

CS 2511: Software Analysis and Design (4)**Catalog Description:**

Techniques for analyzing, designing, and creating medium-scale software through object-oriented design and implementation. Introduction to design patterns. Emphasis on polymorphism and abstraction to increase software modularity, reusability, and flexibility. Includes a medium-scale team-development project.

Textbook: Cay Horstmann, *Object-Oriented Design and Patterns*, Wiley, 2006, (2nd ed.).

Course Goals:

This course follows and requires the freshman introductory sequence CS 1511-1521. It is intended to provide the background necessary to undertake serious programming projects that the student will later encounter in advanced courses, actual jobs, or research. The course takes up where the freshman year leaves off in the discipline of object-oriented design and coding, and introduces Java as the object-oriented programming language. Major topics include interface types and abstract classes, polymorphism and inheritance, design patterns, frameworks, and multithreading. Students will learn how to build graphical user interfaces, and how to systematically test and debug their code. Along the way, they will be introduced to the basic concepts of software engineering.

Prerequisites by Course & Topic

CS 1521: Computer Science II – object-oriented programming, elementary data structures

Major Topics Covered in the Course

Problem analysis and functional requirements specification

Formal methods of object-oriented design using UML

Introduction to design patterns

Java programming

Interface types and polymorphism

Creating graphical user interfaces

Using software frameworks

Unit testing

Class/Laboratory Schedule: Lecture: 3 hours per week, Laboratory: 1, Discussion: 1

Course Outcomes:

1. Students demonstrate a proficiency in problem analysis:
 - a. Produce a functional specification of a software solution.
 - b. Produce use cases for a software solution.
2. Students demonstrate proficiency in object-oriented design:
 - a. Identify object classes for a software solution and the relationships among them.
 - b. Produce UML class diagrams.
 - c. Produce UML sequence diagrams.
 - d. Recognize and exploit known design patterns.
 - e. Maintain timely and clear design documentation.
3. Students demonstrate proficiency in object-oriented coding:
 - a. Realize class diagrams in an object-oriented programming language.
 - b. Take advantage of interface types and polymorphism
 - c. Create effective user interfaces.
 - d. Understand and use software frameworks.
 - e. Employ unit testing methodology.
4. Students demonstrate the ability to work successfully on a team:
 - a. Students meet with fellow team members to choose leadership, delegate responsibility, and see programming projects through to completion.
 - b. Students communicate regularly with team members.

Relationship to Program Outcomes

CS 2511 is a required course taken upon successful completion of the introductory computer science sequence. This course contributes to meeting the following program outcomes:

2. *Students can design, develop, and analyze significant software systems.*
- This course increases a student's comprehension and understanding of the software development process introduced in CS I & II. They use UML to design program solutions, learn about object-oriented design patterns, and extend medium-scale software frameworks. In addition, they use an integrated development environment to build and systematically test programs. Course outcomes 1, 2 and 3 map to this program outcome.
3. *Students understand the fundamentals of computer organization and architecture, data structures and related algorithms, and programming languages.*
Students learn Java and continue to develop their object-oriented programming skills through advanced topics such as interface types, polymorphism, reflection, and multithreading. Course outcomes 1, 2 and 3 map to this program outcome.
4. *Students can apply computer science principles and practices to a variety of problems*
This course requires the solving of a diverse set of problems, including: implementing designs (puzzle-solving applications); creating functional specifications and design models (calendar application); and extending software frameworks (multi-person game application). Course outcomes 1, 2 and 3 map to this program outcome.
5. *Students can work independently and also work effectively in teams.*
This course refines the student's skills and abilities in working in a team-oriented environment. Course outcome 4 maps to this program outcome.

Assessment Plan for Course:

This course is assessed every other year by the instructor and a course assessment document covering all of the course outcomes and their effect on the program outcomes is prepared.

Estimate CSAB Category Content

	CORE	ADVANCED		CORE	ADVANCED
Data Structures			Computer Organization and Architecture		
Algorithms			Concept of Programming Languages	1	
Software Design	3				

Coordinator/Prepared by: T. Colburn