CSE 2221 (Approved): Software I: Software Components

Course Description

Intellectual foundations of software engineering; design-by-contract principles; mathematical modeling of software functionality; component-based software from client perspective; layered data representation.

Prior Course Number: CSE 221, part of CSE 222
Transcript Abbreviation: SW I: Components
Grading Plan: Letter Grade
Course Deliveries: Classroom
Course Levels: Undergrad
Student Ranks: Freshman, Sophomore
Course Offerings: Autumn, Spring
Flex Scheduled Course: Never
Course Frequency: Every Year
Course Length: 14 Week
Credits: 4.0
Repeatable: No
Time Distribution: 3.0 hr Lec, 1.0 hr Lab
Expected out-of-class hours per week: 8.0
Graded Component: Lecture
Credit by Examination: Yes
Exam Types: Departmental Exams
Admission Condition: No
Off Campus: Never
Campus Locations: Columbus
Prerequisites and Co-requisites: CSE 1211 or CSE 1212 or CSE 1221 or CSE 1222 or CSE 1223 or CSE 201 or CSE 202 or CSE 203 or CSE 204 or CSE 205 or EG 167 or CSE Placement Level A; co-req: Math 1151 or Math 1161
Exclusions: Not open to students with credit for CSE 321
Cross-Listings:

The course is required for this unit's degrees, majors, and/or minors: Yes
The course is a GEC: No
The course is an elective (for this or other units) or is a service course for other units: Yes

Subject/CIP Code: 14.0901
Subsidy Level: Baccalaureate Course

Programs

Abbreviation	Description
BS CSE	BS Computer Science and Engineering

General Information

Java is taught and used

Course Goals

Be familiar with the reasons it is important that software be "correct", i.e., why "good enough" is not good enough when it comes to software quality

Be familiar with the reasons for designing software to minimize the impact of change, and why it is difficult to achieve this
Be familiar with using design-by-contract principles to write software that uses existing software components based on their interface contracts
Be familiar with using interface contracts that are described using simple predicate calculus assertions with mathematical integer, string, finite set, and tuple models
Be familiar with extending existing software components by layering new operations on top of existing operations
Be familiar with layering new software components' data representations on top of existing software components
Be familiar with using simple recursion
Be familiar with using simple techniques to test application software, layered implementations of extensions, and layered data representations, including developing and carrying out simple specification-based test plans
Be familiar with using simple techniques to debug application software, layered implementations of extensions, and layered data representations
Be exposed to using basic algorithm analysis techniques and notations to analyze and express execution times of operations whose implementations involve straight-line code and simple loops
Be competent with writing Java programs in a procedural style using the basic control structures, primitive value types, character strings, and input/output
Be familiar with writing Java programs using core language features including interfaces, classes, inheritance, and assertions
Be familiar with writing Java programs that use software components similar to (but simplified from) those in the Java collections framework
Be familiar with using an understanding of the difference between value types and reference types to trace the execution of simple Java code in situations involving both flavors of types, including their use as parameters to method calls
Be familiar with testing using JUnit
Be familiar with illustrating key dependencies between software components using UML class diagrams (or similar)
Be familiar with using the most important features of a modern IDE, e.g., Eclipse

Course Topics

Topic	Lec	Rec	Lab	Cli	IS	Sem	FE	Wor
Introduction to Java; value types; control structures; basic input/output; introduction to Eclipse	6.0		2.0					
Software components; packages; interfaces; design-by-contract; classes; reference types; methods, calls, and parameter passing; equals and toString methods; Text component; Natural component; introduction to UML class diagrams (or similar)	9.0		3.0					
Layered implementations of new Text and Natural methods; introduction to recursion; introduction to specification-based testing and JUnit	6.0		2.0					
Generics; Sequence component; Queue component; Stack component; List component; layered implementations of new Sequence, Queue, Stack, and List methods; more recursion	6.0		2.0					
Set component; Map component; iterators	6.0		2.0					
Layered data representation concepts; representation invariants and abstraction functions; Natural representation using a Stack; Sequence/Queue/Stack representation using a List	6.0		2.0					

Representative Assignments

Layered methods for Text (e.g., replaceSubstring, concatenate, containsSubstring, removeFirstWord, ...)
Layered methods for Natural (e.g., multiply, power, root)
HTML table generator from series of heart-rate measurements

HTML "pretty-print" generator for some simple language (e.g., ancient mark-up for italics, bold, etc.)
HTML glossary generator, with internal links from definitions to other terms appearing in them
Sequence representation using two Stacks

Grades

Aspect	Percent
Homework and Class Participation	8%
Closed Labs	12%
Programming Lab Assignments	30%
Midterm Exam	20%
Final Exam	30%

Representative Textbooks and Other Course Materials

Title	Author
On-line reference materials	

ABET-EAC Criterion 3 Outcomes

Course Contribution		College Outcome
***	a	An ability to apply knowledge of mathematics, science, and engineering.
*	b	An ability to design and conduct experiments, as well as to analyze and interpret data.
***	c	An ability to design a system, component, or process to meet desired needs.
	d	An ability to function on multi-disciplinary teams.
**	e	An ability to identify, formulate, and solve engineering problems.
	f	An understanding of professional and ethical responsibility.
*	g	An ability to communicate effectively.
	h	The broad education necessary to understand the impact of engineering solutions in a global and societal context.
*	i	A recognition of the need for, and an ability to engage in life-long learning.
	j	A knowledge of contemporary issues.
***	k	An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

BS CSE Program Outcomes

Course Contribution		Program Outcome
***	a	an ability to apply knowledge of computing, mathematics including discrete mathematics as well as probability and statistics, science, and engineering;
*	b	an ability to design and conduct experiments, as well as to analyze and interpret data;
***	c	an ability to design, implement, and evaluate a software or a software/hardware system, component, or process to meet desired needs within realistic constraints such as memory, runtime efficiency, as well as appropriate constraints related to economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability considerations;
	d	an ability to function on multi-disciplinary teams;
**	e	an ability to identify, formulate, and solve engineering problems;
	f	an understanding of professional, ethical, legal, security and social issues and responsibilities;

Course Contribution		Program Outcome
*	g	an ability to communicate effectively with a range of audiences;
	h	an ability to analyze the local and global impact of computing on individuals, organizations, and society;
*	i	a recognition of the need for, and an ability to engage in life-long learning and continuing professional development;
	j	a knowledge of contemporary issues;
***	k	an ability to use the techniques, skills, and modern engineering tools necessary for practice as a CSE professional;
**	l	an ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
*	m	an ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices;
***	n	an ability to apply design and development principles in the construction of software systems of varying complexity.

Prepared by: Bruce Weide