

# OBJECT-ORIENTED DESIGN, PROJECT KICK-OFF

CSSE 120—Rose Hulman Institute of Technology

# Software Design

- A process of breaking a problem down into manageable pieces—*components*
  - ▣ Components **interact** to solve the problem
  - ▣ Every component **provides** *services* through *interfaces*
  - ▣ Other components, called *clients*, **use** these services
- The components are *abstractions*
  - ▣ They **hide irrelevant details** from clients
  - ▣ They can be **independently developed** and **improved**
- Components provide *separation of concerns*

# Object-Oriented Design

- The processes of **finding** and **defining** a useful set of classes for a given problem
- Dominant design method for **large** software systems
- A **data-centered** view of computing
  - ▣ Seems to be a good match for how many people break down problems into pieces
- Part art and part science

# Top-down vs. object-oriented design

	Top-down Design	Object-oriented Design
<b>Abstraction mechanism (What sort of thing are the components?)</b>	Functions	Objects
<b>Interface (How do clients interact with other components?)</b>	Formal parameters, return values	<b>Accessor and mutator methods</b> (with formal params, return vals)

## Zelle writes:

"If we can *break a large problem into a set of cooperating classes*, we drastically reduce the *complexity* that must be considered to understand any given part of the program."

# Guidelines for OO design

- Look for *object candidates*
  - ▣ Look for **nouns** in problem statement (card, hand, deck)
  - ▣ Which of them have **interesting behavior** (card, deck)
  - ▣ Which of them **group related data** (hand)
- Identify *instance variables*
  - ▣ What **info** do objects need **to do their jobs**
  - ▣ Find **home** classes for all the data
- Identify *methods*
  - ▣ Look for **verbs** in problem statement (deal, hit, win)
  - ▣ What **operations** should objects provide **to be useful**

# OO Design Process

- Within classes, uses top-down design for refine methods
  - ▣ Break complex methods down into calls to helper methods
  - ▣ Sometimes need to add methods to other classes
- Work iteratively
  - ▣ Add methods, design new classes, change existing classes
- Experiment!
- Keep it simple!

# OOD Case Study: Forest Fire

Watch demo, then read problem statement

# Project Kick-Off



# Getting Started with Quixo

- Instructor will announce project teams
- Get together with teammates and work on the following:
  - Exchange contact information:
    - Email, cell phones, preferred meeting times and places
  - Begin object-oriented design of Quixo
    - On ANGEL: [Lessons](#) → [Project](#) → [Instructions](#)
  - OOD sketches due next meeting