Clean Code - Checklist

Naming

- ◯ Use **descriptive** and meaningful names
  - ◯ **Variables & Properties**: Nouns or short phrases with adjectives
  - ◯ **Functions and Methods**: Verbs or short phrases with adjectives
  - ◯ **Classes**: Nouns
- ◯ Be as **specific** as necessary and possible
- ◯ Use **yes/ no** "questions" for booleans (e.g. `isValid`)
- ◯ **Avoid misleading** names
- ◯ Be **consistent** with your names (e.g. stick to `get...` instead of `fetch...`)

Comments & Formatting

- ◯ **Most comments are bad** - avoid them!
- ◯ Some good comments are **acceptable**
  - ◯ **Legal** comments
  - ◯ **Warnings**
  - ◯ **Helpful explanations** (e.g. for Regex)
  - ◯ **Todos** (don't overdo it though)
- ◯ Use vertical formatting:
  - ◯ Keep related concepts close to each other (**vertical density**)
  - ◯ Add spacing / distance (e.g. blank linkes) between concepts that are not directly related (**vertical distance**)
  - ◯ Write code **top to bottom**: Called functions should come below calling functions (if possible)
- ◯ Use **horizontal** formatting:
  - ◯ **Avoid long lines** - break them into multiple lines instead
  - ◯ Use **indentation** to express scope

Functions

- ◯ **Limit the number of parameters** your functions use - less is better!
- ◯ Consider using objects, dictionaries or arrays to **group multiple parameters into one parameter**
- ◯ Functions should be **small and do one thing**
  - ◯ Levels of abstraction inside the function body should be **one level below the level implied by the function name**
  - ◯ **Avoid mixing levels** of abstractions in functions
  - ◯ But: **Avoid redundent splitting!**
- ◯ Stay **DRY** (Don't Repeat Yourself)
- ◯ **Avoid unexpected side effects**

Control Structures & Errors

- ◯ Prefer **positive checks**
- ◯ Avoid **deep nesting**
  - ◯ Consider using "**Guard**" statements
  - ◯ Consider using **polymorphism** and **factory functions**
  - ◯ **Extract control structures** into separate functions
- ◯ Consider using **"real" errors** (with error handling) instead of "synthetic errors" built with `if` statements

Objects & Classes

- ◯ Focus on building "real objects" **or** data containers / structures
- ◯ Build **small classes** - focus on a **single responsibility** (which does **not** mean "single method"!)
- ◯ Build classes with **high cohesion**
- ◯ Follow the "**Law of Demeter**" for "real objects" (avoid `this.customer.lastPurchase.date`)
- ◯ Especially when doing OOP: Follow the SOLID principles
- ◯ Especially **SRP and OCP** will help a lot with writing clean code (= readable code)