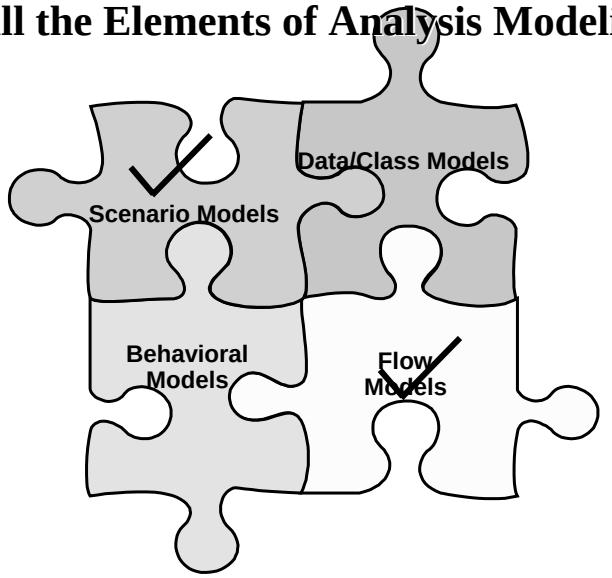
Software Engineering I CSC-382

Lecture 11

Software Engineering I CS-382

- Lecture 11
- What we will cover: (Details of Analysis Modeling)
  - Chapter 8 Sections 8.7, and 8.8 in Pressman
  - Goal is to continue to develop the methods and tools available from Object Oriented Analysis for analysis modeling

Recall the Elements of Analysis Modeling



Recall Our Class-Based Modeling Thru OOA

- Object Oriented Methods view a system as a collection of these objects that communicate to each other thru messages
  - These messages request the various other objects to perform some function or task
- Identify **classes** by examining the problem statement
  - We used our Stereotypes and then our CRC Cards for making this a little easier

Recall Our Class-Based Modeling Thru OOA II

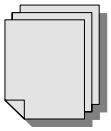
- We then must model:
  - The **attributes** of each class
  - The **operations** that manipulate the attributes
  - Later define **inheritances** and **aggregations** of these classes.
  - Also later on define **associations** and **dependencies** of the classes based on how they collaborate to accomplish the required functionality

Recall Our Simple Set of Stereotypes

Actor Classes:
•People
•Organizations

Interface Classes:
•Screens
•Menus

Business Classes:
•Places
•Things
•Concepts
•Events



Report Classes:
•Printed
•Electronic

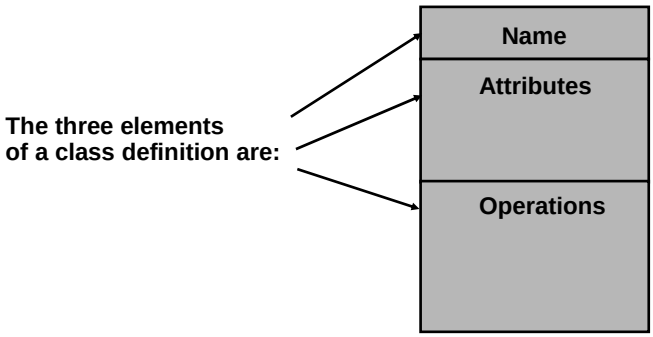
Issues Regarding A Good Design

- Typical factors for a good design:
  - The objects group nicely into ‘neighborhoods’
  - There are few lines of communication between these groupings
  - No single object knows, does, or controls too much
  - The objects behavior matches their desired role
  - An object model solution for one subsystem can be reused for other similar subsystems
  - There are only a few patterns of collaboration that are repeated often in the design

Recall Using the CRC Card for the Class Diagrams

- CRC stands for: Class, Responsibility and Collaborators
  - **Class**: An object is a person, place, thing, event, or concept
  - **Responsibility**: Anything that a class knows or does
    - Therefore the responsibilities will help us define the **attributes** and **operations** of our class model
    - They will also provide us insight into **inheritances**
  - **Collaborator**: A class that another class needs to accomplish its purpose

The Basic Class Diagram



One Prelim. List of Classes for SafeHome

Camera	Home Owner
Sensor	User Screen
Water Sensor	Log-in Screen
CO Sensor	Menu Screen
Fire Sensor	House Status Screen
Window-break Sensor	Change Password Screen
Alarm	Live Video Screen
Visual Alarm	Configuration Screen
Message/Internet Alarm	Alarm Reset Screen
SafeHome Co. (operator)	Maintainer
Emergency Monitor Co. (operator)	Self Tester
Home-to-SafeHome Interface	Power Manager…

In-class Class Diagrams

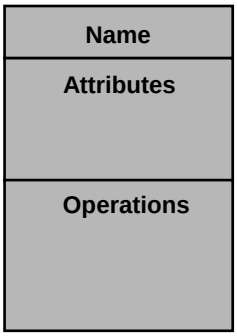
One of our class candidates: Log-in Screen

Attributes:

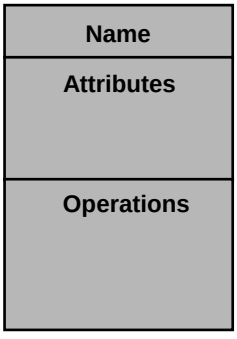
(these are found in the ‘knows’ responsibilities)

Operations:

(these are found in all other responsibilities)



In-class Class Diagrams



Using the CRC Card for Inheritance

- CRC stands for: Class, Responsibility and Collaborators
  - **Class**: An object is a person, place, thing, event, or concept
  - **Responsibility**: Anything that a class knows or does
    - Therefore the responsibilities will help us define the **attributes** and **operations** of our class model
    - They will also provide us insight into **inheritances**

Using the CRC Card for Inheritance II

- Recall the last part of the class model that we need to develop is the **inheritances** and **aggregations, associations**, and **dependencies**
- Recall we said **inheritances** can often be seen through two ways:
  1. The grouping of the CRC cards and
  2. Thru any classes that may have no individual responsibilities
     - i.e. Sensor and Water Sensor, CO Sensor, etc. may all correspond to a particular inheritance structure.

Means for Identifying Possible Inheritances

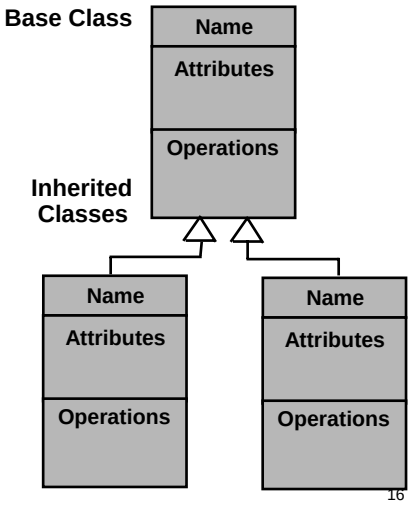
- Look for common categories for the objects
- Look for common roles that all the objects play
  - Try to combine shared responsibilities from multiple classes into unified interfaces
- Then can define an abstract class to hold these shared responsibilities
  - Makes it easier to ensure that other classes can interface smoothly with these related classes.

Example Search for Inheritances

One of our class candidates from our CRC session some of the classes we had trouble filling:

Sensor
Water Sensor
CO Sensor
Fire Sensor
Window-break Sensor

These were classes where we had questions as to which CRC should hold the responsibilities.



In-Class Inheritance Diagrams

For Next Class

- Finish Chapter 8
  - Class Associations