Computer Science Department
cs.salemstate.edu

CSC 110 Software Design and Programming I **4 cr. DII**

Instructor: TBA **Office:** location **Phone:** (978) 542-extension
email: TBA@salemstate.edu **Office Hours:** days and times

Section	Time	Room	Final Exam
nn	days and times	location	date and time
Lnn	days and times	location	

Catalog description:

This course introduces a set of fundamental design principles and problem-solving techniques for the development of computer algorithms and their implementation as programs. Problem solutions are developed with the help of an appropriate modeling language and then coded in an object-oriented programming language. (Consult the Computer Science Department for the languages and tools currently in use.) Topics such as problem specification, object-oriented analysis and design, standard data types, control structures, methods and parameter passing, and design for reuse are presented through a study of specific example problems and solutions. Style, documentation, solution robustness, and conformance with specifications are emphasized throughout. This course is intended for students majoring or minoring in Computer Science. Three lecture hours and three hours of scheduled laboratory per week plus extensive programming work outside of class.

Prerequisites: High school algebra I & II, plus experience with a window-based operating system and the use of email and a word processor. Not available to students who have received credit for ITE 210. Limited to Computer Science majors and minors or permission of the Department Chairperson.

Course Goals:

The purpose of this course is to develop students' understanding of a coherent set of tools and techniques for creating computer solutions to simple problems in data manipulation. Upon completion of the course, a student should be able to do the following:

CG01: analyze a problem statement for completeness and clarity;
CG02: use the methodology of object-oriented design to develop class designs (data descriptions and methods) for a problem solution;
CG03: convert this solution into source code in the designated high-level programming language in accordance with a well-defined set of style rules;
CG04: debug and test the program;
CG05: provide clear documentation for the result.

Course Objectives:

Upon successful completion of this course the student will have:

CO01: demonstrated knowledge of the syntax elements of an object-oriented programming language;
CO02: gained experience in analyzing problem statements for completeness and consistency;
CO03: practiced standard techniques of problem analysis;
CO04: applied the fundamentals of object-oriented design methodology;
CO05: learned and utilized simple techniques for validation and verification of programs;
CO06: created full documentation for several completed projects.

Student Outcome vs. Course Objectives matrix

SO	CO01	CO02	CO03	CO04	CO05	CO06
SO-1	✓	✓	✓	✓	✓	✓
SO-2	✓	✓	✓	✓	✓	✓
SO-3						
SO-4						
SO-5						
SO-6	✓	✓	✓	✓	✓	✓

Notes:

SO-1: Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

SO-2: Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

SO-3: Communicate effectively in a variety of professional contexts.

SO-4: Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

SO-5: Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. Apply computer science theory and software development fundamentals to produce computing-based solutions.

SO-6: Apply computer science theory and software development fundamentals to produce computing-based solutions.

Topics (using Java and UML):

- the "problem-solving universe"
  - operational definition of computer (specifically, electronic stored-program digital computer)
  - components of a typical computer
  - fundamental computer capabilities
- strategies and tools for problem-solving **SDF1(1.5), SE3(0.5), SE5(0.5)**
  - formulating precise specifications for a problem and its solution
  - algorithms
  - modular design of user requirements in measurable units
  - preconditions and postconditions
  - specification of user requirements in measurable terms
- programming languages and programming language paradigms **PL1(0.5)**
- brief history and overview of the Java language **PL1(0.5), PL1(0.5)**
- data types **SDF2(2.0)**
  - basic data types: integer, real, character, boolean
  - literals of each type
- variables and constants **SDF2(2)**
- reference types (including String and pre-defined wrapper classes) **SDF2(3.0)**
- console output **SDF2(0.5)**
- console input **SDF2(0.5)**
- simplified graphical user interfaces (GUIs) **HC1(2), SE2(1)**
  - JOptionPane **HC1(1)**
- object-centered problem analysis **PL1(1.5), SE1(1.5)**
- object-centered design and implementation **PL6(1.5), SE1(1.5)**
- introduction to UML (Unified Modeling Language) **SE3(1.5)**

- classes **PL1(4.0), SE2(0.5)**
  - overview
  - attributes
  - methods **SDF2(1.5)**
    - parameter and argument lists, return values, signatures
    - use of modular design in creating methods
    - visibility rules (scope, context) **PL4(0.5)**
  - objects (instances) **PL1(0.5)**
    - handles
    - copying objects (shallow vs. deep copies, clones)
- selection control structures **SDF2(4.5)**
  - review of Boolean expressions
  - single and double alternative structures
  - multiple-choice (switch) structure
- testing and verification **SE6(0.5)**
- debugging **SE2(0.5)**
- repetition control structures (loops) **SDF2(1), PL4(0.5)**
  - *while* and *do while* structures
  - *for* structure
- object-oriented program design techniques **PL6(2), PL6(1), SE1(1), SE7(0.5)**
  - objects as self-contained entities
  - objects as entities that act upon themselves
    - contrast to other design methodologies in which external agents act upon objects
  - programming for reusability
- collections (conceptual discussion) **PL1(0.5)**
- arrays of one-dimension **SDF3(1.5)**
  - syntax rules
  - static nature of arrays; physical vs. logical size of an array
  - common algorithms: **AL2(1), AL3(2)**
    - storing a value in an array
    - removing a value from an array
    - linear traversal
    - linear search

Note: Programming language features, techniques, and aspects of object-orientation not mentioned above (for example: sorting, binary search, multidimensional arrays, stream and file I/O, inheritance, polymorphism) are normally covered in the subsequent courses CSC 115 Software Design and Implementation II and CSC 260 Data Structures and Algorithms.

Programming assignments: Ten to twelve programming assignments are given. One or more of these may be group projects. Each programming assignment normally involves the design, writing, testing and debugging of a program and the submission of an appropriate laboratory report. Each assignment has a specific due date, with a short grace period during which the assignment may be submitted for reduced credit. When the grace period has expired, the assignment will no longer be accepted.

All programs must be coded in the programming language currently used for instruction in the CSC110/115 sequence no exceptions will be allowed. The version of the language being used will be the currently accepted standard version: any extensions or variations in student-owned compilers must be approved in advance by the instructor, who may choose to forbid their use.

Laboratory exercises: There will be short programming exercises to be completed during weekly scheduled laboratory sessions. Each exercise focuses on a specific language feature or programming technique presented in recent lectures. Performance on these exercises will be incorporated into the course grade.

Exams and quizzes: There will be two examinations, two shorter quizzes, and a comprehensive written two-hour final examination.

Grading: Final grades will be determined on the basis of the following approximate weights: examinations and quizzes - 40%, programming assignments and lab exercises - 60%.

Course Objective / Assessment Mechanism matrix

	Test / Quiz Questions	Homework Problems	Programming Projects	Lab Exercises
CO01	✓		✓	✓
CO02	✓	✓	✓	✓
CO03	✓	✓	✓	✓
CO04	✓	✓	✓	✓
CO05	✓	✓	✓	✓
CO06			✓	

Bibliography:

Bravaco, Ralph; Simonson, Ralph; Simonson, Shai. **Java Programming: From the Ground Up.** McGraw-Hill, 2009.

Carrano, Frank. **Imagine! Java: Programming Concepts in Context. First Edition**. Prentice Hall, 2010.

Dale, Nell; Weems, Chip; Headington, Mark. **Programming and Problem Solving With Java**. **Second Edition.** Jones & Bartlett, 2007.

David J. Barnes and Michael Kolling. **Objects First with Java: A Practical Introduction Using BlueJ.** Sixth Edition. Pearson Publications 2016.

Deitel, Harvey; Deitel, Paul. **Java How to Program: Early Objects Version. Eleventh Edition.** Prentice Hall, 2017.

Flanagan, David. **Java in a Nutshell: A Desktop Quick Reference**. **Seventh Edition.** O'Reilly,2019.

Gaddis, Tony. **Starting Out with Java™: Early Objects, Sixth Edition.** Addison-Wesley, 2017.

Gaddis, Tony. **Starting Out with Java: From Control Structures through Objects. Sixth Edition**. Addison Wesley, 2019.

Gaddis, Tony. **Starting Out with Java: From Control Structures through Data Structures. Fourth Edition**. Addison Wesley, 2018.

Horstmann, Cay. **Big Java: Early Objects**. **Seventh Edition.** John Wiley & Sons, 2019.

Hosch, Frederick A.; Nino, Jaime. **An Introduction to Programming and Object-Oriented Design Using Java**. **Third Edition.** Wiley, 2008.

Liang, Y. Daniel. **Introduction to Java Programming, Comprehensive Version. Eighth Edition.** Prentice Hall, 2010.

Lewis, John; Loftus, William. **Java Software Solutions: Foundations of Program Design**. **Eleventh Edition.** Addison-Wesley, 2017.

Oaks, Scott; Wong, Henry. **Java Threads: Understanding and Mastering Concurrent Programming**. **Third Edition.** O'Reilly Media, 2004.

Schildt, Herbert. **Java: The Complete Reference**. **Eleventh Edition.** Osborne/McGraw-Hill, 2018.

Academic Integrity Statement:

"Salem State University assumes that all students come to the University with serious educational intent and expects them to be mature, responsible individuals who will exhibit high standards of honesty and personal conduct in their academic life. All forms of academic dishonesty are considered to be serious offences against the University community. The University will apply sanctions when student conduct interferes with the University primary responsibility of ensuring its educational objectives." Consult the University catalog for further details on Academic Integrity Regulations and, in particular, the University definition of academic dishonesty.

The Academic Integrity Policy and Regulations can be found in the University Catalog and on the University website (http://catalog.salemstate.edu/content.php?catoid=13&navoid=1295#Academic_Integrity). The formal regulations are extensive and detailed - familiarize yourself with them if you have not previously done so. A concise summary of and direct quote from the regulations: "Materials (written or otherwise) submitted to fulfill academic requirements must represent a student's own efforts". *Submission of other's work as one's own without proper attribution is in direct violation of the University's Policy* and will be dealt with according to the University's formal Procedures. *Copying without attribution is considered cheating in an academic environment - simply put,* ***do not do it!***

University-Declared Critical Emergency Statement:

In the event of a university-declared emergency, Salem State University reserves the right to alter this course plan. Students should refer to www.salemstate.edu for further information and updates. The course attendance policy stays in effect until there is a university-declared critical emergency.

In the event of an emergency, please refer to the alternative educational plans for this course, which will be distributed via standing class communication protocols. Students should review the plans and act accordingly. Any required material that may

be necessary will have been previously distributed to students electronically or will be made available as needed via email and/or Internet access.

Equal Access Statement:

"Salem State University is committed to providing equal access to the educational experience for all students in compliance with Section 504 of The Rehabilitation Act and The Americans with Disabilities Act and to providing all reasonable academic accommodations, aids and adjustments. **Any student who has a documented disability requiring an accommodation, aid or adjustment should speak with the instructor immediately.** Students with Disabilities who have not previously done so should provide documentation to and schedule an appointment with the Office for Students with Disabilities and obtain appropriate services."

Note: This syllabus represents the intended structure of the course for the semester. If changes are necessary, students will be notified in writing and via email.