CSCI 0150: Introduction to Object-Oriented Programming and Computer Science

Course Information and Syllabus
Semester I, 2022–2023

For all relevant course information and lecture zoom links, visit the course website: http://cs.brown.edu/courses/csci0150/. **Our Canvas page just links to the website.**

Lectures	K Hour: 2:30 pm – 3:50 pm on Tuesdays and Thursdays
Room	Salomon 101
Textbook	*None*
Prerequisite	*None*
Instructor	Andries (Andy) van Dam (avd@cs.brown.edu)
Office	CIT 465
Professor's Office Hours	Please reach out to Lisa Manekofsky (ljm@cs.brown.edu) to schedule an appointment
Head TAs	Abigail Marks (amarks3), Adam Mroueh (amroueh), Brandon Diaz (bdiaz2), Emily Hinds (ehinds3), Naafiyan Ahmed (nahmed21)
UTAs	Allie Masthay (amasthay), Anastasio Ortiz (aortiz18), Annabel Roth (aroth7), Ayman Benjelloun Touimi (abenjell), Cannon Caspar (ccaspar), Caroline Hwang (chwang15), Catherine Kim (ckim167), Charles Levy (clevy9), Claire Oberg (coberg1), Damian Wasilewicz (dwasilew), Daniel Fiume (dfiume1), Daniel Liu (dliu58), Daniel Segel (dsegel), Lexi Henrion (ehenrion), Emil Munteanu (emuntean), Fern Tantivess (vtantive), Gabi Becher (gbecher), Grace Marshburn (gmarshbu), Grant Landon (glandon), Haruka Masamura (hmasamur), Heidi Schaefer (hschaef1), Ishika Tulsian (itulsian), Javier Fernandez Garcia (jferna35), Jess Wan (jwan8), Joe Maffa (jmaffa), Juan Garcia (jgarci71), Keya Kilachand (kkilacha), Khaled Abdo (kabdo1), Kris Diallo (kdiallo2), Lynda Umuhoza (lumuhoza), Manny Quezada (mquezad1), Nadya Tan (ntan13), Naomi Park (npark), Orlando Cedeno (ocedeno), Patrick Peng (ppeng4), Pauline Nguyen (pnguye37), Richard Dong (rdong14), Sam Bernstein (sbernst6), Sarah Onderdonk (sonderdo), Sherry Zhang (szhan235), Sophie Zhang (szhan257), Taleena Chandaria (tchandar), Talia Sawiris (tsawiris), Thomas Bui (tbui12), Yabeke Zike (yzike), Yasmine Abdelaziz (yabdelaz), Zach Boston (zboston2), Zyn Yee Ang (zang),
STAs	Nadya Tan (ntan13), Fern Tantivess (vtantive)

TA Office Hours	https://cs.brown.edu/courses/cs015/hours
Time Requirements	In addition to 3 hours per week in class, CS0150 requires 10 sections, which are 120 minutes each. CS0150 also requires that you start working consistently from the time an assignment is handed out. Assignments are closely spaced and each assignment uses concepts from previous work. This makes it difficult to fall behind on one assignment and still complete the next one. Starting early is the key to successful programming in CS0150. Typically, students find that CS0150 requires about 15 hours of coursework a week, in addition to attending lectures.
Goals	This course introduces principles of computer science, emphasizing object-oriented design and programming in Java, an effective modern technique for producing modular, reusable, and internet-aware programs. It also introduces interactive 2D computer graphics, user interface design, some fundamental data structures and algorithms, and computational efficiency. A sequence of successively more complex graphics programs, including Tetris and culminating in a significant final project, helps provide a serious introduction to the field. Each assignment includes an interactive user interface. Therefore, most of the games you program can be played! However, this course is not focused on games nor game design. It uses games as a natural and fun vehicle for learning object-oriented programming and graphical user interfaces. The course takes an interactive approach to programming assignments and an equally interactive approach to lectures. Andy's lectures are supplemented by skits performed by the UTAs (Undergraduate Teaching Assistants) to teach course concepts and to make the class fun and enjoyable! This course is intended for both potential concentrators and those who may take only a single course. No prerequisites and no prior knowledge of programming and mathematics, beyond basic algebra, are required.
Diversity: All are Welcome	Our intent is that this course provide a welcoming environment and community for all students taking CS0150. Our TAs have undergone training in diversity and inclusion; all members of the CS community, including faculty and staff, are expected to treat one another in a professional manner. If you feel you have not been treated in a professional manner by any of the course staff, please contact either Prof. van Dam (the instructor; avd@cs.brown.edu), Ugur Cetintemel (Dept. Chair; roberto_tamassia@cs.brown.edu), Tom Doeppner (Vice Chair; twd@cs.brown.edu), Laura Dobler (diversity & inclusion staff member; laura_dobler@brown.edu), or Kathi Fisler (Associate Director of the CS Undergraduate Program; kfisler@cs.brown.edu). We take all complaints about unprofessional behavior seriously. CS0150 aims to be a community for students within the department, and cultivating an open and inclusive environment is a critical part of this effort.
TopHat	The course uses TopHat quiz questions to allow students to actively engage with material during synchronous lecture. These questions can be answered online or using an app, both of which are available on the TopHat website. If students need an electronic device to access TopHat, they can borrow a laptop from CIS, located in Page-Robinson 510. TopHat responses will be graded on a

	complete/incomplete basis. These responses will, in total, be worth 5% of your grade.
Collaboration Policy	CS0150 has a Collaboration Policy that provides specific guidelines for what you can and cannot do in regard to working with other students. This policy is based on Brown's Academic Code of Conduct, but it is specific to CS0150. Because CS0150 has no exams, we take our collaboration policy very seriously. This policy has two separate phases that will take place throughout the semester. For more detailed information, please read the full collaboration policy here.
Grading	Like many courses in the Computer Science department, CS0150 relies heavily on the role of its Undergraduate Teaching Assistants. In addition to holding approximately 170 TA office hours per week and facilitating programming labs, the CS0150 Undergraduate TAs also grade all student work, based on a carefully crafted grading rubric and with supervision from the Head TAs and Andy. Debugging and Conceptual Hours, held by your undergraduate peers who know the course and the assignments intimately, are there to ensure that you get targeted help with your problems. Your grade in this course will be based solely on your performance on the assignments and TopHat questions, as there are no tests, quizzes, papers, or final exams. Assignments are weighted, each growing in complexity and weight as the semester goes on. Once your work has been graded by a TA, with supervision from the head TAs, you will receive a grade report, with comments, via codePost. Your final grade will be based upon assignment, lab and design section, and TopHat scores. Cutoffs will not be determined until the end of the semester. If you are on the borderline between letter grades, factors such as consistently handing assignments in on time, a general upward grade trend throughout the semester, and a strong finish to the course will be taken into account. Students will receive course credit only if they submit minimally functional (MF) versions of all assigned projects and do not miss more than 2 discussion sections. Details of MF are on each project handout. However, your grade for a particular assignment is determined by how well it meets the standards for design set in the course and the assignment specifications, not simply by whether or not it works. Part of the art of programming involves a structured, disciplined approach to solving problems. Conventions for programming are stated explicitly in the CS0150 Style Guide and in lecture slides through examples. For each project, a large part of the grade will be based on design and style. The table below shows the approximate relative weight of each assignment used in calculating the final grade. Note that the weights may change slightly over the semester. **Assignment | Weight** Andy's Kitchen - 1% AndyBot - 2% Pong - 3%

	TicTacToe - 4% Fruit Ninja - 7% Cartoon - 10% DoodleJump - 13% Tetris - 18% Final Project - 25% Sections (Labs and Discussion) - 12% TopHat - 5% Total - 100% See the Course Missive for further details.
Due Dates	Projects must be handed in by 11:59pm EST on their due dates unless otherwise stated. Mini-assignments are generally due the Monday after they are released. Labs are due before the completion of the next lab section. All due dates (including early and late hand-in dates, if applicable) will be clearly written under the title of the assignments.
Minimum Functionality Requirements	To pass CS0150, you must complete each and all of the 9 programming projects with at least "minimum functionality" (MF), meaning you'll have to hand in an acceptable version for each and all by the end of the semester. Requirements for meeting MF for each project are detailed on each project handout. If you do not meet MF the first time (you will be notified when grades are sent out), you will have to re-submit a working version by the end of the semester to pass, even if your absolute grade is high enough (you must obviously also have a passing final grade to pass the course). Note that only meeting MF requirements will not be enough to earn full credit on a project.
Late Policy	Mini-Assignments for section will not be accepted late. Our late policy for programming assignments is as follows: • Most projects have a "late deadline", posted on the assignment handout on the course website. Programs handed in after the due date but before the late deadline will be penalized 8% of the possible points for that assignment. (A late submission of a program that would have received 94 out of 100 points would instead receive 86 points.) They will also not be eligible for extra credit. • You are entitled to **three** free late days during the semester. Each late day allows you to work on the project for one extra day past the on-time deadline without penalty, though you will still not be eligible for extra credit on the assignment. If you want, you may split the late days among projects, by using one or two late days each on separate projects, but you cannot use all three late days for one project. • At the end of the semester, we will apply your late days to the assignments for which it will be most beneficial to your grade. That policy means individual rubrics will not reflect late credits applied nor late deductions.

	• Anything handed in after the late deadline will receive an NC. Late days will not be accepted. • You may not use late days on the Final Projects. If you submit your Final Project by the late deadline, the late deduction will apply. • Assignments without a late deadline (Andy's Kitchen , AndyBot, and Pong) MUST be handed in by the regular deadline, otherwise they will receive an NC. • If one late day is used on a partner project, then it will count as both students using a late day. For example, say Partner A has 3 and Partner B has 2 late days before the project. If the pair hands in 1 day late, Partner A will then have 2 late days left, and Partner B will have 1. • If one partner is out of late days and an assignment is handed in late, the late penalty will be assigned to them but not their partner assuming their partner has late days remaining (e.g. partner A has 0 late days and partner 0 has 3)
Remote Accommodations	If you find yourself in a situation (COVID or otherwise) where you will need to be remote for some period of time, email Andy (avd@cs.brown.edu) with the circumstances and duration you will be remote. For lectures, you will be exempt from TopHat questions for the duration of your remote period. For Lab/Section email your Lab TAs and they will provide you a zoom link. For TA hours, email the Head TAs (cs0150headtas@lists.brown.edu) and the TAs holding hours at the time you want to attend; they will send you a zoom link. You will only be entitled to remote accommodations if you have the express permission of the professor.
Extensions	Please note that our late day policy is meant to cover short term illnesses and any foreseeable conflicts, such as athletic and artistic performances, travel for internship or job interviews, etc. We will, however, grant extensions upon the provision of a Dean's Note, which one may be able to obtain by speaking to a Dean about their extenuating circumstances. We are aware that granting an extension on one assignment will likely cause you to fall behind on the following assignments on schedule; to counteract this, the extension will be split between the current or requested assignment and the immediately following assignment to give you time to catch up. If you do not use your extension for the following project the extra days **will not carry over** to any future assignment after that. The process of obtaining an Incomplete also requires speaking to a Dean; not handing in all your work by the deadline does not automatically grant you an INC.
Extra Credit	Those of you who may have extra time and a strong interest in computer science are welcome to augment your already-working programs with extra credit extensions. Later project handouts will include a list of possible extra credit extensions that you should keep in mind when designing your programs (but you can invent your own, too). Extra credit is only to be done after the original assignment has been fully completed - if you have not met the requirements, you will not receive extra credit. Extra credit may not redefine the original assignment. Make sure to document anything you believe is extra credit in your header comments. Extra credit is capped at 10 points per project.

Accommodations	If you feel you have physical, psychological, or learning disabilities that could affect your performance in the course, we urge you to contact SAS (https://www.brown.edu/campus-life/support/accessibility-services/). We will do whatever we can to support accommodations recommended by SAS.
Mental Health	The entire CS0150 staff (Andy, the HTAs, and the TAs) wants you to succeed in the course, but not at the expense of your personal well-being. If you feel as though you are under too much pressure or there are other factors keeping you from performing well at Brown, we encourage you to contact Brown's Counseling and Psychological Services (CAPS: https://www.brown.edu/campus-life/support/counseling-and-psychological-services/). They provide confidential counseling. The Brown CS department also provides additional resources through its Health and Wellness initiative. To learn more and meet with student advocates, see their website here: https://browncs-health-and-wellness.github.io/
Lecture Topics	Throughout the semester, CS0150 will cover Object-Oriented Programming Fundamentals, Arithmetic and Flow of Control, Data Structures and Algorithms, and other advanced topics. Below are the specific topics for each lecture. 1. 9/8/22: Welcome to CS0150 + What is Programming? 2. 9/13/22: Calling and Defining Methods in Java 3. 9/15/22: Introduction to Parameters and Math 4. 9/20/22: Working with Objects I 5. 9/22/22: Working with Objects II 6. 9/27/22: Interfaces and Polymorphism 7. 9/29/22: Inheritance and Polymorphism 8. 10/4/22: Math and Making Decisions 9. 10/6/22: Introduction to 2D Graphics in JavaFX 10. 10/11/22: 2D Graphics, Part II 11. 10/13/22: 2D Graphics, Part III 12. 10/18/22: Loops 13. 10/20/22: Arrays and ArrayLists 14. 10/25/22: Design Patterns I 15. 10/27/22: Design Patterns II 16. 11/1/22: Recursion 17. 11/3/22: Big-O Complexity and Sorting 18. 11/10/22: Linked Lists 19. 11/15/22: Stacks, Queues, and Trees 20. 11/17/22: Final Project Overview 21. 11/29/22: Computer Graphics 22. 12/1/22: (Head TA Lecture) Surprise Lectures by Each HTA
Required Reading	If you decide to take CS0150 (and we hope you do!), you will also be held responsible for all of the information in the documents in the 'Required Reading'

	section of the course website, including our course missive and collaboration policy, and for any class announcements made in lecture or via email. • Course Missive • Collaboration Policy • Style Guide

CS200 Learning Outcomes

CS200 is designed to help you gain specific skills. They are organized into the following four theme areas. Each assignment will be labeled with a subset of these outcomes.

Data Structures and Algorithms

This theme focuses on how to organize and process data, with an emphasis on the structure of data and process organization.

Learning Objectives:

- State the key structural and performance characteristics of core data structures: lists, trees, arrays, hashtables, heaps, and graphs
- Select and justify a specific data structure for use in a specific problem, citing alignment with problem needs or resource constraints
- Explain tradeoffs between mutable and immutable data structures in the context of a specific problem
- Understand what makes a data structure mutable or immutable at the level of implementation
- Embed and manipulate tree-shaped data in a linear data structure
- Use dynamic programming to reduce the running time of a program
- Implement reachability as a core computation when appropriate in a graph-based problem

Object-Oriented Programming and Design

This theme focuses on the idioms and mechanisms of designing and implementing good solutions with OO programming

Learning Objectives:

- Create a class hierarchy that makes proper use of interfaces (for variant types or contracts) and abstract classes (for sharing common code)

- Develop classes that protect how their data is used through access modifiers and appropriate choice of provided methods
- Develop programs that allow selecting different concrete data structures through the use of interfaces
- Use concrete examples, invariants, hand tracing, and debugging to explore or explain how a program works
- Comment code effectively with Javadocs

Testing and Validation

This theme focuses on the ability to demonstrate that a program behaves as intended

Learning Objectives:

- Develop a collection of examples and tests that exercise key behavioral requirements aspects of a problem
- Write programs that can compile and run against a provided interface
- Write programs that behave as expected based on a provided description or collection of test cases
- Identify possible logical errors and/or faulty assumptions that can be caught by testing
- Discuss how to detect whether a system has adverse impacts on individual users, populations, or organizations

Technical and Social Impact Analysis

This theme focuses on understanding the impacts of a design or a program on resources (such as time and space), users, and society

Learning Objectives:

- Correctly determine the run-time complexity of an algorithm
- Correctly determine the space complexity of an algorithm
- Identify potential societal impacts of a program on individual users, populations, or organizations