Introduction to Object-Oriented Design

CS 4354
Summer II 2015

Jill Seaman

Review of Object-Oriented Concepts

- **Encapsulation**: combining data and code into a single object.
- **Data hiding (or Information hiding)** is the ability to hide the details of data representation from the code outside of the object.
- **Interface:** the mechanism that code outside the object uses to interact with the object.
  - ✦ The object's (public) functions
  - ✦ Specifically, outside code needs to "know" only the function prototypes (not the function bodies).

Objects and Classes

- **Objects** have state and behavior:
  - ✦State: the information stored by the object
    - – Values of the fields of a Java object
  - ✦Behavior: the operations an object supports
    - – Methods a Java object can perform
- **Class** is a collection of objects with the same behavior and common set of possible states.

From Problem to Code

- ✦Analysis
  - – Completely defines tasks to be solved by the program
  - – Result is a detailed textual description called a Functional Specification
- ✦Design
  - – Structures the programming tasks into a set of interrelated classes
    - ❖ Identify classes (and attributes)
    - ❖ Identify responsibilities of the classes
    - ❖ Identify relationships among the classes
- ✦Implementation
  - – Implements and tests the classes

Identifying Classes

- Look for nouns in the functional specification.
- Focus on concepts, not implementation

- The attributes of the nouns become the fields of the class that represent the state of the objects in that class.
  - Email Message might be a noun from the specification
  - Attributes of a message become the fields:
    - To address
    - From address
    - Subject
    - Text

Identifying Responsibilities

- Look for verbs in the functional specifications
  - Add message to mailbox
  - Remove message from mailbox
  - Set the subject of a message
- Every operation is the responsibility of a single class
- Not always easy to decide which class is responsible:
  - Example: Add message to a mailbox
  - Who is responsible, the message or the mailbox?

Identifying Relationships

- **Aggregation** relationships: (objects of one class **contains** instances of another)
  - Example: Mailbox contains Messages
  - Implemented using a **field** in one or both of the classes. (Mailbox might have a field that is an array of Messages).
- **Dependency** relationships: (objects of one class **uses** instances of another)
  - Example: Mailbox uses a PrintManager object to output a Message to a printer.
  - Mailbox may have a field or a method parameter that is a PrintManager

Setters and Getters and Information Hiding

- Setters and Getters
  - Can also be known as Mutators and Accessors
  - Methods used to change / return the value of a field
- Advantages:
  - Provides controlled access to fields when the fields are made private.
    - Fields are made private to support information hiding
- Disadvantages:
  - Violation of information hiding. If another class can see and set the values of all of your attributes, nothing is hidden.
- Try not to use getters and setters. If you feel you need a setter, there is probably some responsibility that you are giving to the wrong class.