



PROMETEO

PROGRAMACIÓN

Buenas prácticas en programación

Estructura y organización

"Un buen programa empieza con un buen orden."

Buenas prácticas:

- Mantén una estructura clara de carpetas: `src`, `resources`, `test`, `docs`.
- Agrupa las clases por paquetes funcionales: `model`, `controller`, `view`, `utils`.
- Crea un `README.md` con instrucciones.
- Usa `.gitignore` para excluir archivos innecesarios.

Estructura y organización

Nombres descriptivos

"El código debe explicarse sólo."

Buenas prácticas:

- Usa nombres claros y significativos.
- Evita abreviaturas y letras sueltas.
- Usa *camelCase* en variables y métodos.
- Clases con inicial mayúscula (*Persona*, *CalculadoraSegura*).

Estructura y organización

Comentarios y documentación

"Comenta el porqué, no el qué."

Buenas prácticas:

- Explica por qué haces algo, no lo obvio.
- Usa Javadoc para clases y métodos.
- Mantén los comentarios actualizados.

Estructura y organización

Código limpio y legible

"Escribe código como si quien lo fuera a mantener fuera un psicópata que sabe dónde vives."

Buenas prácticas:

- Usa sangrías consistentes (4 espacios o 1 tab).
- Espacios entre operadores (`x + y`, no `x+y`).
- No abuses de líneas largas (>100 caracteres).
- Deja espacios en blanco entre bloques lógicos.

Estructura y organización

Reutilización de código

"No repitas código, reutilízalo."

Buenas prácticas:

- Extrae lógica repetida a métodos o clases.
- Usa herencia o composición cuando sea apropiado.
- Centraliza utilidades comunes (`MyScanner`, `MyException`).

Estructura y organización

Control de errores y excepciones

"Esperar lo inesperado es parte de programar."

Buenas prácticas:

- Captura excepciones específicas.
- Evita `catch (Exception e)` genéricos.
- Valida los datos antes de procesarlos.

Estructura y organización

Control de versiones (Git)

"Tu código sin control de versiones es un accidente esperando ocurrir."

Buenas prácticas:

- Commits pequeños y descriptivos.
- Usa ramas: `feature/`, `bugfix/`, `release/`.
- No subas archivos generados o configuraciones IDE.

Estructura y organización

Separación de responsabilidades

"Cada clase debe tener un solo motivo para cambiar." — Principio SOLID (S)

Buenas prácticas:

- Separa la lógica de negocio, la vista y el modelo.
- Usa patrones MVC cuando sea posible.

Estructura y organización

Pruebas y validación

"Probar tu código no es desconfianza, es profesionalidad."

Buenas prácticas:

- Usa tests unitarios (JUnit).
- Prueba casos límite y errores.
- Usa aserciones (`assertEquals`, `assertThrows`).

Estructura y organización

Recursos y dependencias

"Tu proyecto debe compilar en cualquier ordenador."

Buenas prácticas:

- Usa Maven o Gradle para dependencias.
- No incluyas `.jar` dentro del código.
- Documenta ejecución en el `README.md`.

Conclusión

"El código que escribes hoy lo mantendrá otro mañana. Haz que te dé las gracias, no los nervios."



Recordatorio final:

1. Piensa antes de programar.
2. Escribe limpio y ordenado.
3. Documenta y valida.
4. Reutiliza, no repitas.
5. Hazlo con orgullo profesional.



PROMETEO