CMSC 131

Object-Oriented Programming I

this reference, Encapulation, API

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Overview

- this reference
- Abstraction
- Encapsulation
- API

Code Duplication Avoidance

- For the current project (Medieval Soldiers) and future projects we will be grading for comments and code duplication avoidance
- Code Duplication
  - If there is a fragment of code that appears in several section of your program, then that code should be factored out and placed in an auxiliary method
- Example:
  - ExperimentWithCodeDup.java
  - ExperimentNoCodeDup.java

this Reference

- Current Object
  - Represents the object a non-static method operates on
- **this**
  - Represents a reference to the current object
  - It is a special reference initialized for you
  - It does not make sense in a static method
- When we use it? (See **Example:** CD.java/CDDriver.java)
  - To tell parameters from instance variables
    - Notice that we could refer to instance variables by using this.<instanceVariableName>
  - To call constructors from another constructors
    - **this** must be the first statement in the constructor
  - In equals method implementation
  - To return a reference to the same object (cascading of method calls)
  - To define non-static methods based on static ones
- Eclipse capitalizes on the use of this when automatically defining code
  - Source →"Generate Constructors using fields"

Abstraction (Technique)

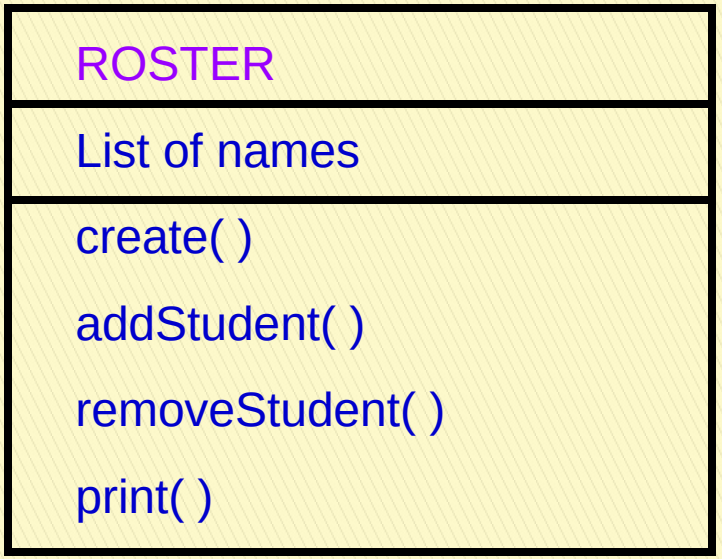
- Abstraction
  - Provide high-level model of activity or data
- Procedural abstraction
  - Specify what actions should be performed
  - Hide algorithms
- Data abstraction
  - Specify data objects for problem
  - Hide representation

Encapsulation (Technique)

- Encapsulation
  - Confine information so it is only visible/accessible through an associated external interface
  - Makes possible **Information hiding**
- Approach
  - For some entity X in program
    - Abstract data in X
    - Abstract actions on data in X
    - Collect data & actions on X in same location
  - Protects and hides X
- Extension of abstraction

Abstraction & Encapsulation Example

- Abstraction of a Roster
  - Data
    - List of student names
  - Actions
    - Create roster
    - Add student
    - Remove student
    - Print roster
- Encapsulation
  - Only these actions can access names in roster



API (Application Programming Interface)

- API → Application Programming Interface
- Interface implemented by a software that enables interaction with other software
- API designed in such a way that …
  - You can develop programs that will not break when the system represented by the API is updated
    - Example: We can change how we represent a phone number internally from String to integers
  - The only thing in the API are things the user will absolutely need
- Example: Java API
  - Project BattleField API
    - http://www.cs.umd.edu/class/fall2010/cmsc131H/projects/MedievalSoldiers/doc/index.html
  - http://download.oracle.com/javase/6/docs/api/index.html

Java Support for API Development

- Visibility via private/public
- The public access specifier allow us to define what will represent the interface
- The private access specifier allow us to encapsulate
- Only make something public if there is a reason to. Why?
  - As long as interface is preserved, class can change without breaking other code
  - The more limited the interface, the less is to maintain
  - To avoid giving object users unnecessary information
  - **Example:** Let's add a getSalary method to the ExperimentNoCodeDup class
    - Let's change it so we use String as a salary instead of double
- Rule of thumb
  - Make instance variables private
  - Implement set/get methods
    - Needed as private instance variables cannot be accessed outside the class declaration
  - Make auxiliary methods private