CS246—Deliverables, Group Project (Spring 2015)

Due Date 1: Friday, July 17, 04:55pm
Due Date 2: Tuesday, July 28, 04:55pm

Your project will be graded based on correctness and completeness (60%), documentation (20%), and design (20%). Grading will proceed in two phases, outlined below:

Correctness and Completeness

The correctness and completeness component of your project will be assessed via a live demo in front of a TA. You will need to sign up for a 20-minute demo slot with a TA; both group members must show up, and you should plan a demo that takes the TA on a tour through all of the required features of the project. Be prepared to explain to the TA how you implemented the various parts of your project. At any time, the TA may ask you to demonstrate a particular element of your project (e.g., whether some boundary case is handled properly). If there is a component of your project that is not working, advise the TA, rather than pretend that it does.

After you have demonstrated the core requirements, you may demonstrate any additional features of the project that you may have implemented, for extra credit. The TA will assess the worth of your additional features after your demo is over.

An announcement regarding when time slots are available for demos will be posted on Piazza. It is your responsibility to book a demo slot once this announcement has been made. Demo slots are chosen on a first come first served basis.

If your project submission on Marmoset does not compile, you will not be allowed to continue with your demo. The TA will not, under any circumstance, change your code, or allow you to change your code, in order to make your submission compile. You will receive a grade of 0 on the correctness and completeness portion of the project.

Documentation and Design

Your documentation and design will be assessed via written documents that you will submit to Marmoset as PDFs.

Sharing code with your partner

To complete the project, you and your partner will have to share source code. We recommend using git and the `git.uwaterloo.ca` service to facilitate this sharing. Refer to sections 1.7 and 1.8 of the course notes for instructions on how to set up a git source code repository and how to set up permissions to allow your partner to access it. Note, only one person in each group should setup the repository. Even if you are working alone on the project, it might be useful to setup a git repository to make use of Version Control functionalities. **Ensure that any git repository you create is configured to be private.**

On Due Date 1

Submit the deadlines.txt document. Your assignment will not be marked if you do not submit this document. Make sure to read this document to understand the deadlines and requirements expected of you.

Submit a plan of attack. This must include a UML that describes how you anticipate your system to be structured (even if you complete the project by due date 1, the UML you submit must be one that you built prior to starting the project).

Your UML must show the classes that make up your project and the relationships between them. You only need to show public methods for the initial UML (i.e, you can leave out private fields and protected/private methods, unless you need to show them to illustrate a point, e.g., a design pattern). You will not be graded on the degree to which you adhere to this model, but you will be asked to account for any differences that arise between this model and your final submission. If during implementation you determine that your UML is inaccurate **do not** try to adhere to the UML. Make the required changes and document them for later explanation to the TA. File: `uml.pdf`.

In addition, your plan of attack must include a breakdown of the project, indicating what you plan to do first, what will come next, and so on. Include estimated completion dates, and which partner will be responsible for which parts of the project. You should try to stick to your plan, but you will not be graded by the degree to which you stick to it. Your initial plan should be realistic, and you will be expected to explain why you had to deviate from your plan (if you did).

Finally, your initial plan of attack must include answers to all questions listed within the project specification itself. You should answer in terms of how you would anticipate solving these problems in your project, even though you are not strictly required to do so. If your answers turn out to be inconsistent with your final design, you will have an opportunity to submit revised answers on Due Date 2.

Your plan should be no more than 5 pages long.

On Due Date 2

On Due Date 2, you must submit all of your code to Marmoset, together with a Makefile, such that issuing the command `make` builds your project. Your executable should be called `cc3k` for ChamberCrawler3000, `bb7k` for BuildingBuyer7000 or `pp9k` for PawnPusher9000,

In addition, you must submit a final design document. It must outline the final, actual design of your project, and how it differed from your design on Due Date 1 (if it did). You must include an updated UML, reflecting the actual structure of your project. This UML must show all fields and methods regardless of visibility. Do this even if it is the same as the original UML.

Your document must provide an overview of all aspects of your project, including how, at a high level, they were implemented. If you made use of design patterns, clearly indicate where. Your document must describe each class used and its relationships to other classes.

Your system must employ good object-oriented design techniques, as presented in class.

Include all answers to questions posed within the project specification, and indicate how they differ from the answers you gave on Due Date 1 (if they did).

You should not expect the TA to read through all of your code. Therefore, your design document should stand alone – the TA should not need to have your code open in order to understand your document. However, if you wish to highlight certain aspects of your design, you should indicate clearly where they can be found in your code, if the TA wants to have a look.

Finally, answer the following questions:

1. What lessons did this project teach you about developing software in teams? If you worked alone, what lessons did you learn about writing large programs?

2. What would you have done differently if you had the chance to start over?