GA7-220501096-AA1-EV01 definir estándares de codificación de acuerdo con plataforma de desarrollo elegida

Presentado por:

Eliana Patricia Elles Torrecilla

Leidy Mariana González Marín

Ficha:

2758339

Presentado a:

Aharon Alexander Aguas Navarro

Modalidad Virtual

Tecnología En Análisis y Desarrollo De Software

Servicio Nacional De Aprendizaje – SENA

2024

Introducción

El desarrollo del software AHNA requiere adoptar estándares de codificación que aseguren coherencia, legibilidad y mantenibilidad del código. Este informe técnico define dichos estándares para el desarrollo de AHNA, un sistema diseñado para gestionar ingresos y salidas de empleados en empresas, basado en la programación orientada a objetos (POO).

Objetivo

Establecer un conjunto claro de estándares de codificación que guíen a los desarrolladores en la creación del software AHNA. Estos estándares incluyen el nombramiento de variables, la declaración de clases y métodos, y otras convenciones esenciales para asegurar un código limpio, entendible y fácil de mantener.

Estándares de Codificación

Nombramiento de Variables:

- **Variables Locales**:
  - **Estilo**: camelCase.
  - **Descripción**: Los nombres deben ser descriptivos y reflejar el propósito de la variable.
  - **Ejemplos**: employeeName, totalSalary.
- **Constantes**:
  - **Estilo**: MAYÚSCULAS con guiones bajos para separar las palabras.
  - **Ejemplos**: MAX_EMPLOYEES, DEFAULT_TIMEOUT.
- **Variables de Instancia**:
  - **Estilo**: camelCase.
  - **Prefijo**: this. para variables de instancia.
  - **Ejemplos**: this.employeeList, this.totalRevenue.
- **Variables de Clase (Estáticas)**:
  - **Estilo**: camelCase.
  - **Prefijo**: static para indicar que son variables de clase.
  - **Ejemplos**: static companyName, static maxCapacity.

Declaración de Clases

- **Nombres de Clases**:
  - **Estilo**: PascalCase.
  - **Descripción**: Los nombres deben ser sustantivos y reflejar la función de la clase.
  - **Ejemplos**: EmployeeManager, AttendanceRecord.
- **Estructura y Organización**:
  - **Archivo Individual**: Cada clase debe estar en su propio archivo.
  - **Nombre de Archivo**: El nombre del archivo debe coincidir con el nombre de la clase.
  - **Ejemplo**: La clase EmployeeManager debe estar en EmployeeManager.java.

- **Documentación**:
    - **Comentarios de Documentación**: Incluir Javadoc para describir la funcionalidad de la clase.
    - **Ejemplo**:

javaCopy code

```
/**
 * Clase para gestionar empleados.
 */
public class EmployeeManager {
    // Código de la clase
}
```

Declaración de Métodos

- **Nombres de Métodos**:
    - **Estilo**: camelCase.
    - **Descripción**: Los nombres deben ser verbos y reflejar la acción que realiza el método.
    - **Ejemplos**: calculateSalary, getEmployeeDetails.
- **Longitud y Complejidad**:
    - **Descripción**: Los métodos deben ser cortos y realizar una única tarea.
    - **Norma**: Evitar métodos con más de 20-30 líneas de código.
- **Documentación**:
    - **Comentarios de Documentación**: Incluir Javadoc para describir el propósito del método, los parámetros y el valor de retorno.

Convenciones de Formato

- **Indentación**:
    - **Espacios**: Utilizar 4 espacios por nivel de indentación.
    - **Tabulaciones**: No utilizar tabulaciones (tabs).
- **Longitud de Línea**:

        - **Norma**: No exceder los 80 caracteres por línea.
    - **Llaves**:
        - **Posición**: Las llaves de apertura deben estar en la misma línea que la declaración.
        - **Ejemplo**:

javaCopy code

```
if (condition) {
   // Código
} else {
   // Código
}
```

Manejo de Excepciones

- **Captura de Excepciones**:
    - **Bloques try-catch**: Utilizar bloques try-catch para manejar excepciones.
    - **Mensajes de Error**: Incluir mensajes de error claros y útiles.
    - **Ejemplo**:

javaCopy code

```
try {
   // Código que puede lanzar una excepción
} catch (IOException e) {
   System.err.println("Error al leer el archivo: " + e.getMessage());
}
```

- **Propagación de Excepciones**:
    - **Throws**: Propagar las excepciones utilizando la palabra clave throws cuando sea apropiado.
    - **Ejemplo**:

javaCopy code

```
public void readFile(String fileName) throws IOException {
   // Código
```

}

Pruebas y Documentación

- **Pruebas Unitarias**:
    - **Requisito**: Implementar pruebas unitarias para todos los métodos públicos.
    - **Herramienta**: Utilizar JUnit para las pruebas.
- **Comentarios**:
    - **Descripción**: Utilizar comentarios dentro del código para explicar secciones complejas o importantes.
    - **Norma**: Mantener los comentarios actualizados con el código.

Estructura del Proyecto

- **Paquetes**:
    - **Organización**: Organizar las clases en paquetes lógicos.
    - **Ejemplo**:

plaintextCopy code

com.empresa.ahna

├── modelos

├── servicios

├── controladores

└── utilidades

- **Archivos de Configuración**:
    - **Ubicación**: Mantener archivos de configuración en un directorio separado.
    - **Ejemplo**:

plaintextCopy code

config/

├── application.properties

└── log4j.properties

Conclusión

La implementación de estos estándares de codificación asegura que el desarrollo del software AHNA se realice de manera estructurada y coherente. Al seguir estas normas, se mejora la legibilidad y mantenibilidad del código, se facilita la colaboración entre los desarrolladores y se garantiza la calidad del producto final. Estos estándares son fundamentales para el éxito del proyecto y para el cumplimiento de los objetivos planteados.