etc., o tipos de datos definidos por el usuario (objetos de una clase). Por lo tanto, el tipo de elemento de la matriz determina qué tipo de datos contendrá el arreglo.

Ejemplos:

int intArray[];
o int[] intArray;

byte byteArray[];
short shortsArray[];
boolean booleanArray[];
long longArray[];
float floatArray[];
double doubleArray[];
char charArray[];

MyClass myClassArray[];

```
Object[]  ao,
Collection[] ca;
```

Aunque la primera declaración establece que int Array es una variable de arreglo, no existe un arreglo real. Simplemente le dice al compilador que esta variable (int Array) contendrá un arreglo del tipo entero. Para vincular int Array con un arreglo “físico” real de enteros, debe asignar uno usando new y asignarlo a int Array.

Instanciando un arreglo en Java

Cuando se declara un arreglo, solo se crea una referencia de un arreglo. Para crear o dar memoria al arreglo, cree un arreglo como este: La forma general de new, tal como se aplica los arreglos unidimensionales, aparece de la siguiente manera:

var-name = new type [size];

Aquí, el tipo especifica el tipo de datos que se asignan, el tamaño determina la cantidad de elementos en el arreglo y var-name es el nombre de la variable del arreglo que está vinculada al arreglo. Para usar new para asignar un arreglo, debe especificar el tipo y la cantidad de elementos para asignar.

Ejemplo:

```
int intArray[ ];
intArray = new int[20];
```

O

```
int[] intArray = new int[20];
```

Nota: Los elementos del arreglo asignados por new se inicializarán automáticamente en cero (para tipos numéricos), falso (para booleanos) o nulo (para tipos de referencia).

La obtención de un arreglo es un proceso de dos pasos. Primero, debe declarar una variable del tipo de arreglo deseado. En segundo lugar, debe asignar la memoria para contener el arreglo, utilizando new, y asignarla a la variable de arreglo. Por lo tanto, en Java, todos los arreglos se asignan dinámicamente.

Arreglo literal

En una situación en la que ya se conocen el tamaño de un arreglo y las variables del arreglo, se pueden usar literales de arreglo.

```
int[ ] intArray = new int[ ]{ 1,2,3,4,5,6,7,8,9,10 };
```

En este caso:

- La longitud de este arreglo determina la longitud del arreglo creado.
- No es necesario escribir la nueva parte int[ ] en las últimas versiones de Java.

Accediendo a un arreglo en Java

Se accede a cada elemento del arreglo a través de su índice. El índice comienza con 0 y termina en (tamaño total del arreglo)-1. Se puede acceder a todos los elementos del arreglo usando ciclos en Java.

```
for (int i = 0; i < arr.length; i++)
  System.out.println("Element at index " + i +
                                " : "+ arr[i]);
```