CS230 Data Structures, Wellesley College

Final Project Description

During the rest of the semester, you will have the opportunity to work on a programming project in an application area of interest to you. There are four stages to the project, each of which requires you to submit work that will be graded. This page provides overall guidelines for the project and for the stages, and lists some possible project ideas that you may consider. **It is required that you work in teams of 2 or 3 on this project**, and you will need to document the contributions of each partner. Working in teams is the norm in the programming world, so you will benefit from this process. If, for some reason, you cannot work with another classmate, you will need to get permission from your professor. To view some project ideas visit the CS230 Gallery.

General Guidelines

For the programming project, you will **design, implement, test and document** an entire program largely from scratch. You can receive advice and help from the Instructors on your design, implementation, and testing, but you should complete the programming on your own as much as possible. You are expected to write a fairly extensive program that is larger and more challenging than the programs that you completed in assignments during the semester. You are allowed to use any of the standard classes that are in your textbook, Java Foundations, the Java SDK, or any code that has been created or provided in CS230 this semester (for examples, the implementations of various Abstract Data Types that you use in our labs and handouts). It is expected that all other parts of the program will consist of code that is developed by you from scratch. Unless you are given explicit permission from one of the Instructors, **it is not acceptable to use code from other resources**, such as other books, the Internet, or previous semesters of CS230, in either an original or modified form. Use of code from these other sources without permission would be a violation of the Honor Code.

Your program should use **at least two** of the following data structures (**ADTs**) covered in class: **LinkedList, Stack, Queue, Tree, Graph, Hash Table and Heaps/PQs**. Your program should also include multiple class definitions. Although this will vary among projects, there should be **at least three new classes** that you define (in addition to any classes that are part of the implementation of the ADTs). All projects will require a graphical user interface (**GUI**). If your project involves the manipulation of data, you may want to store this data in text file(s).

To help you in the development of the project, we have divided it into four phases.

Phase 0: Project Proposal

Deliverables: Email with project title and one-paragraph description of your project [5 points]
Due: 11/23/15 by 11:59 pm

At this stage you should form a team of 2-3 students. Feel free to use this Google Doc for coordinating team members. To make sure that you are moving in the right direction as a team, you should contact your instructors (email us or visit our offices) with ideas of projects you may have well in advance of Phase 1. This way we can help you choose and shape a good project. By the date above you should add to the this document the **title** of your project and a one paragraph **description** of what your project is about. We will acknowledge your submission and give you permission to proceed or ask you to adjust your project accordingly.

Phase 1: Specifications

Deliverables: User's Manual and Technical Report [10 points]
Due: 11/30/15 by 11:59pm

For phase 1 of the project, you should store in a directory named **phase1** in each of your CS230 directories named **final-project** a description of your project and an outline of the overall structure of the program and its use of data structures. This initial description can be written in English **jointly by all team members** as a Google Doc. It is OK to describe some of these components in Java. The description should include the following two documents:

1. The **User's Manual** of what you expect will be the overall behavior of your program. For the GUI (graphical user interface), draw a picture of what you expect the interface to look like. Explain how the user will be able to interact with your program. Of course, you can update this manual at a later phase, but it will be very useful to have it in place this early in your design.

2. A **Technical Report** with description of:

2a. The **ADTs** that will be used and what information they will store. Include a brief justification for each of your choices.

2b. A list of the important **classes** that you expect to define for your project with a brief description of the purpose of each class. Some of these classes should capture the basic objects that exist in the problem. There may also be classes that embody the graphical user interface, or the main() method. This list should include the classes that implement the ADTs that you plan to use. Note that as you proceed with your program development, you may discover other classes that would be useful to define for your application.

2c. A list of some of the main **actions** that you expect to be embodied in methods in your new class definitions (you do not need to include the basic operations defined for the ADT classes that you plan to use). As you proceed with your program development, you will probably discover additional useful methods to define for various classes.

If you are unsure of what to include in the two documents documents above, please ask one of the Instructors. In fact, before the due date for Phase 1 of the project, you are encouraged to meet with an instructor to discuss your choice of project topic and data structures. No need to hand in hard copies of your two documents, but make sure you save them as PDF (no doc). **All members** of a team should include these documents in their final-project directories, though they are expected to be identical documents.

Phase 2: In-class Presentation of the GUI

Deliverables: In-class progress presentations, presentation slides and labor-plan [10 points]
Due: 12/3/15 by 10:00 am

During these class periods, each group will share your project ideas with the rest of the class. It is understood that, even though you will not have a completed program at this point, yet you can describe your general problem or application, choice of data structures, and user interface. You might also comment on what aspects of the project are especially interesting or challenging.

1. You should prepare in advance a Google Slides **presentation** and store it in a directory named **phase2** inside your final-project directories. In addition add the url of your presentation to this document. Share the presentation with the instructors.

2. In class each team will also give a 7 minute presentation that clearly conveys **to your classmates**, not to your instructors, what you will be doing in your project. **IMPORTANT:** You are expected to be present at **all** presentations in your lecture period, and give feedback to your classmates.

3. In your **phase2** directory include also a document named **labor-plan.txt** (soft copy only) that describes the plan on how the programming effort will be distributed between the partners. Some programming for a joined project can be done together, but there should be parts that each individual student is largely responsible for developing.

Phase 3: Completed Project

Deliverables: Final documents with javadoc-compliant documentation, and meeting with Instructors [75 points]
Due: To be completed no later than the last day of exam period

In this final phase, you should submit your **final, completed, and working program**. Place all your documents in a directory named **phase3** in your **final-project** directory. You should also arrange a time to meet with your instructors to demonstrate your final program and talk about your experience with its development.

Bring to the meeting a nice, professional looking folder containing a hardcopy of the user manual, all of the code files in your project, as well as a description of how to run your program. If parts of your program do not work, or are not working as originally planned, provide some additional notes about this and create screenshots demonstrating the problems (if applicable, to help us determining the severity of the bug).

Your final program should encompass principles of good program design, such as the effective use of classes and methods, informative names for variables and methods, and code that is efficient and concise. With regard to documentation, you should provide a comment at the beginning of every method, describing what the method does, and at least one comment at the top of every class file, briefly describing the class and its purpose. You are encouraged to use javadoc-compliant comments.

As this is a joint project, the efforts of each team member should be documented in the comments to the code (i.e. each class file should include a comment stating which student was primarily responsible for its implementation). This kind of additional planning and documentation is common in real-world software development projects that are implemented by a team of programmers. In your phase3 directory, include also a file named **labor-division.txt** (soft copy only) that describes the contributions of each team member.

Project Ideas

The following are some initial project ideas. They may be helpful in getting you thinking about the project you want to implement. You are welcome to choose a project outside of this list, but should discuss your idea with an Instructor. To see some of the better projects that previous CS230 students have created, take a look at **our gallery of past cs230 projects.**

Games, puzzles...

Build a program that allows a user to play a game. If you want to do a card game, you need to choose one that is much more complex than the War card game, because this existing implementation already provides large pieces of code that are relevant to many card games. If your favorite game is too complex to implement, you may change the rules to something more manageable.

Search Problems

The general task in search problems is to find a path from some point of origin to some destination point. The maze search problem was one example. You could build a GUI around a maze-solving program that would allow the player to solve a set of mazes. When the player gets stuck, the program could suggest a move to help the user. Other examples include a program that uses a database of flight information for an airline to determine a sequence of flights that a traveler could take to get from a city of origin to a desired destination, or a program that solves puzzles of some sort.

Computer Simulation

Write a program that simulates the behavior of a complex system, analogous to the printer simulation but much more elaborate. Some example systems that you could model include traffic flow at a 4-way intersection, flow of planes and passengers at an airport, or flow of patients in a large hospital.

Mini Expert System

Build a program that embodies expert knowledge about a domain, and can use this knowledge to make inferences from data provided by the user. For example, a simple medical expert could collect data about a patient's health and try to diagnose the cause of the patient's illness. An expert system could also identify a class of objects based on observed properties, or make decisions about actions to perform in response to a current situation. Many expert systems build their knowledge into simple rules, such as those used in the Zoo animal identification program.

Maintaining a Database

Write a program that allows a user to work with a database of stored information about a particular topic. (**Warning**: There is a strong and long theory on databases you do not know about yet (take CS304!), so this is not the ideal type of a project; but if you absolutely want to create a database with a few specific operations, it is possible.) The user should be able to add new entries to the database, modify entries, and retrieve information, and interact with the program through a graphical user interface. Some sample application areas include maintaining a database of student records, a store inventory, or an address book. (A calendar is not advised as a project, since it has many complications.) You need to think about how you can add an aspect of complexity to the application so that it goes beyond the straightforward application of the basic table operations that we discussed in class. Such complexity may be incorporated through the nature of queries that the user can make about the data or knowledge stored in the database.