DEFINIR ESTÁNDARES DE CODIFICACIÓN DE ACUERDO A PLATAFORMA DE DESARROLLO ELEGIDA. GA7-220501096-AA1-EV02

PRESENTADO POR
UBEIMAR BEDOYA ARANGO

PROFESOR
CRISTIAN ARIAS

SENA
REGIONAL CALDAS
ANÁLISIS Y DESARROLLO DE SOFTWARE 2721447
MEDELLÍN
2024

INTRODUCCIÓN LENGUAJE DE PROGRAMACIÓN JAVA

Java es un lenguaje de programación de propósito general, orientado a objetos, que fue desarrollado por Sun Microsystems en 1995. Desde entonces, ha ganado una gran popularidad y es ampliamente utilizado en diversas aplicaciones, desde desarrollo de aplicaciones web hasta dispositivos móviles y sistemas embebidos. Algunas características de este lenguaje son:

1. **Orientación a Objetos:** Java es un lenguaje orientado a objetos, lo que significa que se basa en el concepto de clases y objetos. Las clases son plantillas que definen propiedades y comportamientos, mientras que los objetos son instancias concretas de esas clases.
2. **Independencia de Plataforma:** Una de las características más destacadas de Java es su capacidad de ser "write once, run anywhere" (escribir una vez, ejecutar en cualquier lugar). Esto se logra mediante la máquina virtual de Java (JVM), que permite ejecutar código Java en cualquier dispositivo o sistema operativo compatible con la JVM.
3. **Portabilidad:** Los programas Java son portables, lo que significa que pueden ejecutarse en cualquier máquina que tenga una JVM instalada, independientemente de la arquitectura de hardware o el sistema operativo.
4. **Compilación y Interpretación:** Java utiliza un enfoque de compilación e interpretación. El código fuente se compila en bytecode, un formato intermedio que es ejecutado por la JVM. Esto proporciona flexibilidad y portabilidad.
5. **Garbage Collection:** Java tiene un sistema de recolección de basura que gestiona automáticamente la liberación de memoria, eliminando la necesidad de que los programadores se ocupen explícitamente de la gestión de la memoria.
6. **Seguridad:** Java incorpora características de seguridad, como la gestión de permisos y el sandboxing, que ayudan a prevenir la ejecución de código malicioso.
7. **Librerías Estándar:** Java cuenta con una amplia biblioteca estándar (Java Standard Edition API) que proporciona funcionalidades para tareas comunes como entrada/salida, manipulación de cadenas, redes, gráficos, entre otros.
8. **Desarrollo Web:** Java es utilizado extensamente en el desarrollo web a través de tecnologías como JavaServer Pages (JSP) y servlets. También es la base de muchos marcos de trabajo web, como Spring.
9. **Desarrollo Móvil:** Android, uno de los sistemas operativos móviles más populares, utiliza Java como uno de sus lenguajes principales.

10. **Comunidad Activa:** Java tiene una comunidad de desarrolladores activa y una amplia cantidad de recursos en línea, lo que facilita el aprendizaje y la resolución de problemas.

❖ Estándares de codificación en JAVA

Los estándares de codificación en Java son pautas y prácticas recomendadas que ayudan a mantener un código limpio, legible y consistente. Estas normas facilitan la colaboración entre desarrolladores y mejoran la mantenibilidad del código a lo largo del tiempo. Algunos de los estándares de codificación comunes son:

- **Convenciones de Nombres:** Se siguen convenciones de nomenclatura, como camelCase para nombres de variables y métodos, UPPER_CASE para constantes y CamelCase para nombres de clases.

```
int miVariable;
void miMetodo() {

}
```

- **Indentación:** Se utiliza una indentación consistente para mejorar la legibilidad del código. La indentación típica es de cuatro espacios.

```
if (condicion) {
   // código indentado
}
```

- **Longitud de Línea:** Se sugiere limitar la longitud de línea a un número específico de caracteres (por ejemplo, 80 o 120) para facilitar la lectura del código.

- **Comentarios:** Se utilizan comentarios para explicar secciones de código complejas o proporcionar documentación del código. Se evita el uso excesivo de comentarios obvios.

```
// Esto es un comentario explicativo
int variable = 42; // Comentario en línea
```

- **Uso de Espacios en Blanco:** Se utilizan espacios en blanco de manera consistente para mejorar la legibilidad del código.

  ```
  int x = 5;
  int y = 10;
  ```

- **Manejo de Excepciones:** Se siguen prácticas adecuadas para el manejo de excepciones, como evitar el uso excesivo de bloques catch y manejar las excepciones de manera específica en lugar de genérica.

  ```
  try {
     // código que puede lanzar excepciones
  } catch (TipoDeExcepcion1 e) {
     // manejo de la excepción TipoDeExcepcion1
  } catch (TipoDeExcepcion2 e) {
     // manejo de la excepción TipoDeExcepcion2
  }
  ```

- **Usar Generics y Tipos Parametrizados:** Se fomenta el uso de generics para escribir código más genérico y reutilizable.

  ```
  List<String> lista = new ArrayList<>();
  ```

- **Evitar Uso de Variables Globales:** Se evita el uso excesivo de variables globales y se favorece la encapsulación utilizando modificadores de acceso.

  ```
  private int variableLocal;
  ```

❖ Nomenclatura para nombramiento de variables y constantes

- ✓ Variables
  - ➢ **Camel Case:** Para nombres de variables, se utiliza el estilo camel case, donde la primera letra de la palabra se inicia con minúscula y las primeras letras de las palabras subsiguientes se inician con mayúscula.

    ```
    int miVariable;
    String nombreCompleto;
    ```

- **Significado Descriptivo:** Los nombres de variables deben ser descriptivos y reflejar el propósito de la variable.

  ```
  int edad;
  String nombreUsuario;
  ```

✓ Constantes

- **Mayúsculas con Guiones Bajos:** Los nombres de constantes se escriben en mayúsculas con guiones bajos para separar palabras.

  ```
  final int LONGITUD_MAXIMA = 100;
  final double PI = 3.14159;
  ```

- **Palabras en Mayúsculas:** Las constantes a menudo se escriben completamente en mayúsculas para indicar que son valores constantes que no deben cambiar durante la ejecución del programa.

  ```
  final int VALOR_MAXIMO = 100;
  ```

- Ejemplos combinados

  ```
  // Variables
  int edadUsuario;
  String nombreCompleto;

  // Constantes
  final int LONGITUD_MAXIMA = 100;
  final double PI = 3.14159;
  ```

❖ Nomenclatura para nombramientos de clases, métodos, comentarios

✓ Clases

- **Camel Case con Inicial Mayúscula:** El nombre de una clase debe seguir el estilo camel case, pero con la primera letra en mayúscula.

  ```
  public class MiClase {
     // código de la clase
  }
  ```

- ✓ Métodos

  - ➢ **Camel Case con Inicial Minúscula:** Los nombres de métodos siguen el estilo camel case, comenzando con minúscula.

    ```
    public void miMetodo() {
       // código del método
    }
    ```

  - ➢ **Significado Descriptivo:** Al igual que con las variables, los nombres de los métodos deben ser descriptivos y reflejar el propósito del método.

    ```
    public void calcularTotal() {
       // código para calcular el total
    }
    ```

- ✓ Comentarios

  - ➢ **Comentarios en Línea:** Los comentarios en línea se inician con // y se utilizan para explicar brevemente líneas de código.

    ```
    int edad; // Variable que almacena la edad del usuario
    ```

  - ➢ **Comentarios de Bloque:** Los comentarios de bloque, entre /* y */, se utilizan para describir secciones de código más extensas o proporcionar información adicional.

    ```
    /*
    * Este es un comentario de bloque que explica
    * varias líneas de código.
    */
    ```

  - ➢ **Javadoc para Documentación:** Para documentar clases y métodos, se utiliza Javadoc. Los comentarios Javadoc comienzan con /** y proporcionan información sobre el propósito, parámetros y valores de retorno de un método, así como detalles sobre la clase.

```
/**
 * Clase que representa a un estudiante.
 */
public class Estudiante {
    /**
     * Calcula el promedio de las calificaciones del estudiante.
     *
     * @param calificaciones Array de calificaciones del
estudiante.
     * @return Promedio de las calificaciones.
     */
    public double calcularPromedio(int[] calificaciones) {
        // código para calcular el promedio
    }
}
```

Estas son convenciones estándar en Java que ayudan a mantener la consistencia y la legibilidad en el código. Al seguir estas pautas, se facilita la comprensión del código para otros desarrolladores que puedan trabajar en el proyecto.