COMP 142 — Computer Science II: Objected-Oriented Programming — Fall 2023
CRNs 13612/13613

Instructor:	Nate Phillips
Times:	(Mon/Wed 10–10:50 and Fri 9–10:50) or (Mon/Wed 11–11:50 and Fri 11–12:50)
Classroom:	Briggs 001
Course website:	http://www.cs.rhodes.edu/142
Email:	phillipsn@rhodes.edu (please include "CS 142" somewhere in the subject)
Office:	Briggs 210
Office hours:	See website for scheduled office hours. I am also available by appointment.

Official Course Description: An introduction to the fundamental concepts and practices of object-oriented programming. The object-oriented programming paradigm is introduced, with a focus on the definition and use of classes as a basis for fundamental object-oriented program design. Other topics include an overview of programming language principles, simple analysis of algorithms, basic searching and sorting techniques, and an introduction to software engineering issues.

Unofficial Course Description: CS142 is the second course in the sequence for computer science majors or minors and ideally should be taken immediately after CS141. CS142 offers a new perspective on software design through an introduction of the object-oriented paradigm. Special emphasis is placed on the process of building hierarchies of abstractions to hide implementation details through a careful and systematic analysis of problems of moderate complexity. Various design approaches will be explored with the goal of identifying the situations for which each approach is applicable.

This course will use the Java programming language as the vehicle for exploration of fundamental computer science concepts. However, this is not a course about Java; it is about the structure and interpretation of computer programs.

The particular Java environment that will be used in this course is available in the computer labs on Rhodes College campus, and can also be installed on personal computers. You are free to develop the code for the assignments on your own computer using an environment of your choice. However, keep in mind that the source code that you submit for assignments must compile and run successfully using the environment from class.

Course Objectives: At the end of this course, you should be able to

- Analyze problems of moderate complexity and solve such problems by writing code using the Java programming language,
- Apply principles of good program design, especially the uses of data abstraction and modular program composition,
- Understand the fundamental design, analysis, and implementation of basic data structures and algorithms,
- Assess how the choice of data structures and algorithm design methods impacts the performance of programs,
- Apply your programming skills to diverse applications in science and engineering

Course Topics: (not necessarily in this order)

- Java basics: syntax, data types, arrays, strings, functions, file reading
- Objects and classes
- Object-oriented design
- Inheritance and polymorphism
- Recursion
- Asymptotic analysis
- Linked lists
- Optional topics as time permits: generics, other data structures (stacks, queues)

Text: There is no required textbook for this class. Materials will be distributed in class or online. That said, the following are recommend resources for the class:

- Introduction to Java by Liang (textbook)
- Introduction to Programming in Java by Sedgewick and Wayne (textbook)
- Official Java tutorials at https://docs.oracle.com/javase/tutorial/
- Introduction to Programming Using Java (free online textbook) at http://math.hws.edu/javanotes/index.html

Prerequisites: The course assumes successful completion of CS141 or significant programming experience. Please come see me if you have not had CS141.

Coursework:

	Tentative weight	Tentative date
Programming projects	40%	
Labs and homework	20%	
Midterm 1	12.5%	Monday, October 3, in class
Midterm 2	12.5%	Monday, November 14, in class
Comprehensive final exam	15%	Mon, Dec 12, 8:30am (10am M/F section) Fri, Dec 9, 1pm (11am M/F section)

Grades of A–, B–, C–, and D– are guaranteed with final course grades of 90%, 80%, 70%, and 60%, respectively. If your final course grade falls near a letter grade boundary, I may take into account participation, attendance, and/or improvement during the semester.

Assignments are due at 11:59pm CST unless otherwise specified. In general, late will work will not be accepted without an extension obtained prior to the due date.

Office Hours: In addition to regular office hours, I am also available immediately after class for short questions. You never need an appointment to see me during regular office hours; you can just come by. Outside of regular office hours, feel free to stop by my office, and if I have time, I'll try to help you. If I don't have time at that moment, we'll set up an appointment for a different time. Don't be shy about coming by my office or sending me email if you can't make my regular office hours. I set aside time each week for "unscheduled" office hours.

Attendance: Attendance is expected for each class. If you know ahead of time that you will be absent from class for any reason, please discuss this with me at the beginning of the semester or as early as possible. Otherwise, if your attendance deteriorates, you will be referred to the dean and asked to drop the course. Attendance, participation, and apparent overall

improvement trend may be considered in assigning a final grade. Attendance will be checked each class lecture period. After five unexcused absences, each additional absence will reduce the final grade for the course by one letter grade.

Workload: It is important to stay current with the material. You should be prepared to devote at least 2–3 hours outside of class for each in-class lecture. In particular, you should expect to spend a significant amount of time for this course working on a computer trying example programs and developing programming assignments. Do not wait to the last minute to start your programming assignments.

You are encouraged to form study groups with colleagues from the class. The goal of these groups is to clarify and solidify your understanding of the concepts presented in class, and to provide for a richer and more engaging learning experience. However, you are expected to turn in your own code that represents the results of your own effort.

Programming Assignments:

- All programs assigned in this course must be written in Java, unless otherwise specified. When turning in assignments, submit only the Java source code files (`.java`); do not submit any files generated by the IDE (e.g., `.class`).
- Back up your code somewhere as you're working on your assignments. Computer crashes or internet downtime are not valid excuses for missing a deadline.
- Programming grades will be graded on correctness of the program output, efficiency and appropriateness of the algorithms used in the code, and style and documentation of the source code.
- Grades are assigned to programs as follows by this general guideline:
  - A (100 pts): The program is carefully designed, efficiently implemented, well documented, and produces clearly formatted, correct output.
  - A- (93 pts): The program is an 'A' program with one or two of the minor problems described for grade 'B.'
  - B (85 pts): The program typically could easily have been an 'A' program, but it may have minor/careless problems such as poor, inadequate, or incomplete documentation; several literal values where symbolic constants would have been appropriate; wrong file names (these will be specified per program assignment); sloppy code format; minor efficiency problems; etc. (This is not an exhaustive list.) You would be wise to consider 'B' the default grade for a working program — this might encourage you to review and polish your first working draft of an assignment to produce a more professional quality final version of your program.
  - C (75 pts): The program has more serious problems: incorrect output or crashes for important special cases (the "empty" case, the "maxed-out" case, etc.), failure to carefully follow design and implementation requirements spelled out in the assignment, very poor or inefficient design or implementation, near complete absence of documentation, etc.
  - D (60 pts): The program runs, but it produces clearly incorrect output or crashes for typical cases. Or, it may deviate greatly from the design or implementation requirements stated in the assignment description.

- F (35 pts): Typically, an 'F' program produces no correct output, or it may not even run. It may "look like a program" when printed as a hard copy, but there remains much work to be done for it to be a correct, working program.

Rules for Completing Assignments Independently

- Unless otherwise specified, programming assignments handed in for this course are to be done *independently*.
- Talking to people (faculty, other students in the course, others with programming experience) is one of the best ways to learn. I am always willing to answer your questions or provide hints if you are stuck. But when you ask other people for help, sometimes it is difficult to know what constitutes legitimate assistance and what does not. In general, follow these rules:
  - **Rule 1: Do not look at anyone else's code for the same project, or a different project that solves a similar or identical problem.**
    Details: "Anyone else" here refers to other members of the class, people who have taken the class before, people at other schools enrolled in similar classes, or any code you find online or in print. "Similar or identical problem" here should allow you to look at code that uses techniques applied in different situations that you can then adapt to your project. However, if you find yourself copying-and-pasting code or directly transforming code line by line to fit into your program, then that is considered plagiarism.
    Exception: You may help someone else debug their program, or seek assistance in debugging yours. However, this requires the person writing the code being debugged to have made a good-faith attempt to write the program in the first place, and the goal of the debugging must be to fix one specific problem with the code, not re-write something from scratch.
  - **Rule 2: Do not write code or pseudocode with anyone else.**
    Details: You must make a good faith effort to develop and implement your ideas independently before seeking assistance. Feel free to discuss the project *in general* with anyone else before you begin and as you're developing your program, but when you get to the level of writing code or pseudocode, you should be working independently.

A violation of these rules constitutes plagiarism and is not acceptable behaviour for Rhodes students. Such violations will be dealt with harshly and will have consequences that are reflected in your grades, which can include failing the assignment, dropping a letter grade for the entire course, or even failing the course. However, if you have any questions about what is acceptable for this course, please send me an email with the details relating to your specific case.

At any point, you may be asked to explain your code or ideas or to reflect on aspects of your coding style, approach, or the assignment itself. This self-analysis is part of your assignment, but, even more, thoughtful responses will help you better understand your own code and design practices!

The underlying idea is that course work and outside assistance should genuinely help you to learn the material (as opposed to just getting the assignment done). Programming assignments are graded as a benefit to you; they are your chance to show what you have learned under circumstances less stressful than an exam. In return, I ask only that your work fairly reflect your understanding and your effort in the course.

Coding Style: Designing algorithms and writing the corresponding code is not a dry, mechanical process, but an art form. When you design and write programs, it is important to write thoughtful code and comments so that you can understand your own work and answer these questions! In previous classes, you have mostly used short, targeted programs; this course represents the start of a transition to larger, more complex projects. As projects become larger, it becomes even more important to design clear code and documentation.

Programming assignments will be graded based on correctness and style. To receive full credit for graded programs, you must adhere to good programming practices. Therefore, your assignment must contain the following:

- A comment at the top of the program that includes the author of the program, the date or dates, and a brief description of what the program does
- Concise comments that summarize major sections of your code, along with a comment for each function in your code that describes what the function does.
- Meaningful variable and function names
- Well-organized code
- White space or comments to improve legibility
- Avoidance of large blocks of copy-and-pasted code

Class Conduct:

- I encourage everyone to participate in class. Raise your hand if you have a question or comment. Please don't be shy about this; if you are confused about something, it is likely that someone else is confused as well. Teaching and learning is a partnership between instructor and students, and asking questions not only helps you better understand the material, but also helps me better connect with you in the future.
- When in class, only use cell phones or other electronic devices for classwork and keep the volume on silent.
- If you cannot make it to class for whatever reason, make sure that you know what happened during the lecture that you missed. The information from that class is your responsibility. (A good way of doing this is to ask a classmate!)
- If you have to leave a class early, please inform the instructor in advance.

Additional policies: On the class webpage, there are additional class and college policies covering accommodations, academic integrity, diversity, sexual misconduct disclosure, and recording lectures.