Activity 7-3: The Interface Segregation Principle

Introduction

03:24-03:32 Overview of last video (LSP)

Interfaces

14:28-15:41

- What is the problem with the `Switch –> Light` dependency?

15:41-19:35

- Interfaces have more in common with their clients, not their implementers.
- **physical coupling** and **logical coupling**. -"Inheritance is the strongest of the physical couplings but the weakest of the logical couplings".

19:35-21:00

- Name interfaces after their clients.

21:00-24:40 (skip) rant about interfaces.

Fat classes

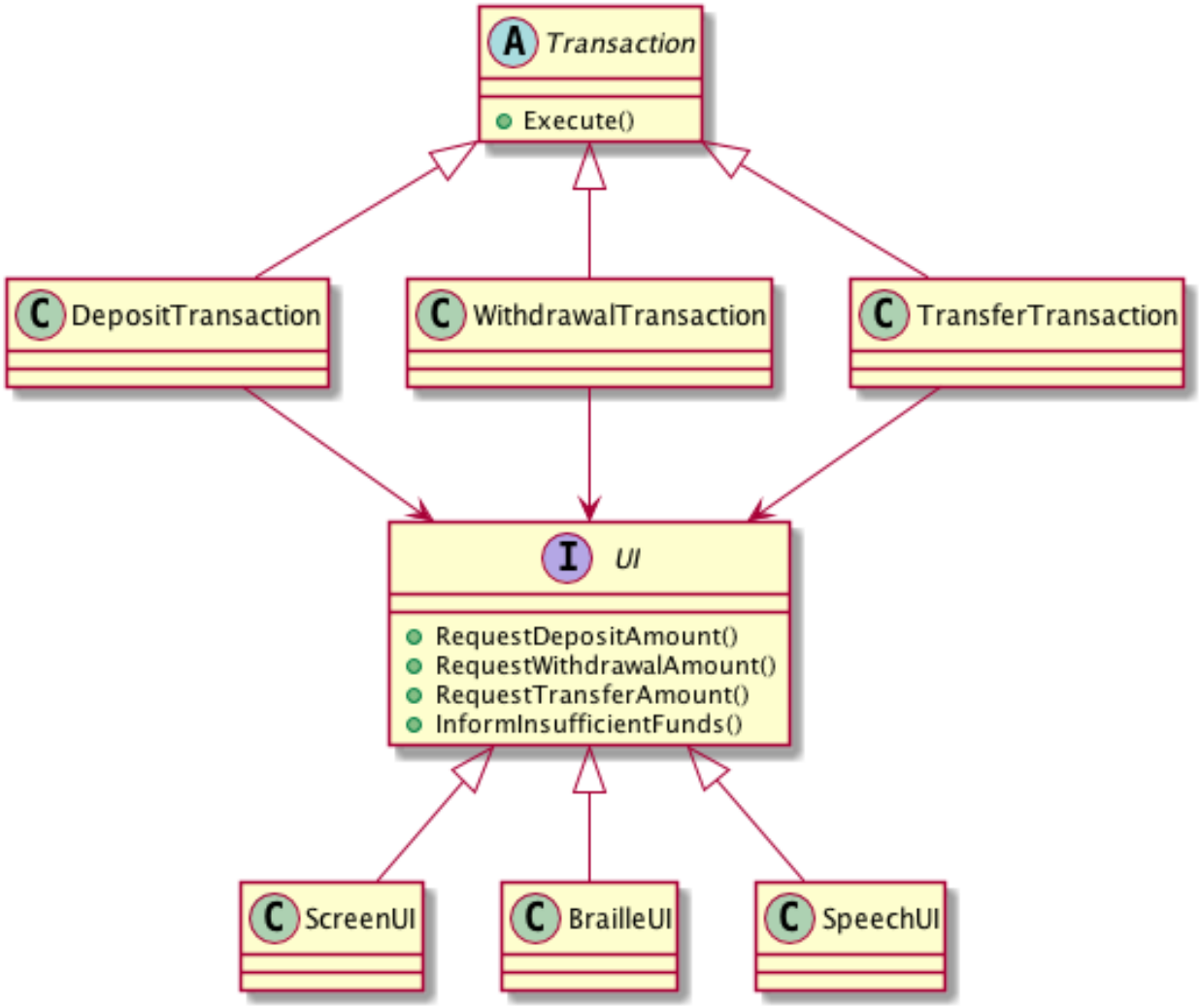
25:50-33:42 fat classes example: the job class

- Fat classes have large fan-in.

33:42-36:30 the problem with fat classes

- Fat classes can be protected from their clients by creating a separate interface for each client.

Consider the following example from an ATM application.



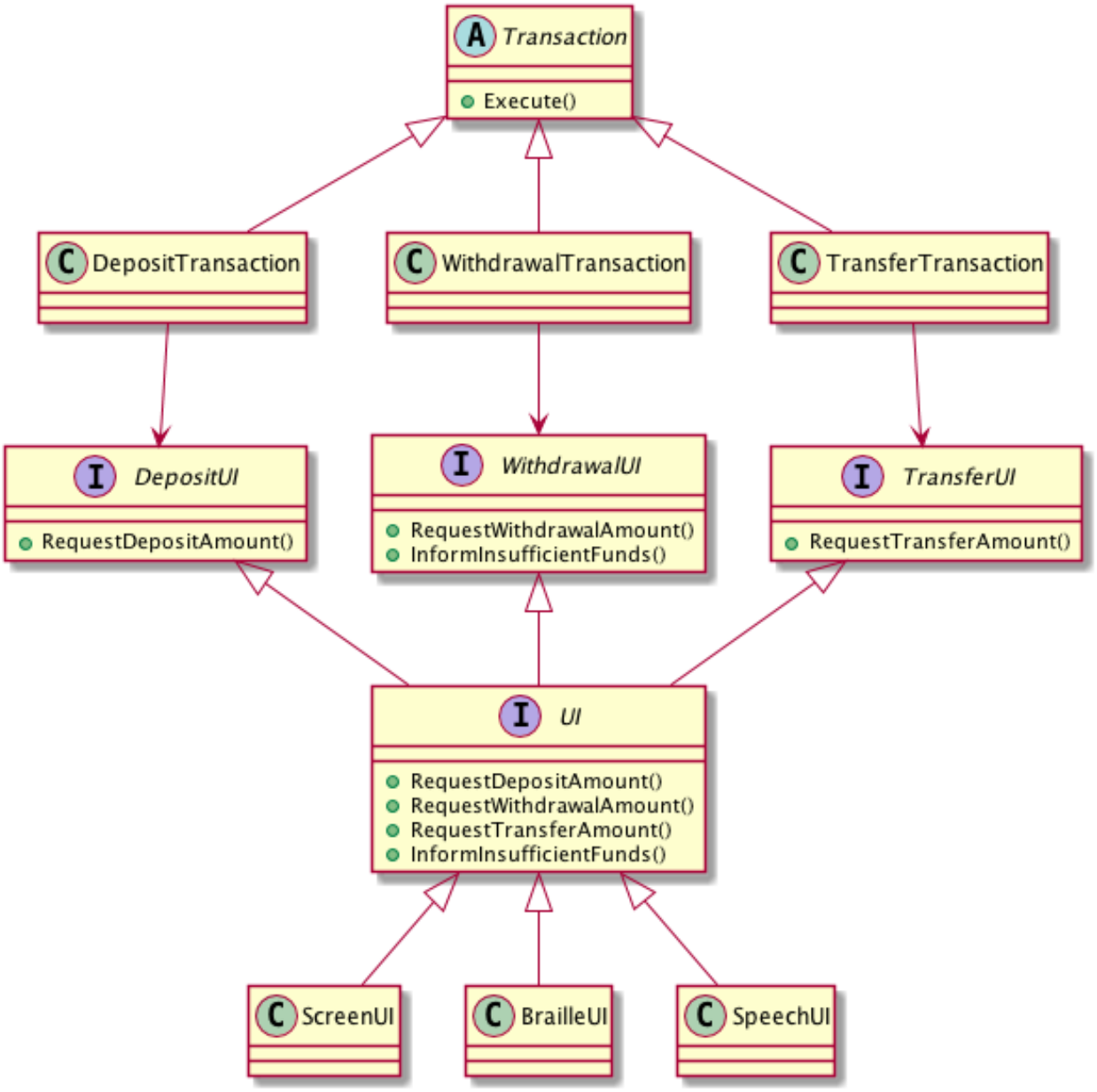
Each of the different transactions uses different parts of the interface. However as it stands they all depend on the interface.

Question: If the TransferTransaction needs a new method added to the interface, which parts of the system will need to be recompiled?

Conclusion: A module that knows about more than it needs to may end up being affected by changes to other modules that it does *not* depend on, because those modules may ask for changes to other modules that they both depend on.

Common dependencies cause transitive coupling between the modules that share the dependency.

Alternative setup, solves this problem.



The Interface Segregation Principle

45:00- 50:00 the interface segregation principle

- Interface Segregation Principle: Don't depend on things you don't need.

54:00-58:25 the need to know