#### The O of SOLID

The Open/Closed Principle

Maximilian Meffert

#### The Open/Closed Principle

- Probably around since the late 1980's
  - Bertrand Meyer, *Object-Oriented Software Construction*, 1st edition. Prentice-Hall 1988, ISBN 0-13-629031-0
- What does it say?
  - "Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.
    - A module will be said to be open if it is still available for extension. For example, it should be possible to add fields to the data structures it contains, or new elements to the set of functions it performs.
    - A module will be said to be closed if [it] is available for use by other modules. This assumes that the module has been given a well-defined, stable description (the interface in the sense of information hiding)" (Bertrand Meyer)
  - "You should be able to extend the behavior of a system without having to modify that system." (Robert C. Martin)
- What does it mean?
  - Software entities should be **both** open for extension **and** closed for modification
  - Open for Extension = It is possible to extend behavior
  - Closed for Modification = It is not necessary to modify manifestation (i.e. source code, compilation, etc.)

#### The Open/Closed Principle

- Why is it a "good" thing to adhere to?
  - Change of requirements is immanent through the life cycle of most software
  - Decreases change impact, i.e. number of modules to alter
    - This number should tend to 0 since you are just add new modules (replacing others)
    - Allows new features without the need to re-engineer / re-arrange exiting ones.
  - Decreases risk of regression because of human error, i.e. developer faults
    - Since change impact decreases
  - Increases a software designs capability for adaption to change
    - ... uhm, "agile"?

Higher-order functions

```
const twice = (f, v) \Rightarrow f(f(v));

const add3 = v \Rightarrow v + 3;

twice(add3, 7); // 13
```

```
Func<func<int,int>, Func<int,int>> twice = f => x => f(f(x));

Func<int,int> plusThree = x => x + 3;

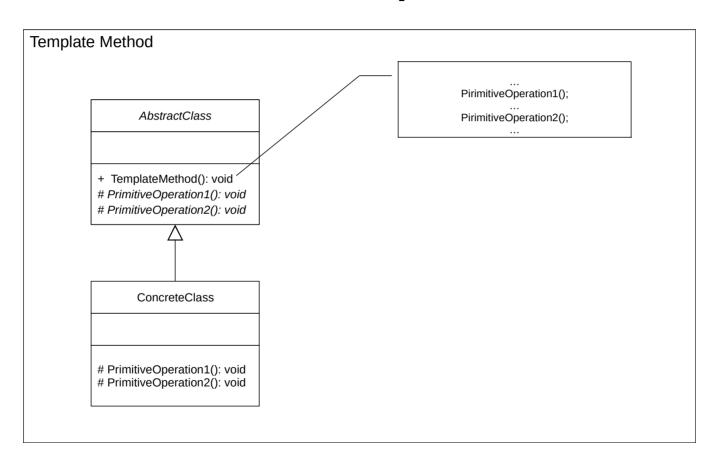
Console.WriteLine(twice(plusThree)(7)); // 13
```

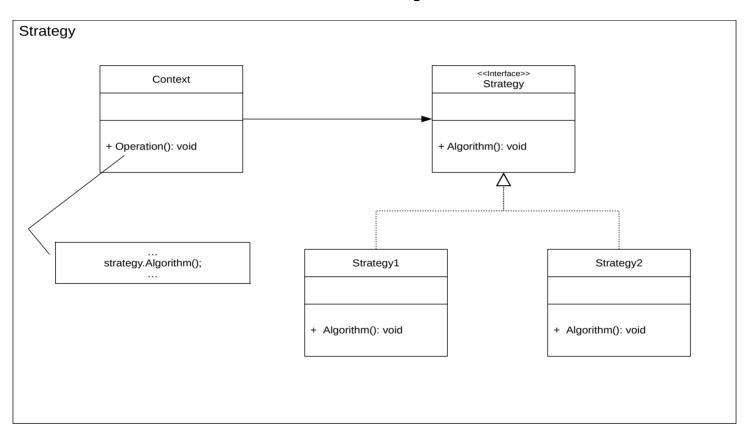
#### Higher-order functions

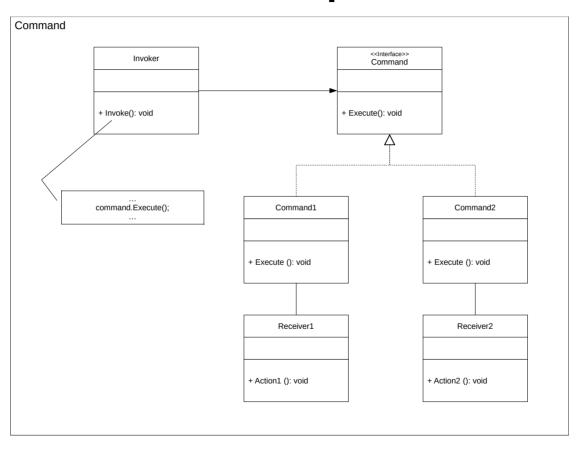
## Example A non-exclusive list of Open/Closed

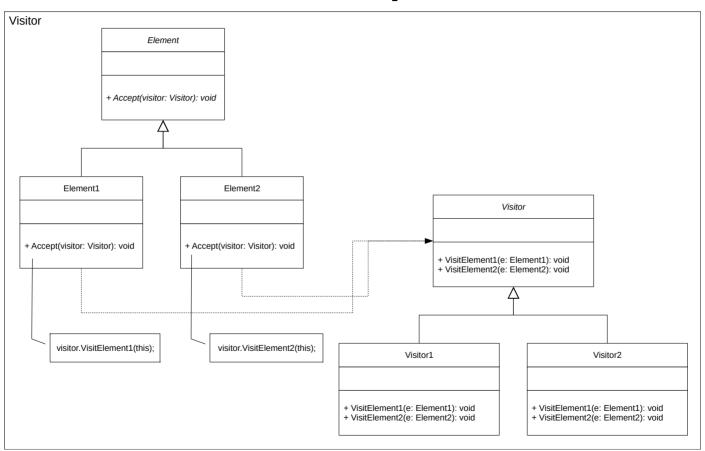
Clusive list of Open/Closed

Gang of 4 Patterns

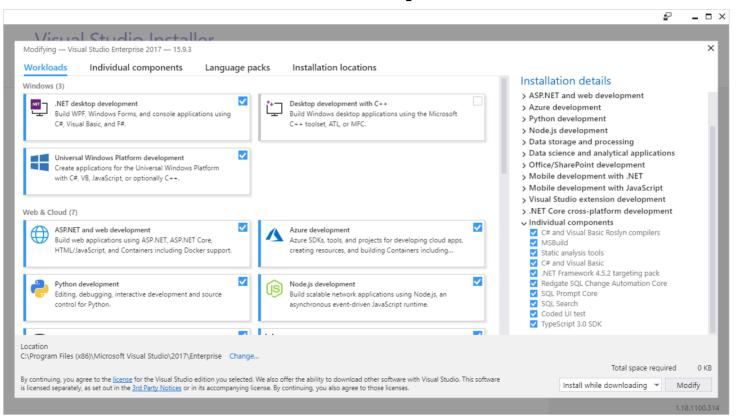




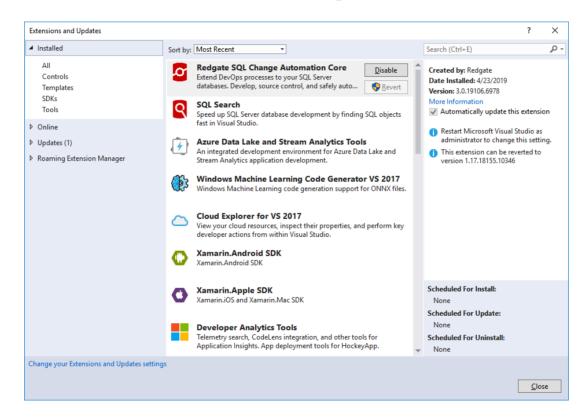




# Example Plugin Architectures



Visual Studio

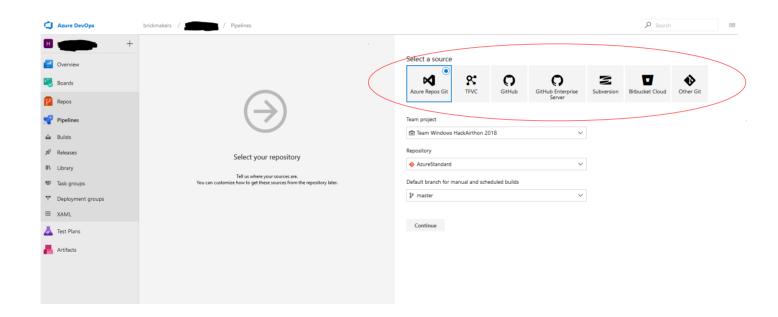


Visual Studio (Marketplace)

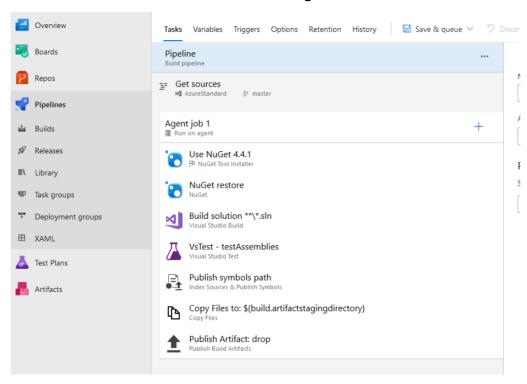
```
File Edit Selection View Go Debug Terminal Help
                                     ≝ ··· ≣ lexer.pl ×
                                                                                                                                                                                                                                                       ហ ⊞
      ✓ ENARIED
       C# for Visual Studio Code (powered by Omni...
                                                       lexer([Token|T]) --> token(Token), !, lexer(T)
       C# IDE Extensions for VSCode
      C# FixFormat 0.0.81
                                                      lexer(String, Tokens) :
       Fix format of usings / indents / braces / empt...
                                                          lexer(Tokens,Chars,[]).
       erlang 0.5.2
       Erlang language extension for Visual Studio C...
       Pierrick Gourlain
       Prolog language support for Visual Studio Co...
       Peng Lv
       VSC-Prolog 0.8.23
                                                 21 token(token(string, Value)) --> tString(Value)
                                                 22 token(token(float,Value)) --> tFloat(Value).
                                                 23 token(token(integer, Value)) --> tInteger(Value).
                                                      token(Token) --> tIdentifier(Identifier), {
                                                          reservedWord(Identifier,Value) -> Token = Value ; Token = token(identifier,Identifier)
     Notepad++ keymap 1.0.2
                                                 30 token(token("!=")) --> "!="
                                                 32 token(token(">=")) --> ">=".
                                                      token(token("=")) --> "="
                                                      token(token("*")) --> "*".
                                                      token(token(".")) --> "."
                                                     token(token(",")) --> ","
                                                                                                                                                                                                                                          ▼ ≝ 🖁 🖰 ∧ ×
                                                Cannot lint the prolog file. The Prolog executable was not found. Use the 'prolog.executablePath' setting to configure
                                                                                                                                                                                                       ⚠ Git not found. Install it or configure it using the 'git.path'
                                                                                                                                                                                                       Source: Git (Extension)
                                                                                                                                                                                                                                    Download Git Don't Show Again
⊗0 A 0
                                                                                                                                                                                                                     Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Prolog 🙂 🔔
```

Visual Studio Code

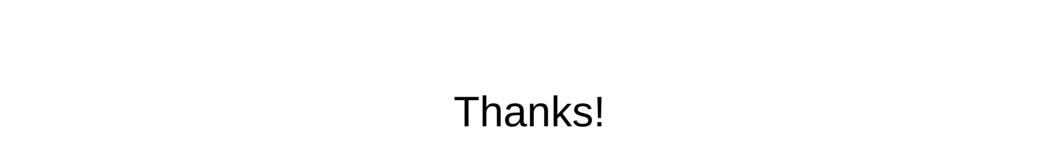
- Other (modern) IDEs also make heavy use of plugin architectures
  - IntelliJ
  - eclipse
  - Rider
  - **—** ...



Azure DevOps Build Pipelines: VCS



Azure DevOps Build Pipelines: Agent Jobs



#### References

- Bertrand Meyer. 1995. Object-oriented software construction, New York: Prentice Hall.
- Robert C. Martin. 2003. Agile Software Development, Principles, Patterns, and Practices, Prentice Hall.
- Robert C. Martin. 2014. The Open Closed Principle. (May 2014). Retrieved April 25, 2019 from http://blog.cleancoder.com/uncle-bob/2014/05/12/TheOpen ClosedPrinciple.html