# CSC 241 Assignment 2
# Abstract Data Types and Programming Methodology
# Due: Thursday, February 24

## 1 Introduction

In this semester, we are going to develop a Java program called "***GradeManager***", which manages students' grades. This program provides functions such as adding/deleting/editing grades of certain students. All the grades are stored in designated file(s) and updated as grading data change. It also supports to manage students, *e.g.*, adding a new student and his/her grading data or deleting a grading record of a particular student. This program will be built up through several assignments, in each of which you will be asked to apply what you will learn in lectures. At the end of semester, you will have a Java program which utilizes various OOP techniques and diverse data structures.

The assignments will be handed in an order for completing a final program. So, you MUST follow instructions and achieve requirements when you work on an assignment. Java code for each assignment should be errorless and submitted in the Blackboard course shell. Since a next assignment usually asks you to add more functions or edit what have been made in the prior assignment, you should keep the previous Java code(s). If the prior program is submitted with errors or runs unsuccessfully, it must be corrected before it goes to the next assignment.

## 2 Goal for This Assignment

The aim of the **second** assignment is to build classes to manage students' grade in a particular course. To define a class, abstract class, and super/sub-class you will use what you learned in class – lecture02 and lecture 03. This program uses certain methods and procedures made in Assignment 1, for instance, loading data from a given file, reading users' input and so on. Thus, you may copy your assignment 1 and modify it as the assignment 2 requests.

## 3 Instructions

### A. Template file

Each assignment should be built in a package. The names of package and class for this assignment are below.

**Package**: Assignment2

**Class**: GradeManager

```
package Assignment2;
        …
public class GradeManager {
        …
```

In this assignment, a template package (***Assignment2*>*GradeManager.java***) is provided. You first unzip the ***Assignment2.zip*** which is in the Blackboard course shell.

### B. Data Files

In this assignment, two kinds of data files are provided. The first kind is for grading data, which were used in the assignment 1. They will be reused.

The second type is for course information such as name, crn, capacity, code, time, and number of sections. The assignment package includes a file for course info, named "`cs241.txt`". The course information is below.

| | |
|---|---|
| **Name** | Abstract Data Types and Programming Methodology |
| **CRN** | 14607 |
| **Capacity** | 24 |
| **Code** | cs241 |
| **Time** | 13:50 |
| **Number of Sections** | 1 |

Note that all data files are organized in a hierarchy of folders. First, files for course info are in a folder named "data" under the package (i.e., Assignment2). For example, the "`cs241.txt`" is placed in `src/Assignment2/data`. Since Windows uses a different file separator, the path is `src\Assignment2\data\cs241.` Second, files for students' grade are located under a folder named the course code. For example, all grading data files are placed in `src/Assignment2/data/cs241`. Therefore, if the file path was `src/Assignment1/David.txt` in the assignment 1, it should be changed to `src/Assignment2/data/cs241/David.txt` in the assignment 2.

### C. Developing Environment

Your program should be **implemented in Java only**. The program in another language will not be graded.

### D. Submission

You will submit your Java package. Zip the package **Assignment2** again and upload it in the Blackboard course shell. The assignment will give you **two weeks**, so it is **due on Thursday, February 24**. All submission **by 11:59 PM** on that day will be accepted without any penalty. On the due date, Blackboard may be suffering of too much network traffics and be unstable. There is no excuse about the issue, therefore you are strongly recommended to submit earlier than the due date.

## 4 Requirements

### A. Abstract Class Initialization (course)

To manage grades of students, you will design a class called `course`, in which students have enrolled. The `course` class will be initialized by using the information in a course info file (e.g., `cs241.txt`). For this, user enters a course code when he/she asks shown in **Figure 1**. Review the lecture01 which introduces methods for loading files if necessary.

```
Enter a course code:
```

**Figure 1.** User enters a course code to initialize the course in console. In the assignment package, there is a course info file named ***cs241.txt*** under data folder.

Once a course code is entered, a course object is initialized with information in its course file. Note that `Course` (you can locate it in the assignment package) is an abstract class which is covered in Lecture03. It is extended by class `Section`. Those who are not sure about *extends* keyword or inheritance, need to review Lecture02c. After the initialization, the course information will be shown as illustrated in **Figure 2**.

```
Enter a course code: cs241
Name: Abstract Data Types and Programming Methodology | CRN:
14607 | Code: ccs241 | Capacity: 24 | Time: 13:50
```

**Figure 2.** Course information for cs241 is printed.

### B. Array of Class (student)

In this assignment, you will design a class called `Student`, which includes all grades of a student. `Student` object must be declared in `Course` class or `Section` class. You may decide in which class `Student` object is instantiated. Since there are multiple students in a class, you should have an array of `Student`. In this assignment, you assume that **four** students enrolled, since there are four files for students. Once a course object is created in the section A, your will open all files in the course-named folder.

`Student.java` is included in the assignment package. You should complete the class with proper methods and may add more variables if you need.

### C. Data Management

After grading data are loaded, you will manage the grading data of students or quit the program. You will enter a pre-defined menu shown in **Figure 3**. Currently there are two menus but there will be more menus in the following assignment.

```
Select menu [edit | quit]?
```

**Figure 3.** User selects what he/she wants to do. There are currently two functions – edit and quit. If "quit" is chosen, then program will be terminated.

### D. Edit Data

If "edit" is chosen, you will edit score for a particular student. Your program will ask as shown in **Figure 4**. First, you are asked to enter a student's name you want to find. Then you will see grades for the students.

```
Enter a student: John
Name: John | ID: 123456789 | Q1: 10 | Q2: 8 | Q3: 8 |
Midterm:87 | Final: 91
```

**Figure 4.** After entering a student's name John, you will see his current grade.

Once you chose a student, you should indicate a particular coursework you want to edit and a new score. You will enter a course and a new score, as shown in **Figure 5**.

```
Enter a coursework you want to edit (q1,q2,q3,mid,final): q1
Enter a new score: 9
Name: John | ID: 123456789 | Q1: 9 | Q2: 8 | Q3: 8 |
Midterm:87 | Final: 91
```

**Figure 5**. If user chooses to edit, he/she would enter a coursework and its new score. This example demonstrates how q1 has changed.

This change must update a grading data which had the old data. Thus, once a course has a new data, its corresponding file must be immediately updated by writing the new data in the original file. You also may review the method for writing a file in the lecture01.

After changing the grade and updating the grading file, your program should ask you to select menu shown in **Figure 3**.

# 5 Grading

### A. Grading criteria

The lab is assigned **30** points, which is 10% of the final grade. It will be graded by evaluating the requirement. Any missing and unsatisfiable criteria will take off points. The tentative and brief criteria are below.

- Compilation: **5** points
- Execution: **5** points
- Proper output: **20** points

### B. Late penalty

Late submission will take off **10% per day** after due date. **Thus, submission after 10 days will not be accepted in any circumstances**.

# 6 Academic Integrity

Any dishonest behaviors will not be tolerated in this class. Any form of plagiarism and cheating will be dealt with according to the guidelines on the Academic Integrity Policy on-line at http://www.oswego.edu/integrity. For more information about university policies, see the following online catalog at:

http://catalog.oswego.edu/content.php?catoid=2&navoid=47#stat_inte_inte

**Student who is against the honor code will not have any credits in this project.**