Department and Course Number	**CEG 461**	Course Coordinator	Thomas C. Hartrum
Course Title	**Object-Oriented Programming and Design**	Total Credits	4

BS CE: Elective; BS CS: Elective.

This document was prepared by: Thomas C. Hartrum Date: May 10, 2005

Catalog Data

Study of object-oriented design and programming. Programming topics emphasize the core concepts of encapsulation, inheritance, polymorphism, and dynamic binding. Additional topics include class organization, software maintenance, and design of reusable components. **Prerequisite**: CEG 460.

Text Books and Other Source Materials

1. Page-Jones, ***Fundamentals of Object-Oriented Design in UML***, Addison Wesley, 2000, ISBN 0-201-69946-X.

2. Dattatri, *C++*: ***Effective Object-Oriented Software Construction***, 2nd ed., Prentice Hall, 2000, ISBN 0-13-086769-1.

Home Page: http://www.cs.wright.edu/~thartrum/CEG661WI05/intro661.html

Course Objectives

The student should have learned the following:
1. The object-oriented programming paradigm.
2. Reuse mechanisms in object-oriented languages.
3. Assessment of object-oriented models (coupling, cohesion, & encumbrance).
4. Object-oriented analysis and design methodologies.
5. Design guidelines for inheritance and polymorphism.
6. The Unified Modeling Language (UML).

Prerequisites by Topic

1. The software lifecycle.
2. Object-oriented models: classes, associations, state-models.
3. Software requirements and analysis.
4. Object-oriented design and programming.
5. Syntax and semantics of C++.
6. Data abstraction with classes, encapsulation and data hiding.

Course Content

1. Introduction to OO, Review of OOA and UML
2. OO with C++; C++ inheritance

3. Encapsulation and Connascence
4. Associations, Domains, Encumbrance, and Cohesion
5. Mastering Data Abstraction
6. Midterm Exam; Inheritance and Polymorphism
7. State Space & Behavior; Type conformance
8. C++ Polymorphism, Perils of Polymorphism
9. Advanced Topics
10. Advanced Topics

Class/Laboratory Schedule:

Each week has two lectures of 75-minutes each. There is no scheduled lab. Students are expected to work in open labs for no less than 2 hours a week.

Laboratory Projects:

There is one quarter-long team project, turned in and graded in three phases. Phase I consists of an abstract design, including UML diagrams and a full data dictionary, and takes 2.5 weeks. Phase II consists of implementation in C++ code, and takes 2 weeks. Phase III involves extending the design and code to handle a user's extension request, and takes 2 weeks Projects are done by two- or three-person teams. Grading of code is based on execution (20%), meeting requirements (30%), documentation (20%), and style (30%).

Contribution to Professional Component

CEG 461 contributes 4 hours to Criterion 4(b), including engineering design.

Course Contribution to Program Educational Objectives

CEG 461 contributes to Objectives 1 and 2. By studying various abstract object-oriented design concepts it develops design skills in the students. Detailed examination of object-oriented concepts in C++, along with the projects, extends the students' implementation abilities. The students acquire design and programming skills applicable to other, more advanced courses.

Course Contribution to Program Outcomes and Assessment

a	b	c	d	e	f	g	h	i	j	k
PXX	0	PXX	0	PX	0	PX	0	PX	PX	PXX

Estimate ABET CAC Category Content

	Core	Advanced		Core	Advanced
Data Structures		0.5	Concepts of PL		0.5
Algorithms			Comp Organization + Architecture		
Software Design		3.0	Other		

Oral and Written Communications

There are no oral presentations. Students submit reports on the projects, which include a textual data dictionary. However, these are graded based on technical content and not grammar, spelling, or style (although spelling errors are frequently marked).

Social and Ethical Issues

None.

Theoretical Content

Strength of predicates and type conformance are discussed, including contravariance and covariance. Approximately 3 hours are spent on this material, along with homework and exam questions.

Problem Analysis

For the project, students are given a requirements specification including use cases, UML diagrams, and a data dictionary. They need to analyze this in order to come up with design decisions.

Solution Design

Focus of this course is on abstract design, using UML, and implementation. Based on the provided requirements specification, students design an object-oriented solution, then implement it in C++

Outcomes

The student should be able to

1. Design object-oriented solutions from unified modeling language specifications.
2. Analyze object-oriented solutions for independence and information-hiding.
3. Design good class inheritance hierarchies to support reusable classes.
4. Design abstract classes to form reusable object frameworks.
5. Map object-oriented designs in UML to a specific object-oriented language.
6. Organize and contribute to team programming projects.

Outcome Measures and Assessment

Students are assessed by several homework problems (10%) and two exams (25% each). In addition, application of the concepts covered in class is assessed by the project design and implementation (40%).
There is a self-assessment conducted at the beginning of the course, and another at the end. These forms are included below.