

# Data-driven Coding and Patterns

---



**Simon Robinson**

Software Developer

@TechieSimon [www.SimonRobinson.com](http://www.SimonRobinson.com)



# Data-driven Code

Program logic depends on the data being used



# Example: Business Rules (Charging a Customer)



```
switch (orderDetails)
{
    // lots of cases
```



# C# Patterns

Special C# syntax to test if an instance satisfies a (possibly complex) condition

Allowed in:

`is` clauses

`switch` statements

`switch` expressions



# Demo: Backup App



App to backup files,  
but can also clean up  
to preserve disc space



# The Business Rules for this App:

ControlFlow-EarlyDraft

File Home Share View

« m09 Data-driven » Code » ControlFlow-EarlyDraft

Search ControlFlow-EarlyDraft

Name	Size	Type
.hg		
.vs		File folder
.vscode		File folder
bin		
obj		
.hgignore	0 KB	HGIGNORE File
ControlFlowDemo.csproj	1 KB	C# Project file
ControlFlowDemo.sln	2 KB	Visual Studio Solu...
FolderProcessor.cs	2 KB	C# Source File
Program.cs	1 KB	C# Source File
Sequences.cs		
TemporaryFile.tmp		
UserInputNumber.cs	1 KB	C# Source File
VeryBigVideo.mp4	500,719 KB	

Ignore .hg directories  
(These are part of Mercurial source control)

Delete bin and obj directories

Ignore TemporaryFile.tmp files

Ignore files > 100MB



# Demo



## Create the backup/cleanup app

- Implement the business rules using C# patterns and switch



# Demo



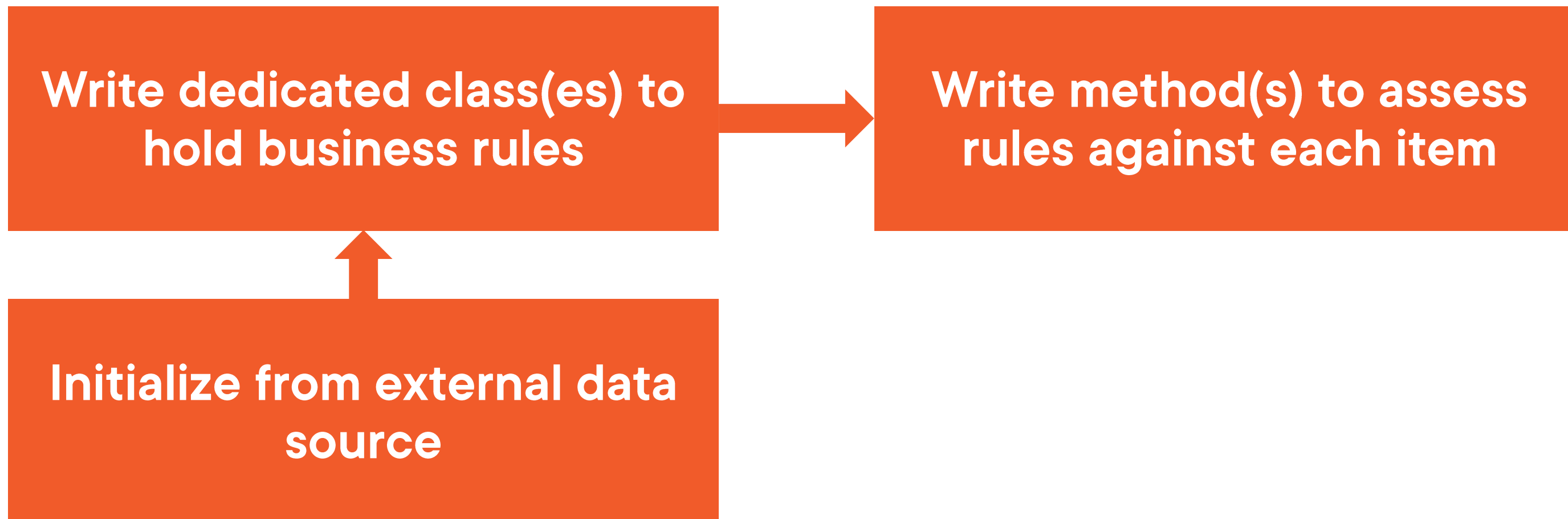
**Import rules from a file (so they aren't hard-coded)**

- To allow changing the rules without re-compiling
- This will prevent using C# patterns





# Write Data-driven Code:



Items were file system objects in the demo,  
but might equally be clients or orders etc.



# Summary



## Data-driven code

- Patterns are great, but only if the data is constant
- If data is loaded externally, you can't use patterns
- Create class(es) to hold business rules
- Write code to use whatever logic is required to test objects against business rules

