En Java, los **literales booleanos** son los valores que representan verdadero o falso en el lenguaje, y son las únicas dos opciones posibles para variables de tipo booleano. En Java, estos valores son:

- `true` (verdadero)
- `false` (falso)

Ejemplo en Java:

```
boolean esMayorEdad = true;
boolean esMenorEdad = false;
```

Aquí, `true` y `false` son literales booleanos. No son cadenas de texto ni números, sino valores propios del tipo de datos `boolean`. En Java, este tipo de dato es especialmente útil para condiciones y expresiones lógicas en estructuras de control como `if`, `while` o `for`.

Ejemplo en una condición:

```
int edad = 18;
boolean esMayorEdad = edad >= 18;

if (esMayorEdad) {
    System.out.println("Es mayor de edad.");
} else {
    System.out.println("Es menor de edad.");
}
```

En este ejemplo, `esMayorEdad` almacena un literal booleano (`true` o `false`) según el resultado de la condición `edad >= 18`.

Aplicación ejemplo de tipos de variables

```
public class ejemplovariables {
    final double PI =3.1415926536;
    int x;
    int obtenerX(int x) {
            int valorantiguo = this.x;
            return valorantiguo;
    }
    public static void main(String[] args) {
    }
}
```

¡Vamos a desglosar el código línea por línea!

```
public class ejemplovariables {
```

Esta línea define una **clase pública** llamada `ejemplovariables`. Al ser pública (`public`), esta clase puede ser accedida desde cualquier otra clase de cualquier paquete. Es un contenedor que puede tener variables, métodos y el método principal para ejecutar el programa.

```
    final double PI = 3.1415926536;
```

Aquí se declara una **constante** `PI` de tipo `double`, que es un número decimal de alta precisión. La palabra clave `final` significa que el valor de `PI` no podrá cambiar después de ser asignado, es decir, será constante a lo largo del programa.

- **Uso:** `PI` podría usarse como una referencia constante al valor de pi en cálculos matemáticos, como el área o el perímetro de un círculo.

```
    int x;
```

Se declara una **variable de instancia** llamada `x` de tipo `int` (número entero). Al ser una variable de instancia, cada objeto de la clase `ejemplovariables` tendrá su propia copia de `x`. Por defecto, si `x` no se inicializa, tomará el valor `0` en Java.

- **Uso:** `x` podría representar cualquier valor entero relacionado con el contexto de la clase, como una posición, un índice o un contador.

```
    int obtenerX(int x) {
```

Este es el encabezado de un **método llamado** `obtenerX`, que devuelve un valor de tipo `int` y toma un parámetro llamado `x` de tipo `int`. Este método se utiliza para obtener el valor actual de la variable de instancia `x`, pero a través de una variable de referencia (con el mismo nombre) pasada como parámetro.

```
        int valorantiguo = this.x;
```

Aquí se declara una **variable local** llamada `valorantiguo` y se le asigna el valor de `this.x`. La palabra clave `this` se usa para referirse a la variable de instancia `x` de la clase (no a la `x` que se pasa como parámetro).

- **Uso:** `valorantiguo` almacena el valor actual de `x` antes de cualquier posible cambio.

```
        return valorantiguo;
```

Esta línea **retorna el valor de** `valorantiguo`, es decir, el valor de `x` antes de cualquier modificación. Este valor es devuelto al método que llame a `obtenerX`.

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

Este es el **método principal** `main`, el punto de entrada de cualquier aplicación Java. Aquí es donde podrías crear una instancia de `ejemplovariables` y llamar a sus métodos. Sin embargo, en este caso, el `main` está vacío y no realiza ninguna acción.

¿Para qué se puede usar esta clase?

La clase `ejemplovariables` puede servir como un modelo básico para manejar una variable entera (`x`) y un valor constante (`PI`). Podría extenderse para incluir métodos que realicen cálculos usando `PI` y manipulen la variable `x` con lógica adicional.
