Software Engineering

DESCRIPTION

Review of methods and tools used in software development. Object-oriented design and open software architectures. Requirements analysis, design, implementation, testing, maintenance, and management. Engineering applications.
This course assumes that you are familiar with object-oriented programming using Java. If you feel you have weaknesses in this area, for instance, if you scored a C or below in Comp 132 or Comp202, the first few weeks of the course are a good time to brush up on your skills. The class will build upon knowledge of these fundamentals and will emphasize object-oriented analysis and design, and good programming discipline. A software development project to be carried out in groups forms the backbone of the course. We will see a software product through the project lifecycle, from analyzing customers' requirements to design, modeling, implementation, and testing. We will have design reviews for each major phase of your project. There will be regular meetings with each project group and a final presentation for your product at the end of the semester.

COURSE LEARNING OUTCOMES

- Develop an understanding of the key concerns and challenges of engineering a large software system.
- Develop an understanding of the issues of requirements, specifying, architecting, designing, and testing a software system in order to provide modularity, adaptability, and maintainability.
- Analyze, design, and develop software using object-oriented analysis and design, and design patterns.
- Develop teamwork management skills.

OTHER

Required textbook:

Applying UML and Patterns, Edition: 3E, Larman Pearson Education.

Head First Design Patterns, Freeman, Robson, Bates, Sierra O'Reilly.

Reading (recommended):

- Program Development in Java: Abstraction, Specification, Liskov and Guttag Addison-Wesley.
- Object-Oriented Software Engineering, Bruegge&Dutoit
- Java Concurrency: http://docs.oracle.com/javase/tutorial/essential/concurrency/
- Refactoring: Improving the Design of Existing Code, Fowler
- More material will be added as lectures covered relating to Git, Agile, Testing, Advanced Java topics, etc.

Grading Policies

Assessment: (Tentative, there could be up to -+5 changes).

- Class Participation %5
- Midterms %30
- Group Project %65

Class Participation: Attendance will be taken. Some of the project discussions will be conducted in the class as well. You are expected to attend classes at least 70% to get full points. There will be no makeup for missed attendances beyond that. 70% 5 points, 60% 3 points, 50% 2 point, less than 50% 0 points.

Bonus: Several ways to get bonus points: Code review, best demo, in-class participation

We will be using **Discussion Board** for anything related to the course. Subscribe to the discussion forums to be notified by email if someone posts or responds. **DO NOT send emails**, answering emails in a timely manner is not guaranteed.

Midterms: There will be one midterm. The midterm is a closed-book exam and will be conducted face-to-face. There will be only one makeup exam during the last week of the semester for those who missed a midterm due to health reasons (the report is required). The format of the makeup exam will be given at least one week before the exam - not necessarily in the same format as the midterms.

Final: There is no final exam in this course. It is a project-based course.

Project organization and grading:

You are required to form project groups of size GROUP_SIZE. GROUP_SIZE is FIVE students (only a few of four or six class size is not multiple of five).

- If you are not part of a team by the third week of the semester, you will be put in a team randomly.
- Similarly, if you form a team of size less than GROUP_SIZE, you will get new members (chosen randomly).
- **During the semester, if some of your team members drop the course, the remaining members are still responsible for the work, and no compensation will be given. Choose your team wisely.**
- If the class size is not multiple of GROUP_SIZE, some groups might have +/-1 members (randomly assigned if no team volunteers).

Group members are expected to take part in every aspect of the project, and your grade for each phase will be based on your contribution to the group's work. **If you fail the group project (i.e. score less than 60 out of 100 points), you will fail the class regardless of your grades on the midterm.**

The project requires the students to work together in a team from the beginning to the end. You should plan on committing your time and effort to teamwork. Teams that do not work together but individually produce very poor results and score poorly! **Teamwork, teamwork, teamwork!** Keep this in mind. Make sure that you perform well in your team. The peer evaluations submitted by your peers will affect your project scores. The details of peer evaluation will be announced during the project.

- The weekly meetings will be graded based on group work and individual contributions. The objection to weekly meeting grades must be done within a week after the grade is announced. No objections will be considered after that.
- Teams or team members should report to the instructor as soon as possible if there are problems in the team that will affect teamwork.
- The health report should be more than one week to be counted as an excuse for the project's weekly participation. Maximum **TWO** meetings will be excused for health reports.

Weekly Project Meeting Schedule: We will have around 10 meetings. Each meeting will be conducted preferably online involving all the group members and the TA. The meeting time will be scheduled so that every group member and the TA can meet at the same time. We try to schedule the meetings towards the end of the week. The meeting time **does not have to be during lab hours**. If no time can be found that fits all, it has to be during lab time.

We will schedule online meetings/breakout rooms for each group and the TA.

Your project will have **intermediate milestones with corresponding deadlines**. Work submitted late will receive no credit. **This is a strict policy**. If you can't complete the work by the deadline, reduce the scope of the work you have to do. You have to submit a coherent set of documents and/or software by the deadline even if it fulfills only part of the requirements for the deadline.

Further details of project administration, schedule, and deadlines will be announced after the class roster is finalized.

Tool tutorials: Throughout the semester, you will need to use software engineering and project management tools such as UML modeling tools, version management tools (GIT), and testing tools such as JUnit. Links to material on these tools will be available on the course website. Please familiarize yourself with these tools as soon as possible.

Note: the grading weights for midterms and the project is **tentative**, and may change **slightly** depending on their relative difficulties.

Tentative Schedule

- Introduction to software engineering and iterative development.
- Requirements analysis and capture. Use cases, UML use-case diagrams. Non-functional requirements. Domain models and UML.
- Sequence diagrams. Operation contracts. UML class, object, and interaction diagrams.
- Architectural Design.
- Objects with responsibilities. GRASP design patterns.
- GRASP: Polymorphism, Pure Fabrication, Indirection, Protected Variation. GoF Design Patterns.
- GoF Design Patterns: Adapter, Factory, Singleton
- GoF Strategy, Composite, Facade, Observer.
- Concurrent/Multithreaded Programming in Java
- Concurrent/Multithreaded Programming in Java, Activity Diagrams, State Diagrams.
- Data abstraction, Type hierarchies, polymorphic abstraction, Specification.
- Specification, testing, and verification.Test-driven development, Unit testing, Refactoring.
- More on GoF patterns.
- Term Project discussions.
- Term Project demos.