



SWINBURNE
UNIVERSITY OF
TECHNOLOGY

SWE30010

Development Project 2: Design, Planning and Management

Lecture 5

Software Design



Commonwealth of Australia
Copyright Act 1968

Notice for paragraph 135ZXA (a) of the *Copyright Act 1968*

Warning

This material has been reproduced and communicated to you by or on behalf of Swinburne University of Technology under Part VB of the *Copyright Act 1968* (the *Act*).

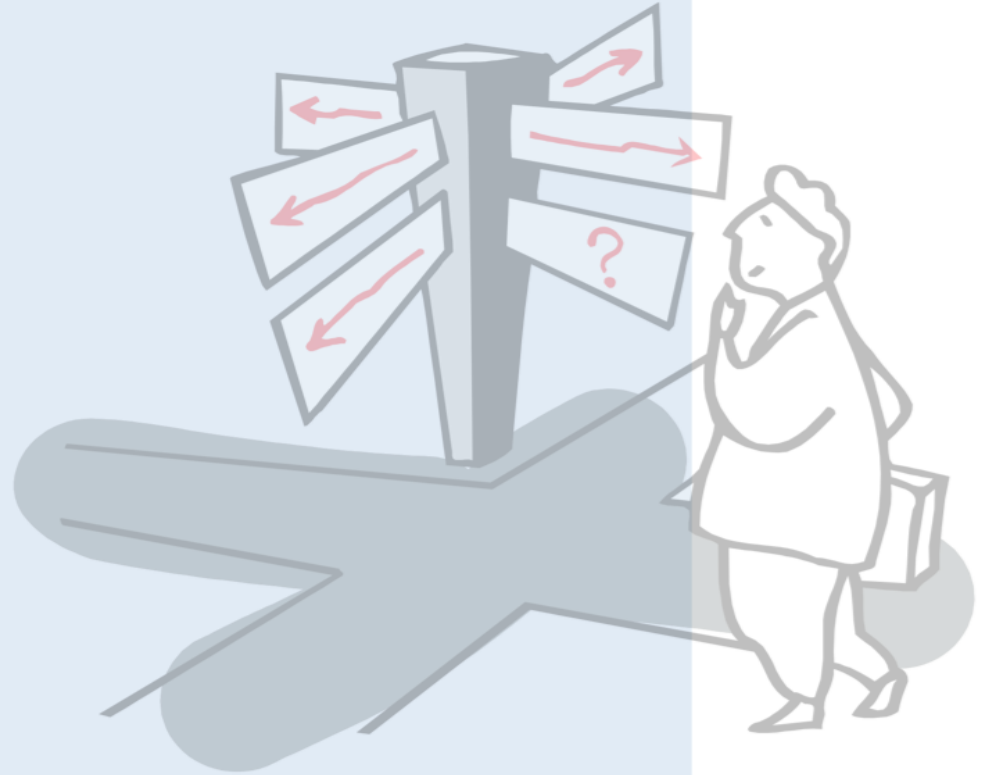
The material in this communication may be subject to copyright under the *Act*. Any further reproduction or communication of this material by you may be the subject of copyright protection under the *Act*.

Do not remove this notice.

Roadmap



- High Level Design
- Design Principle
- Design Pattern



High Level Design



- Software Components (e.g. classes / packages / namespaces)
- Where to put each software component
- How they interact to solve the business requirements
- Roles and Responsibilities of each software component

Design Principle (Recap)



Pre-OO era

- Strong Cohesion
- Loose Coupling

Design with OO principles

- Encapsulation
 - Inheritance
 - Polymorphism
 - Information hiding
-
- OO principles “promotes”
Strong cohesion and Loose coupling

Cohesion (= Intra-dependency)



- Intra-dependencies of the components in a software unit (e.g. class, method, module)
- Want Strong cohesion
 - ☐ Meaning that “separating” these components apart to different units will cause issues
- Weak cohesion
 - ☐ Those components can be easily separated into different units without causing problems
- Refactoring (weak cohesion → stronger cohesion)

Cohesion – Examples



Weak Cohesion

■ PRS_ex1

- PRS_AddNewAssessment.java

Strong Cohesion

■ PRS_ex2

- PRS_AddNewAssessment.java
- AddAssessmentForm.java
- Assessment.java

■ PRS_GUI

- PRS_AddNewAssessment_GUI.java
- AddAssessmentForm_GUI.java
- Assessment.java

Coupling (= Inter-dependency)



- Inter-dependencies of different software units (e.g. class, method, module)
- Want Loose coupling
 - Meaning that the units do not depend on others very much
 - So replacing one unit with “a compatible one” will not cause issues
- Strong coupling
 - Those units “depend” on each other so much that replacing one with “a compatible one” will cause troubles due to some dependencies
- Refactoring (strong coupling → loose coupling)

Coupling – Examples



Strong Coupling

■ PRS_ex1

- PRS_AddNewAssessment.java

Loose Coupling

■ PRS_GUI

- PRS_AddNewAssessment_GUI.java
- AddAssessmentForm_GUI.java
- Assessment.java

■ PRS_GUI2

- PRS_AddNewAssessment_GUI.java
- AddAssessmentForm_GUI2.java
- Assessment.java

OO Principles (Recap)



■ Inheritance

- ☐ Super-class and Sub-class

■ Encapsulation

- ☐ Prevent data being accessed / changed by others

■ Information hiding

- ☐ Provide flexible design choice

■ Polymorphism

- ☐ Provide single interface for different types
- ☐ Examples: operator overloading, [Java] Generic, a superclass with different sub-classes

Design Pattern (Recap)



- Well known solution for a particular programming situation
- Well known patterns
 - ☐ Model View Controller (MVC)
 - ☐ Façade pattern

Model-View-Controller



- Model – data model
- View – presentation of the model
- Controller – controls the flow / interactions of the view and model

MVC – Example – Balance Transfer



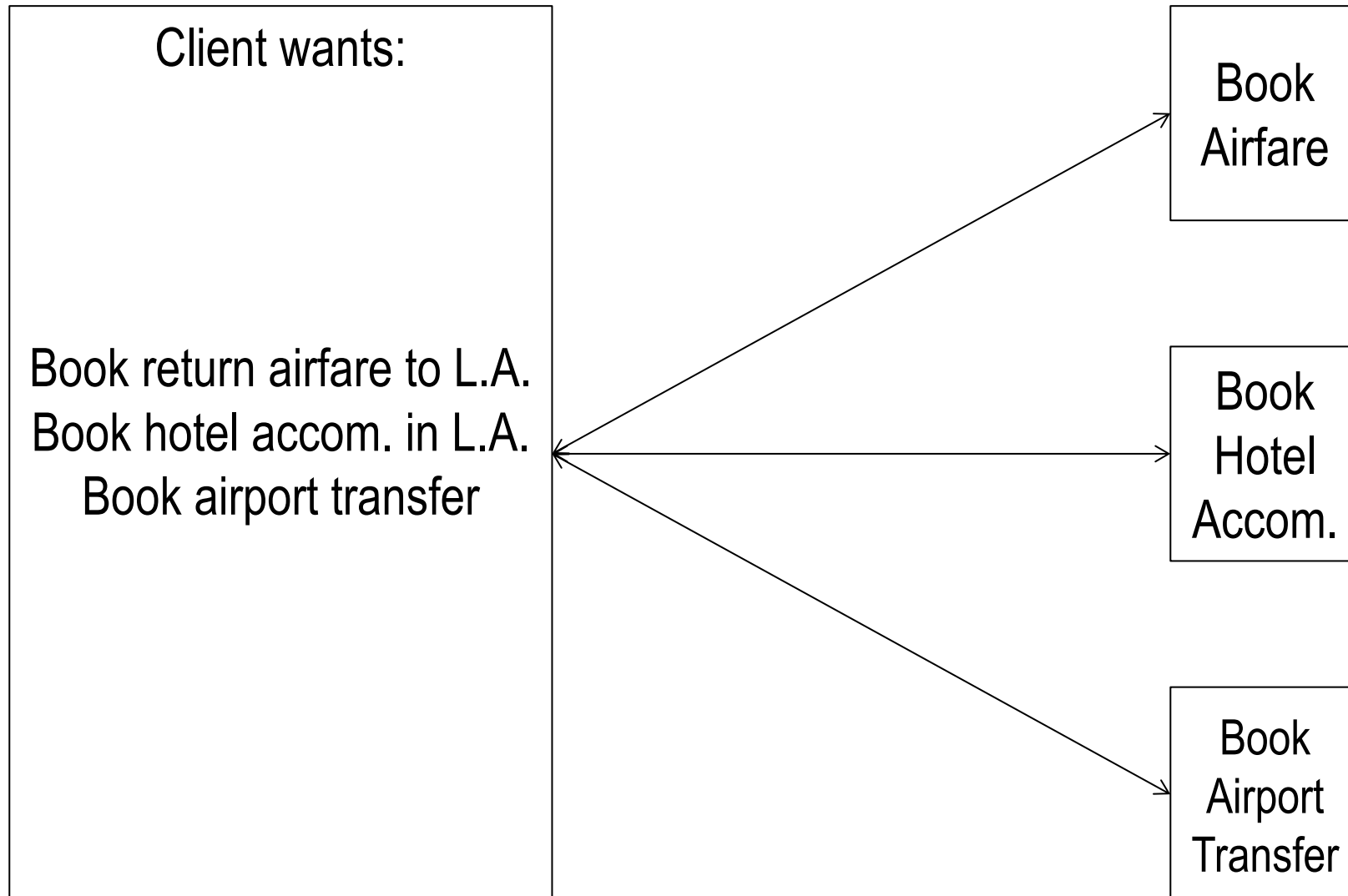
- Model: Bank_Account
- Views: Presentation Form and Result
 - ☐ Form to collect the required information
 - ☐ Responses with respect to the transfer
- Controller:
 - ☐ Check the business logic
 - ☐ Enough balance for transfer; transfer amount within daily limit; ...
 - ☐ Both accounts exist and active
 - ☐ ...
 - ☐ Control the process flow according to the “requirements”

Façade



- A common frontend for several inter-related operations
- Provide a unified interface to a set of interfaces of a subsystem
 - Usually for backend processing
- Usually: provide a higher-level interface that makes the subsystem easier to use
 - ... backend processing hidden from other developers

Façade – Example – Travel Booking(Analogy)



Façade – Example – Travel Booking (cont'd)

