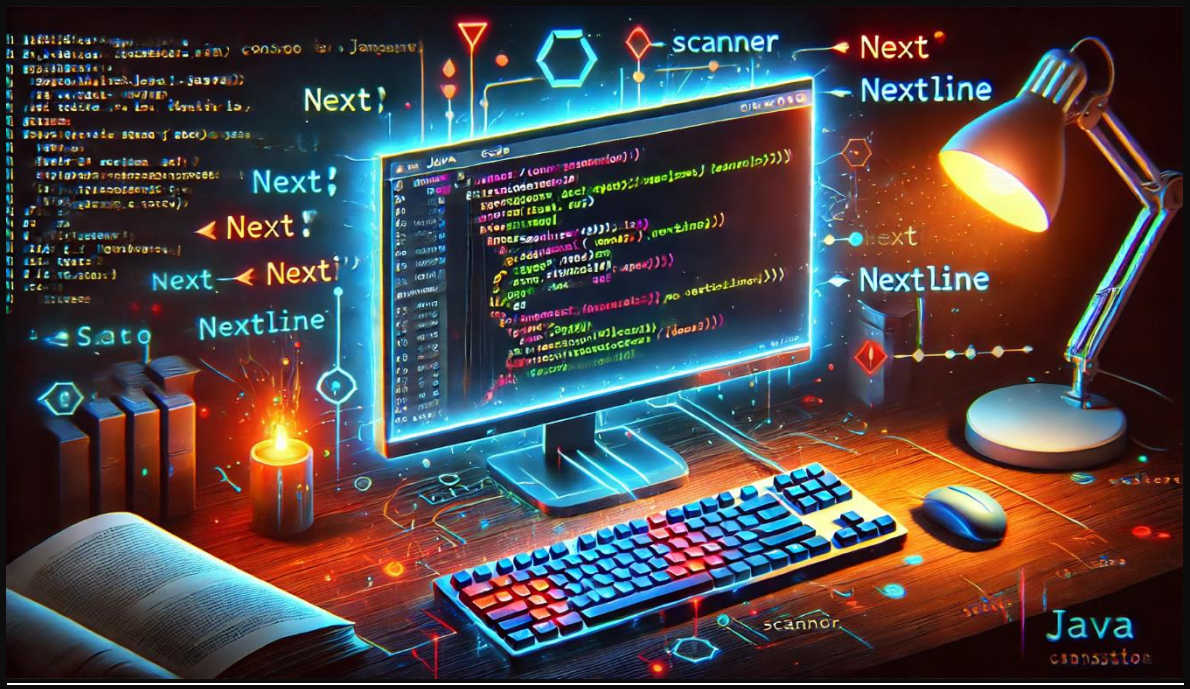
Conversión de Tipos de Datos por Consola en Java



Ejemplo Completo de Clases Wrapper en Java con Entrada desde la Consola

En Java, las clases wrapper (clases envolventes) proporcionan una forma de utilizar tipos de datos primitivos como objetos. Estas clases se encuentran en el paquete `java.lang` y cada tipo primitivo tiene una clase wrapper correspondiente. A continuación, se muestra cómo utilizar estas clases wrapper para convertir datos ingresados desde la consola.

Clases Wrapper Disponibles en Java

1. **Integer**: Para convertir cadenas en enteros.
2. **Float**: Para convertir cadenas en números de punto flotante de precisión simple.
3. **Double**: Para convertir cadenas en números de punto flotante de doble precisión.
4. **Boolean**: Para convertir cadenas en valores booleanos.
5. **Long**: Para convertir cadenas en números largos.
6. **Short**: Para convertir cadenas en números cortos.
7. **Byte**: Para convertir cadenas en bytes.
8. **Character**: No tiene un método `parse`, pero puede ser utilizado para convertir un único carácter.

Código Completo con Ejemplos

```java
import java.util.Scanner;

public class ClasesWrapperEjemplo {
    public static void main(String[] args) {
        Scanner consola = new Scanner(System.in);

        /* Integer.parseInt() */
        System.out.print("Proporciona un valor entero: ");
        int entero = Integer.parseInt(consola.nextLine());
        System.out.println("entero = " + entero);

        /* Float.parseFloat() */
        System.out.print("Proporciona un valor flotante: ");
        float flotante = Float.parseFloat(consola.nextLine());
        System.out.println("flotante = " + flotante);

        /* Double.parseDouble() */
        System.out.print("Proporciona un valor doble: ");
        double doble = Double.parseDouble(consola.nextLine());
        System.out.println("doble = " + doble);

        /* Boolean.parseBoolean() */
        System.out.print("Proporciona un valor booleano (true/false): ");
        boolean booleano = Boolean.parseBoolean(consola.nextLine());
        System.out.println("booleano = " + booleano);

        /* Long.parseLong() */
        System.out.print("Proporciona un valor largo: ");
        long largo = Long.parseLong(consola.nextLine());
        System.out.println("largo = " + largo);

        /* Short.parseShort() */
        System.out.print("Proporciona un valor corto: ");
        short corto = Short.parseShort(consola.nextLine());
        System.out.println("corto = " + corto);

        /* Byte.parseByte() */
        System.out.print("Proporciona un valor byte: ");
        byte byteValue = Byte.parseByte(consola.nextLine());
        System.out.println("byte = " + byteValue);

        /*
        Character no tiene un método parse,
        pero se puede usar para un único carácter
        */
        System.out.print("Proporciona un solo carácter: ");
        char caracter = consola.nextLine().charAt(0);
        System.out.println("caracter = " + caracter);
    }
}
```

Explicación del Código

1. **Integer.parseInt()**:
    - Convierte una cadena en un entero.

    - `int entero = Integer.parseInt(consola.nextLine());`
2. **Float.parseFloat()**:
    - Convierte una cadena en un número de punto flotante de precisión simple.
    - `float flotante = Float.parseFloat(consola.nextLine());`
3. **Double.parseDouble()**:
    - Convierte una cadena en un número de punto flotante de doble precisión.
    - `double doble = Double.parseDouble(consola.nextLine());`
4. **Boolean.parseBoolean()**:
    - Convierte una cadena en un valor booleano.
    - `boolean booleano = Boolean.parseBoolean(consola.nextLine());`
5. **Long.parseLong()**:
    - Convierte una cadena en un número largo.
    - `long largo = Long.parseLong(consola.nextLine());`
6. **Short.parseShort()**:
    - Convierte una cadena en un número corto.
    - `short corto = Short.parseShort(consola.nextLine());`
7. **Byte.parseByte()**:
    - Convierte una cadena en un byte.
    - `byte byteValue = Byte.parseByte(consola.nextLine());`
8. **Character**:
    - Para obtener un carácter, utilizamos el método `charAt` en la cadena.
    - `char caracter = consola.nextLine().charAt(0);`

Conclusión

El manejo de las clases wrapper en Java es esencial para convertir y manipular tipos de datos primitivos como objetos. A través de ejemplos prácticos, hemos demostrado cómo utilizar cada una de las clases wrapper disponibles en Java para leer y convertir datos ingresados desde la consola.

Recursos Adicionales

- Java Documentation - Wrapper Classes

Estos recursos proporcionan información adicional sobre el uso de las clases wrapper en Java.

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx