Computer Science Department
cs.salemstate.edu

SYLLABUS **Fall, 2022**

CSC 115 Software Design and Programming II **4 cr. DII**
Prerequisite(s): CSC110 or ITE210

Instructor: Beifang Yi **Office**: MH 208B **Phone**: (978) 542-7246
Email: byi@salemstate.edu **Hours**: TWRF (12:25-1:40) **Website**: http://weblab.salemstate.edu/~byi/

Section	Time	Room	Final Exam
01	T & R 1:40-2:55pm	MH 346	**December 20, Tuesday, 2:00pm-4:00pm MH 346**
L21	T & R 3:05-4:20 pm	MH 210	
Office Hours		12:25—1:40 (Tuesday, Wednesday, Thursday, Friday), MH208B	

Catalog description:

This course extends the treatment of object-oriented methodologies, languages and tools begun in CSC110. The emphasis is on the analysis of complex problems, particularly those involving multiple design alternatives, and the use of class libraries. Fundamental strategies for algorithm design are presented and discussed. Specific topics include inheritance, polymorphism, recursion, stream and file I/O, exceptions, and graphical interface programming. Style, documentation, solution robustness, and conformance with specifications are emphasized throughout. Three lecture hours and three hours of scheduled laboratory per week, plus extensive programming work outside of class

Prerequisite: CSC110 or 201J.

Goals:

The purpose of this course is to enhance and extend students' understanding of tools and techniques for object-oriented software development. 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 diagrams (data descriptions and methods) for a problem solution;
CG03: demonstrate understanding of and apply fundamental strategies for algorithm design;
CG04: convert this solution into source code in the designated high-level programming language in accordance with a well-defined set of style rules;
CG05: debug and test the program;
CG06: provide clear documentation for the result.

Objectives:

By the end of the course students will have:
CO01: gained a deeper understanding of object-oriented design methodology;
CO02: learned to recognize situations in which multiple design alternatives are possible;
CO03: applied fundamental algorithm design strategies;
CO04: learned to recognize and apply design patterns;
CO05: learned and utilized techniques for validation and verification of programs;
CO06: gained experience in judging the effectiveness and cost of a software design;
CO07: gained experience in choosing among competing design alternatives;
CO08: gained experience in the use of the UML modeling language;
CO09: extended their knowledge of an object-oriented programming language, including graphical user interfaces, event-driven programs, file-based input/output, and the use of libraries;

CO10: produced full documentation for multiple completed projects, including formal class diagrams;
CO11: participated in one or more group projects.

Course Topics:

A detailed topics list and a general course bibliography can be found from the Computer Science Department; contact the department office or Computer Science Curriculum Director more information.

Text:

(Required) Java How to Program: Early Objects, 11th Edition, by Deitel & Deitel. Prentice-Hall, 2017 (ISBN: 978-0-13-474335-6).

Required Material:

(Required) Thumb (flash) drive, 16 GB minimum or online storage (for saving your projects and coursework) in addition to your personal computer/laptop.

Additional references:

- Course teaching materials: http://weblab.salemstate.edu/~byi/CSC201J_202J_Deitels/index.html .
  - This website is password-protected and ask the instructor of the password (or log into Canvas for it).
- Course online system (Canvas): https://elearning.salemstate.edu/.
  - Access to this site via the username and password given/assigned by SSU.
  - Detailed course information, such as all types of assignments, including labs, projects, tests, teaching materials and references, and etc., is available on CSC115 through Canvas.

Software:

(Required) Java SE (version 8 or later). Download the JDK (i.e., Java Development Kit) at https://www.oracle.com/java/technologies/javase-downloads.html. You may use your preferred IDE for working on Java programming assignments.

Cell phones:

Turn the ringer off, or, better yet, *turn the phone off*.

Lecture/Lab Attendance:

Class policy is that of the Registrar's office - see the University catalog for details. Lecture will start promptly at the scheduled time, so please make a serious effort to not be late. While class attendance does not *directly* affect your overall final grade, some of the material covered in class is not found (in the same form) in the text, so class attendance and notes are very important. Note that you are at all times responsible for materials and assignments discussed in class. We will use SSU's online course management system, Canvas (https://elearning.salemstate.edu/) to post all types of assignments (including different formats/types of tests), grades, and announcements regarding the course topics and progress. You will need to visit Canvas (with your SSU Navigator use-name and password) for the course activities. Canvas uses your ***SSU-stored email*** box for the communication between you and the instructor and thus you ***must use this email*** address. Each student is responsible for completing all course requirements and for keeping up with all that goes on in the course (whether or not the student is present).

Class attendance is **strongly recommended**. **Lab exercises, some of the programming assignments, and some mini-tests, will be tested, administered, and graded in the lab hours**. **Some mini-tests will be held in both class and lab hours or other times (online).**

Some of the class/lab hours will be used to review important course topics, to discuss and investigate Java implementation details that time may not permit to be fully presented in the texts (for design and implementation drills, for programming exercises) to assist with design and implementation problems that arise in project exercises, to discuss and review important topics/questions regarding the tests, and to check/examine/grade the exercises and homework.

Student-Instructor Communication:

If you have any questions regarding course material, and *in particular if you are having problems with a programming assignment*, the most effective way to get assistance is to *discuss with the instructor* (*either in the class, in the lab, or outside the classroom*).

Please **note**: Canvas is used for submission of the assignments/labs/projects and for posting grades. If you ask questions through

Canvas-grading-submission features, they usually do ***not*** go to the instructor directly. Please ask questions in the class/lab or send emails (you may send email via Canvas) to the instructor!

Final Grade:

Final grade will be determined using the following grading weights:

written assignments	8%
lab exercises	26%
programming projects	26%
mini-tests	20%
final examination	20%

Attendance is not used to calculate the final grade; however, note that you are at all times responsible for all types of assignments and materials presented in class.

The following table shows how the course work is assessed against the course objectives:

	Tests	Assignments (including programming projects)	Lab Exercises
CO01	✓	✓	✓
CO02	✓	✓	✓
CO03	✓	✓	
CO04	✓	✓	✓
CO05	✓	✓	✓
CO06	✓	✓	✓
CO07		✓	✓
CO08	✓	✓	✓
CO09		✓	
CO10		✓	
CO11		✓	

Written Assignments:

There will be various types of written assignments.

These assignments are designed to help understand the course topics, prepare for programming practices, and get prepared for the tests (i.e., the mini-tests and final examination). And **more important**: many of the test questions will be much like these assignment questions! Reading of the text related with the assignment topics must be done before working the assignment questions.

The assignments will be assigned from the text on a regular basis: for the maximum benefits; do the readings before the material is covered in class, and you will find that most of the assignment questions are based on the reading of the text.

There will be various types of questions (such as true-false, fill-in-black, multiple-choice, short-answer questions) which are designed to test your understanding of the course topics and which will prepare for your tests.

Lab Exercises:

Lab exercises are designed not only for help on the understanding of the course topics but also for the preparation of the programming projects. Lab exercises must be completed**, tested and graded** (by the instructor) **in the lab hours**. Lab submissions to Canvas only (by the deadline) does ***not*** guarantee lab credits—you need to have them tested by the instructor by the deadline.

Please note that lab exercises and programming projects are *different* assignments; the credits earned from the programming projects do not count for your lab assignment grades, and vice versa.

Programming Projects:

Programming projects will be assigned throughout the semester. *Most of them will have pre-lab activities to be completed prior to the implementation of the assigned tasks*. They *will definitely require* ***significant*** *programming time outside of scheduled lab*. Programming assignments may have different full score points, depending on the difficulty and the amount of the work of the exercises.

Each programming project must be submitted at Canvas (https://elearning.salemstate.edu/) and **must be tested/examined /checked in the lab hours by the instructor for full credits** of the project.

Please note: to get *full credits* for each project, **you must (1) submit your solution (that is program source code for programming projects) to Canvas by the deadline, (2) have the instructor test your code in the lab hours, (3) answer the questions provided by the instructor correctly during the testing, (4) and modify/update your code correctly as a solution for a programming question that is very similar to the one you have submitted/shown for the project.** Also note that the instructor will post the assignment or project grades regularly (usually within one week after the assignment/project's due date) on Canvas and you are encouraged to check your assignment/project grades regularly.

- Given the lengthy steps of testing your solution code, please do submit your answers by the deadline and be sure that you are very familiar with your solution—the Java code of *your own work*!
- If the submitted Java code cannot get compiled, **zero** (0) points will be assigned as the grade.
- If during the test you cannot answer the provided questions correctly or make necessary correct changes upon your submission for a varied problem (which is very similar to the project problem), your scores for that project will be severely affected (which can be as low as **0 points**).
- Testing and grading will be proceeded *only after* the deadline. If the testing is done and the grading is posted and you would improve on your grades by submitting new work, the updated grades will be based on the new submission time (i.e., subject to late-submission penalty). So please start working on the projects early and have the instructor test your solutions well before the deadline and you may have chance of improving on the projects (based on the test feedback) and also on the grades.
- **Important**: your submission must follow **project submission requirements** which will be given in class/Canvas; otherwise, you may get **as low as zero** for your project.

Tests (mini-tests and final examination):

There will be mini-tests administered throughout the semester and one *comprehensive* final examination. Please refer to Final Grade above for the grading weights of these tests.

The mini-tests will be like closed-book written quizzes (with questions in the form of multiple-choice, true-false, fill-in-blank, and short-answer questions) and programming problems. The tests must be completed during the lab hours or other designated time periods. The programming mini-tests will be much like lab exercises and/or programming projects on a smaller scale.

The final examination will be a comprehensive examination, i.e., covering all the topics given in the class with emphasis on assignments (reading/writing) and programming skills/algorithms.

Missed Tests:

Tests (mini-tests and final-exam) may not be made up except for *documented/emergency* situations. If a test must be made up, arrangements must be made with the instructor to take the test before it is discussed in class (usually within a week of the test being administered).

Due Dates/Time:

- Late submission of assessed assignments, labs, or projects will result in penalties defined in the following:
  - **25 percentage** points will be deducted for being late, up to 24 hours.
  - **50 percentage** points will be deducted for being late, more than 24 hours and up to 48 hours.
  - **75 percentage** points will be deducted for being late, more than 48 and up to 72 hours.
  - **100 percentage** points will be deducted for being late more than 72 hours (i.e., a grade of zero will be given).
  - A grade of **zero** will be given for any assignments including labs or projects which are submitted after the final examination time. That is: **no assignments (including labs/projects/semester-project) will be accepted after the final examination**.
- Should there be an emergency that prevents you from completing/submitting your assignments/labs/projects on time, you will need to send *an email request* for the extension on the coursework submission. The instructor will reply to this request email with a specific number of days for the extension period or a new deadline for the assignment and you will need to keep *this email as a record of the extension approval*. Sending only a request email does ***not*** guarantee the extension approval.
- **Please double-check** your submissions (since your assignment submissions are usually graded after their deadlines): to guarantee your successful and correct submissions, you would need to download your submissions and examine the

downloaded materials; as for the projects, you would need to check the downloaded programs and then compile/run the code.

Study Groups:

While I strongly encourage study groups, I require that students hand in their answers in their own words - if two answers are highly similar to each other, **neither** will receive credit.

When working on your programming projects, you may discuss with others the project topics, the algorithms and methodologies related to the project; but when you work on writing the code, this coding work should be 100% of your own work. **If two answers/written code segments come out exactly the same or highly similar, neither will receive credit and/or further actions will be taken** (such as reporting to the department and/or university).

Academic Integrity:

Academic Integrity Policy and Regulations can be found in the University Catalog and on the University's 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.

All students are expected to be familiar with the academic regulations, including those regarding Academic Integrity, for Salem State University as published in the college catalog. In addition, each student is responsible for completing all course requirements and for keeping up with all that goes on in the course (whether or not the student is present).

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 Disability Services and obtain appropriate services.

Students must comply with any Covid-19 Health and Safety Protocols for the 2022-2023 Academic Year. Students should review the information found at https://www.salemstate.edu/covid19.

In the event of a university declared critical emergency, Salem State University reserves the right to alter this course plan. Students should refer to Salem State 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 located at Canvas (https://elearning.salemstate.edu/) where the course contents such as announcements, assignments are posted. Students should review the plans and gather all required materials before an emergency is declared.

Please remember that if, for any reason, you decide to drop this course, you **MUST** do so officially through the Registrar's office. The last day to withdraw from a course this semester is **November 28th**.

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 all regular class communication mechanisms (class discussion, emails, and/or the course link at Canvas https://elearning.salemstate.edu).

Course Schedule for CSC115

(Subject to Change)

Week	Dates	Contents (textbook chapters and others)
1	9/7—9/9	Programming Basics (Ch 1, 2, 3, 4)
2	9/12—9/16	Design & Implementation Basics (Ch 5, 6, 7)
3	9/19—9/23	Design & Implementation Basics (Ch. 8) OOP: Inheritance (Ch 9)
4	9/26—9/30	OOP: Inheritance (Ch 9)
5	10/3—10/7	OOP: Polymorphism & Interface (Ch 10)
6	10/10—10/14	OOP: Polymorphism & Interface (Ch 10)
7	10/17—10/21	Exceptions & Exception Handling (Ch 11 & others)
8	10/24—10/28	Java GUI (Ch 2, 3, …, 11)
9	10/31—11/4	Java GUI (Ch 2, 3, …, 11) Strings, Characters (Ch14)
10	11/7—11/11	Files, I/O Streams (Ch15)
11	11/14—11/18	Generic Collections (Ch16) Lambdas and Streams (Ch17)
12	11/21—11/25	Recursion (Ch18) **(Thanksgiving Recess)**
13	11/28—12/2	Searching, Sorting (Ch19)
14	12/5—12/9	Generic classes/Methods: A Deeper Look (Ch20) Custom Generic Data Structures (Ch21)
15	12/12—12/13	Applets & Junit Reviews
15/16	12/15—12/22	**Final Examination** Dec 20th , Tuesday, 2:00pm—4:00pm, MH346