# CSC251 JCF Project—Students and Gradebooks

The purpose of this project is to gain experience creating, combining and navigating various collections in the Java Collections Framework.

## The Program

This program is a simple Student Body and Course Gradebook text based application. The user can create courses, create students, add students to courses, create and add assignments to courses, and analyze course/student/assignment relationships in various ways.

To accomplish these tasks, the program creates and maintains a collection of *Student* objects in a singleton class *StudentBody*. It represents courses with a class *Course*. A *Course* creates and maintains a roster of its students, and a gradebook for recording student *Assignment* scores.

A substantial portion of the program is supplied to you in the attached zip file. The zip file also contains a directory *doc* that contains Javadoc-style descriptions of both the provided classes and the classes and methods that you are to write. To view the documentation, extract the *doc* directory (it must be extracted—browsers cannot work inside a zip file) and use a browser to view *doc/index.html*.

## Objectives

This project will allow students to demonstrate the ability to

* Use elements of the Java Collections Framework to store student and grade information in a collection of courses
* Navigate the JCF elements to extract information from various collections
* Combine information from various collections to present user-chosen views
* Use JCF features to sort and manipulate elements in various ways
* Write JUnit tests

## Getting Started

Extract the code from the attached zip file into your IDE. Run the program. It won’t do much, but you can get the flow of execution. Note the functions that the program offers, but are not implemented.

As you examine the code, note the comments

*// TODO*

Most (I cannot guarantee “all”) of the enhancements you must make are identified with TODO comments. Both Eclipse and IntelliJ denote TODO comments on the right of their source editor panes. In Eclipse’s case, the indicator is a small blue horizontal rectangle. For IntelliJ, it is a small blue horizontal bar. Other information in this area of the pane indicates the location of compile problems, warnings, unused variables, etc.

Note that successfully completing this project will require you to “connect the dots” scattered across the documentation and provided code.

## Data Structures

The class *StudentBody* is a singleton containing the complete list of all students in the college. It is a *Map*, with the *key* being an *Integer studentId*, and the *value* being a *Student* object.

A *Course* keeps a roster, which is a *Map* having a *key* of *studentId* and with its *value* being a *List* of *Assignments*.

## Optional Method in Course

Completing *Course.filterRoster* is optional. It is worth 10 extra credit points for the assignment. In addition, having it available can greatly reduce the work necessary for doing other *Course* methods such as *getAssignment*.

## Grading Elements

The program

* Allows the user to create any number of courses, maintains all courses created
* Allows the user to populate any course with students, add assignments for all students of a course
* Implement the various main menu items displaying student, course and assignment information
* The GradebookApp main method uses its static methods where possible
* Appropriate collections from the JCF are chosen for the various functions (e.g., we don’t expect that very many courses will be created and manipulated by a user of this application, so a simple List will suffice, but we expect many, many students and assignments could be created, and linear searching is unacceptable)
* The code meets the corresponding JavaDoc specification
* The supplied JUnit tests all pass
* Supply at least one JUnit test for the following GradebookApp methods:
  + getStudentsTakingEverything
  + courseAverageForAssignment
  + calculateStudentAverageInOneCourse
* All scores and averages are reported to the user as a percentage
* Optionally implements Course.filterRoster for 10 additional points
* Follows your instructor’s policy on project submissions, code quality and conventions