Object-Oriented Programming Intro

Department of Computer Science
University of Maryland, College Park
Object-Oriented Programming (OOP)

• Approach to improving software
  • View software as a collection of objects (entities)

• Motivated by software engineering concerns
  • To be discussed later in the semester

• OOP takes advantage of two techniques
  • Abstraction
  • Encapsulation

• Abstract Data Type
  • Implementation independent interfaces
  • Data and operations on data
Techniques – Abstraction

- 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
Techniques – Encapsulation

• Encapsulation
  • Confine information so it is only visible / accessible through an associated external interface

• 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

<table>
<thead>
<tr>
<th>ROSTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of names</td>
</tr>
<tr>
<td>create( )</td>
</tr>
<tr>
<td>addStudent( )</td>
</tr>
<tr>
<td>removeStudent( )</td>
</tr>
<tr>
<td>print( )</td>
</tr>
</tbody>
</table>
Java Programming Language

- Language constructs designed to support OOP
  - Interfaces
    - Specifies a contract
    - Provides abstract methods (no implementation)
  - Two views
    - Enforcing implementation of methods
    - Defining an IS-A relationship
- Class
  - Implements/defines contracts
  - Supports encapsulation of implementation (e.g., via private)
  - Class extending another class
    - Allows new class to inherit everything from original class
    - Defines an IS-A relationship
- Class libraries designed using OOP principles
Object & Class

• Object
  • Abstracts away (data, algorithm) details
  • Encapsulates data
  • Instance exist at run time

• Class
  • Blueprint for objects (of same type)
  • Exists at compile time
Java Collections Framework

- Collection
  - Object that groups multiple elements into one unit
  - Also called container
  - Example: ArrayList
- Collection framework consists of
  - Interfaces
    - Abstract data type
  - Implementations
    - Reusable data structures
  - Algorithms
    - Reusable functionality
Java Collections Framework

- Collection – Java Interface
  - See Java API entry for Collection
    - [http://download.oracle.com/javase/6/docs/api/index.html](http://download.oracle.com/javase/6/docs/api/index.html)
  - Example (CollectionExample.java)

- Collections – Class
  - [http://download.oracle.com/javase/6/docs/api/index.html](http://download.oracle.com/javase/6/docs/api/index.html)