Foundations of Programming

Bulletin Description

A first formal course in computer programming required (e.g. Comp 110, UNC). Intermediate programming: objects, pointers, classes, interfaces, packages, object composition, inheritance, visibility, delegation, observers, MVC (model view controller), window systems and user-interface toolkits, collections, generics, threads, , recursive descent, exceptions, assertions.

General Course Info

Term:	Fall 2018
Department:	COMP
Course Number:	401
Section Number:	002
Time:	TR 11-12:15PM
Location:	Room AR0121
Website:	http://www.cs.unc.edu/~dewan/comp114/current/

Communication with Instructors

Any question about the technical content should be posted as a public Piazza message. This way, the answer is available to all students. Technical questions sent as private messages will be ignored. For example, if you do not understand an assignment requirement, make a public post.

Communication of personal nature (such as class absences) should be sent as private messages on Piazza. How to post a private message is given here.

If you feel uncomfortable posting private messages to instructors, you can use the help alias: **help401-002-f18@cs.unc.edu.** The problem here is that one has to reply-all to make sure the communication is seen by all instructors. **Reply all when you get an email from an instructor so it goes back to all of us.**

Use an individual instructor email only when the communication is of interest only to that instructor. **The ~~strikethrough~~ font tries to make the point that personal email communication is not recommended, and may be ignored.**

Instructor Info

Name:	Prof. Prasun Dewan
Office:	FB150
Email:	~~dewan@cs.unc.edu~~, help401-002-f18@cs.unc.edu
Phone:	919 5906123
Web:	http://www.cs.unc.edu/~dewan

Office Hours: Tuesday 12:15 to 13:30, Thu 15:30 to 16:30

Teaching and Lab Assistants

Name	Office	Email	Office Hours
Andrew Vitkus		~~vitkus@cs.unc.edu,~~ help401-002-f18@cs.unc.edu	

Learning Assistants (Tentative)

Name	Office	Email	Office Hours
Jed Huang		~~jedhuang@live.unc.edu~~ help401-002-f18@cs.unc.edu	
Sarah Chipman		~~chipms@live.unc.edu~~, help401-002-f18@cs.unc.edu	
Jacob Smith		~~jsmith00@live.unc.edu~~ help401-002-f18@cs.unc.edu	
Gordon Ji		~~jiyu2@live.unc.edu~~ help401-002-f18@cs.unc.edu	
Neeley Holroyd		~~nholroyd@live.unc.edu~~ help401-002-f18@cs.unc.edu	
Colleen Ni		~~zni@live.unc.edu~~ help401-002-f18@cs.unc.edu	

Textbooks and Resources

Word chapters, PPT slides, YouTube videos on the covered material will be accessible from the course home page. These should suffice for the course.

Mixed Lectures and In-Class Exercises

Lecture time will be used for in-class team exercises, quizzes, and lecturer-led discussions on the topics of this course, listed below. The exercises, called praxes, will involve a "discovery" process in which students reason about, modify, and run instructor-provided software demonstrating the concepts in some topic. For each topic, the software will embed, as comments, a script to be followed to perform the discovery. (Thus, this process is similar to one in

which students observe instructors modifying code in-class, with the difference that the students will actively take the actions the instructor would have taken rather than passively following instructor actions.) Each praxis on some topic will be preceded by a short discussion motivating the topic, and followed by a Sakai quiz testing what was learnt in it, and a class Q&A discussion of the subtle issues raised by it.

The hope with praxes is threefold: (a) you learn material in a hands-on and cooperation-oriented approach that makes it stick better, (b) you can participate in-depth in the Q&A discussion on the associated topic, and (c) you learn powerful commands of the programming environment (in our case, Eclipse) that practitioners use to improve the programming process.

Remember that a lack of certainty is a natural part of the process, and that you are not expected to master the concept from the praxes alone. When you come across a concept you are unsure about, experiment with the code, talk to people around you, form hypotheses, write down and ask questions in the discussion that follows, and look into the detailed explanations of concepts given online. The idea is that once you do gain an understanding of a concept, it will be a more grounded one, spurred by experience and necessity.

So explore, be inquisitive, and do not be afraid to not know everything! Turn your uncertainty into a tool of learning by asking questions and experimenting.

The discovery-based (somewhat laborious, by definition) praxes are yet another means at your disposal to learn the material**. Some of you will prefer other means such as the word docs, ppt pdfs, video recordings, which is fine.** Use what works best for you to answer the Sakai quiz associated with each topic. Our hypothesis is that the praxes probably help you remember the material. Ultimately, the assignments will be the most effective method for retaining the information.

Course Description

This course is intended for people who have learned to program. Its goal is to teach how to program well. The common programming strategy of beginners is to write the first solution they can think of without carefully identifying and weighing different alternatives. For all but the simplest problems, this approach of writing ''quick and dirty'' programs will take you to the debugging stage very quickly, but will make debugging slow. For large, complex programs, you need to identify multiple alternative solutions to the problem, choose an alternative that most directly solves the problem, and think carefully what your solution does, and how it works. The claim is that, although "quick and dirty'' programming may produce a program faster, the

concepts we teach will help you produce a correct program faster. Moreover, they will lead to programs that are easy to change and reuse.

We assume you have learned the following basic programming concepts: primitive types (integers, real numbers, Booleans), variables, constants, assignments, comments, expressions, arrays, loops, arrays, and procedures/functions/methods. These concepts are taught in most, if not all, introductory programming courses regardless of whether they teach conventional or object-oriented programming. This course will teach you the next-level programming concepts. These include objects, classes, pointers, interfaces, packages, structures, inheritance, delegation, design patterns, exceptions, and assertions. These concepts will not help you solve new problems; rather, they will help you solve problems in new ways. The skills that will enable you to use these concepts will form a large part of the challenge you face in this course. After this course, you will have a much deeper understanding of the programming and learn some of the ideas that can make programming a science. We will be using Java as a vehicle for learning these concepts.

Target Audience

As mentioned in the course description, this course is intended for people who have learned to program. Its goal is to teach how to program well.

Prerequisites

As mentioned in the course description, we assume you have learned the following basic programming concepts: primitive types (integers, real numbers, Booleans), variables, constants, assignments, comments, expressions, arrays, loops (both counter-based and event-based), arrays, and procedures/functions/methods. These concepts are taught in most if not all introductory programming courses regardless of whether they teach conventional or object-oriented programming.

Goals and Key Learning Objectives

As mentioned in the course description, the goal is to teach how to program well. The common programming strategy of beginners is to write the first solution they can think of without carefully identifying and weighing different alternatives. For all but the simplest problems, this approach of writing ''quick and dirty'' programs will take you to the debugging stage very quickly, but will make debugging slow. For large, complex programs, you need to identify multiple alternative solutions to the problem, choose an alternative that most directly solves the problem, and think carefully what your solution does, and how it works. The claim is that, although "quick and dirty'' programming may produce a program faster, the concepts we teach

will help you produce a correct program faster. Moreover, they will lead to programs that are easy to change and reuse

Course Requirements

The students must attend lectures and recitations, implement in-class recitation assignments, implement a semester-wide project, and take a midterm and a final.

The material you learn makes less sense when you write small programs. This is why, in this class, you will exercise it in a large project you will build incrementally over the course of the semester. Each increment will be an assignment that builds on the software you have implemented as part of previous assignments. Because this is a programming course, unless all programs are submitted and work, you may not pass. Point values of assignments and programs will be in accordance with their length and difficulty.

You must submit the source code of your program (with pledge signed) and screens showing executions of the program on test data.

Examinations are closed book, notes and program listings; computers and collaboration are not allowed either.

Key Dates

Midterm: Tuesday Oct 16th, 2018 (in class)
Final: 12pm, Thu Dec. 13, 2018 (in class)

Grading Criteria

Final grades will be based on the following formula:
Regular credit:
- 25% Midterm Exam (Closed Book)
- 15% Final Exam (Closed Book)
- 10% Lecture Quizzes (Open Book)
- 3% Recitation Quizzes (Open Book)
- 47% Regular Credit Part of First 9 Assignments

Extra Credit:
- Class Participation:
  - Up to 5%: In-Class Answers by You
  - Up to 5%: Reports of In-Class Answers by You or Others
- Early Submission (5% of each assignment)
- Extra Credit parts of First 9 assignments, and Last Three Assignments
  - Point values will be given for each credit part/assignment

Up to 5%: Fudge Factor
Extra Credit Quizzes: Point given in quizzes

There is no fixed mapping between overall percentage and final grade.

Walking in late and leaving class early (without prior instructor permission), talking during lectures, and other disruptive behavior will result in negative points for In-Class Work.

Bring your laptops to class and recitation so you can do exercises.

Course Policies, Class Activities, and Early Rewards/Late Penalty

Students are required to attend each class unless there are extenuating circumstances. If such circumstances occur, you should access the class material posted for missed classes, and contact classmates to become aware of the announcements that were made.

The "lectures" will use a combination of regular and flipped structure. Each topic will be introduced by a lecture, a lecture-quiz, and a post-quiz discussion. There should be enough time during class to take the quiz – but it will be due later at a time announced to you. The recommended method for mastering the material is to do a "hands-on" praxis in which instructor-provided code is studied, changed, and executed to discover the concepts in the topic. It will be possible to answer many if not all of the questions (that have non-zero points) by reading written material and watching online videos and associated with the topic. The time devoted to praxes can be used to clarify concepts and requirements, and get help with assignments.

The post-quiz discussion will involve question and answer segments. You need to maintain a diary of the questions and answers to receive points for the class participation extra credit. If you answer a question and note down the question and your answer in the diary, your will receive points for "In-class answers by you" part of class participation. If you note down a question and its correct answer (in your own words), you receive points for the Reports of In-Class Answers by You or Others part of the grade.

The diary will be a single Piazza post you edit repeatedly. The details of the format are given here.

The recitations will help bridge the gap between the examples in the praxis and other class material and the assignments, providing a template for each assignment. These quizzes, together with the lecture quizzes, will serve to help keep students on track.

If we get enough LAs, to further ensure that students are on track, we will assign an LA (learning assistant) to a group of students. The LA will serve as a mentor/coach of the group of students. The LA will note: (a) if each student in the group comes to class, (b) the progress they have made since the last class, (c) the class-discussion they have assimilated, and (d) the questions they have answered. Based on these observations, the LA will reach out to the shy students who are not keeping up to understand the issues they are facing and try to resolve these issues with the help of other instructors. We are lucky to have students in the Peer Teaching class, Comp 227, sign up to provide LA duties, so each group should be relatively small.

Each assignment is associated with a submission date and an early-submission date. Except for the last assignment, an assignment is also associated with a first and second submission date. The early submission is associated with a reward of 5% extra points. The first and second submission dates are associated with penalties of 10% and 25%, respectively. An assignment submitted after the second submission date will get no credit as there will be ample time to finish it with help. However, if for some reason, you get behind and cannot finish an assignment by the second late day, you can shift the due dates of subsequent assignments, that is, can move the due date of each remaining assignment^{i} to the due date of asignment^{i+1}, *as long as you sacrifice an assignment from the end*. Thus, if you cannot finish assignment 3 by its due date, you can move its due date to the due date of assignment 4, the due date of assignment 4 to the due date of assignment 5, and so on, as long as you sacrifice the last assignment. You can shift and sacrifice multiple times. Thus, if you cannot complete assignment 6 by the due date of assignment 7, you can shift its due date to the due date of assignment 8 by now sacrificing the second last assignment. A sacrificed assignment may not be graded, even if you finish it.

The early dates for assignments are the target assignment completion dates for those who are planning on doing the extra credit assignments. As a result, the difference between the regular date and early submission date varies – for A1 the difference is two days and for A9 – the last regular assignment – it is about a month.

There will be ample time to finish the quizzes, and no late submissions will be accepted. In addition, there will be some extra credit quizzes, which you can take to improve your grade or make up for quizzes you missed to submit in time.

Request for meetings outside office hours should be made help401-002-f18@cs.unc.edu and only if you can show that your schedule prevents you from meeting at one of the publicized hours - otherwise it is your responsibility to schedule your work on this assignment so you can meet when you are able to. And remember office hours are meant for

clarifications and hints and not tutoring - so do not request 1-2 hour blocks, which are impossible for us to satisfy. By coming to office hours you are implicitly pledging that you have taken the lecture and recitation quizzes relevant to the material for which you are asking for help. Not following this rule is an honor court violation.

Collaboration Allowed and Honor Court

1. You are encouraged and expected to discuss the assignments among yourselves.
2. You are permitted to discuss all aspects of the Java programming language with anyone.
3. You are permitted to discuss solutions at the design level *but not at the code level.* For example, you are allowed to tell others that you have separate classes in your program for scanning and evaluating expressions, or that you are using a loop instead of recursion for scanning, but are not allowed to show them Java interfaces, classes, while loops or other Java code in your solution. A general rule of thumb is that if you are communicating using a natural language, you are discussing at the design level, but if you are communicating using pseudo or actual code, you are discussing at the code level. If you have successfully finished some part of an assignment, then it should be ok to look at someone's code to help then debug it, as there is little danger you will incorporate their code in your solution.
4. Copying any kind of text (code, comments, progress described in notes passed to Las) is plagiarism.

For details on how the Honor Code applies to programs, consult the handout 'Honor Code Observation in Computer Science Courses.'

DI(STRESS)

A programming-intensive course such as this one can cause stress and even lead to distress, especially if you have a heavy workload. Please come talk to one or more of the instructors if you are facing difficulty of any kind. One of the goals of a mentor is to help identify situations that can lead to (di) stress. In addition, several other resources are available at UNC for those facing advanced stress:

Counseling and Psychological Services: Can provide mental health care and evaluation for students. Consult with staff M – F, 8a – 5p by phone or in person, with or without student | 919-966-3658

After usual office hours: Campus Health Services is open M – F from 5p - 6p and weekends from 8a – 5p. Calling Campus Health after hours connects you with a nurse who can reach on-call mental health support. Or go directly to an Emergency Room | 919-966-2281

Course Schedule

If possible, a schedule of topics covered by the course organized by course date or week number.

1. Course Information, Introduction, and Scanning
2. The Dual Roles of a Class, Constructors and Pointers
3. Programming Interfaces
4. User Interfaces
5. Composite Objects and Shapes, Trees, DAGs and Graphs
6. Inheritance and Collections, Variable Inheritance
7. IS-A and Inheritance-based Type Checking
8. Collection Kinds
9. Model View Controller and Component Notifications
10. User-Interface Toolkits and Graphics
11. Abstract Classes, Generics, Adapters, Basic Delegation
12. Assertions and Exceptions

Disclaimer

The professor reserves to right to make changes to the syllabus, including project due dates and test dates. These changes will be announced as early as possible.