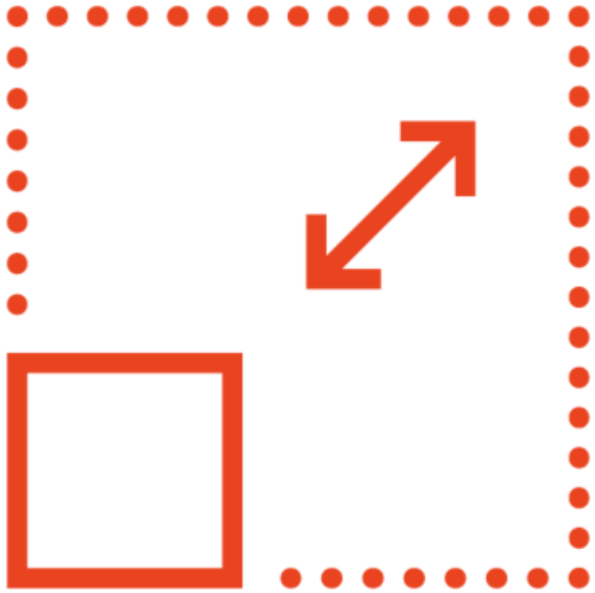
Dealing with Change Preventers

Andrejs Doronins
TEST AUTOMATION ENGINEER

Change Preventers

When code change in one place forces you to change code in many other places

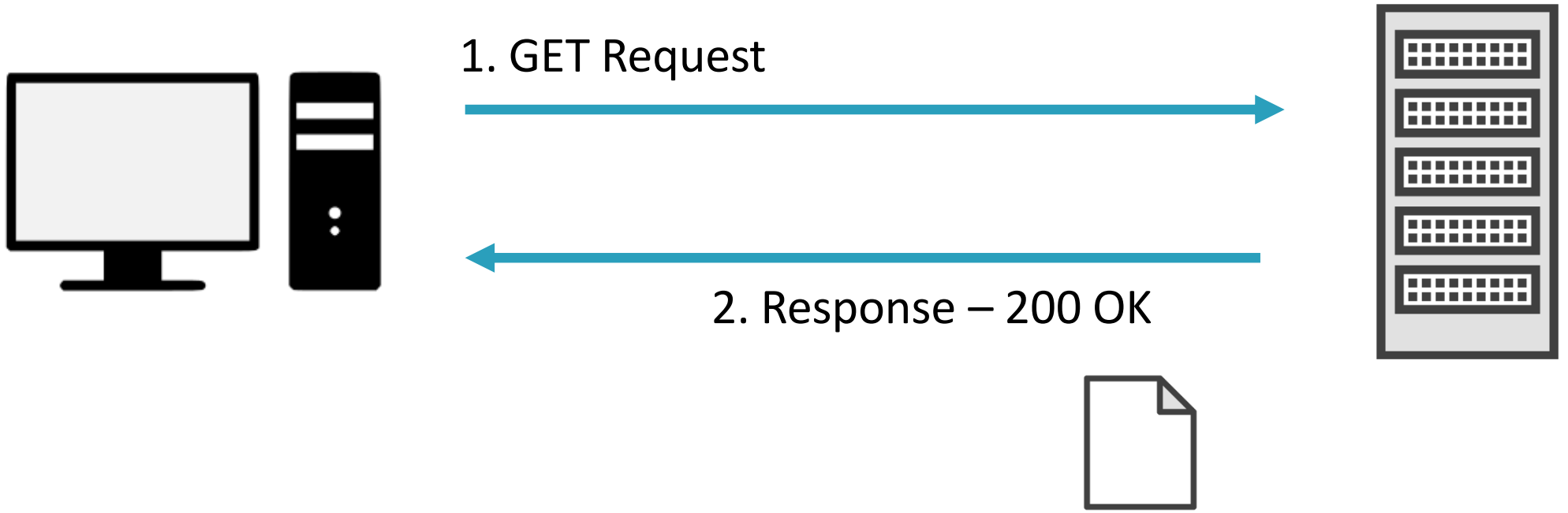
Change Preventers



Divergent change

Solution sprawl & Shotgun surgery

Parallel inheritance hierarchies


1. GET Request
2. Response – 200 OK

Divergent Change

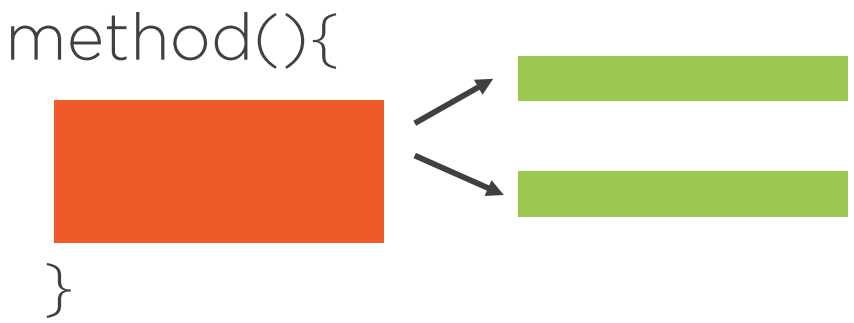
Changing several unrelated things within the same class

Divergent Change Issues

Requires more typing

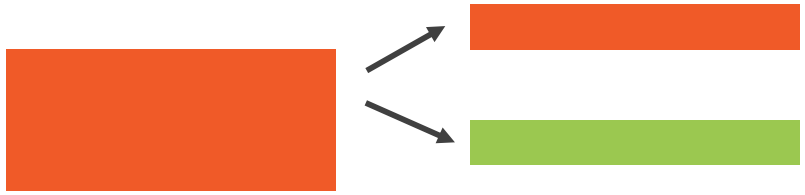
Requires additional knowledge of what to change where

How to Fix Divergent Change



Extract Method

(split a method into several smaller methods)



Extract Class

(Split into several smaller classes)

New Requirements

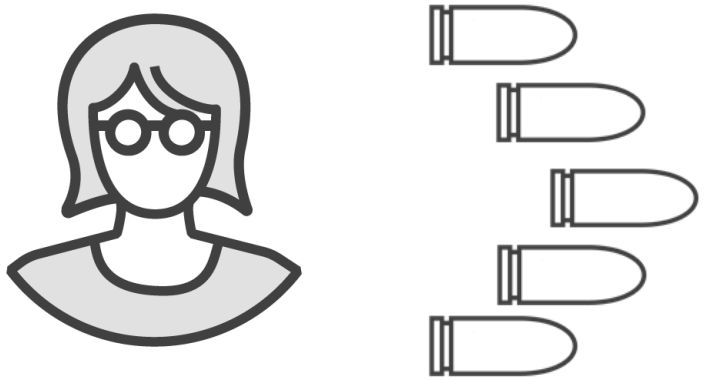
Display the currency rates in a side UI widget

Benefits Achieved

- Reduces code duplication
- Simplifies maintenance

Solution Sprawl

A solution is broken into multiple classes or places



Shotgun Surgery

An update requires additional changes in multiple classes or modules

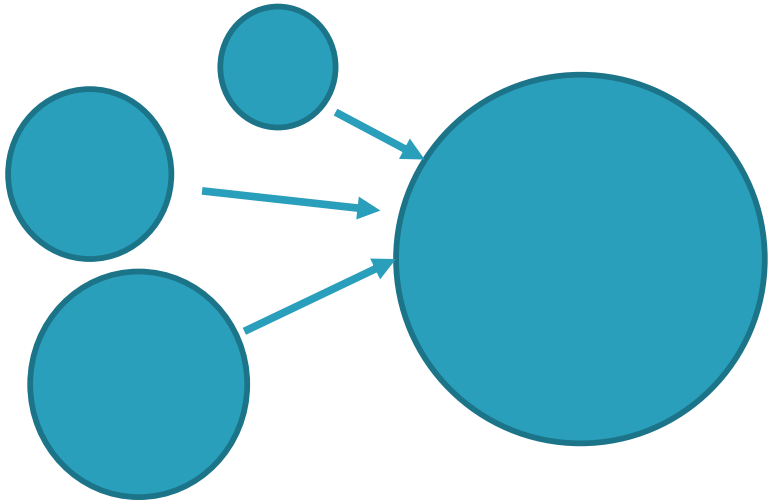
Solution Sprawl leads to Shotgun Surgery

Shotgun Surgery Issues

Difficult to remember all the interconnected places

New team members are likely to make a mistake by forgetting to update one of the places

How to Fix Solution Sprawl



Combine into one

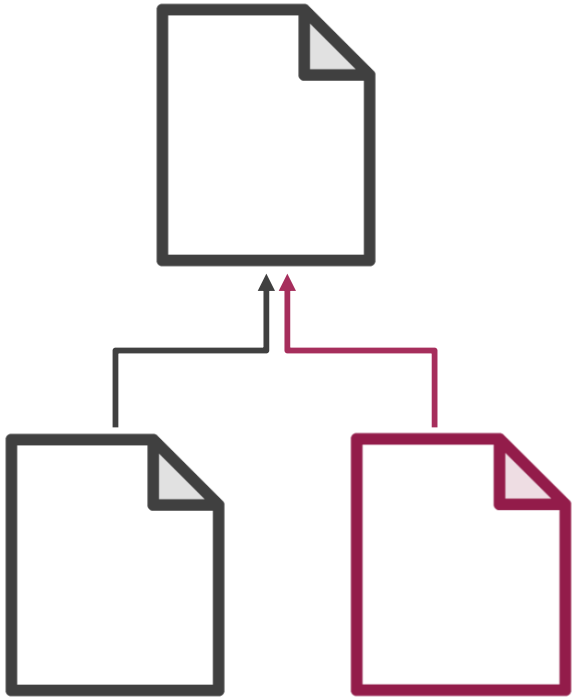
Change until you have a class with a single responsibility that encapsulates related changes

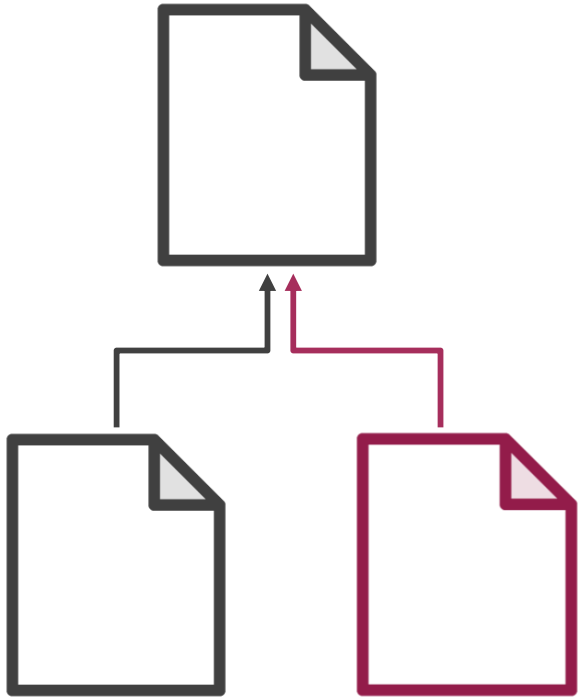
Benefits Achieved

Reduced possibility of mistakes (and bugs)

Parallel Inheritance Hierarchies

You create a subclass in one inheritance tree. This forces you to create another subclass in another tree.



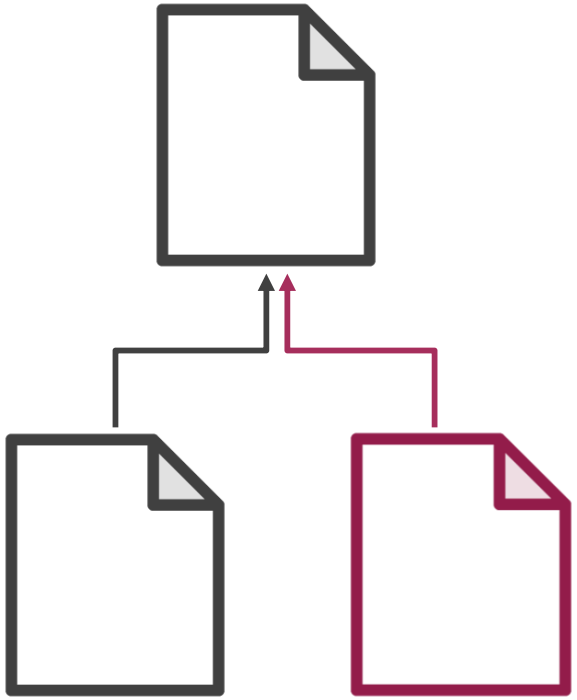


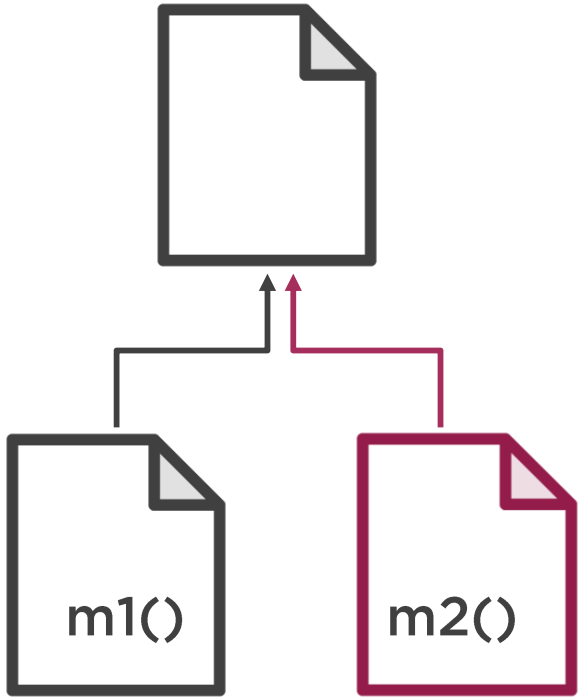
Parallel Inheritance Hierarchies are a special case of Shotgun Surgery

Parallel Inheritance Issues

Making changes becomes harder as the hierarchies grow

Sometimes you must choose the lesser evil

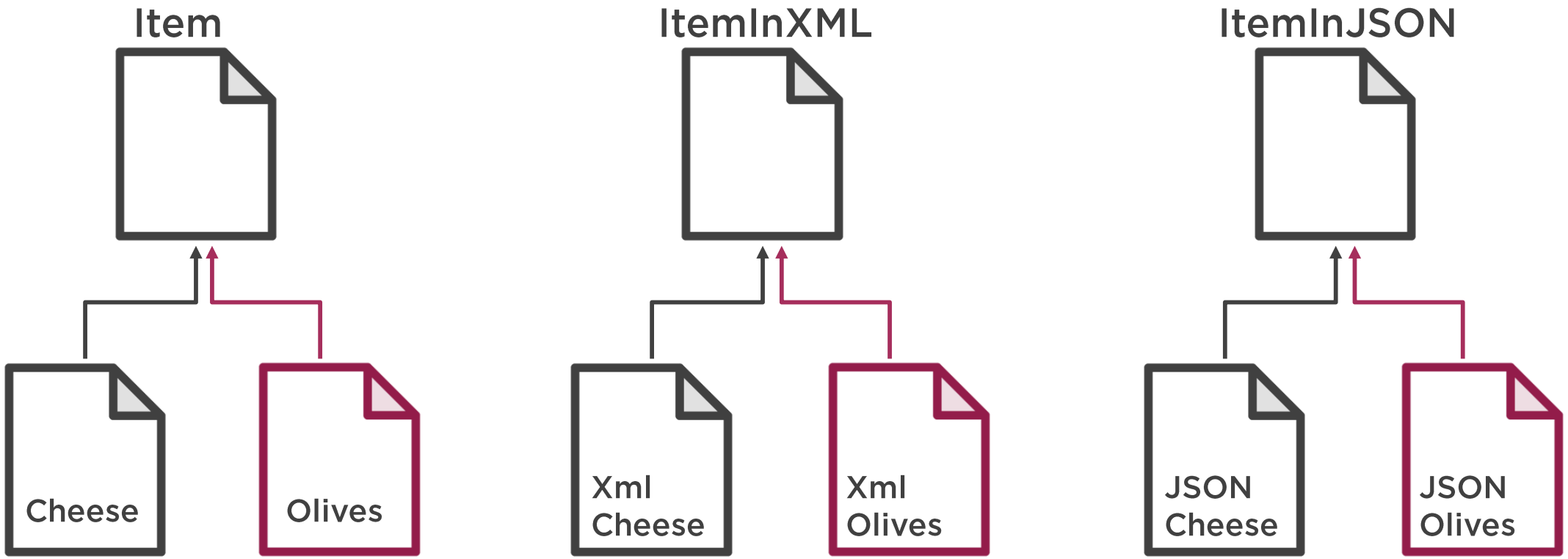



m1()
m2()

Suitable Patterns

Visitor Pattern

Bridge Pattern


Item
Cheese
Olives
ItemInXML
Xml
Cheese
Xml
Olives
ItemInJSON
JSON
Cheese
JSON
Olives

Benefits Achieved

- Reduces code duplication
- Cleaner code organization

Summary

Classes should be highly coherent to avoid Divergent Change

Solutions should be consolidated to avoid Solution Sprawls

Avoid parallel inheritance