Weisfeld, The Object-Oriented Thought Process, Addison-Wesley, ISBN 9780321861276

Course Description:

This is a course in object-oriented programming using the C++ programming language. The course will look at the entire software project life-cycle. That is, we will consider the analysis, design, coding, testing, and evaluation of object-oriented software.

Software analysis is the modeling of user requirements for a system. Traditional structured analysis attempts to describe how data should flow within the system. Object-oriented analysis (OOA) describes the essential objects of the problem, their internal behavior, and the communication between objects.

The product of object-oriented analysis should serve as the basis for object-oriented design (OOD). Analysis is concerned with what the system must do; design is concerned with the structure of the system. With a design one should end up with a framework or architecture that describes the logical and physical relations between objects in the system. We will use UML (Unified Modeling Language) as a tool for thinking about and documenting the design process. Rational Rose is the software that we will use for creating UML diagrams.

Object-oriented programming (OOP) is the process of converting the results of the design into actual code (typically using a language such as C++, Smalltalk, Eiffel, Java, etc.). This course will focus on C++ as the course programming language. Students are free to use any compiler they wish, so long as it has the advanced features of C++ discussed in class. These include STL, RTTI, and exception handling. Visual C++, version 5.0 or higher, will suffice for this course.

This course will also discuss the testing of object-oriented software (a still somewhat hazy subject) and also discuss software metrics as they do (or don't apply) to OOP.

Students in this course are assumed to have some prior knowledge of C++, but we will cover the basics of C++ syntax. Students are also expected to have previous knowledge of data structures (stacks, lists, queues). The principal programming concepts which will be covered are: C++ classes, inheritance, polymorphism, iostreams, templates, exception handling, and run-time type identification (Note: some of these concepts are not necessarily supported by current C++ compilers).

The concepts of C++ will be applied to both specific and generic programming examples. The concept of reusability, especially as it applies to library design, will form a principal motif in
the course. A major focus of the course will be the discussion of the STL (Standard Template Library). We will also briefly examine other object oriented languages such as Java, Python, Ruby, and C#.

At the conclusion to the course students will have an excellent foundation in both the theoretical and practical aspects of object-oriented programming.

**Course Requirements:**

1. Homework assignments/Quizzes. Regular homework assignments (and/or quizzes) will be given covering topics from C++ and object-oriented programming. The lowest homework grade will be dropped, the remaining homework assignments averaged together will count for 20% of the course grade.

2. Mid-term. It will be open book(s), open notes. The mid-term will focus primarily on features of object-oriented programming. It will be worth 25% of the course grade.

3. Final exam. Also open book. The final exam, which is not comprehensive, will focus on features of analysis and design. It will be worth 25% of the course grade.

4. Analysis/Design Project. There will be a major term group project due at the end of the semester. The point of the project is that the student is doing a careful analysis of a significant problem and the design of a system to solve that problem. Students will be allowed to work in groups of two or three. The project will be worth 25% of the course grade.

**COURSE OUTCOMES**

The following are the outcomes for this class and what assignments are relevant for the assignment

<table>
<thead>
<tr>
<th>Outcome</th>
<th>Project</th>
<th>Homework</th>
<th>Midterm</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome 1. Students will understand the object model and the canonical features of object-oriented programming</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Outcome 2. Students will understand class construction in C++</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Outcome 3. Students will understand advanced features of C++ such as templates, abstract classes, virtual functions</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Outcome 4. Students will understand object-oriented analysis and design</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Course Topics
(Note: topics will not necessarily be discussed in the order shown below. Time and class interest will determine the precise order)

(I) Objects
(A) C++ objects
(B) The Object Model
   (1) Abstraction
   (2) Encapsulation
   (3) Modularity
   (4) Inheritance
(C) Typing in C++
   (1) Strong vs. Weak Typing
   (2) Static vs. Dynamic binding
(D) Persistence of Objects

(II) Object-Oriented Analysis
(A) The Nature of an Object
   (1) State
   (2) Behavior
   (3) Identity
(B) Relations Between Objects
   (1) Inheritance (is-a relationship)
   (2) Aggregation (has-a relationship)
   (3) Using

(III) An Overview of C++
(A) Classes
   (1) Interface
   (2) Implementation
   (3) Canonical Form
(B) Inheritance
   (1) Derived Classes
   (2) Abstract Classes
   (3) Virtual Functions and Polymorphism
   (4) Multiple Inheritance
(C) Memory allocation operators (new and delete)
(D) Iostreams

(IV) Object-Oriented Design
(A) UML
   (1) Use Cases
   (2) Types of UML Diagrams
(B) Design Components
   (1) Problem Domain
   (2) Human Interface
   (3) Task Management
   (4) Data Management
(V) Templates
   (A) Concrete and Abstract Types
   (B) Templates vs. Inheritance
   (C) STL

(VI) Other C++ Features

Electronic Forms of Communication: In accordance with College policy, I will use your Ramapo College email address (@ramapo.edu) to communicate with you about all course-related matters. The syllabus and other course-related files will be posted on Luminis.

Attendance Policy: You are expected to attend all classes. Should you miss a class, you are still responsible for the work covered and any announcements made such as test dates. To do this, it is best if you have the contact information of at least two other students in the class, in advance.

Policy on Academic Integrity: Students are expected to read and understand Ramapo College’s academic integrity policy, which can be found in the College Catalog. Members of the Ramapo College community are expected to be honest and forthright in their academic endeavors. Violations of the policy will be sanctioned.

Students with Disabilities: Any student who needs course adaptation or accommodations because of a documented disability should make an appointment with me as soon as possible. (Note: you must be registered with the Office of Specialized Services.)