

Using ArchUnit to test your architecture

Matt Ho

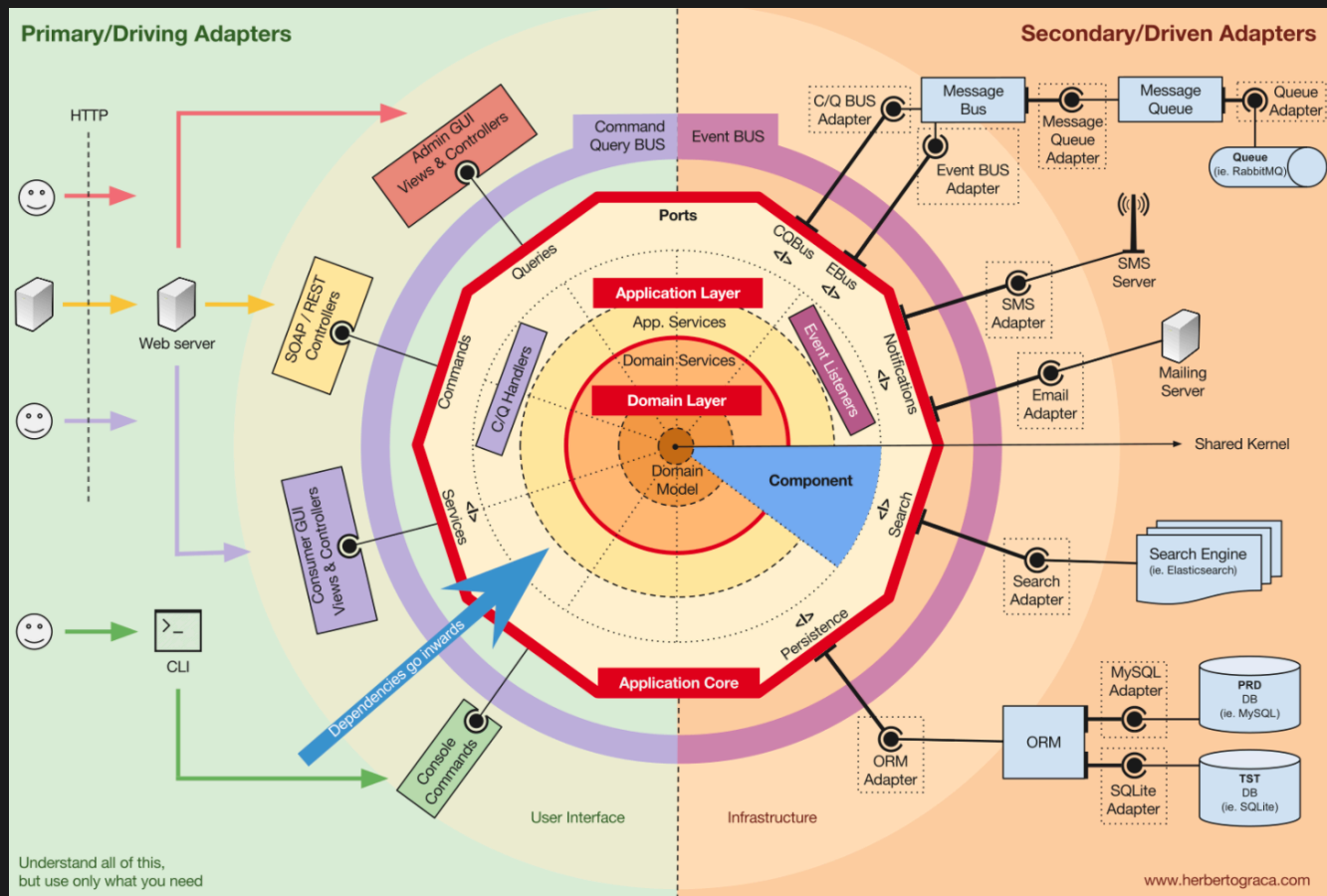


About me

- Hi, I'm Matt 🖐️
- 松凌科技 Architect / R&D
- [JCConf Speaker](#)
- methodho@gmail.com
- <https://github.com/shihyuho>



放心, 這不是在講架構



本節在分享

架構已經在那了 &&

該如何確保在持續開發/維運的過程中, 應用程式仍然遵循著架構走

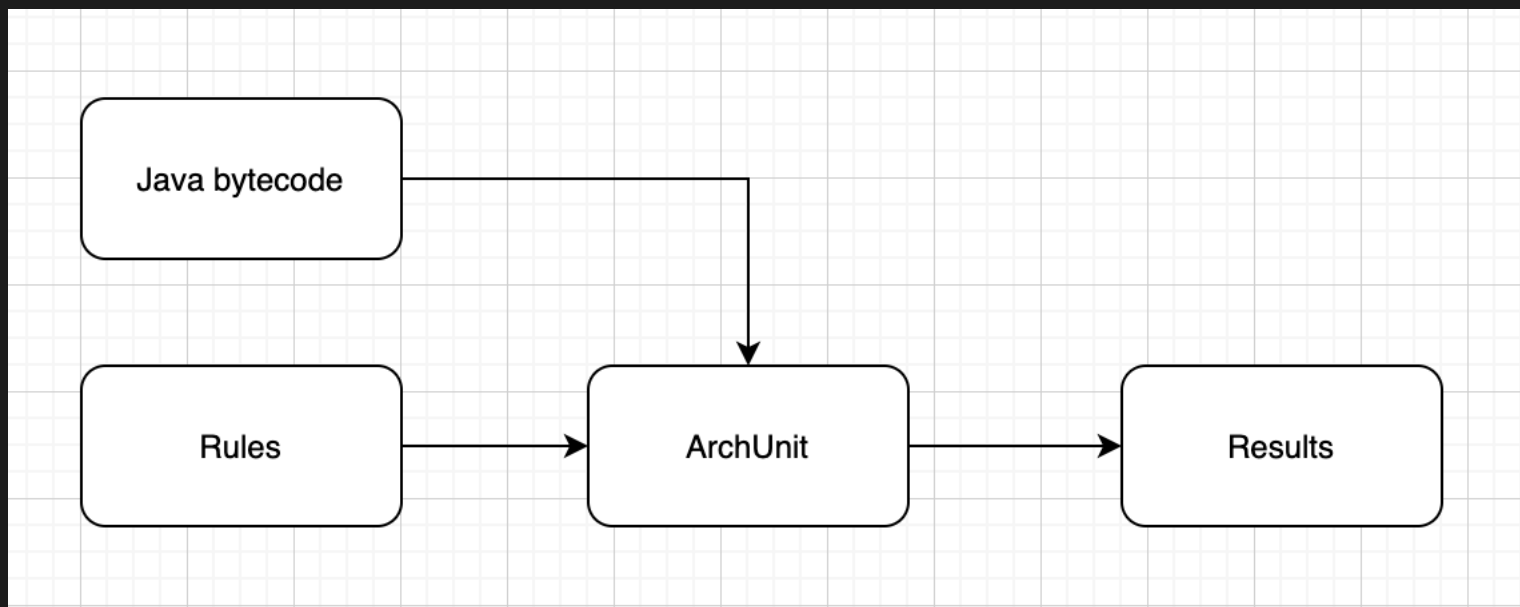


<https://www.archunit.org/>

ArchUnit Overview

- 是一個測試套件, 整合了 JUnit 4, JUnit 5 等
- 良好的 Fluent API
- 容易延伸擴充
- Apache-2.0 license

運作過程



起手式

```
@AnalyzeClasses(packagesOf = MyApplication.class, importOptions = {  
    DoNotIncludeTests.class,  
    DoNotIncludeJars.class  
})  
class _01_StartTests {  
    @ArchTest  
    static final ArchRule my_rule = ArchRuleDefinition.classes()...
```

決定從哪邊開始匯入 bytecodes

起手式

```
@AnalyzeClasses(packagesOf = MyApplication.class, importOptions = {  
    DoNotIncludeTests.class,  
    DoNotIncludeJars.class  
})  
class _01_StartTests {  
    @ArchTest  
    static final ArchRule my_rule = ArchRuleDefinition.classes()...
```

定義 Rules

起手式

```
@AnalyzeClasses(packagesOf = MyApplication.class, importOptions = {
    DoNotIncludeTests.class,
    DoNotIncludeJars.class
})
class _01_StartTests {

    @ArchTest ← ArchUnit 執行得到結果
    static final ArchRule my_rule = ArchRuleDefinition.classes()...
}
```

起手式 - Rule Pattern

ArchRuleDefinition.GIVEN_OBJECTS
 .that(). PREDICATE
 .should(). CONDITION

Class Naming

Controller 命名必須以 'Controller' 結尾

```
@Controller  
public record MyController(MyService service) {  
  
}
```



Class Naming

Controller 命名必須以 'Controller' 結尾

```
@Controller  
public record MyController(MyService service) {  
  
}
```

Class Naming


Controller 命名必須以 'Controller' 結尾

```
@Controller  
public record MyController(MyService service) {  
  
}
```

```
@ArchTest  
static ArchRule controllers_should_have_name_ending_with_controller  
= ArchRuleDefinition.classes() GivenClasses  
    .that().areAnnotatedWith(Controller.class) GivenClassesConjunction  
    .should().haveSimpleNameEndingWith(Controller.class.getSimpleName());
```

Class Naming

Data Access 命名必須以 'Dao' 結尾



```
public interface MyDao extends Repository<MyEntity, Long> {  
    }  
}
```

Class Naming

Data Access 命名必須以 'Dao' 結尾

```
public interface MyDao extends Repository<MyEntity, Long> {  
    }  
}
```

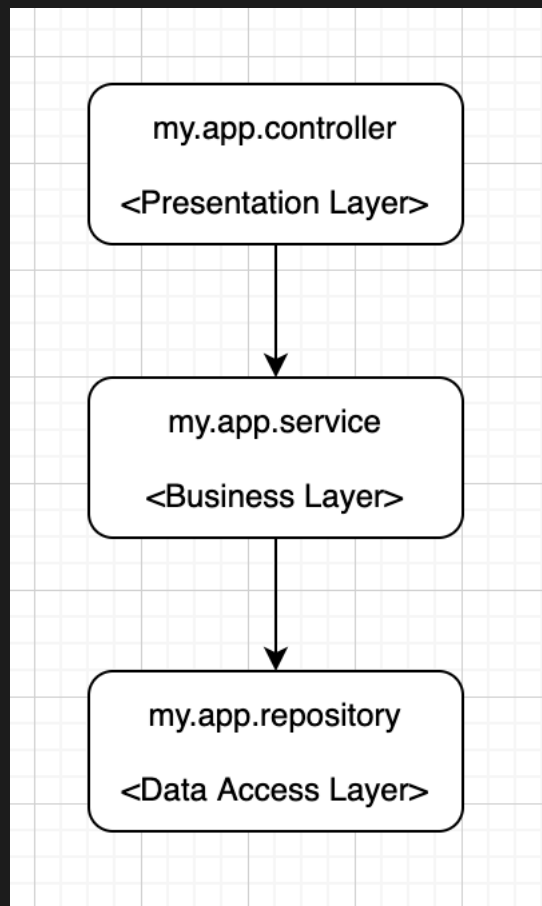

Class Naming

Data Access 命名必須以 'Dao' 結尾

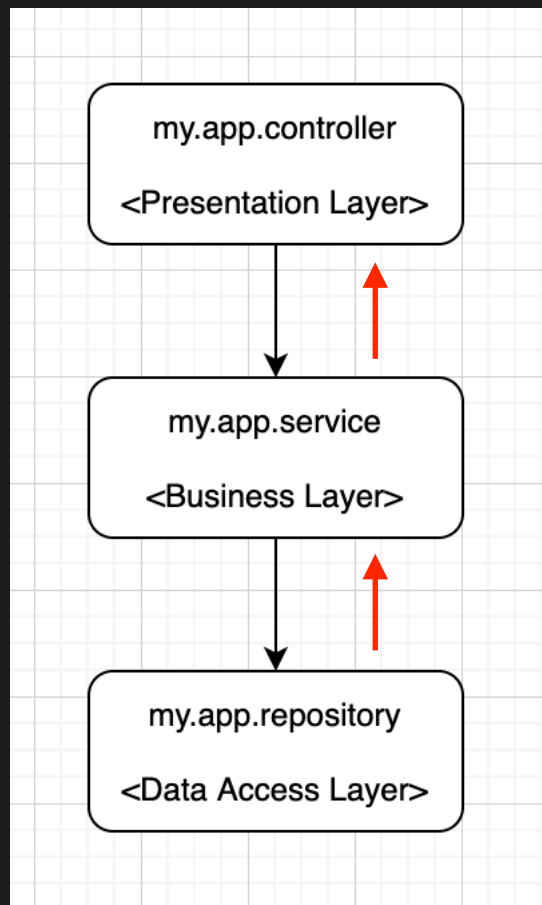
```
public interface MyDao extends Repository<MyEntity, Long> {  
    }  
}
```

```
@ArchTest  
static ArchRule repositories_should_have_name_ending_with_dao  
    = ArchRuleDefinition.classes() GivenClasses  
        .that().implement( type: Repository.class) GivenClassesConjunction  
        .should().haveSimpleNameEndingWith( suffix: "Dao");
```

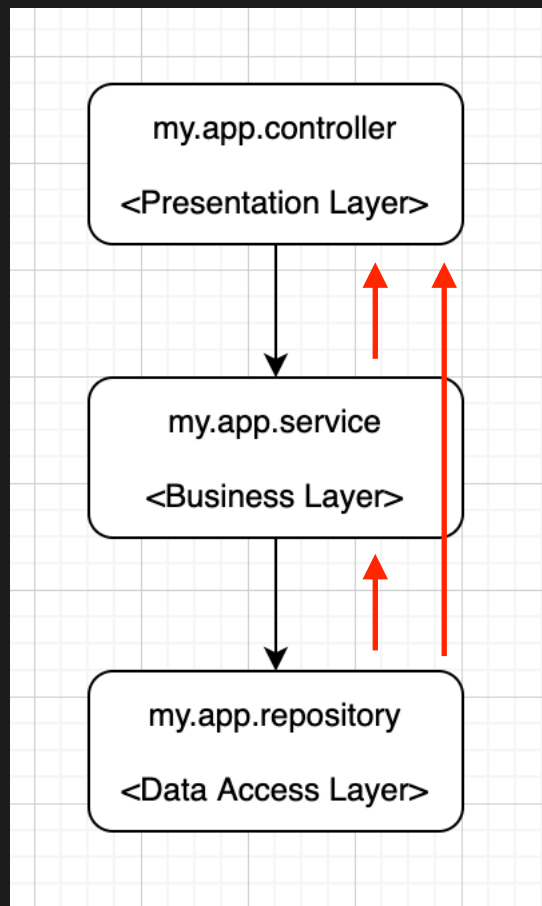
Layered



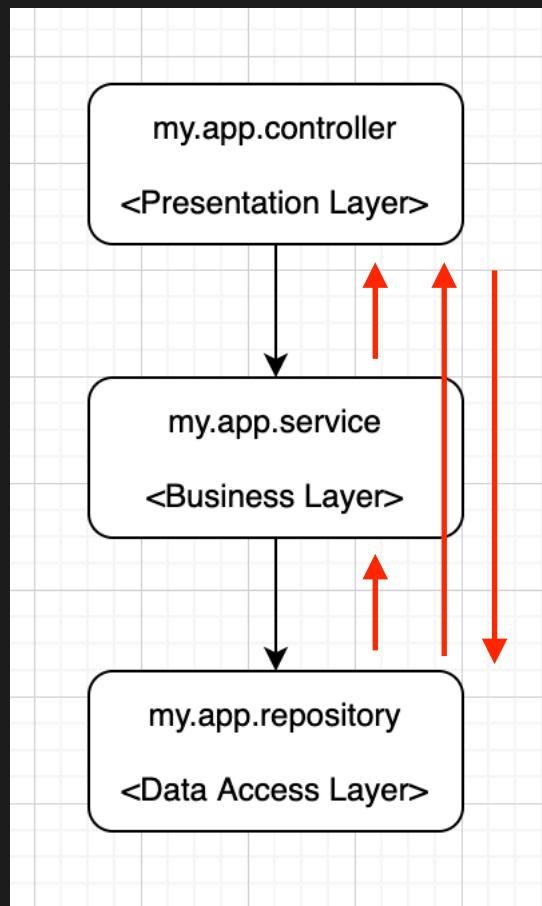
Layered



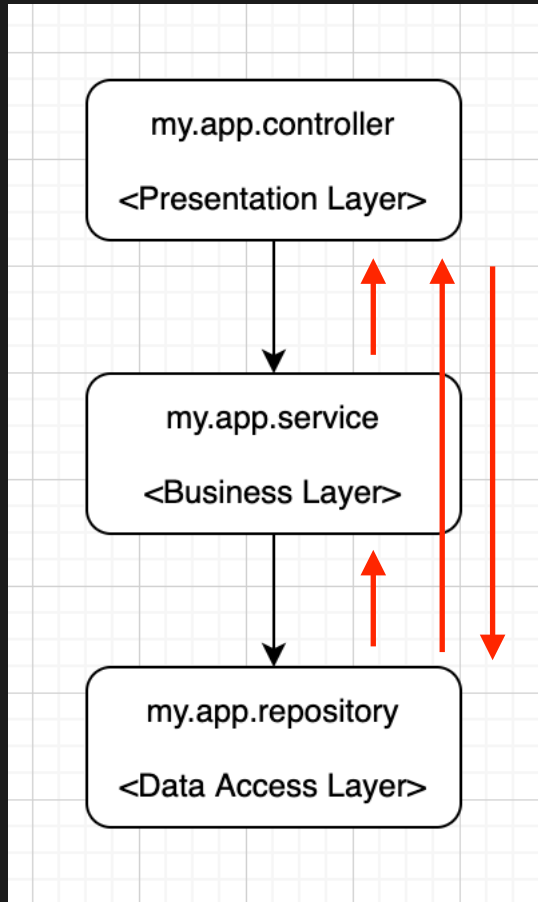
Layered



Layered



Layered



Architectures.layeredArchitecture()
.layer(). PREDICATE
.whereLayer(). CONDITION

Layered

```
@ArchTest
static ArchRule three_tier_layered = Architectures.layeredArchitecture()
    .consideringAllDependencies()
    .layer("Presentation").definedBy("..controller..")
    .layer("Business").definedBy("..service..")
    .layer("DataAccess").definedBy("..repository..")

    .whereLayer("Presentation").mayNotBeAccessedByAnyLayer()
    .whereLayer("Business").mayOnlyBeAccessedByLayers("Presentation")
    .whereLayer("DataAccess").mayOnlyBeAccessedByLayers("Business");
}
```

定義 3 層 layer

定義依賴關係

General

8.3.1. GeneralCodingRules

The class `GeneralCodingRules` contains a set of very general rules and conditions for coding. For example:

- To check that classes do not access `System.out` or `System.err`, but use logging instead.
- To check that classes do not throw generic exceptions, but use specific exceptions instead.
- To check that classes do not use `java.util.logging`, but use other libraries like Log4j, Logback, or SLF4J instead
- To check that classes do not use JodaTime, but use `java.time` instead.
- To check that classes do not use field injection, but constructor injection instead.

Freeze

- 套用在舊的系統中, 超多條違規改不動 ;-(

Freeze

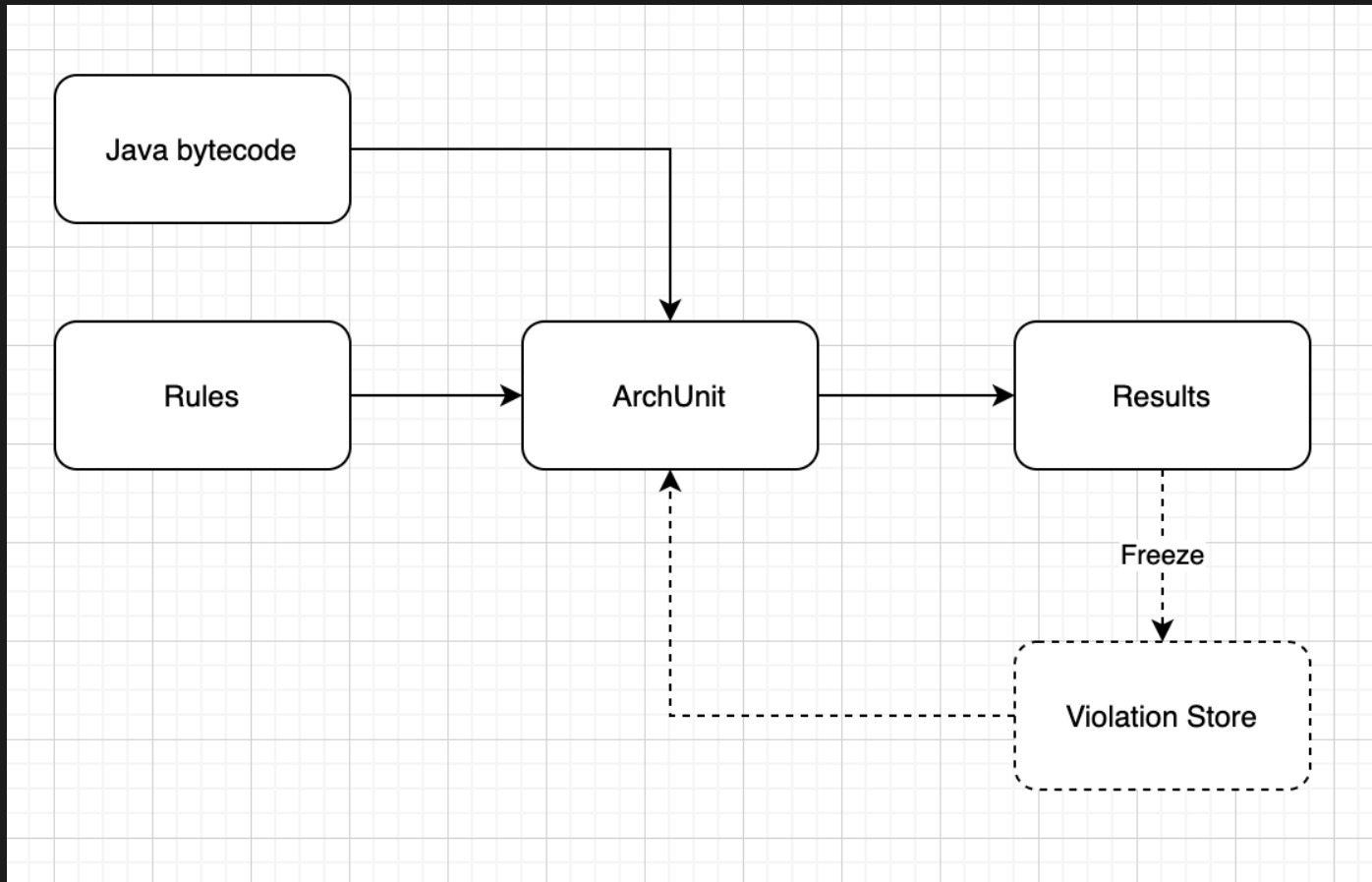
- 套用在舊的系統中, 超多條違規改不動 ;-(
- 好吧, 那就現在開始的程式才檢查 :-)

Freeze

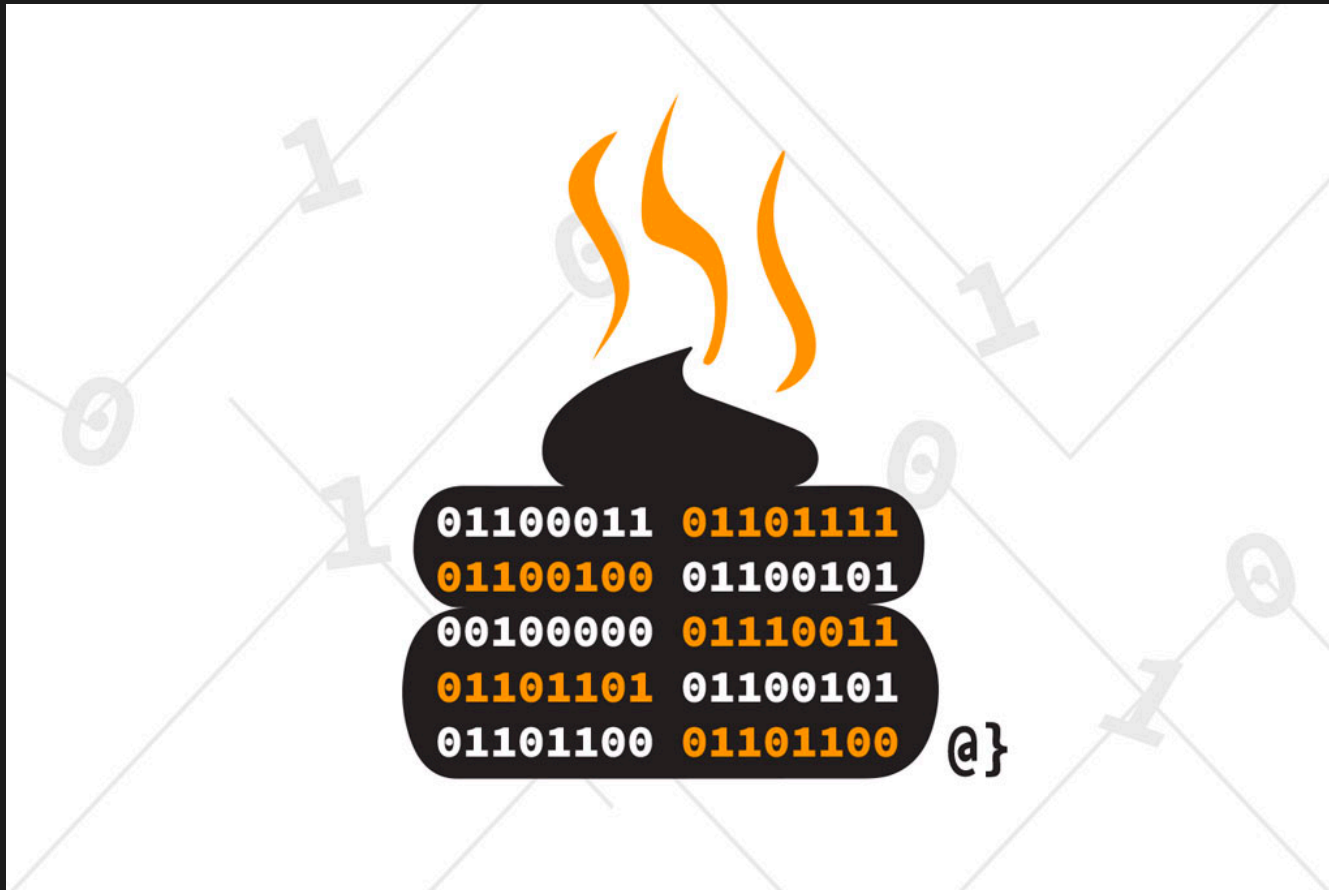
- 套用在舊的系統中, 超多條違規改不動 ;-(
- 好吧, 那就現在開始的程式才檢查 :-)

```
FreezingArchRule.freeze(  
    ArchRuleDefinition.GIVEN_OBJECTS  
        .that(). PREDICATE  
        .should(). CONDITION  
)
```

Freeze



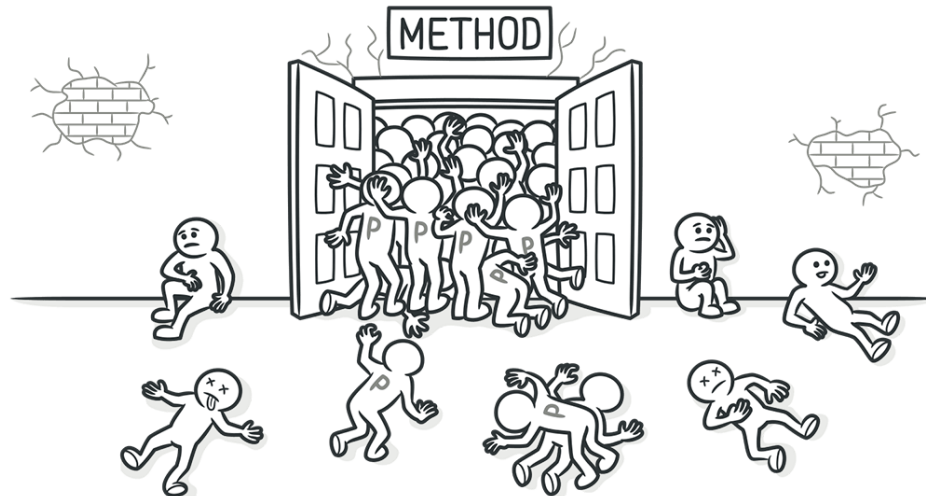
Code Smells



Long Parameter List

Signs and Symptoms

More than three or four parameters for a method.

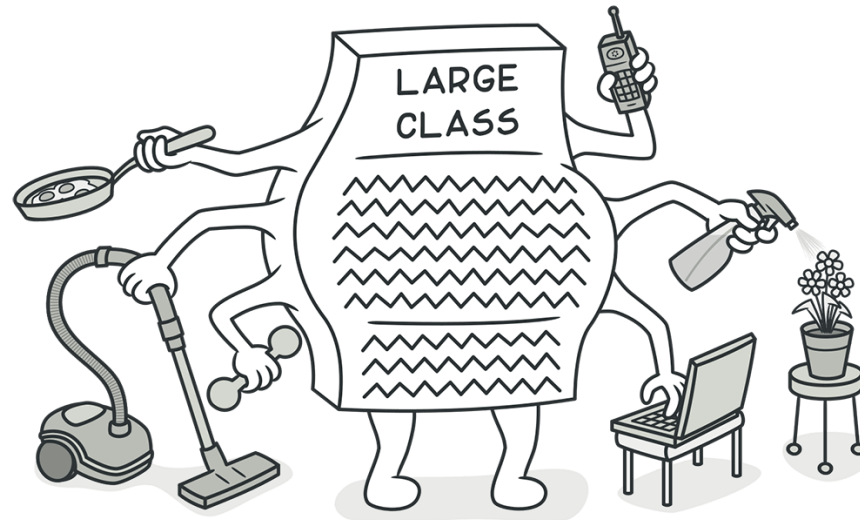


<https://refactoring.guru/smells/long-parameter-list>

Large Class

Signs and Symptoms

A class contains many fields/methods/lines of code.

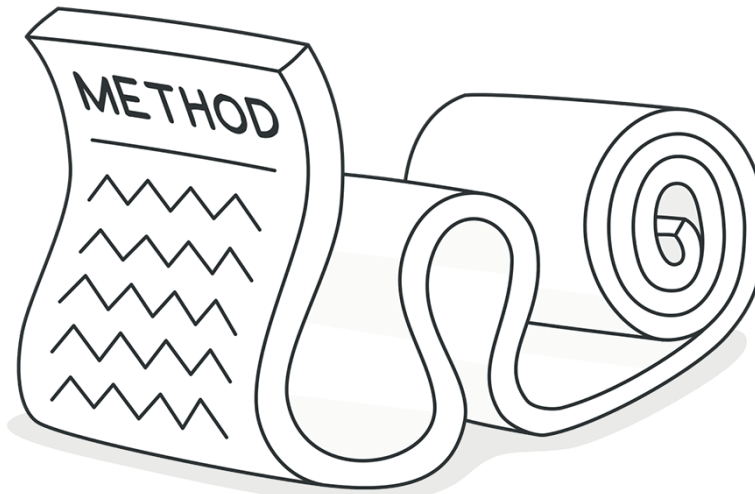


<https://refactoring.guru/smells/large-class>

Long Method

Signs and Symptoms

A method contains too many lines of code. Generally, any method longer than ten lines should make you start asking questions.



<https://refactoring.guru/smells/long-method>

Recap

- 起手式
- Naming Rules
- Layers Rules
- General Rules
- Freezing Rules
- Custom Rules - Code Smells



Thank you 🙌