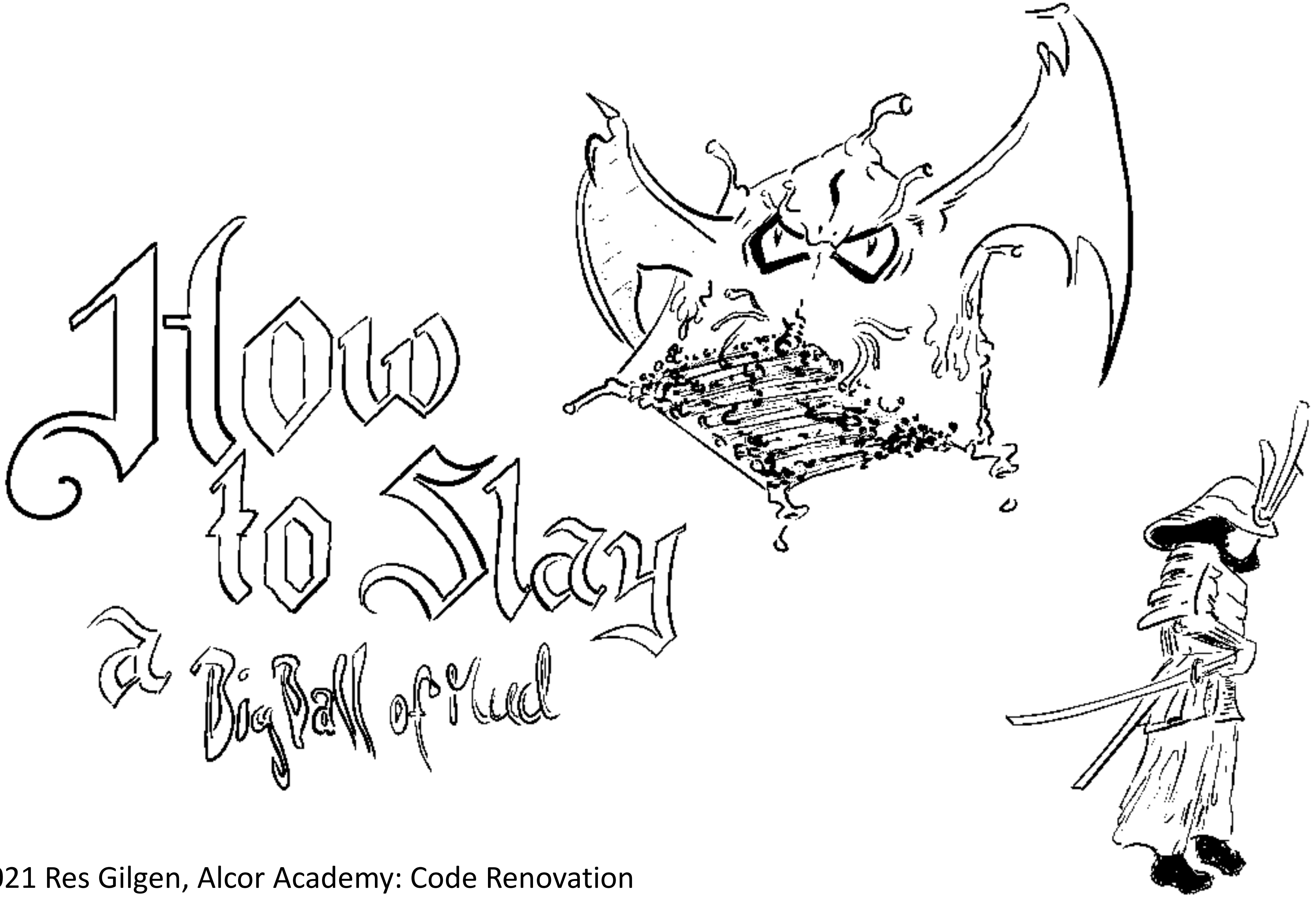


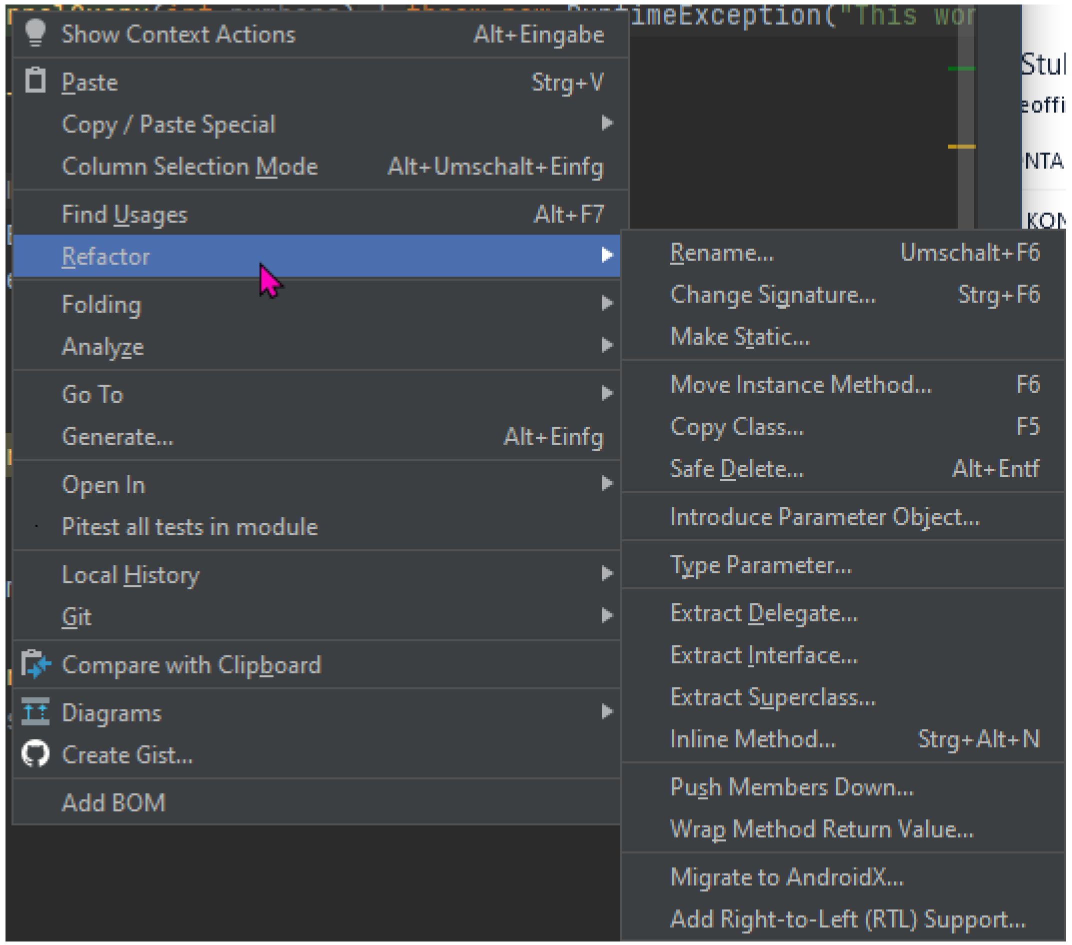
06.05.2021 Res Gilgen, Alcor Academy: Code Renovation

Checklist

- Equip yourself
  - Weapon
  - Armour
  - Skills
- Know your «Dragon»
  - Weakness
  - Desire
  - Habitat
- Prepare yourself
  - IDE-Refactorings, Seams, Pattern
  - Tests
  - Practice
- Know your Mud
  - Bugs
  - Future Projects depends on…
  - Time consuming Tasks

IDE

- Use Automated modification
  - it usually doesn't break the behaviour
- Be carefully with
  - Injections
  - reflection!



Seams

- Pre-processing seams
  - Runs before the compiler
  - Injects behavior at that stage
- Link seams
  - Linker combine code in other files
  - This allows you to link it different
- Object seams
  - Use inheritance/polymorphism
  - Change the behaviour to test a part of the productive code

- Pre-processing seams
  - Decreases code clarity
  - Not in production code
- Link seams
  - Very hard to notice
  - Make production vs test obvious
- Object seams
  - Refactor to access the relevant code to test
  - TestableClasses inherit from the original and override the disturbing functions

Pattern (Refactoring Calisthenics)

- Refactor: extract Method, extract Class, extract Interface
- Refactor: Introduce Parameter, introduce Class(encapsulate),
- Refactor: static setter (expose but: package private)

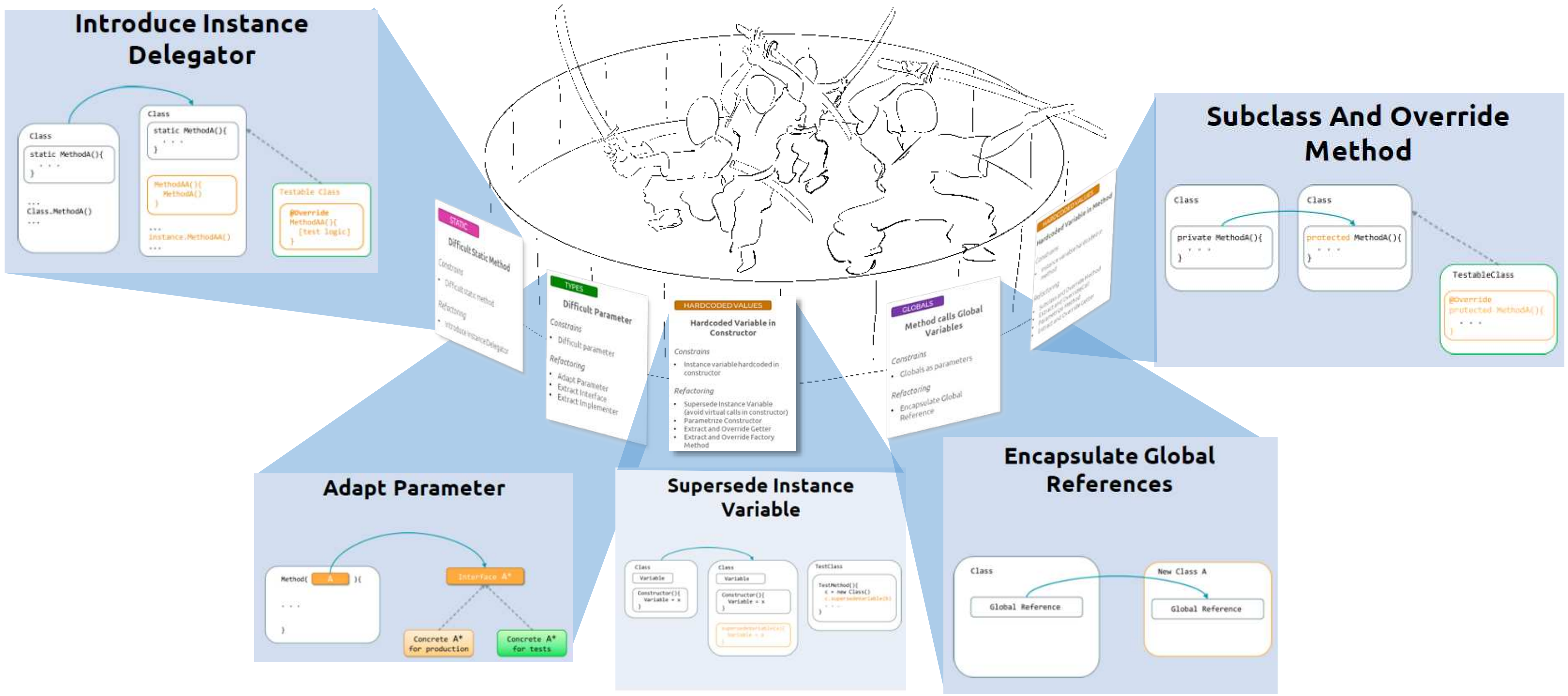
- Supersede: substitute by a testclass that inherit from the original
- Supersede: override Method in test to remove unused dependencies
    - (Peel & Slice)

→ Recognize with the Legacy Code Smells, tackle with the Cheetsheet

Tests

- Characterization Tests: Lock down the current behaviour
  - Golden Master: ApprovalTests by Llewellyn Falco
- Mutation Tests: Identify what is not yet tested
  - PIT: http://pitest.org

Practice


Introduce Instance Delegator
Class
static MethodA(){
. . .
}
...
Class.MethodA()
...
Class
static MethodA(){
. . .
}
MethodAA(){
MethodA()
}
...
instance.MethodAA()
...
Testable Class
@Override
MethodAA(){
[test logic]
}
STATIC
Difficult Static Method
Constrains
Difficult static method
Refactoring
Introduce Instance Delegator
TYPES
Difficult Parameter
Constrains
Difficult parameter
Refactoring
Adapt Parameter
Extract Interface
Extract Implementer
HARDCODED VALUES
Hardcoded Variable in Constructor
Constrains
Instance variable hardcoded in constructor
Refactoring
Supersede Instance Variable (avoid virtual calls in constructor)
Parametrize Constructor
Extract and Override Getter
Extract and Override Factory Method
GLOBALS
Method calls Global Variables
Constrains
Globals as parameters
Refactoring
Encapsulate Global Reference
HARDCODED VALUES
Hardcoded Variable in Method
Subclass And Override Method
Class
private MethodA(){
. . .
}
Class
protected MethodA(){
. . .
}
TestableClass
@Override
protected MethodA(){
. . .
}
Adapt Parameter
Method(A){
. . .
}
Interface A*
Concrete A* for production
Concrete A* for tests
Supersede Instance Variable
Class
Variable
Constructor(){
Variable = x
}
TestClass
TestMethod(){
c = new Class()
Encapsulate Global References
Class
Global Reference
New Class A
Global Reference

```
[illegible]
```

HARDCODED VALUES

Hardcoded Variable in Constructor

Constrains

- Instance variable hardcoded in constructor

Refactoring

- Supersede Instance Variable (avoid virtual calls in constructor)
- Parametrize Constructor
- Extract and Override Getter
- Extract and Override Factory Method

HARDCODED VALUES

Hardcoded Variable in Method

Constrains

- Instance variable hardcoded in method

Refactoring

- Subclass and Override Method
- Extract and Override Call
- Parametrize Method
- Extract and Override Getter

GLOBALS

Method calls Global Variables

Constrains

- Globals as parameters

Refactoring

- Encapsulate Global Reference

STATIC

Difficult Static Method

Constrains

- Difficult static method

Refactoring

- Introduce Instance Delegator

Lessons learned

- Use your IDE to refactor safely (I just thought, it is handy)
- Recognize the code smells by the cheat sheet
- Use Refactoring Callistenics to make it testable
- Start using Characterization Tests and Mutation Tests
- You can’t save your code in one day

Sources

- Alcor Academy https://alcor.academy/code-renovation