Estructura e Indentación del código - Estructuras de control: secuencial-condicional-iterativa

Estructura e Indentación del código

Introducción

En Java, la indentación del código se refiere a la forma en que se organizan las líneas de código dentro de un programa de manera que sea fácil de leer y entender. La indentación ayuda a destacar bloques de código relacionados y a identificar fácilmente el inicio y el final de cada bloque.

Hay algunas reglas básicas a seguir al indentar código en Java:

Use sangrías o tabulaciones para indentar el código en lugar de espacios en blanco. La mayoría de los editores de código tienen opciones para configurar la cantidad de espacio que se utiliza para cada nivel de indentación.

Indente el código dentro de bloques de código, como los bloques de código de una clase, método o estructura de control de flujo (if, for, etc.).

Alinee el código en bloques relacionados. Por ejemplo, alinee los puntos y coma de las declaraciones de variables en una clase o los puntos y coma de las instrucciones en un método.

Evite utilizar una indentación excesiva. Un nivel o dos de indentación suele ser suficiente para la mayoría de los programas.

Ejemplo de código indentado en Java:

```
public class Main {
	public static void main(String[] args) {
		int num1 = 5;
		int num2 = 10;

		if (num1 > num2) {
			System.out.println("num1 es mayor que num2");
		} else {
			System.out.println("num1 no es mayor que num2");
		}
	}
}
```

En Java, la indentación es netamente con fines de legibilidad. Sin embargo, en desarrollos de mínima exigencia técnica, los indentados tienen qué ser correctos.

Existen otras tecnologías, donde el indentado impacta netamente en el código, un ejemplo de esto es el lenguaje “YML”, donde cada indentación se asocia a distintas instancias

```
spring
        datasource
                url=jdbc:mysql://localhost:3306/employeemanager
                username=root
                password=root
        jpa
                show-sql=true
                hibernate
                ddl-auto=update
                properties
                        hibernatedialect=org.hibernate.dialect.MySQL5Dialect
```

Estructuras de control

Las estructuras de control nos permiten modificar el flujo de un programa, ante ciertas condiciones o variaciones. Existen los siguientes tipos:

- secuenciales
- condicionales o selectivas
- repetitivas o de iteración.

Secuenciales

Las instrucciones de un programa se ejecutan por defecto en orden secuencial. Esto significa que las instrucciones se ejecutan en secuencia, una después de otra, en el orden en que aparecen escritas dentro del programa.

La estructura secuencial es el orden natural de ejecución. Las instrucciones que componen esta estructura se ejecutan en orden una a continuación de la otra. La mayoría de las instrucciones están separadas por el carácter punto y coma (;).

Las instrucciones se suelen agrupar en bloques. El bloque de sentencias se define por el carácter llave de apertura ({) para marcar el inicio del mismo, y el carácter llave de cierre (}) para marcar el final. Java en sí mismo opera de modo secuencial.

Condicionales o selectivas

Se basan en comparaciones.

Bloque If:

Para comparaciones sencillas, usamos if, donde establecemos una condición para ser evaluada, y si la misma se cumple se efectúa alguna acción. Opcionalmente, podemos declarar un bloque else, para que se ejecuten otras acciones, en caso de que no se cumpla la primera condición.

```
If (condición) {
        //instrucciones
}else {
        //instrucciones
}
```

Por último está la mezcla de ambas, if-else. Para secuencias más complejas se pueden emplear múltiples bloques como los mencionados, de manera anidada.

Bloque if-else anidados:

```
If (condición) {
        //instrucciones
} else if(condición) {
 //instrucciones
 } else {
        //instrucciones
}
```

Equivalente a usar if anidados, es la estructura switch, en ella se establecen múltiples condiciones y acciones a efectuarse ante cada una de ellas. También es posible implementar una condición por defecto (default), que se ejecutará si no se cumple ninguna de las condiciones previas.

Bloque switch

```
switch (expresión) {
        case valor1:
                //instrucciones
                [break]
        case valor2:
                //instrucciones
                [break] ..
        .. default:
```

```
        //instrucciones
}
```

Estructuras repetitivas

Permiten reiterar las veces que el desarrollo lo amerite. Es sumamente importante evitar bucles infinitos indeseados. Asimismo, tenemos que tener en cuenta que si la condición no se cumple nunca, las instrucciones no se ejecutarán jamás. En casos complejos, también podemos anidar.

Bloque while:

```
        while (condición) {
//instrucciones
 }
```

Si necesitamos que al menos una vez el bloque de instrucciones se ejecute, dejando de lado la condición, es necesario recurrir al bloque do-while.

Bloque do-while:

```
do {
//instrucciones
} while (condición)
```

Por último encontramos el bucle for. Su estructura consta de una expresión inicial, una condición de salida, y una tasa de incremento/decremento para nuestro contador, tal como muestra el bloque anterior.

Bloque for:

```
for(expresión Inicial; condición; incremento/decremento) {
//instrucciones
}
```

Lectura requerida:

Lectura ampliatoria: