Group Assignment

Due: May 16, 23:59 PT

Overview

There are 6 problems, each problem contains (some of) the following levels of difficulty:

- Level 0 - Easy (1 person)
- Level 1 - Standard - G/F (2 ~ 3 people)
- Level 2 - Advanced - G/F (3 ~ 4 people)
- Level 3 - Challenging - G/F (3 ~ 4 people)

For level 1 - 3, G means GUI is the harder part; F means the functionalities is the harder part.

Pick ONE level of any ONE of the problems, either work in a team (preferred) or individually. For each level, you can work in less than the suggested number of people, but no more than.

The lower the level is, the stricter the grading will be. For level 0, you can only get up to 12 / 15 points; while for level 3, you can get up to 1 extra credit. The grading will NOT be based on the number of people in the group.

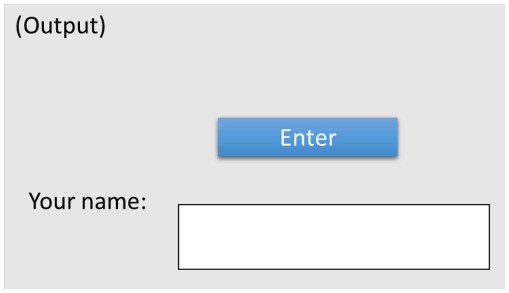
Since the code will not be auto-graded, you can manage the project, package, classes, methods, etc. as you need. Feel free to use the code in the previous assignments. You may need to modify them (return type, parameters, etc.) based on the problem.

It's ok to use code from other sources if cite them (mention in the code comment). And here, other sources do not include websites like Chegg, where other people write code for you. **If you didn't cite the source or you asked others to write your code, you will get 0 on the assignment and one letter grade off.**

For the design, you don't have to follow too closely to the requirement. For example, the problems all ask you to provide a button, and show the output after the user clicks the button. Alternatively, you can have "live update", that is, the output will change as the use enters the information. As long as the output is correct, the detailed design is up to you. You can even have different functionalities.

Also, it's up to you to design how to handle the invalid input. You may don't want to close the application though; instead, display a corresponding error message.

For the layout, you don't have to be fancy, but it should be reasonable and easy to follow. At least align everything properly. The demonstration below will be considered as a "below expectation" layout:



Submission

(Mandatory) On Gradescope

Submit the following, ONE submission for each group (every team member will get the same score):

- Source code (.java files)
- A .jar for your application (steps shown in lesson 1)
- A backlog (using the template, more details in lesson 16)
  - Please copy to your own drive using File -> Make a copy. Do NOT use copy and paste.

As usual, if you submit via GitHub, you will get 0.5 extra points. Make sure all files are submitted though.

After you submit, remember to add your team members, which means, if any of the team members didn't contribute, just don't add him/her. See how-to here: https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members

(Optional) In person meeting

If you encounter any questions, or just want some suggestions, or want to discuss the way to improve it, feel free to schedule a meeting. You may get some extra points for this only if you didn't get full points at the end. The deadline for the meeting is May 16, at 23:59 PT.

Grading Rubrics

You will get 3 / 15 points as long as you submitted everything. The rest 12 points will be graded as follows:

	Excellent (4 pts)	Good (3 pts)	Needs Improvement (2 pts)	Below Expectation (1 pts)
Correctness	Able to handle most of invalid user input, clearly error messages	Able to handle some of the invalid user input or unclear error messages	Works ok but will crash on invalid user inputs	Some functionalities not working
Coding Style	Well-formatted, well-documented, no unused code, clear logic	Well-formatted, has documentation, no unused code	Well-formatted but no documentation, includes unused code	Not well-formatted, code hard to understand
Overall polish	Pretty and easy to follow layout, friendly and clear prompts	Reasonable layout, clear prompts	Layout not so easy to follow	Confusing layout/prompts

And you may get some extra credits (submit via GitHub, choose level 3, etc.)

Problem 1 - Time Difference

Write an application with GUI to calculate time difference between two times. No level 1/3 for this problem.

Level 0 - Easy (1 person)

The application includes text fields to take the user input for the two times. Once user clicks the "calculate" button, the time difference will show on screen. The rule is the same as assignment 1.

Level 2 - Advanced - G (3 ~ 4 people)

The application includes text fields to take the user input for the two times, and the number of days. By default (if user didn't input the number of days), it's 1 day as assignment 1 if the second time is less than the first time. Once user clicks the "calculate" button, the time difference will show on screen.

And there should also be two clocks to show the times based on the user input.

Hint:

- Modify the printTimeDifference method so it also takes an int for the number of days. Then instead of adding 24 hours to the second time 2, add 24 * number of days.
- Have a class as a component to draw the Clock (extends JComponent). Have another class as a panel (extends JPanel) that includes the Clock and text fields to take user inputs.

Problem 2 - Card

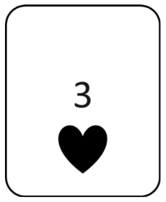
Write an application with GUI to display a card based on the shorthand. No level 2/3 for this problem.

Level 0 - Easy (1 person)

The application includes 1 text fields to take the user input for the shorthand of a card. Once user clicks the "get full description" button, the full description of the card will show on screen. The rule is the same as assignment 1.

Level 1 - Standard - G (2 ~ 3 people)

The application includes 1 text fields to take the user input for the shorthand of a card. Once user clicks the "get card" button, the corresponding card will be drawn on screen. For simplicity, you can draw a simple card like below, instead of a more realistic card (if you designed fancier cards, the grading may be less strict). The rule is the same as assignment 1.



Hint:

- Make getSuit and getRank public so you can use them in other classes.
- For each suit, define a class as a component to draw the suit (extends JComponent).

Problem 3 - Matrix

Write an application with GUI for matrix calculations. Since taking a matrix as the user input is not so straight-forward, this problem starts with level 1. Check How to Use Tables for more details. Alternatively, you can have a grid layout of text fields.

You can either use/update your Matrix class, or use an existing package such as JAMA (remember to cite if you use external package).

Level 1 - Standard - G (2 ~ 3 people)

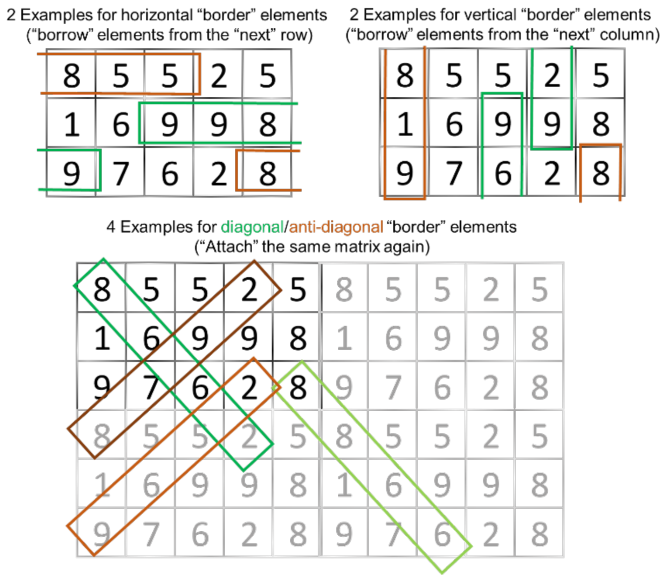
The application takes two matrices from user input. You can ask user to enter the size they want first, or display two fixed sized table and allow them to add more rows/columns later. Once user clicks the “multiply” button, the resulting matrix will show on screen.

Level 2 - Advanced - F (3 ~ 4 people)

The application takes two matrices from user input. The user can select to do the multiply or addition. The user can also transpose one of the matrices, or get the sum of diagonals if the matrix is a square matrix. Other rules same as level 1.

Level 3 - Challenging - F (3 ~ 4 people)

The application takes two matrices from user input. It includes all functionalities mentioned in level 2. The user can also select to compute the maximum product of 4 horizontal, vertical, diagonal, or anti-diagonal adjacent elements if the size of the matrix is over 3 * 3 (inclusive). Note that the rule is different from what you can find online. Treat the "border" elements as follows ("borrow" the elements in another row/column):


2 Examples for horizontal “border” elements (“borrow” elements from the “next” row)
2 Examples for vertical “border” elements (“borrow” elements from the “next” column)
4 Examples for diagonal/anti-diagonal “border” elements (“Attach” the same matrix again)

This level will focus on functionality, so the grading for layout design will be less strict if your functionality is correct. Highly suggest you think about this problem to have a better understanding for 2-d arrays.

Hint:

- Use modulus.
- Vertical is the transpose of horizontal; anti-diagonal is the transpose of diagonal.

Problem 4 - Write Primes

Write an application with GUI that will write the primes from user input. No level 2/3 for this problem.

Level 0 - Easy (1 person)

The application includes text fields to take the user inputs. Once user clicks the “find primes” button, all the primes in the user inputs will display on the screen. You can use other delimiters instead of “,” and lines.

Level 1 - Standard - G (2 ~ 3 people)

The application allows user to select a file for the inputs, other than using text field. That is, the user can choose to enter text in the text field, or select a file. The user can also choose to save the result into a file, instead of displaying the result on the screen. (Hint: use JFileChooser)

Other than find all primes in user input, the application should allow user to enter 1 number, and choose:

- Check the number is prime or not. If not, print out all its factors.
- Print all prime numbers that are smaller than or equal to the number.

Problem 5 - Appointment

Write an application with GUI that manages user's appointments. Since the Appointment class and its subclasses are more complex than other parts, the lowest level for this problem is level 2.

Level 2 - Advanced - F (3 ~ 4 people)

The application allows the user to add an appointment, delete an appointment based on the description, display all appointments and display appointments that occurs on the selected date. When displaying the appointments (all or just the selected date), user can choose to sort by start date or description.

The application also has options to save and reload the appointments. The other rules are the same as assignment 4 part 3.

Hint: use the AppointmentManager class you implemented in assignment 4 part 3:

- Modify the print appointment method so it will print the assignment based on different comparator.
- Add a method for printing the appointments that occurs on the date passed in. That is, go over your appointment collection, only prints the appointment if occursOn returns true.
- Add a method to save the appointments. You can write them to a file. Each line includes the information of one appointment. Take advantage of toString method of Appointment class
- Add a method to load the data from the file. Read each line, based on how you store them.

Level 3 - Challenging - G (3 ~ 4 people)

The application shows a calendar, so when the user clicks a specific date, it will show the appointments occur on that day. Other functionalities and rules same as Level 2.

Hint: use JTable, DefaultTableModel, and GregorianCalendar, or even use an existing package such as LGoodDatePicker (remember to cite if you use external package).

Problem 6 - Gradebook

Write an application with GUI that manages a gradebook. Since the Student class and Gradebook class involve more comprehensive materials, the lowest level for this problem is level 2.

Level 2 - Advanced - F (3 ~ 4 people)

The application displays a gradebook (a table with student id, name, and grade) and allows the user to add a student, delete a student or modify the grade for a particular student. User can also choose to sort the gradebook by name, or by the grade.

The application also has options to save and reload the grades. The other rules are the same as assignment 4 part 2. But you can choose any data structure you like as long as it's reasonable.

Hint:

- Check How to Use Tables if needed. Alternatively, you can have a grid layout of text fields.
- Use the Student class and Gradebook class you wrote in assignment 4 part 2:
  - Add a method to save the students. You can write them to a file. Each line includes the information of one student. Take advantage of toString method of Student class
  - Add a method to load the data from the file. Read each line, based on how you store them.

Level 3 - Challenging - F (3 ~ 4 people)

The application displays a gradebook with different assignments. The user can add an assignment with specified full points, and record the points each student earned for each assignment. The grade should be calculated based on the percentage of the assignment score, instead of let user enter them (for simplicity, ABCDF already enough). Of course, the use can change a score anytime and the grade will be recalculated. Other functionalities and rules same as Level 2 (except for the save and reload functionality - you can still include it if you want though).

Hint:

- You can have an Assignment class that holds the full points, and current point.
- Here is an example of the gradebook table (assignment points should be editable):

ID	Name	Grade	Assignment 1 (10 pts)	Assignment 2 (10 pts)
1	Bob	B	9	8
2	Alice	A	9	10
3	Trudy	C	7	7