Search this site

Home

Welcome to Software Design at Olin College, Fall 2013.

Instructor: Allen Downey

Meetings: Tuesdays and Fridays in AC128, 10:50am to 12:30pm.

Textbook: *Think Python*. You can buy a hard copy from the bookseller of your choice, or read the electronic version.

Mailing list: Sign up for the class mailing list here. You also might want to sign up for the Computer Science at Olin mailing list.

Class web page: Short URL for this page is tinyurl.com/sd13fa

Policies

SUBPAGES (3): POLICIES PROJECT PROJECT WIKI

SITEMAP

Search this site

Policies

Coursework

Your work in this course will include

- Homeworks.
- Quizzes.
- Exams: Oct 4, Nov 1, Dec 6.
- Project.

Homework

We'll have 11 homeworks, mostly due on Tuesdays, worth 10 points each. You can earn up to 100 points for homework.

We will grade homework face-to-face as often as possible, in order to provide timely and substantive feedback. We will deduct points for errors in function and style, but you will have a chance to correct them.

We will deduct one point for each day a homework is late, up to 4 points. That means you can get up to 6 points even if a homework

is very late. But try to stay on schedule.

Every file you turn in must have a top-level comment identifying you as the author and listing anyone who made a substantial contribution. This comment is your assertion that the homework represents your own work.

Quizzes

We'll have about one quiz per week, except when we have exams, usually on Fridays. Quizzes are at the beginning of class, as incentive for punctuality. We'll go over solutions immediately and you will have a chance to note errors.

Because quizzes are frequent and we go over them immediately, it is impractical for me to administer make-up quizzes. However, I will drop your lowest quiz score, so you can miss one with impunity.

If you miss more than one quiz, I will generally fill in a low passing grade to avoid the impact of averaging in zeros. In any case, quizzes are primarily intended to be developmental, not evaluative, so they make up a small part of your final grade.

Exams

We will have three exams, roughly every four weeks. They will be in class exams, probably using a full class period.

Each exam covers new material and material from the previous exams. Because the exams are cumulative, you can use the score from a later exam to replace the score from a previous exam.

Project

During the second half of the semester you will work will a small team on a project of your design. At the end of the semester you will present your project to the class and turn in your code and a final report. I will provide details as we go along.

Participation and professionalism

We will do lots of fun and useful things during class time. You should plan to be there!

As students, you have some responsibility for creating and maintaining a classroom atmosphere that is conducive to everyone's learning and enjoyment. I hope you will think about how your participation contributes to the learning environment.

Some things you can do to help:
1) Come to class on time! I will do my best to use class time effectively. Late arrivals are disruptive.
2) Come to class prepared. Make sure you have always at least skimmed the reading.
3) Try not to fall behind. If we are all working on the same stuff at the same time, everything works

better.
4) Take care of your brain. Eat well, sleep well, get some exercises. Come to class ready to work.
5) Be professional.
6) Be respectful toward me, the NINJAs and your fellow students.
7) Be generous with your ideas and your time. Help each other.
8) Be reflective. Think about what's working and what's not, and take responsibility for making the class work for you.
9) Be honest. Some elements of this class create moral hazards. Avoid them.
10) Have fun!

Use of code from external resources

In general, you are encouraged to search for and use code from external resources, with appropriate attribution. Guidelines for using external code depend on the context and your use of the code.

For example, if you find snippets of code that make up a small part of an assignment, and assemble them into a solution, that's perfectly fine. If the snippets demonstrate common use of Python features, you can use them without attribution. Something more substantial or unusual should be attributed.

The goal of homeworks is to give

you an opportunity to develop programming skills, so you should avoid using online solutions that undermine that goal.

The goal of quizzes and exam is to evaluate your progress, so you should not use external resources in a way that undermines that goal.

All material that you turn in, including homeworks, quizzes, and exams, is presumed to be your own work, unless you explicitly attribute it to another source. Representing someone else's work as your own is a violation of engineering ethics and Olin's Honor Code.

Comments

You do not have permission to add comments.

Search this site

Project

Overview

In the second half of the semester, you will work on a project on a team of 2-4 other students.

Some students work on projects that address the needs of a client, which can make the experience more effective, but you are also free to define a project that serves your interests and educational goals.

Resources

You can assume that by the time we get into the project you will know how to write programs with

1) graphics, animation, and sound,
2) graphical user interfaces, and
3) networking.

This class will not cover everything you need to write a Web app, but if you know some HTML/CSS, and possibly some JavaScript, or you are willing to pick up what you

need, then writing a Web app is a viable project.

Where to go looking for ideas:

1) Scratch your own itch. Automate a repetitive task? Facilitate communication/interaction/sharing? Build specialized courseware for a class? Make a game? Mash up online services?

2) Find a client. Talk to faculty and staff on campus. Talk to parents and relatives with real jobs.

3) Make a positive difference in the world. See broken things and fix them.

Requirements

1) The project should involve a substantial software design component; that is, you should have to make non-trivial design decisions. For example, your program should include more than one class, and you should be designing interfaces between classes. If you are writing 0 or 1 classes, something is almost certainly wrong.

2) You should work with 1-3 other people. Software development processes are qualitatively different when there is more than one person involved. Larger teams are possible, but in that case you will need to show that

the work can be divided in a way that makes it possible for everyone to make progress.

Schedule

Here is the schedule of deadlines:

October 25: Proposal due.

November 8: Design Due.

November 22: Refinement due.

Week of December 9: Mandatory NINJA check-in (no deliverable, but you must talk to us).

Finals: Report due; Presentation

These are hard deadlines; you will lose credit if you miss them.

Project proposal

Your proposal should be a short document (1-2 pages) with answers to the following questions:

- Who's in the group?
- What's the project?
- What's the minimum/maximum deliverable?
- What's your first step?
- What's the biggest problem you foresee or question you need to answer to get started?
- Where is the git repo for the project?

Notes:

1) One of you should create a repo for the project on GitHub and invite the other members of the group as collaborators.

2) *Do not include the bulleted list of questions in your proposal.*

3) The intention of the minimum/maximum deliverable is that you should specify a lower bound you are almost certain you can achieve and a stretch goal you probably won't achieve.

4) As the project proceeds, you will be editing this document and turning in updates. Think of them as successive drafts of the final report.

5) The documents you turn in at various stages do not have to be polished, or formatted beautifully. But please don't take that as a license to make them incoherent, or verbose, or silly. You should try to present your ideas clearly and concisely, with the goal of producing a final report that you would be proud to show to the outside world.

Design proposal

At this point in the project you should have at least a rough idea of the design you are planning to implement, you should have chosen the packages you will use,

and be familiar with them, and you should have written some code that at least tests out the packages. You might also have a prototype of some kind.

The design proposal should be a revised version of your project proposal, so it should include all of the elements described above in addition to the new material described below.

1) The design proposal should explain at least one part of your design by telling a software design story. There are two common structures for this kind of story:

We had a problem.
There were at least two alternatives.
Here are the pros and cons of the alternatives.
Here is the justified reason we chose the one we chose.
OR
We wrote a first draft.
It had a problem.
Here is what we changed.
Here is how that solved the problem.

Try to make the logical structure of your story clear. You don't have to document every design decision you made, but you should choose at least one that you think is interesting.

2) At this point you might have chosen some of the data

structures you will use in your project. If so, please describe them (ideally using a UML state diagram) and explain the most important algorithms these data structures support.

3) Finally, your design proposal should include a development plan. What have you done so far, and how do you plan to proceed? How will the work be divided among the members of the team? If you are planning to split up the work, how will you manage the interfaces between the parts of the project? If you are planning to specialize, I suggest that you make some effort to ensure that all members of the team understand all parts of the design at an appropriate level of abstraction.

4) Your design proposal should include the material from the previous reports, edited to bring it up to date.

Design refinement

At this point you should have at least a prototype, and possibly you will have achieved your minimum deliverable. You should turn in an updated version of your design proposal that includes any improvements I suggested during the previous iterations, and any refinements you have made in your design.

In particular, I would like to know

if there are **any significant changes you made in the design** while you were in the process of implementing it, and why. Examples might include:

- You added new classes to the design, or consolidated classes, or changed the relationships among the classes.
- You discovered a performance problem and changed your data structures and/or algorithms.
- You started out with the intention of using some package, but you needed something it didn't have or you found something better.

Also, this report should **include a discussion of abstraction and language**: as you develop your design, you should find yourself developing abstractions (like new classes and collections of functions and methods) and vocabulary for talking about your program, and for expressing the computation that makes up your program. What are the elements of this language, and how are they reflected in your design? Before you write this section, you should read Chapter 1 of Domain Driven Design.

Why am I asking about abstraction and language? Because programming is about communication: you communicate with your teammates, but you also

translate your ideas into code, and translate code into ideas. These communications are mediated by the language that you design; for example, when you create a class and give it a name, that name becomes part of the shared vocabulary of your team.

If you design your language well, it facilitates communication and allows you to express complex ideas concisely. If you design it badly, it creates confusion and can lead to errors.

So I am asking you to reflect on this issue and write a few sentences about the vocabulary you have developed and how well it is working.

Also, your design refinement must **include a UML Class diagram**. If there are no classes in your design, that might be a sign that you need to rethink your design.

If your project involves any significant data structures or algorithms, you should describe them at an appropriate level of abstraction.

Reminder: this status report should include all the material from the previous reports, updated and refined, plus the new material. In particular, **make sure you include the location of the GitHub repo for the project**.

Final report

Your final report should include the following elements:

- All the material from the previous documents, edited to bring them up to date. Sections that are no longer relevant can be removed.
- A final refinement of the design, including any changes since the previous iteration and an explanation of why the change occurred.
- An analysis of the outcome. Did you achieve your minimum deliverable? If not, why not? Did you achieve your maximum deliverable? If so, what went right (that is, why was the project easier to complete than you thought)?
- A reflection on your design. What was the best decision that you made? What would you do differently next time?
- A reflection on the division of labor. Were you able to divide the project in a way that allowed the members of the team to work independently and then integrate their work into a whole?
- Bug report: what was the most difficult bug you had to find? How did the bug manifest itself? What did you do to find it? How long did it take? What was the cause of the problem? In retrospect, what could you

have done to find it faster?

Of course, the organization of these elements will depend on your project, and some may more more applicable than others.

The final report is due on demo day. Please check it into your repository in PDF format; put it in the top-level directory with the name `final_report.pdf`. Also, please drop off a hard copy for me or send it by campus mail (arriving the next day is fine).

Demos

During the final exam period, you will have the opportunity to demonstrate your project for the class.

We will figure out the schedule later, but you should aim to be ready to demonstrate your minimum deliverable on the first day of demos.

Here are some suggestions for running a good demo [coming soon].

Expo

Depending on what else you are doing this semester, you might choose to present your Software Design project at Expo.

If you choose to present your project as a poster, it should include a description of the project as a whole, and also an explanation of what part of the project you worked on. This part of the poster should be targeted to a general audience with no programming experience. Your poster should also include an explanation of at least one design decision; this part of the poster can be targeted to an audience of programmers.

In my opinion, the most important criterion of a good poster is that it should pass the 15-second test. Someone should be able to walk up to your poster, know immediately where to start reading, and, in the first 15 seconds, they should learn what kind of project it is (that is, a software design project), who worked on it, and what you did.

After that (unfortunately) most people will walk away. For the ones who stay, your poster should tell a short, coherent story about your project. The flow of the poster should be obvious (that is, what to read first, and so on) and there should be a small number of conclusions (like one) that are clear and apparent. The total number of words should be small. The figures should be clearly labeled, and there should be text that explains what the figures are meant to show. You should design the poster as if you will not

be there to explain it.

Comments

You do not have permission to add comments.