Tipos de Datos en Java

Tipos de Datos en Java

Los **tipos de datos** en Java son una categorización de las variables que se utilizan en el lenguaje de programación. Estas categorizaciones determinan qué tipo de valores pueden almacenarse en las variables y cómo se pueden manipular estos valores. Java es un lenguaje de programación fuertemente tipado, lo que significa que cada variable se declara con un tipo específico.

Java tiene dos categorías principales de tipos de datos:

1. **Tipos de Datos Primitivos**
2. **Tipos de Datos de Referencia**

1. Tipos de Datos Primitivos

Java tiene 8 tipos de datos primitivos. Son los tipos de datos más básicos y no derivan de ningún otro tipo.

Tipo de Dato	Tamaño (bits)	Rango de Valores	Descripción
`byte`	8	-128 a 127	Un entero de 8 bits con signo.
`short`	16	-32,768 a 32,767	Un entero de 16 bits con signo.

Tipo de Dato	Tamaño (bits)	Rango de Valores	Descripción
`int`	32	-2,147,483,648 a 2,147,483,647	Un entero de 32 bits con signo.
`long`	64	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807	Un entero de 64 bits con signo.
`float`	32	3.40282347 x 10^38 (positivo y negativo)	Un número en coma flotante de precisión simple de 32 bits, sigue la IEEE 754.
`double`	64	1.79769313486231570 x 10^308 (positivo y negativo)	Un número en coma flotante de precisión doble de 64 bits, sigue la IEEE 754.
`char`	16	'\u0000' a '\uffff'	Un carácter Unicode de 16 bits.
`boolean`	1 (tamaño no definido)	`true` o `false`	Representa un valor booleano.

Ejemplos de Tipos Primitivos

```java
public class TiposPrimitivos {
    public static void main(String[] args) {
        // Enteros
        byte unByte = 127;
        short unShort = 32000;
        int unInt = 2147483647;
        long unLong = 9223372036854775807L;

        // Punto flotante
        float unFloat = 3.14f;
        double unDouble = 3.141592653589793;

        // Carácter
        char unChar = 'A';

        // Booleano
        boolean unBoolean = true;

        // Imprimir valores
        System.out.println("byte: " + unByte);
        System.out.println("short: " + unShort);
        System.out.println("int: " + unInt);
        System.out.println("long: " + unLong);
        System.out.println("float: " + unFloat);
        System.out.println("double: " + unDouble);
        System.out.println("char: " + unChar);
        System.out.println("boolean: " + unBoolean);
    }
}
```

2. Tipos de Datos de Referencia (Tipos Object)

Los tipos de datos de referencia en Java incluyen:

- **Clases**: Definen objetos y sus métodos.
- **Interfaces**: Definen métodos que pueden ser implementados por las clases.
- **Arreglos**: Colecciones de elementos del mismo tipo.

Los tipos de referencia los vamos a ver a detalle en el tema de Clases y Objetos.

Valores por Defecto

Los valores por defecto son los valores asignados a las variables si no se inicializan explícitamente.

- **Primitivos**:
  - `byte`, `short`, `int`, `long`: `0`
  - `float`, `double`: `0.0`
  - `char`: `'\u0000'`
  - `boolean`: `false`
- **Referencia**:
  - `null`

Conceptos Clave

1. **Variables Primitivas**: Contienen valores simples y directos.
2. **Variables de Referencia**: Contienen referencias (punteros) a objetos.

Usando Clases para Conocer Más Detalles de los Tipos Primitivos en Java

En Java, puedes utilizar clases envolventes (wrapper classes) para obtener más detalles y funcionalidad adicional para los tipos de datos primitivos. Las clases envolventes proporcionan métodos útiles y permiten trabajar con tipos primitivos como si fueran objetos.

Clases Envolventes

Para cada tipo de dato primitivo, existe una clase envolvente correspondiente en el paquete `java.lang`. Estas clases permiten tratar los valores primitivos como objetos y proporcionan métodos para convertir y manipular datos.

Tipo Primitivo	Clase Envolvente
`byte`	`Byte`
`short`	`Short`

Tipo Primitivo	Clase Envolvente
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

Ejemplos y Métodos Útiles

1. Integer (para `int`)

La clase `Integer` proporciona métodos para trabajar con valores enteros.

Métodos útiles:

- `Integer.parseInt(String s)`: Convierte una cadena a un entero.
- `Integer.toString(int i)`: Convierte un entero a una cadena.
- `Integer.MAX_VALUE`: Valor máximo de un `int`.
- `Integer.MIN_VALUE`: Valor mínimo de un `int`.

Ejemplo:

```
public class DetallesInteger {
    public static void main(String[] args) {
        int numero = 42;

        // Convertir a cadena
        String numeroStr = Integer.toString(numero);
        System.out.println("Número como cadena: " + numeroStr);

        // Valor máximo y mínimo
        System.out.println("Valor máximo de int: " + Integer.MAX_VALUE);
        System.out.println("Valor mínimo de int: " + Integer.MIN_VALUE);

        // Parsear cadena a entero
        int parsedNumero = Integer.parseInt("123");
        System.out.println("Número parseado: " + parsedNumero);
    }
}
```

2. Double (para `double`)

La clase `Double` proporciona métodos para trabajar con valores de punto flotante.

Métodos útiles:

- `Double.parseDouble(String s)`: Convierte una cadena a un `double`.
- `Double.toString(double d)`: Convierte un `double` a una cadena.
- `Double.MAX_VALUE`: Valor máximo de un `double`.
- `Double.MIN_VALUE`: Valor mínimo de un `double`.

Ejemplo:

```
public class DetallesDouble {
    public static void main(String[] args) {
        double numero = 3.14159;

        // Convertir a cadena
        String numeroStr = Double.toString(numero);
        System.out.println("Número como cadena: " + numeroStr);

        // Valor máximo y mínimo
        System.out.println("Valor máximo de double: " + Double.MAX_VALUE);
        System.out.println("Valor mínimo de double: " + Double.MIN_VALUE);

        // Parsear cadena a double
        double parsedNumero = Double.parseDouble("2.71828");
        System.out.println("Número parseado: " + parsedNumero);
    }
}
```

3. Character (para `char`)

La clase `Character` proporciona métodos para trabajar con caracteres.

Métodos útiles:

- `Character.isDigit(char ch)`: Verifica si el carácter es un dígito.
- `Character.isLetter(char ch)`: Verifica si el carácter es una letra.
- `Character.toUpperCase(char ch)`: Convierte un carácter a mayúscula.
- `Character.toLowerCase(char ch)`: Convierte un carácter a minúscula.

Ejemplo:

```
public class DetallesCharacter {
    public static void main(String[] args) {
        char letra = 'a';

        // Convertir a mayúscula
        char mayuscula = Character.toUpperCase(letra);
        System.out.println("Letra en mayúscula: " + mayuscula);

        // Verificar si es una letra
        System.out.println("Es una letra: " + Character.isLetter(letra));

        // Verificar si es un dígito
        char digito = '1';
        System.out.println("Es un dígito: " + Character.isDigit(digito));
```

```
    }
}
```

Tipos de Datos `char` en Java y Unicode

El tipo de dato `char` en Java soporta Unicode. En Java, un `char` es un entero de 16 bits que puede representar cualquier carácter del conjunto de caracteres Unicode.

¿Qué es Unicode?

Unicode es un estándar de codificación de caracteres diseñado para soportar el texto de la mayoría de los sistemas de escritura del mundo. Unicode asigna un código único a cada carácter, independientemente de la plataforma, el programa o el idioma. Esto permite que los sistemas informáticos representen y manipulen texto de manera consistente.

- **Rango de Unicode**: El estándar Unicode puede representar más de 1.1 millones de caracteres, utilizando varios planes de codificación, pero la mayoría de los caracteres comunes se encuentran en el rango de 0x0000 a 0xFFFF, que es lo que soporta el tipo `char` en Java.

Características del Juego de Caracteres Unicode

1. **Universalidad**: Unicode incluye caracteres de prácticamente todos los sistemas de escritura del mundo, así como símbolos técnicos, puntuación y caracteres especiales.
2. **Interoperabilidad**: Al usar Unicode, los sistemas y aplicaciones pueden intercambiar texto de manera confiable y consistente.
3. **Compatibilidad**: Unicode es compatible con otros estándares de codificación de caracteres, permitiendo la conversión entre diferentes codificaciones.

Ejemplo de Uso de `char` con Unicode en Java

```
public class UnicodeExample {
    public static void main(String[] args) {
        // Carácter Unicode para la letra 'A'
        char letra = 'A';
        System.out.println("Carácter: " + letra);
        System.out.println("Código Unicode: " + (int) letra);

        // Carácter Unicode para el símbolo del euro (€)
        char simboloEuro = '\u20AC';
        System.out.println("Carácter: " + simboloEuro);
        System.out.println("Código Unicode: " + (int) simboloEuro);

        // Carácter Unicode para un carácter chino (中)
        char caracterChino = '\u4E2D';
        System.out.println("Carácter: " + caracterChino);
        System.out.println("Código Unicode: " + (int) caracterChino);
    }
}
```

Juego de Caracteres Unicode:

https://home.unicode.org/

Conclusión

Conocer los tipos de datos en Java es fundamental para cualquier programador. Estos tipos determinan cómo se almacenan los datos y cómo se pueden manipular en un programa. Los tipos de datos primitivos son simples y eficientes, mientras que los tipos de datos de referencia permiten crear estructuras de datos complejas y objetos.

Las clases envolventes en Java proporcionan una manera poderosa de trabajar con tipos primitivos, ofreciendo métodos útiles para conversión, manipulación y obtención de información adicional sobre estos tipos. Esto permite aprovechar tanto la eficiencia de los tipos primitivos como la funcionalidad de los objetos en Java.

Recursos Adicionales

Estos recursos proporcionan información detallada y ejemplos adicionales sobre cómo utilizar y manipular los diferentes tipos de datos en Java.

- Java Documentation on Primitive Data Types
- Java Documentation on Reference Data Types
- https://docs.oracle.com/en/java/javase/21/

Estos recursos proporcionan información detallada sobre las clases envolventes y sus métodos, ayudándote a comprender mejor cómo trabajar con tipos de datos en Java.

- Documentación de la clase Integer
- Documentación de la clase Double
- Documentación de la clase Character

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx