# Sección 1 y 2: Clean Code y Deuda técnica

* Deuda técnica

Es la falta de calidad, de cierto aspecto de nuestro código. Es la falta de calidad en nuestro código, o falta de calidad en algún proceso de nuestro desarrollo de software.

No siempre es código, pueden ser temas de documentación, pruebas, etc, sin embargo, se paga con una refactorización.

Da como consecuencia, en costos económicos, y en otros apuntados.

Imagen que contiene interior, cuchillo

Descripción generada automáticamente

Tipos de Deudas:

* 1. Imprudente
  2. Prudente y deliberada (si no se paga a tiempo, más intereses se pagan después)
  3. Inadvertido (desconocimiento y falta de experiencia)
  4. Prudente e inadvertida (ocurre terminando el proyecto, visualizando posibles mejoras, y preguntándose si vale la pena construir algo de nuevo)

Tabla

Descripción generada automáticamente

Caer en la deuda técnica es a menudo inevitable, y es normal.

¿Cómo se paga una deuda técnica?

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Clean Code

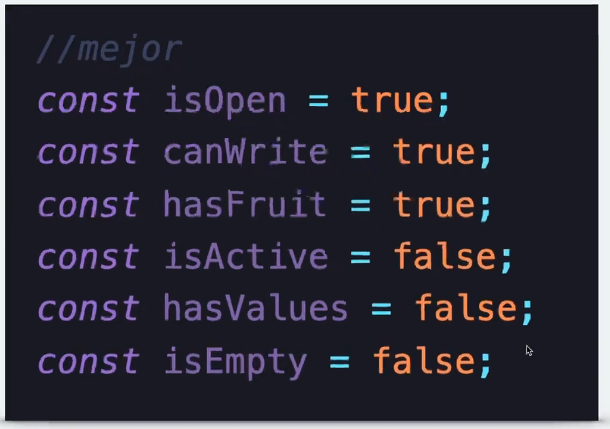
Cliente o código limpio es aquel que se ha escrito con la intención de que otra persona o tú mismo lo entienda en el futuro.

1. Definir nombres claramente en variables o constantes, sin ausencia de palabras o letras.
2. Solo ausentar en los nombres de las variables, los nombres técnicos de los lenguajes.

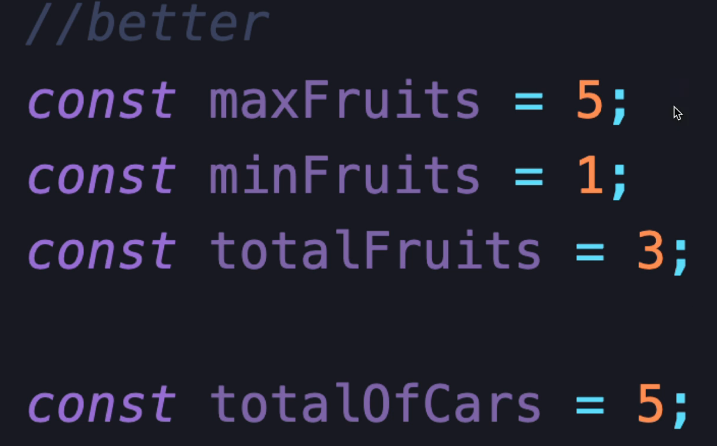
Texto

Descripción generada automáticamente

Opciones de nombres para variables booleanas



Opciones de nombres para variables numéricas



Opciones de nombres para funciones

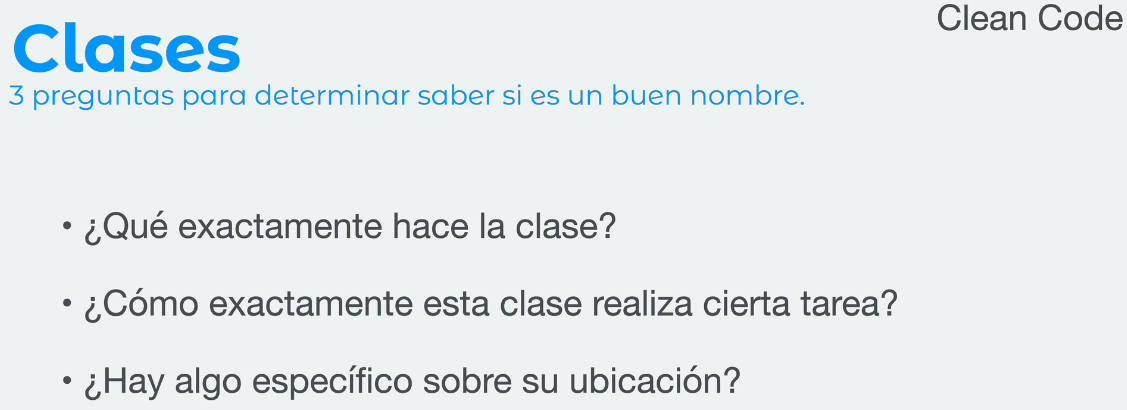
Deben ser acciones, pero describiendo lo necesario, corto y preciso.



Preguntas para poner un nombre correcto a una clase, debe ser un sustantivo, y no ser uno genérico.

“Si algo no tiene sentido, remuévalo o refactoriza”

Mas palabras no da a ser un mejor nombre



Diferencias entre argumentos y parámetros.



Se recomienda que sean solo funciones con 3 parámetros máximo, no más, porque están muy aglomeradas.

Recomendaciones para mejorar funciones con muchos parámetros.

Se recomienda igual ordenarlos de manera alfabetica.

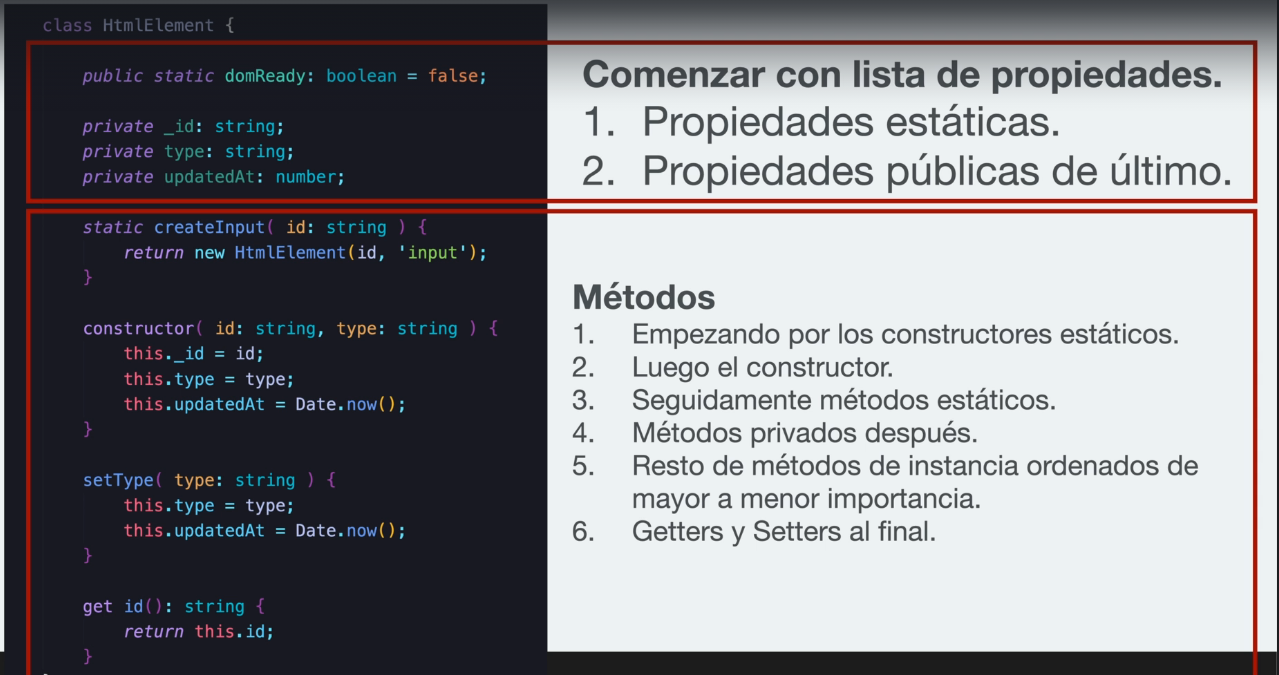


## Principio del DRY (Don’t Repeat Yourself)

* Simplemente es evitar tener duplicidad de código.
* Simplifica las pruebas.
* Ayuda a centralizar procesos.
* Aplicar el principio DRY, usualmente lleva a refactorizar.

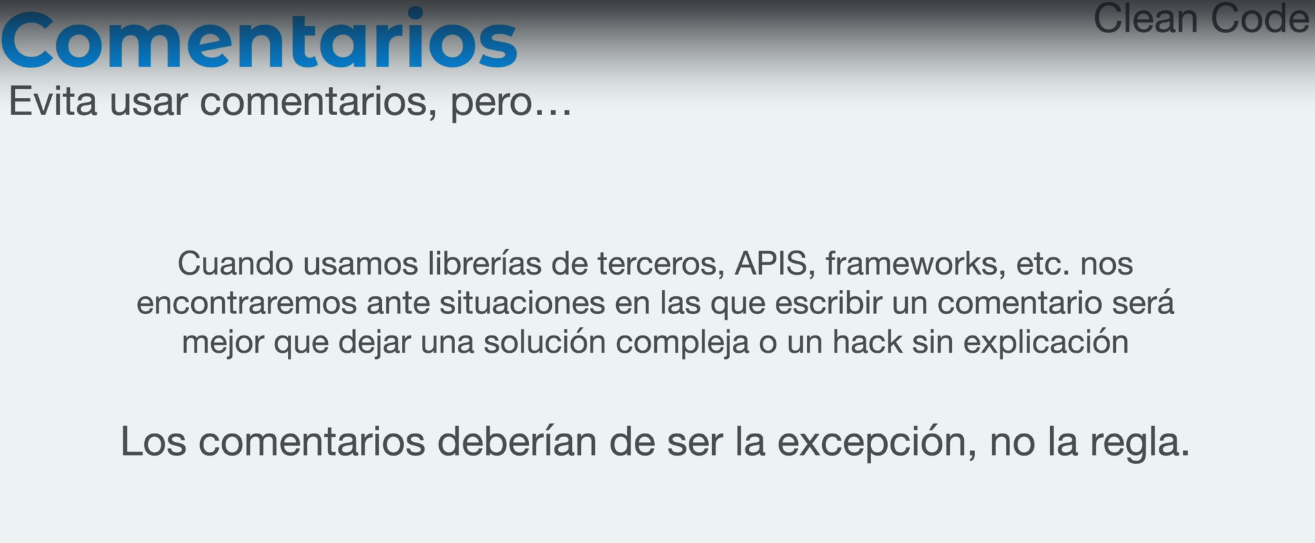
# Sección 3: Clean Code – Clases y Comentarios

Mantener una estructura de creación de clases. Se menciona una recomendación en el orden de creación que seria el siguiente en la imagen.



Evitar usar comentarios, solo en caso de ser necesario, por librerías, apis, etc.

Se debería de comentar, “¿El por qué?”, y no el ¿qué? O, el ¿cómo?



***“No comentes el código mal escrito, reescríbelo”***