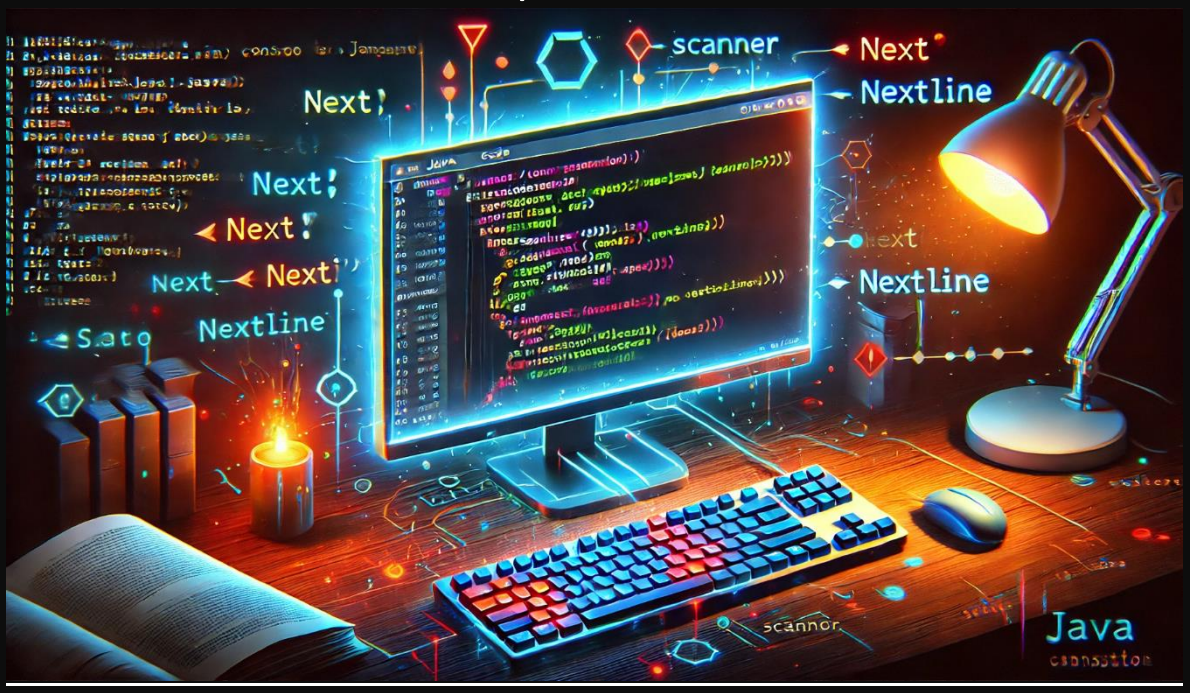
Entrada de Datos por Consola en Java



Leer datos de la consola usando la clase Scanner en Java

Introducción a `Scanner`

La clase `Scanner` es parte del paquete `java.util` y se utiliza para leer la entrada del usuario desde varias fuentes, como el teclado, archivos, cadenas y flujos de entrada.

Sintaxis Básica

Para utilizar la clase `Scanner`, primero debes importarla y luego crear una instancia de `Scanner` asociada a la fuente de entrada.

Importación de la clase `Scanner`:

```
import java.util.Scanner;
```

Creación de una instancia de `Scanner`:

```
var scanner = new Scanner(System.in);
```

Ejemplos de Uso

Leer Datos Primitivos

1. **Leer un `int`:**

```
import java.util.Scanner;

public class LeerInt {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce un número entero: ");
        var numero = scanner.nextInt();
        System.out.println("El número introducido es: " + numero);
    }
}
```

2. **Leer un `double`:**

```
import java.util.Scanner;

public class LeerDouble {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce un número decimal: ");
        double numero = scanner.nextDouble();
        System.out.println("El número decimal introducido es: " + numero);
    }
}
```

3. **Leer un `boolean`:**

```
import java.util.Scanner;

public class LeerBoolean {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce un valor booleano (true/false): ");
        var valor = scanner.nextBoolean();
        System.out.println("El valor booleano introducido es: " + valor);
    }
}
```

Leer Cadenas (Strings)

1. **Leer una línea completa:**

```
import java.util.Scanner;

public class LeerLinea {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce una línea de texto: ");
        var linea = scanner.nextLine();
        System.out.println("La línea introducida es: " + linea);
```

```
    }
}
```

2. **Leer una palabra**:

```
import java.util.Scanner;

public class LeerPalabra {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce una palabra: ");
        var palabra = scanner.next();
        System.out.println("La palabra introducida es: " + palabra);
    }
}
```

Conversión de Datos

A veces es necesario convertir los datos leídos a otro tipo de datos. Aquí algunos ejemplos:

1. **De `String` a `int`**:

```
import java.util.Scanner;

public class ConversionStringAInt {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce un número: ");
        var entrada = scanner.nextLine();
        var numero = Integer.parseInt(entrada);
        System.out.println("El número convertido es: " + numero);
    }
}
```

2. **De `String` a `double`**:

```
import java.util.Scanner;

public class ConversionStringADouble {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("Introduce un número decimal: ");
        var entrada = scanner.nextLine();
        var numero = Double.parseDouble(entrada);
        System.out.println("El número decimal convertido es: " + numero);
    }
}
```

Buenas Prácticas

1. **Cerrar el `Scanner`**:
    - Siempre cierra el `Scanner` cuando ya no lo necesites para liberar los recursos.

```
scanner.close();
```

2. **Manejo de Excepciones**:
    - Usa bloques `try-catch` para manejar posibles excepciones al leer datos. Este tema lo estudiaremos a detalle en la lección de manejo de excepciones, así que solo es un adelanto de esta lección y cómo se puede aplicar al trabajar con la consola de Java.

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class ManejoExcepciones {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        try {
            System.out.print("Introduce un número entero: ");
            var numero = scanner.nextInt();
            System.out.println("El número introducido es: " + numero);
        } catch (InputMismatchException e) {
            System.out.println("Entrada no válida. Introduce un entero.");
        } finally {
            scanner.close();
        }
    }
}
```

Estos recursos te proporcionarán más información y ejemplos sobre cómo utilizar la clase `Scanner` en Java.

Uso de `Scanner` en Java y Problemas con el Buffer

Problema Común con `nextInt` y `nextLine`

Cuando se utilizan métodos como `nextInt`, `nextDouble`, etc., y luego se utiliza `nextLine`, a veces parece que `nextLine` no funciona correctamente. Esto ocurre porque los métodos como `nextInt` no consumen el carácter de nueva línea (Enter) que queda en el buffer de entrada.

Explicación del Problema

Cuando se llama a `nextInt`, `nextDouble`, etc., se lee el valor numérico, pero no se consume el carácter de nueva línea (Enter) que el usuario presiona después de ingresar el valor. Esto deja el carácter de nueva línea en el buffer, y cuando se llama a `nextLine`, éste lee el carácter de nueva línea restante en lugar de la siguiente línea de entrada.

Solución: Vaciar el Buffer

Para solucionar este problema, se debe consumir el carácter de nueva línea restante en el buffer antes de llamar a `nextLine`. Esto se puede hacer llamando a `nextLine` inmediatamente después de `nextInt`, `nextDouble`, etc.

Ejemplo de Código

A continuación, se muestra un ejemplo de cómo manejar este problema:

```
import java.util.Scanner;

public class ScannerExample {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);

        // Leer un número entero
        System.out.print("Introduce un número entero: ");
        var numero = scanner.nextInt();

        // Consumir el carácter de nueva línea restante
        scanner.nextLine(); // Esto limpia el buffer

        // Leer una línea de texto
        System.out.print("Introduce una línea de texto: ");
        var linea = scanner.nextLine();

        System.out.println("Número: " + numero);
        System.out.println("Texto: " + linea);

        scanner.close();
    }
}
```

Explicación del Ejemplo

1. **Lectura de un Entero**:
   - `var numero = scanner.nextInt();` lee un número entero del usuario.
2. **Consumo del Carácter de Nueva Línea**:
   - `scanner.nextLine();` se usa para consumir el carácter de nueva línea restante en el buffer.
3. **Lectura de una Línea de Texto**:
   - `var linea = scanner.nextLine();` lee la siguiente línea de entrada del usuario correctamente.

Otras Buenas Prácticas

- **Uso de `try-with-resources`**: Cierra automáticamente el `Scanner` al final del bloque, evitando fugas de recursos. Este tema lo estudiaremos a detalle cuando veamos el Manejo de Excepciones. Esto es sólo un adelanto de cómo se puede aplicar cuando usamos la consola en Java.

  ```
  try (Scanner scanner = new Scanner(System.in)) {
      // Código para leer entradas
  }
  ```

- **Validación de Entradas**: Siempre valida las entradas del usuario para manejar entradas no válidas y evitar excepciones. El tema de sentencias de decisión como if-else lo estudiaremos a detalle en la lección de Sentencias de Decisión, esto es sólo un adelanto de cómo se puede utilizar cuando trabajamos con la consola de Java.

```
if (scanner.hasNextInt()) {
    var numero = scanner.nextInt();
} else {
    System.out.println("Entrada no válida.");
}
```

Conclusión

La clase `Scanner` es una herramienta poderosa y flexible para leer datos de entrada en Java. Conociendo su sintaxis y métodos, puedes manejar diferentes tipos de datos de manera eficiente. Siguiendo las mejores prácticas, te aseguras de que tu código sea robusto y fácil de mantener.

Recursos Adicionales

- Java Documentation - Scanner
- Common Scanner Issues - Stack Overflow

Saludos!

Ing. Ubaldo Acosta

Fundador de GlobalMentoring.com.mx