En Java, **los tipos primitivos** son los bloques básicos de construcción de datos. Cada tipo primitivo tiene un **literal** específico que se utiliza para representar valores directamente en el código. A continuación, se presentan ejemplos de uso de los literales de cada tipo primitivo en Java, acompañados de una explicación detallada de cada línea y concepto.

Tipos Primitivos en Java

Java cuenta con **8 tipos primitivos**:

1. **byte**
2. **short**
3. **int**
4. **long**
5. **float**
6. **double**
7. **char**
8. **boolean**

1. byte

- **Descripción:** Tipo entero de 8 bits con signo. Rango de -128 a 127.
- **Literal:** Entero decimal dentro del rango permitido.

Ejemplo:

```
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        byte edad = 25;
        System.out.println("Edad: " + edad);
    }
}
```

Explicación:

- `byte edad = 25;`
  - Declara una variable `edad` de tipo `byte` y le asigna el valor literal `25`.
- `System.out.println("Edad: " + edad);`
  - Imprime: `Edad: 25`

2. short

- **Descripción:** Tipo entero de 16 bits con signo. Rango de -32,768 a 32,767.
- **Literal:** Entero decimal dentro del rango permitido.

Ejemplo:

```
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        short temperatura = -15;
        System.out.println("Temperatura: " + temperatura + "°C");
    }
}
```

Explicación:

- `short temperatura = -15;`
  - Declara una variable `temperatura` de tipo `short` y le asigna el valor literal `-15`.
- `System.out.println("Temperatura: " + temperatura + "°C");`
  - Imprime: `Temperatura: -15°C`

3. int

- **Descripción:** Tipo entero de 32 bits con signo. Rango de -2,147,483,648 a 2,147,483,647.
- **Literal:** Entero decimal, binario, octal o hexadecimal.

Ejemplo:

```java
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        int numeroDecimal = 100;
        int numeroBinario = 0b1100100; // 100 en binario
        int numeroOctal = 0144;         // 100 en octal
        int numeroHexadecimal = 0x64;   // 100 en hexadecimal

        System.out.println("Decimal: " + numeroDecimal);
        System.out.println("Binario: " + numeroBinario);
        System.out.println("Octal: " + numeroOctal);
        System.out.println("Hexadecimal: " + numeroHexadecimal);
    }
}
```

Explicación:

- `int numeroDecimal = 100;`
  - Asigna el valor decimal `100` a `numeroDecimal`.
- `int numeroBinario = 0b1100100;`
  - Utiliza el prefijo `0b` para representar el número binario `1100100`, que equivale a `100` en decimal.
- `int numeroOctal = 0144;`
  - Utiliza el prefijo `0` para representar el número octal `144`, que equivale a `100` en decimal.
- `int numeroHexadecimal = 0x64;`
  - Utiliza el prefijo `0x` para representar el número hexadecimal `64`, que equivale a `100` en decimal.
- `System.out.println(...)`
  - Imprime los valores convertidos a decimal:

```
Decimal: 100
Binario: 100
Octal: 100
Hexadecimal: 100
```

4. long

- **Descripción:** Tipo entero de 64 bits con signo. Rango de -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.
- **Literal:** Entero decimal con sufijo `L` o `l`.

Ejemplo:

```java
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        long distancia = 10000000000L;
        System.out.println("Distancia: " + distancia + " metros");
    }
}
```

Explicación:

- `long distancia = 10000000000L;`
  - Declara una variable `distancia` de tipo `long` y le asigna el valor literal `10000000000L`. El sufijo `L` indica que es un literal de tipo `long`.
- `System.out.println("Distancia: " + distancia + " metros");`
  - Imprime: `Distancia: 10000000000 metros`

5. float

- **Descripción:** Tipo de punto flotante de 32 bits. Precisión de hasta 6-7 decimales.
- **Literal:** Número decimal con sufijo `F` o `f`.

Ejemplo:

```java
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        float pi = 3.14159F;
        System.out.println("Valor de Pi: " + pi);
    }
}
```

Explicación:

- `float pi = 3.14159F;`
  - Declara una variable `pi` de tipo `float` y le asigna el valor literal `3.14159F`. El sufijo `F` indica que es un literal de tipo `float`.
- `System.out.println("Valor de Pi: " + pi);`
  - Imprime: `Valor de Pi: 3.14159`

6. double

- **Descripción:** Tipo de punto flotante de 64 bits. Precisión de hasta 15 decimales.
- **Literal:** Número decimal (por defecto) o con notación científica.

Ejemplo:

```
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        double e = 2.718281828459045;
        double masa = 1.67e-27; // 1.67 × 10^-27
        System.out.println("Valor de e: " + e);
        System.out.println("Masa de un protón: " + masa + " kg");
    }
}
```

Explicación:

- `double e = 2.718281828459045;`
  - Declara una variable `e` de tipo `double` y le asigna el valor literal `2.718281828459045`.
- `double masa = 1.67e-27;`
  - Utiliza la notación científica `1.67e-27` para representar `1.67 × 10^-27`, asignándolo a la variable `masa`.
- `System.out.println(...)`
  - Imprime:

    ```
    Valor de e: 2.718281828459045
    Masa de un protón: 1.67E-27 kg
    ```

7. char

- **Descripción:** Tipo de carácter de 16 bits que representa un solo carácter Unicode.
- **Literal:** Carácter entre comillas simples `' '`. Puede usar secuencias de escape.

Ejemplo:

```
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        char letra = 'A';
        char simbolo = '\u2764'; // Corazón Unicode
        char apostrofe = '\'';    // Comilla simple escapada

        System.out.println("Letra: " + letra);
        System.out.println("Símbolo: " + simbolo);
        System.out.println("Apostrofe: " + apostrofe);
    }
}
```

Explicación:

- `char letra = 'A';`
  - Declara una variable `letra` de tipo `char` y le asigna el carácter literal `'A'`.
- `char simbolo = '\u2764';`
  - Utiliza la secuencia de escape Unicode `\u2764` para representar el símbolo de corazón (❤).
- `char apostrofe = '\'';`
  - Utiliza la secuencia de escape `\'` para representar una comilla simple dentro de un `char`.
- `System.out.println(...)`
  - Imprime:

```
Letra: A
Símbolo: ❤
Apostrofe: '
```

8. boolean

- **Descripción:** Tipo que representa valores lógicos `true` o `false`.
- **Literal:** `true` o `false`.

Ejemplo:

```java
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        boolean esJavaDivertido = true;
        boolean estaLloviendo = false;

        System.out.println("¿Es Java divertido? " + esJavaDivertido);
        System.out.println("¿Está lloviendo? " + estaLloviendo);
    }
}
```

Explicación:

- `boolean esJavaDivertido = true;`
  - Declara una variable `esJavaDivertido` de tipo `boolean` y le asigna el valor literal `true`.
- `boolean estaLloviendo = false;`
  - Declara una variable `estaLloviendo` de tipo `boolean` y le asigna el valor literal `false`.
- `System.out.println(...)`
  - Imprime:

```
¿Es Java divertido? true
¿Está lloviendo? false
```

Resumen de Literales de Tipos Primitivos

Tipo Primitivo	Literal	Ejemplo
`byte`	Entero decimal	`byte b = 100;`
`short`	Entero decimal	`short s = -15;`
`int`	Decimal, binario, octal, hexadecimal	`int i = 0x64;`
`long`	Entero con sufijo `L`	`long l = 10000000000L;`
`float`	Decimal con sufijo `F`	`float f = 3.14F;`
`double`	Decimal o notación científica	`double d = 1.67e-27;`
`char`	Carácter entre comillas simples	`char c = 'A';`
`boolean`	`true` o `false`	`boolean b = true;`

Consejos Adicionales

- **Uso de Sufijos:** Para `long` y `float`, es recomendable usar sufijos `L`/`l` y `F`/`f` respectivamente para evitar ambigüedades con `int` y `double`.
- **Notación Binaria y Hexadecimal:** Facilita la representación de valores en diferentes sistemas numéricos, lo cual es útil en programación de bajo nivel y manipulación de bits.
- **Secuencias de Escape en `char`:** Permiten representar caracteres especiales y símbolos Unicode que no pueden ser escritos directamente.
- **Legibilidad:** Al usar literales con subrayados (`_`), puedes mejorar la legibilidad de números grandes (disponible desde Java 7).

Ejemplo de Literales con Subrayados:

```java
public class LiteralesPrimitivos {
    public static void main(String[] args) {
        int numeroGrande = 1_000_000;
        long distanciaAstronomica = 9_223_372_036_854_775_807L;
        System.out.println("Número Grande: " + numeroGrande);
        System.out.println("Distancia Astronómica: " + distanciaAstronomica);
```

```
    }
}
```

Explicación:

- `int numeroGrande = 1_000_000;`
  - Usa subrayados para separar los dígitos y mejorar la legibilidad.
- `long distanciaAstronomica = 9_223_372_036_854_775_807L;`
  - Similarmente, usa subrayados en un número grande.
- `System.out.println(...)`
  - Imprime:

```
Número Grande: 1000000
Distancia Astronómica: 9223372036854775807
```

Conclusión

Comprender y utilizar correctamente los **literales de los tipos primitivos** en Java es fundamental para la correcta representación y manipulación de datos en tus programas. Cada tipo primitivo tiene sus propias reglas y convenciones para los literales, lo que permite una mayor flexibilidad y precisión en el manejo de diferentes tipos de datos.