CSE 2231.01 (Approved): Software II-Transition: Software Development and Design

Course Description

Transition from quarters to semesters, Resolve/C++ to Java; data representation using hashing, search trees, and linked data structures; sorting; using trees for language processing; component interface design; best practices in Java.

Prior Course Number: Part of CSE 201, CSE 321, and CSE 421
Transcript Abbreviation: SW II: Dev & Dsgn
Grading Plan: Letter Grade
Course Deliveries: Classroom
Course Levels: Undergrad
Student Ranks: Freshman, Sophomore
Course Offerings: Autumn
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: No
Admission Condition: No
Off Campus: Never
Campus Locations: Columbus
Prerequisites and Co-requisites: CSE 2222 or CSE 222 or CSE 222H; co-req: CSE 2321
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

This is a transition course to be offered ONLY in Au 2012. It is intended for students who have completed CSE 222 (prior to Su 2012) or who have completed CSE 2222 (in Su 2012).

Course Goals

Be competent with using design-by-contract principles and related best practices, including separation of abstract state from concrete representation
Be competent with using interface contracts, representation invariants, and abstraction functions that are described using simple predicate calculus assertions with mathematical integer, string, finite set, and tuple models
Be competent with extending existing software components by layering new operations on top of existing operations
Be competent with layering new software components' data representations on top of existing software components
Be familiar with the reasons for designing software to minimize the impact of change, and why it is difficult to achieve this
Be competent with using simple recursion
Be competent with using simple techniques to debug application software, layered implementations of extensions, and typical data representations
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 illustrating key dependencies between software components using UML class diagrams
Be familiar with using basic algorithm analysis techniques and notations to analyze and express execution times of operations whose implementations involve straight-line code and simple loops, and simple recursion (e.g., in manipulating binary trees)
Be competent with writing Java programs in a procedural style using the basic control structures, primitive value types, character strings, and input/output
Be competent 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 competent with writing Java programs using core language features including interfaces, classes, inheritance, and assertions
Be familiar with using JUnit for testing
Be competent with writing Java programs that use software components similar to (but simplified from) those in the Java collections framework
Be familiar with using the most important features of a modern IDE, e.g., Eclipse
Be familiar with working as part of a team on a software project with multiple milestones
Be exposed to using a version control system, e.g., CVS or SVN

Course Topics

Topic	Lec	Rec	Lab	Cli	IS	Sem	FE	Wor
Introduction to Java; value types; main differences between Java and RESOLVE/C++; Java's control structures; basic input/output; introduction to Eclipse	3.0		1.0					
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	6.0		2.0					
Layered implementations of new Text and Natural methods; recursion; introduction to specification-based testing and JUnit	3.0		1.0					
Generics; Sequence component; Queue component; Stack component; List component; layered implementations of new Sequence, Queue, Stack, and List methods; more recursion	3.0		1.0					
Array component; Set and PartialMap representations using an Array of Queues with hashing; representation using a BinaryTree with binary search tree algorithms	4.0		1.5					
Linked representations of Stack/Queue/List components and variations; singly-linked and doubly-linked lists;	4.0		1.5					
Tree component; language processing using trees; elaboration of small programming language compiler team project (with related programming lab assignments continuing beyond this module); introduction to version control	4.0		1.5					

Topic	Lec	Rec	Lab	Cli	IS	Sem	FE	Wor
Component interface design principles and practices	4.0		1.5					
Advanced Java language constructs and uses; best practices in Java	9.0		3.0					
Review and exams	3.0		1.0					

Representative Assignments

PartialMap representation using a BinaryTree with binary search tree algorithms
List representation using a doubly-linked-list data structure
Various components of a simple programming language compiler
Simple component design (including interface contract) to meet stated requirements

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;
*	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: Neelam Soundarajan