Comp 212

Alan L. Cox
alc@cs.rice.edu

Dung X. Nguyen
dxnguyen@cs.rice.edu

January 19, 2000

High-Level Overview

This course...

- provides the transition from the functional paradigm (Comp 210) to the object-oriented paradigm.

- introduces OOP using Java as the implementation language.

- emphasizes proper formulation and abstraction of the problem domain in the programming process in order to build programs that are robust, flexible, and extensible.

- teaches how design patterns help formulate and implement abstractions in effective and elegant ways.

Low-Level Overview

This course...

- covers data structures and algorithms to manipulate them that are essential to programming: lists, stacks, queues, trees, tables, and graphs.
  - These structures are implemented as systems of cooperating objects using appropriate design patterns.
- covers both stream I/O and event-driven I/O.

Objectives

- The fundamental concepts of OOP: encapsulation, inheritance, and polymorphism.
- The Unified Modeling Language (UML) diagrams, industry standard to describe program designs.
- The Java programming language to implement OO program design.
- Common design patterns, such as composite, interpreter, command, singleton, strategy, state, visitor, adapter, decorator, factory, and iterator.
- The formulation and implementation of basic data structures, such as lists, stacks, queues, trees, tables, and graphs.

Objectives (cont.)

- Fundamental algorithms on data structures, such as searching, sorting, and structural traversal.

- Rudiments of complexity analysis, such as asymptotic behavior and big O notation.

OOP: Key Terms

What is...

- an *Object*
  - a *Method*
- a *Message*
- a *Class*
- *Inheritance*
  - a *Superclass*
  - a *Subclass*

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 deep as desired.

- In general, the further down in the hierarchy that a class appears, the more specialized its purpose.

Course Mechanics

Course web site:

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

Contains

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

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.

- Graphical User Interface and Event-driven programming, strategy pattern, command pattern, the Model-View-Controller (MVC) paradigm.

Schedule (cont.)

...

- 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.

- Graphs and graph algorithms, ...

Assignments

Major programming assignments (45%)

Occasional written and minor programming assignments (10%)

Three equally weighted exams (45%)