


Buenas prácticas para un código limpio en Java


En muchas ocasiones al ingresar a un nuevo equipo de trabajo nos damos cuenta que cada equipo trabaja y se comunica diferente y esto es por diversos factores, la cuenta, la experiencia de cada uno de los integrantes y el DM, etc.


Pero hay algo con lo que en la mayoría de los equipos bien organizados cuenta y son las buenas practicas dentro del desarrollo de SW, con lo cual les quiero compartir algunas de ellas, para que puedas mejorar tu desarrollo y el de tu equipo si aún ni cuentas con ellas dentro de su día a día.


Buenas prácticas para un código limpio en Java 🍷


- 👉 Declaración de clases e interfaces empleando la nomenclatura de camel-case.
- 👉 Declaración de paquetes con todas las letras en minúsculas.
- 👉 Declaración de métodos y variables empleando la nomenclatura camel-case, con la excepción de ser la primera letra de la primera palabra en minúscula.
- 👉 Declaración de constantes con todas las en mayúsculas separando cada palabra con guiones bajos.
- 👉 Empleo de variables, métodos y clases descriptivas y significativas, en lugar de emplear comentarios en todos lados.
- 👉 Las clases y funciones deben de ser pequeñas y hacer una sola cosa, evitar duplicar código.
- 😊 Los métodos no debes de consumir parametros a diestra y siniestra, siempre tratar en enviar objetos que contengan los datos con los cuales va a trabajar el método.
- 📄 Uso de una plantilla como formato de codificación, todo el equipo debe tener acceso a dicha plantilla.
- 👉 Declarar variables con el menor nivel de acceso posible (solo en dentro del método que trabaja con la variable).
- 👉 Evitar la declaración de variables que solo se emplean una vez en el código.
- ✖ Eliminar código innecesario, obsoleto y/o comentado por completo.
- 🙄 No usar **System.out.println("")**; para la "creación" de logs, siempre usar un 😊
Framework adecuado para el logback.
- 😎 Hacer clases finales y de objetos inmutables para el uso seguro de hilos.
- 🦵 Restringir el acceso a paquetes, clases y miembros de clase cuando sea posible, empleando los modificadores de acceso **private**, **protected**, **(default)**, **public**.
- 👉 Uso de interfaces para la implementación de clases.


 **U**so correcto de los tipos de datos.


 **I**mplementar de manera correcta la sobre escritura de los métodos **equals**, **hashCode**, **toString**.

 **E**vitlar la aparición de **nullPointerException** dentro del código, devolviendo colecciones vacías o empleando excepciones cuando el resultado sea nulo (**null**).

 **S**iempre manejar las excepciones, emplear las excepciones en lugar de regresar códigos de respuesta.

 **N**o presentar en los logs información sensible.

 **L**iberar recursos siempre que no sean requeridos.

 **N**o mostrar información sensible al momento que ocurra una excepción.

 **R**eusar objetos como se recomienda en el patrón de diseño "flyweight".