CS2704: Object Oriented Software Design

Topic 2: Abstraction and Separation

Constantinos Phanouriou
Spring 2000

General Info

WWW: http://courses.cs.vt.edu/~cs2704/
Newsgroup: vatech.class.cs2704

Homework1 (due January 31st Midnight):
http://courses.cs.vt.edu/~cs2704/spring00/homework.html
Object-Oriented Design

- Identify objects and classes
  *strategies*: abstraction and separation
- Identify how objects interact in system
  *strategy*: composition
- Identify hierarchies of related classes
  *strategy*: generalization

Object-Oriented Design Strategies

- Abstraction – modeling essential properties
- Separation – treat what and how independently
- Composition – building complex structures from simpler ones
- Generalization – identifying common elements
Course Overview

- C++ classes
- Design notations
- Composition
- Design
- Generalization
- Design
Outline

- Abstraction
- Separation
- Identifying objects and classes

Abstraction

- General concept: describing an entity in terms of its aspects (qualities or features)
- Modeling entities in software
- Only essential aspects should be captured
  - Attributes
  - Behavior
- What is essential depends on situation
Abstraction

A named collection of attributes and behavior relevant to modeling a given entity for some particular purpose
A “Passenger” Abstraction

- Abstraction of a passenger for a flight reservation system

Attributes:  
Behaviors:

Properties of a Good Abstraction

- well named – clearly identifies abstraction
- coherent – sensible description
- accurate – only attributes of entity
- minimal – no irrelevant attributes
- complete – everything needed
Mapping Abstraction to Software

real-world | abstraction | software
---|---|---
entity | attributes | {data, data,…}
behavior | | {method, method,…}

Mapping Abstraction to a Class

className

public

private

{data, data, ….}

{method,method, ….}
Separation

- Separation of *what* a component does from *how* it does it (ex. a procedure)
- Define classes by independently specifying the interface for objects in that class, and the implementations of that interface

Separation of Interface and Implementation
Interchangeability of Implementations

- Interface
  - Implementation 1
  - Implementation 2

Interchangeability of Implementations

- Text Interface
  - Draw Text
- Shape Interface
  - Draw Shape
- Graphics
  - Draw Text
  - Draw Shape
General Structure of a Class

className

{data, data, ....}

{method, method, ....}

General Structure of an Object
Multiple Instances of a Class

Identifying Objects and Classes

- Study features of system
- Look for nouns (people, places, things)
- Example features:
  - “add course grade to student record”
  - “enter rental equipment description”
  - “add frequent flyer miles to customer record”
Good Classes

• Class should represent a set of objects although sometimes only use one
• Behaviors (methods) of class should be meaningful
  – Should a chess piece move itself?
  – What behaviors does a piece of “Data” have?