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 2016
Department: COMP
Course Number: 401
Section Number: 002

Time: TR 2-3:15PM
Location: Room GS 200
Website: http://www.cs.unc.edu/~dewan/comp114/current/

Instructor Info

Name: Prof. Prasun Dewan
Office: FB150
Email: dewan@cs.unc.edu
Phone: 5906123
Web: http://www.cs.unc.edu/~dewan
Office Hours: TR 15:30 – 16:30

Teaching Assistants (TBA)

<table>
<thead>
<tr>
<th>Name</th>
<th>Office</th>
<th>Email</th>
<th>Office Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Textbooks and Resources

I plan to provide Word chapters, PPT slides, and YouTube videos on the material I cover accessible from the course home page. These should suffice for the course. They will not be posted on Sakai, which will be used however for submitting programs.
Flipped Classes

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.) These praxes will be followed by a Sakai quiz testing what was learnt in them and a class discussion of the subtle issues raised by them.

The hope with praxes is twofold: that by the time we discuss a concept, you have enough investment, questions, or background to learn it in a way that gives it context (since you have seen the concepts in code yourselves); and that you learn it in a hands-on and cooperation-oriented approach that makes it stick.

With that in mind, 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 praxis 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 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. 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, interfaces, packages, inheritance, delegation, design patterns, exceptions, assertions, pointers, and formal correctness. 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, 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 non-cumulative final.

The material you learn makes little 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. You may also do demos at certain stages of your project.

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

Key Dates

Midterm: Tuesday Oct 18th, 2016 (in class)
Final: 12pm, Friday Dec. 10 (in class)

Grading Criteria

Final grades will be based on the following formula:
22% Midterm Exam
28% Final Exam
15% In-Class Work (Recitations, Lecture Quizzes, Class Participation)
35% Project (Programming Assignments)

The above percentages add up to 100%, but the instructors reserve the right to apply a 5% fudge factor to give consideration to things such as good (in terms of
quality and quantity) class/piazza participation, stellar programs, thoughtful in-class exercises. In addition, students will get ample opportunity to get extra credit through early submission and extra work. 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 points being deducted from In-Class Work.

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

**Course Policies 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.

Each assignment will be due at 11:55pm on a Friday. Submitting it on the Wednesday of that week will result in a 5% extra credit. Submitting it on the Monday (Friday) of the next week will result in a penalty of 5% (15%). Submitting it after that day will result in a penalty of 20% on the auto-graded parts. No credit will be given on parts that are manually graded or for mistakes made by the auto grader.

Request for meetings outside office hours should be made through help401-002@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 in-class 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.

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

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. Assertions, Threads and Command Objects
12. Abstract Classes, Parsing and Grammars
13. Synchronized Methods, Wait and Notify
14. Generics, Adapters, Basic Delegation, 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.