

# APS 105 – Computer Fundamentals

## Project Part 2: Specification

Fall 1999

To be completed *before* your lab period, week of November 22. The specification is worth 30% of the final project mark, and must be completed or the remainder of your project will not be marked.

### 1 Overview

The purpose of the *specification* is to clearly communicate to your ‘customer’, i.e. your TA, that you have been actively working on your project and that you are on track for completing it on time. Your specification forms a *contract* with your TA, and is a promise or commitment of what you *must* deliver to them on the demonstration day.

It is very important that you complete your specification. **If you do not have a specification prepared to be marked, the TA will be unable to mark your final implementation report or your demonstration.**

### 2 Details

Your specification should indicate to the marker that you have put thoughtful planning into the implementation of your project. It should demonstrate your understanding of the problem, and describe the approach you will take to solve the problem. Any feasibility tests, prototyping, and implementation trials should be complete by the time you write the specification.

The specification is an accurate description of exactly what your project will do, what features it contains, how it is organized (broken up into methods), and what data structures you will use. It is a combination of a *requirements specification*, which promises a list of features, and a *design specification*, which technically describes how your code is to be organized and fit together to work.

All of your research, experimentation, or prototyping must be done by this point, so you should be familiar with any new concepts you will be using in your project such as reading files, loading images, writing Applets, understanding encryption technology, and so forth. You must also have organized your project into a number of well-defined classes and methods, in a similar way to how the Ordered Set ADT was organized. By breaking your project up into classes and multiple methods, each with clear goals, you will find it much easier to complete your whole project.

**It is far better to get a simplified version of your final project working than it is to have a very large, complex project that does nothing. If there is nothing working for the TAs to test, it will be difficult for them to give you any grades. Taking the time to write a good specification will help you identify the important parts of your project so that you can get them working first.**

In writing your specification, you should separate it into two main sections. Each of these sections will be described below. Keep in mind that the both sections combined must fit on one page of typewritten text, and can share one page of hand or computer drawn figures, pseudocode, interface specifications, etc.

### 3 Requirements Specification Section

This section of your report is a firm commitment of the list of features your program must have. It should also clearly describe how the user would use the program to access these features. For example, you must clearly indicate if you are using applets or not. You must describe what options the user has at each point in your program. You must precisely describe the effect that each option has on the program.

For example, in a game of chess, the user simply specifies the source and destination squares on an alphanumeric grid that overlays the chess board. However, you must also indicate whether the computer is automatically checking whether a king is in check, and if it will prevent the user from making illegal moves while in check. Also, you must indicate whether the program would enforce rules for castling or for capturing pawns *en passant*. Additionally, you must indicate whether the user will be given options such as undo, and how many levels of undo are supported.

As well, the requirements should describe what happens if the user does something unexpected or erroneous, such as enter incorrect an grid location on the chess board.

The requirements specification should include as many of the following items as are relevant to your topic.

1. The format and source of all input and techniques for obtaining it.
2. The format and destination of all output and techniques for displaying it.
3. An overview of actions resulting from any input.
4. Actions, possibly corrective, as a result of any errors.
5. Special resources the program will need or use (files etc.).
6. Limitations of the program and any restrictions on its use.

Your requirements section describes a firm contract that you are committed to fulfilling on demonstration day. As a result, you must be careful not to include promises that you cannot meet, such as ‘the computer will play chess against the user.’ Also, it is very important that you do not under-specify your program, and allude to features that will be included ‘if you have time.’ Imagine that your TA is your customer, and they may not purchase your product if it is too weak. For example, the following type of statement is unacceptable: ‘My project will display a chess board and, if I have time, it will be able to enforce the common legal chess moves for each piece.’

### 4 Design Specification Section

This section of your report is a description of how you are planning to organize your program to meet the requirements. Since you have already done careful planning and prototyping of your project, only minor deviations from this specification will be allowed. Major deviations from the design specification are an indication of poor planning, so your marks will be reduced accordingly. Your design will describe how you plan to break up your project into methods and classes, how these classes and methods relate to each other, significant algorithms or pseudocode, the data structures you are using to store information, and so forth. The design specification should include as many of the following items as are relevant to your topic.

1. Algorithms you plan to use.
2. Data structures you plan to use.

3. Classes you plan to use (built-in to Java).
4. Classes you plan to use (of your own design), and their interfaces.
5. Descriptions and/or the organization of the objects you will use or implement.
6. Details of complicated code segments.
7. Solutions to anticipated problems you will encounter during implementation.

In the chess game example, you would describe that the `ChessBoard` would be an object containing a two-dimensional array of `ChessSquare` objects. To the user, the columns are described by the letters 'a' through 'h', and the rows are described by numbers 1 through 8. The row and column values must be mapped to array index values of 0 through 7.

Each `ChessSquare` object would know its location on the chess board. As well, it would contain an integer to describe whether it is empty, a pawn, a queen, etc. The method `canMoveTo()` (`ChessSquare`) would indicate whether 'this' piece can move to another given square (i.e., the given square must be empty, and a valid move for this piece). Of course, the state of the chess board may restrict motion (if a king is in check, or another piece is blocking the path). Consequently, the `ChessBoard` object must have methods for checking the state of the board, such as `findKing()`, which would be used by `isKingInCheck()`, which in turn would be used by `isValidMove()`.

The chessboard example above is still very loosely defined: the relationship between the methods is still not very clear, and other required methods have been left out. In fact, there is probably a better way to organize the chess game, which would only be discovered by planning and prototyping — your first ideas are probably not your best ideas, so be prepared to throw them out and start over *before* completing your design specification.

Your project design specification must be more complete, accurate and precise than this chess example, so that the TA can understand exactly what you are going to do. In fact, if the TA reads only your specification document, he or she should be able to independently complete your project in Java and end up with almost exactly the same thing that you do.

With the addition of a few more comments to better describe how things work and fit together: the `Ordered Set ADT` (Lab 9) can be another example of good planning for a design specification.

## 5 Writeup Requirements

Your specification report is a summary document that you will use to present your ideas to the TA in an organized fashion. It is also a contract of commitment that allows you to proceed to the final report and demonstration stage.

For this part of the project, you are to create a two page project specification, containing a *Requirements Specification* section and a *Design Specification* section. Your specification is not to exceed two pages in length, in a font no smaller than 11 point.

There may be no more than one page of actual text, with the remainder of the submission consisting of sketches, diagrams, Java, and pseudo code. Your submission will be truncated by the marker if you exceed these length restrictions. Handwritten submissions will not be accepted but supporting material (diagrams, flowcharts etc.) need not be computer generated. **Clearly display your topic, full name and student number and TA Name** on the top of the first page of the specification and not on a separate cover sheet.

Since you have only one page, do not waste space with a large fancy title. Also, do not waste space with a long introduction or justification for why you chose your topic — you have already done that with your proposal. Instead, concentrate on the important details. Be clear and concise.

When you submit your specification for marking, make sure that you keep a second copy for yourself so that you can refer to it when implementing your program and writing your final report. The teaching assistant who marked your proposal is the only one who is permitted to mark your specification so ensure that you submit your document to the correct marker.

## 6 Grading

Your mark will not be based on how fancy your specification report looks — your grade is not based only on the written report itself. Rather, your mark is based on how well you convince the TA that you have done excellent preparation for your project, in the forms of thinking, research, planning, and design.

In your report and during the lab, you must communicate to your TA that you have been working on your project, that you have done appropriate experiments and prototyping, and that you have come up with a good design towards completing it on time. You must be able to orally justify and perhaps demonstrate how you arrived at your design specification. Your requirements specification must be very clearly laid out. There should be no ambiguities.

The TA should be convinced that all of the intellectual portion of your project is done, and that only a bit of "mundane coding" remains. There should be no doubt in your mind or the TA's about exactly what you are doing and exactly how you will do it. In this respect, it helps if your report is very clear and concise.

If your report is too long, the TA will not be able to read it in time, and you will receive a poor mark. If your report is unclear, too vague, your commitments are too weak or ambiguous, or you have not done enough planning, you will not receive a good grade.

The effort you put into your specification will also be reflected in your final *project demonstration* and *implementation report* marks. If you do not do proper planning beforehand, you may not complete your project on time. As well, if your final implementation deviates significantly from your specification, you will lose marks for non-conformance.