**Due Feb. 27.**

[**Project 1**](https://ccsu.blackboard.com/webapps/assignment/uploadAssignment?content_id=_657411_1&course_id=_26583_1&group_id=&mode=cpview)

**Write a program in Java according to the following specifications:**

* + The program reads a text file with student records (first name, last name and grade on each line).
	+ Then it prompts the user to enter a command, executes the command and loops. The commands are the following:
		- "all" - prints all student records (first name, last name, grade).
		- "excellent" - prints students with grade > 89.
		- "ok" - prints students with grade <= 89.
		- "end" - exits the loop the terminates the program.

For example, if the input text file is [students.txt](http://www.cs.ccsu.edu/~markov/ccsu_courses/CS152/students.txt) and the user enters "all" the program prints the following:

John Smith 90

Barack Obama 95

Al Clark 80

Sue Taylor 55

Ann Miller 75

George Bush 58

John Miller 65

If the user enters "excellent" the program prints the following:

John Smith 90

Barack Obama 95

If the user enters "ok" the program prints the following:

Al Clark 80

Sue Taylor 55

Ann Miller 75

George Bush 58

John Miller 65

**Requirements and restrictions:**

* + Use the [Students.java](http://www.cs.ccsu.edu/~markov/ccsu_courses/CS152/Students.java), [Excellent.java](http://www.cs.ccsu.edu/~markov/ccsu_courses/CS152/Excellent.java), and [Ok.java](http://www.cs.ccsu.edu/~markov/ccsu_courses/CS152/Ok.java) classes from the course website and make the following modifications/additions:
		- Create an interface Student and implement it with classes Excellent and Ok to represent the excellent and ok students correspondingly.
		- Create an ArrayList and fill it with objects of classes Excellent and Ok to store all students from the file.
		- Use the ArrayList to print all, excellent and ok students. Use the [instanceof](https://docs.oracle.com/javase/tutorial/java/nutsandbolts/op2.html) operator to distinguish between excellent and ok objects.
	+ When you write your program:
		- use proper names for the variables suggesting their purpose.
		- format your code accordingly using proper indentation and spacing.
		- use multiple line comment in the beginning of the code and write your name, e-mail address, class, and section.
		- for each line of code add a short comment to explain its meaning.

**Submit the following files as attachments to this assignment:**

The files with Java source code.

The text file used for input.

And the output.

**Due March 20**

**Project 2**

**Write a program in Java according to the following specifications:**

The program reads a text file with student records (first name, last name and grade on each line). Then it prompts the user to enter a command, executes the command and loops. The commands are the following:

* + "print" - prints the student records (first name, last name, grade).
	+ "sortfirst" - sorts the student records by first name.
	+ "sortlast" - sorts the student records by last name.
	+ "sortgrade" - sorts the student records by grade.
	+ "end" - exits the loop the terminates the program.

For example, if the input file includes the following data:

John Smith   90

Sarah Barnes 75

Mark Riley   80

Laura Getz   72

Larry Smith  95

Frank Phelps 75

Mario Guzman 60

Marsha Grant 85

the program may proceeds as follows (user input is shown in bold face):

Enter command: **print**

John Smith   90

Sarah Barnes 75

Mark Riley   80

Laura Getz   72

Larry Smith  95

Frank Phelps 75

Mario Guzman 60

Marsha Grant 85

Enter command: **sortlast**

Students sorted by last name

Enter command: **print**

Sarah Barnes 75

Laura Getz   72

Marsha Grant 85

Mario Guzman 60

Frank Phelps 75

Mark Riley   80

John Smith   90

Larry Smith  95

**Restrictions and implementation:** Use a regular array (not ArrayList) of **objects** of **class Student** to store the student records. Define a **class Student** that represents a student record (student's first name, last name and grade) and implements the **Comparable interface**. Use the **bubble sort** algorithm for sorting objects **implementing the Comparable** interface. **Hint:** Define a static instance variable in class Student to specify the type of sorting and use it in the **compareTo** method to determine what to compare - first names, last names, or grades.

**ExtraCredit (maximum 2 points):** Implement "add" and "delete" commands for adding and deleting students given their names and grades.

**Documentation:** Run your program and make sure it is designed  and works exactly as described above. Then add your name in the beginning of the code as a comment. Add also comments to explain the classes and methods used. Note that comments and the way you format your program will be graded too.

**Submission:** Submit the following files as attachments:

The source code (.java) of all classes used.

The input file with student names and grades that you used to test the program.

A screen copy (an MS Word file with the image included) of the **Programs window** showing all classes used with the relations between them.

A screen copy or a text file with the content of the **Terminal Window** showing an example run of your program.

**Due April 17th**

[**Project 3**](https://ccsu.blackboard.com/webapps/assignment/uploadAssignment?content_id=_657407_1&course_id=_26583_1&group_id=&mode=cpview)

Write a program in Java that computes the number of words in a text entered in a text field. Use the following graphical objects:

* + a **text field** for the input text (one line of text)
	+ a **button** that, when pressed, computes the word count
	+ a **button** that, when pressed, clears the text field
	+ **labels** for the input and output and for showing the word count

**Requirements and Restrictions:** Choose proper background, color, and layout and arrange the graphical objects in a way  **that scene is shown very clear** . Make sure the**layout does not change** when resizing the window.

**Extra credit (maximum 2 points)** will be given for additional text statistics (for example, maximum, minimum and average word size, number of numeric tokens, word frequencies etc.).

**Documentation:** Run your program and make sure it compiles without errors and works as described above. Then add your name in the beginning of the code as a comment. Add also comments to explain the classes and methods used. Note that comments and the way you format your program will be graded too.

**Submission:** Submit the following files as attachments:

The source code (.java) of all classes used.

A screen copy of the**main window** showing all classes used with the relations between them.

A screen copy showing an example run of your program.

Due May 3rd

Project 4.

1. Design and implement an application that draws a circle using a rubberbanding technique. The circle size is determined by a mouse drag. Use the original mouse click location as affixed center point. Compute the distance between the current location of the mouse pointer and the center point to determine the current radius of the circle.
2. Design and implement a program whose background changes color depending on where the mouse pointer is located. If the mouse pointer is on the left half of the program window, display red; if it is on the right half, display green.