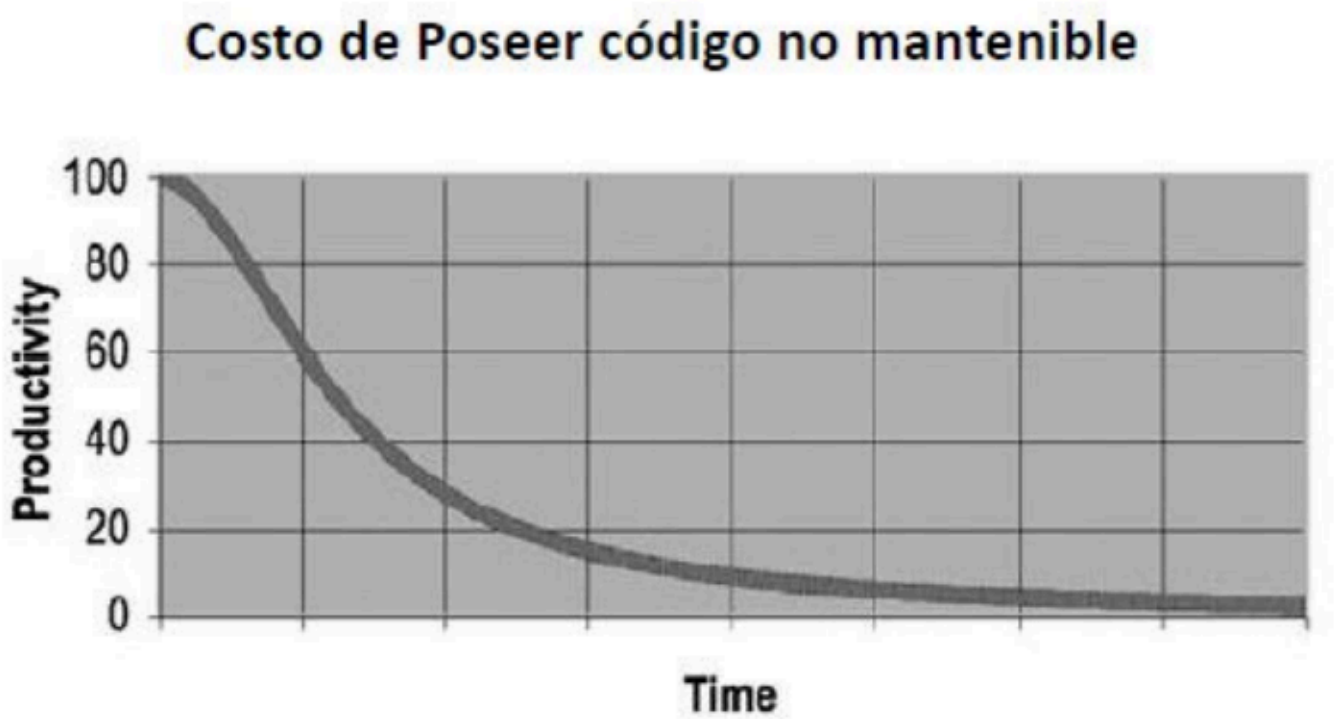
3. Clean code



El costo de poseer código mantenible es que cada vez cuesta más el seguir desarrollando. la productividad se reduce.

> Buscamos

- mantenibilidad
- simplicidad
- flexibilidad
- claridad
- legibilidad

  > Seguimos:

- La regla del Boy scout: dejamos el código más limpio de lo que lo encontramos

nombres

- significativos y pronunciables
- revelan la intención de la función (por ejemplo)
- si es necesario agrego anotaciones (los IDEs hacen que ya no lo sea)
- evito datos hardcodeados que nadie sabe de donde salieron

funciones

- Cortos
- Solo hacen una cosa: *Single Responsability Principle*
- Solo tocan un **nivel de abstracción**: no mezclo abstracciones de alto nivel con sockets por ejemplo. *Single Responsability*
- Legible de arria a abajo como un periódico

caracteristicas generales

- evita switches (*single responsabilty*)
- pocos argumentos (uno o cero)
- en vez de flags (bools por parámetro) usa polimorfismo o nuevas funciones
- no repetir código (*DRY*)
- Principio de minimo asombro: el código hace lo que se se esperaría

Comentarios

pueden ser correctos si:

- aclaran temas legales, informativo o explicactivos *de intención*
- advierten de consecuencias
- TODO
- amplia de información que no está ya descrita en el código

no deben de ser:

- redundantes
- erroneos
- de tipo diairio
- código comentado (muerto)

excepciones

- preferibles antes de mensajes de error

test unitarios

prueban una sola cosa por test (único assert por cada caso)

Test F.I.R.S.T

- Fast (para que sea prático ejecutarlo todo el tiempo)
- Independent
- Repetable (con la misma entrada siempre deberían de obtener el mismo resultado)
- Self validating (no requiere que alguien lea logs para saber si falló)
- Timely (son oportunos: antes o durante la implementación del código. TDD)