How can it be so hard to do something so simple?

KISS and YAGNI

- Keep It Simple Stupid
- You Aren't Gonna Need It.
- Over engineering and under engineering

Good Naming

- Communicates to the reader
- Show intent
- Describe the context
- Doubles as good documentation

Bad naming

- Causes confusion
- Can lead to unnessesary comments
- Misleading
- Makes domain knowledge important

Code Smells

- Low reusability
- High cost of change
- Fixing a bug required a lot of changes several places
- Long long list
- Shotgun surgery
- Bloaters
- Primitive obsession

Solid++ Principles

- Single Responsibility
- Open/Close
- Liskov substitution
- Interface segregation
- Dependency inversion
- Balanced Abstraction
- Least astonishment (WTF)

Single Responsibility

- Readability for obvious reasons
- Testability -> small modules are easier to test
- Reusability -> Easy to read and tested, therefore easy to reuse
- Maintainability -> Easy to maintain and easier handovers

Open/Closed principle

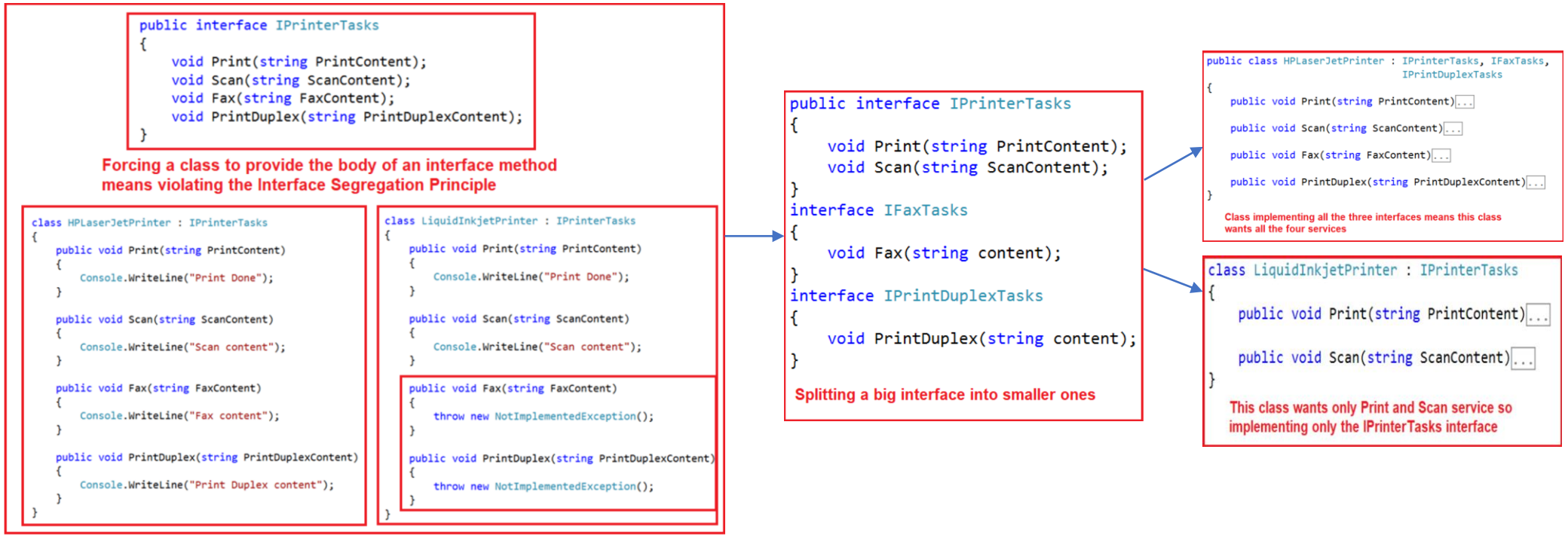
- Should be open for extension
- Should be closed for modification
- Add new features by adding new code, not modifying old code

Liskov substitution

- Substitute a class with any of its subclasses without breaking the system
- Implementations of the same interface should never give different results

Interface Segregation

- Clients should not depend on methods they don’t use
- Developers favours thin, focused interfaces



Dependency inversion

- High-level and low-level modules should depend on abstractions
- Abstractions should not depend on details. Details should depend on abstractions

Conclusions

- Naming is key
- Code smells should always be in mind. Especially when doing code reviews
- Solid principles makes code cleaner, more flexible and easier to change.
- Modules should not be thightly coupled.
- Modules should be highly cohesive to work towards the same goal.

Questions?

Sources

- https://www.c-sharpcorner.com/article/solid-principles-in-c-sharp-liskov-substitution-principle/
- https://sourcemaking.com/refactoring/smells
- https://dotnettutorials.net/lesson/interface-segregation-principle/
- https://code-maze.com/open-closed-principle/

Thank you!

vetle.horpestad

Vetle Horpestad

vetle.horpestad@bouvet.no