Comp 212

Alan L. Cox
alc@cs.rice.edu

John Greiner
greiner@cs.rice.edu

January 17, 2001

Course Objectives

- To transition ...
  - from the functional paradigm of Comp 210 to the object-oriented paradigm.
- To introduce ...
  - Object-Oriented Programming (OOP) including ...
    * the fundamental concepts of encapsulation, inheritance, and polymorphism, and
    * Unified Modeling Language (UML) diagrams, an industry standard to describe OO program designs.
  - Java as a language for OO program implementation.
  - ...

Course Objectives (cont.)

- To introduce (cont.) ...
  - common design patterns for various purposes ...
    - creational: factory and singleton.
    - structural: adapter, composite, and decorator.
    - behavioral: command, interpreter, iterator, state, strategy, and visitor.
  - the formulation and implementation of basic data structures, such as lists, stacks, queues, trees, tables, etc.
  - fundamental algorithms on data structures, such as searching and sorting.
  - ...

Course Objectives (cont.)

- To introduce (cont.) ...
  - the rudiments of complexity analysis, such as asymptotic behavior and big O notation.
  - both stream and event-driven I/O.

OOP: Some Basics

- An *Object* may model, or be an abstaction of, a real-world object. It is a collection of fields (variables) and methods.
- A *Method* is an operation on the object.
- Objects are said to interact through *messages*. A message is directed at an object and specifies the method that that object should perform on itself. The sender knows nothing of how the method is performed.
- A *Class* is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind.
- A class can be derived from, or be an extension of, another class. Specifically, it can *inherit* variables or methods from its parent, or superclass.

Inheritance

- Simplifies the reuse of existing software.
  - Variables and methods of the superclass are transparently provided to the subclass.
  - Only the additional or replacement variables and methods of the subclass are spelled out.
- The inheritance tree, or *class hierarchy*, can be as broad or deep as desired.
  - In general, the further down in the hierarchy that a class appears, the more specialized its purpose.

Schedule

- The list of topics in (approximate) order...
  - Transition from functional to object-oriented programming:
    - primitive types, String type,
    - classes, fields, methods, constructors,
    - UML diagrams,
    - abstract classes, interfaces, inheritance, polymorphism,
    - functional lists and the composite pattern,
    - static fields, static methods,
    - the singleton pattern, the visitor pattern, the interpreter pattern,
    - exception handling.

Schedule (cont.)

- ...
  - Graphical User Interface and Event-driven programming:
    - strategy pattern, command pattern, the Model-View-Controller (MVC) paradigm.
  - Common data structures:
    - mutable lists and the state pattern, stacks, queues,
    - binary trees, binary search trees, self-balancing trees,
    - adapter pattern, decorator pattern,
    - arrays,
    - searching and sorting, quick sort, merge sort, heap sort, bucket sort,
    - template pattern, iterator pattern,
    - priority queues,
    - hashing,
    - complexity analysis.

Course Mechanics

- Course web site:

  `http://www.owlnet.rice.edu/~comp212`

- Contains
  - On-line lecture notes
  - Tutorials
  - Assignments
  - Administrivia

Assignments

- Major programming assignments (45%)
  - Expect three.
  - There will be graded milestones to help diagnose and correct problems early.
- Occasional quizzes during the tutorials and minor programming assignments (10%)
- Three equally weighted exams (45%)

First Assignment

- Sign up for a tutorial section at the course web site:

  `http://www.owlnet.rice.edu/~comp212/01-spring/labs/`