Graphical User Interfaces

Design Fundamentals

COMP2603
Object Oriented Programming 1

Week 8

Outline

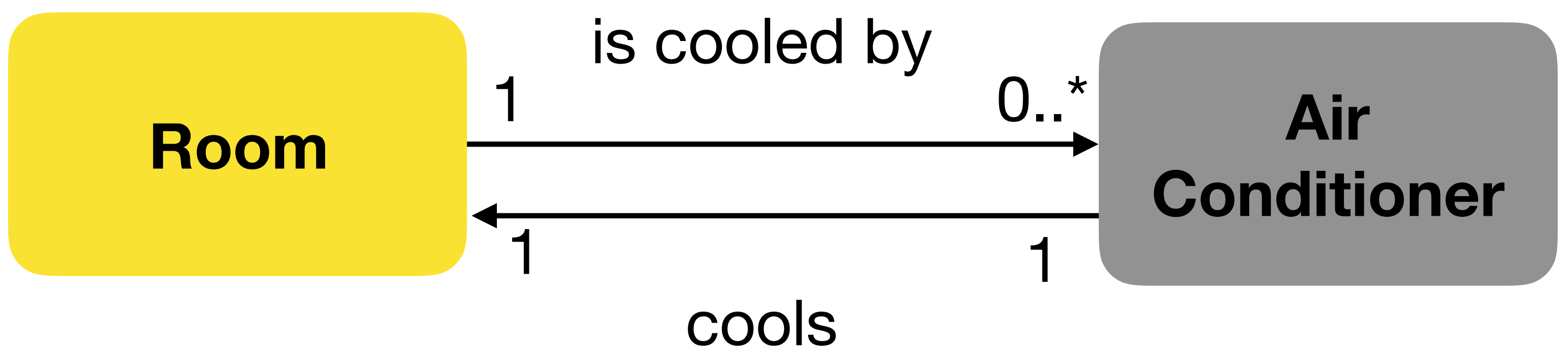
- Three-Tier Architecture
- Model-View Separation
- Design Principles
    - Coupling and Cohesion

Reading: Chapter 4 - Information Hiding and Design Principles: P. Mohan 2013
Reading: Chapter 17 - Intro. to Graphical User Programming: P. Mohan 2013

Object Interactions

An object-oriented program consists of a set of objects that collaborate to achieve some goal.

Two objects collaborate when one object requests a service from the other.



Object Interactions

It is important to control these interactions in order to create high quality programs.

Two notions affect the quality of these programs:

- **Coupling**
- **Cohesion**

Reading: Chapter 4 - Information Hiding and Design Principles, Page 70 P. Mohan 2013

Coupling

Object-Oriented coupling refers to the degree or strength of interconnection among classes.

Loosely coupled classes are desirable since each class can be handled in a relatively independent manner.

Tightly coupled classes are not desirable.

A class can become coupled to another class if it knows about that class in some way.

Loose Coupling

Loosely coupled classes facilitate:

- The replacement of one class by another so that only a few classes are affected by the change.
- The speedy debugging of errors since it is easier to track down an error and isolated the defective class causing the error.

Cohesion

Cohesion concerns the inner strength of a class, i.e., how strongly related the parts of a class are.

A class whose parts are strongly related to each other and to the concept being represented is said to be strongly cohesive.

Strongly cohesive classes are easier to understand, debug and maintain.

Weak Cohesion

In contrast, a weakly cohesive class is one whose parts are hardly related to each other.

These classes tend to embody more than one unrelated concept and takes on responsibilities that could be best handled by another class.

Eg. An Account class that stores customer information such as telephone number, customer name, address etc.

Three-Tier Architecture for Object Oriented Software

Typically, object-oriented programs consist of several layers of software.

Each layer serves a particular purpose.

It is common to use a three-layered architecture when designing these kinds of programs.

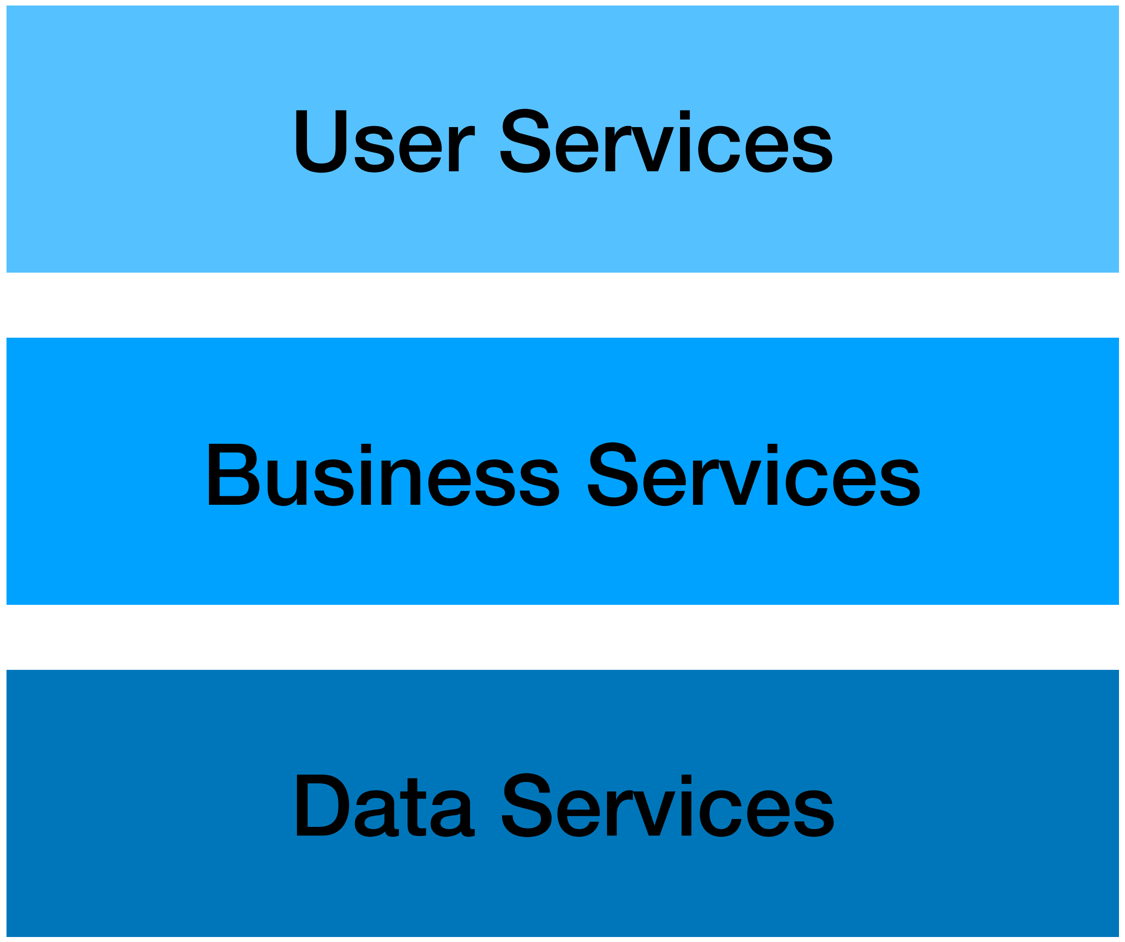
This is known as the **Three-Tiered Architecture**

Three-Tiered Architecture

The Three-Tiered Architecture consists of 3 vertical layers:

1. **User Services:** visual interface for presenting information and gathering data
2. **Business Services:** tasks and rules that govern application processing
3. **Data Services:** persistent storage mechanism to maintain, access and update data

Layers in the Three-Tier Architecture



Model View Separation

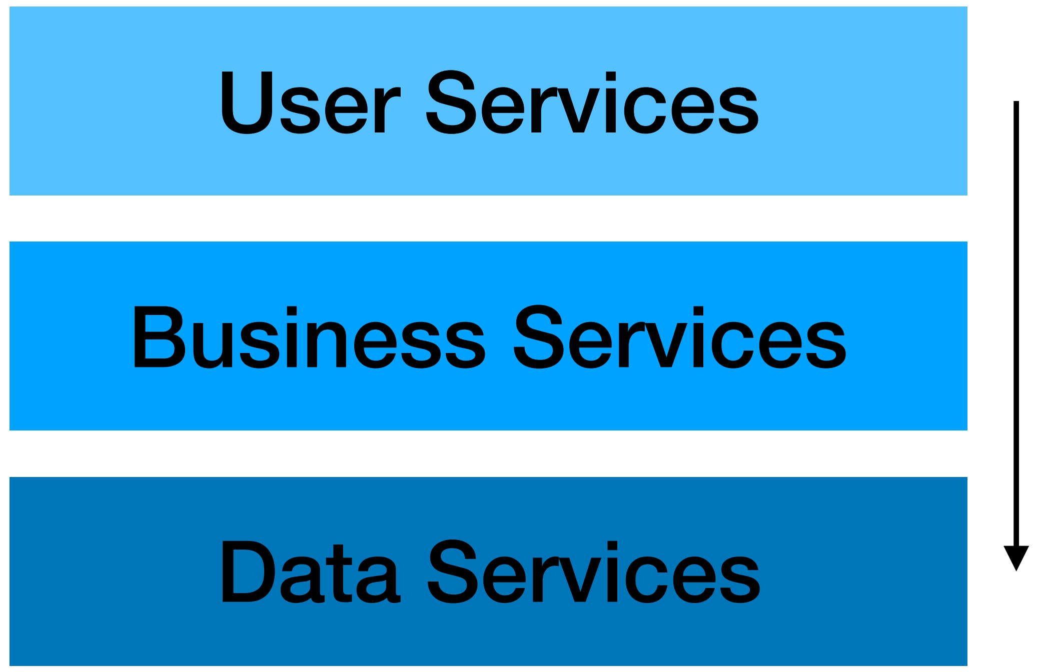
The Model-View Separation design pattern states that model object (domain objects or objects in the Business Services tier) should not have direct knowledge of or be directly coupled to view objects (user interface objects).

Communication between Domain Objects and User Interface Objects

In order for the objects in the user services tier to display information, methods must be invoked on the relevant domain objects.

The data is then displayed using GUI components.

This approach is called **pull-from-above.**



Advantages of Model-View Separation

Advantages of using the Model-View Separation:

- Model can be developed independently of the user interface
- New views can be easily connected to an existing model without affecting the objects in the model
- Multiple, simultaneous views of the same model can be developed (desktop, smart phone, web form views)
- The classes in the model can be easily ported to another user interface