CS2704: Object Oriented Software Design

Topic 1: Introduction

Constantinos Phanouriou
Spring 2000

Objectives

• Study OOP concepts; develop basic skills in software design
• Emphasize practices for design, construction, testing, and debugging of OO software systems
• Examine OO features of C++
• Study the primary principles and language features – objects, classes, inheritance, and polymorphism.

GTAs

Peter Schoenhoff (pschoen@csgrad.cs.vt.edu)
Office: 128McB
Office Hours: 8 – 12 MW, 8 – 10 F

Vinayak Kamath (vkamath@csgrad.cs.vt.edu)
Office: 128McB
Office Hours: 3:30 – 7 WF, 4:30 – 7:30 H

Texts

• A Practical Introduction to Software Design with C++ by Steven Reiss

Online

• All students are required to have email and WWW access
• List-server: Announcements only
• Newsgroup: vatech.class.cs2704

Grading

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
<th>Tentative Dates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td>40%</td>
<td>TBA</td>
</tr>
<tr>
<td>Homework / Quizzes</td>
<td>15%</td>
<td>TBA</td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>20%</td>
<td>Early March</td>
</tr>
<tr>
<td>Final Exam</td>
<td>25%</td>
<td>Tuesday, May 9</td>
</tr>
</tbody>
</table>
Test Environments

• All programming assignments submitted are required to compile under either:
  – Microsoft Visual C++, version 6.0
  – GNU g++ compiler (installed on the FreeBSD machines in McB 124)
• Programs will be tested under either Windows NT or FreeBSD

Procedural Programming

• Problem is divided into sequence of sub-problems to be solved
• Program is sequence of procedure calls
• Think in terms of task and subtasks
• Languages: C, Pascal, Fortran, COBOL, …

Program Demonstrations

• Demonstrations will take place in the McBryde CS Dept. Computer Lab for some assignments
• GTAs will NOT go to dorms to grade programs on specific systems neither can the students bring their systems to the lab (All programs must run under Windows NT or FreeBSD)

Procedural Design

• Key: identify simple tasks that can be programmed easily
• Top-down: break tasks into subtasks
• Bottom-up: write small useful routine and compose into larger procedures
• Diagrammed as structure charts

Backups

• Students are responsible for making backup copies of all their work in the course
• Backup copies should be maintained until after the end of the term
• Remember: Computer systems are mechanical devices. Systems fail. Plan accordingly.

Problems in Procedural Programs

• Large programs made up of many small procedures
• Not clear which does what to what data
• No enforced control over access to data
• Difficult to fix bugs, modify, and use procedures in other programs
Software Engineering Goals

- **Reusability** – components can be used in many applications
- **Extensibility** – ease of change
- **Flexibility** – modifications do not “break” system

Object-Oriented Programming

- Think of building program from parts
- Parts are *objects* that interact to solve the problem
- Define *classes* of objects that can be reused
- Think in terms of objects and interactions
- Languages: Java, C++, Smalltalk, …

Generalization in OOP

- Two approaches to defining classes in terms of others
  - Inheritance – “inherit” properties of other classes
  - Parameterized classes (*templates*) – class defined in terms of parameter classes
- Design patterns are solutions to common design problems

Why Object-Oriented?

- Object interactions are defined by methods, which are just procedures
- Programs still sequences of “procedure” calls, but think of as interacting objects
- Procedures organized around objects
  - Data access easier to understand
  - Data access easier to control
- “Easier” to avoid design problems

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
Homework 1

- Due midnight Monday, January 31\textsuperscript{st}
- Submit Electronically using the "curator" system
- http://courses.cs.vt.edu/~cs2704/spring00/homework.html
- Project 1 will build on Homework 1