Syllabus

General Info

Course CS321 Software Development Practicum

Instructors
- Charilaos Skiadas (skiadas at hanover dot edu)
- Theresa Wilson (wilsont at hanover dot edu)

Term Spring 2018-2019

Offices LYN 102

Office Hours By appointment

Book
- *Agile Software Development, Principles, Patterns, and Practices*, by Robert Martin
- online ACM resources[1]

Websites
- https://skiadas.github.io/SoftwareDevelopmentPracticumCourse/ (for most course materials)
- https://moodle.hanover.edu[2]

Class times MTWRF 10:15am-5:00pm, in LYN120A, with break for lunch

Course Description

Software Development is a complicated process. Of course you must write code that solves a problem, but this is only a small part of the work when bringing a software product from initial idea to releasable application. As part of the process, you will invariably have to revisit code written in the past in order to improve it or to fix bugs. You will need to maintain "development" versions of your code along with "production" versions. You will need ways to keep track of bugs in your code as well as proposed enhancements, and you will need to track your progress as you work towards make these changes. Above all, you need will to work with other people's code, and they will need to work with yours.

This course is a survey and practicum on all of these aspects of software development. It moves on parallel across 5 categories of topics:

1. *Java and Object-Oriented-Programming Fundamentals* reviews and reinforces the key aspects of OOP with a discussion of classes, inheritance and delegation, interfaces, encapsulation etc.
2. *Design Concepts and Principles* discusses the SOLID principles, which help to keep large code bases clean.

[1] http://learning.acm.org
[2] https://moodle.hanover.edu/course/view.php?id=825

3. *Design Patterns* discusses tried-and-true solutions to common design problems.

4. *Development Practices* covers standard development techniques such as agile methods, user stories and CRC Cards, version control fundamentals, test-driven development, issue management, and UML diagrams.

5. *Coding Practices* discusses topics related to the particulars of writing clean and readable code, focusing on general values and principles as well as more specific topics such as naming of variables and methods, commenting and formatting principles, the law of Demeter, exception handling, and refactoring techniques.

Along the way, students are also engaged in a collaborative project that will allow them to reinforce all these concepts while working with their peers on a large-scale project.

Textbook

There are two main sources of material for the course. The first is the main course textbook, *Agile Software Development, Principles, Patterns, and Practices* by Robert Martin. This textbook covers the topics of *Design Concepts and Principles* and *Design Patterns*.

The second source is a list of electronic resources available to members of the ACM (Association for Computing Machinery). The ACM delivers resources that advance computing as a science and a profession. This includes the ACM Learning Center[3], which provides access to hundreds of online books and videos related to computing. To access these electronic resources, you will need to purchase an ACM student membership (about $20 per year). The cost of a student membership is modest given the weath of resources it provides access to. We encourage you to continue your membership every year and to take advantage of the opportunities and resources it offers.

A catered list of readings and videos from the ACM Learning Center will be provided. These will cover the remaining topics: *Java programming and OOP Fundamentals*, *Development Practices* and *Coding Practices*. Sections from the following will be covered:

- *Clean Code*, by R. Martin
- *Clean Code* Video Series, by R. Martin
- *Design Patterns* Video Series, by R. Martin
- *Test-driven development, by example*, by K. Beck
- *Implementation Patterns*, by K. Beck

[3] https://learning.acm.org/

Course Components

Class Preparation

In the class schedule page[4] for each class day you will find a list of links to "prep" items. These may be videos to watch or book sections to read, possibly with a short assignment to help you learn the material. You are expected to do this prep work before coming to the next class.

Class Participation

You are expected to attend every class meeting. The accelerated nature of Spring term and the collaborative nature of the team development project require constant engagement with the material and your classmates.

Each class day is typically divided into 5-6 *activities*. Missing an activity will result in a reduction of your final score by one percentage point, up to a total of 10 points. Emergencies will be dealt with on an individual basis. There are very few reasons that would qualify as an excuse for an absence. If you have known conflicts discuss them with us by the end of the first day.

Class meeting times The class meets for 10am to 5pm with a break for lunch. This is the required amount of time expected of a student for a 1-unit course. You will do *most* of the work for the class during class time, including *all* the work on the project.

Exams

There will be three midterms: Monday 5/6, Monday 5/13 and Wednesday 5/22.

Project

For a large part of the course you will be engaged in a collaborative project with a group of your classmates. Project work will typically be as follows:

- All work on the project will be done during the class meeting times, in the 10am-5pm block. We expect that you will NOT work on the project when out of class.
- We will be working in 2-3 day "development iterations" during which the group will work to implement a set of features for the project.
- Each feature is "owned" by one student; that student is in charge of seeing the feature through to its completion.
- At any given time, you will be either developing a feature you own or pairing up with another student to help with their feature.

[4] skiadas.github.io/SoftwareDevelopmentPracticumCourse/site/schedule.html

- Your grade is not determined by the "number of features *you* have completed" (beyond a minimum requirement). A large part of the grade is based on the success of the project as a whole. So assisting others with their features is just as important as working on your own features. Paired programming is both required and expected.

The project will overall count for 45% of the final grade, broken down as follows:

- Group Assessment
  - Evaluation of the overall quality of the project's codebase, based on a development rubric. (20%)
  - Assessement of the success of the project based on the delivered functionality. (10%)
- Individual Assessment
  - Leading development of a required number of features. (5%)
  - Effective paired programming with teammates on their features. (5%)
  - Peer assessment of your contribution to the team. (5%)

Phone policy

Your phone is required to be in your bag, set to silent, at all times, except during scheduled breaks (there will be at least one 5-minute break every hour). This includes both lecture and other in-class activities as well as programming/development time. If phone use becomes a problem, we will confiscate your phone until the end of the class day.

Grading

Your final grade depends on class participation, the development project, and three exams, broken down as follows:

Components	Percent
Participation	10%
Project (Group)	30%
Project (Individual)	15%
Exam 1	15%
Exam 2	15%
Exam 3	15%

This gives a number up to 100, which is then converted to a letter grade based roughly on the following correspondence:

Letter grade	Percentage Range
A, A-	90%-100%
B+, B, B-	80%-90%
C+, C, C-	70%-80%
D+, D, D-	60%-70%
F	0%-60%