CSCI 221: Computer Programming II Due, Sat.February 22, 2020, 10pm
HW 4

Collaboration Policy: This is an individual effort. We have covered everything you need to complete this assignment, however Dr. McCauley and Anuja are happy to answer your questions and guide you along to complete this assignment.

You may not:

1. Work with others to complete this assignment
2. Show you code to anyone (student or otherwise) other than Dr. McCauley, Anuja, or a CSL tutor.
3. You may not view anyone else's (student or otherwise) code.
4. You may not google solutions to these problems.
5. Use concepts not yet covered in class or text readings.
6. Post code to Slack.

You may:

1. Talk to anyone about how Java works (independent of your specific solution).
2. Google java language topics such as "how to use Scanner", "how to write if statements", etc.
3. Post questions to Slack.

Dr. McCauley and Anuja want to assist you. Please drop by or post to Slack. We do not promise to be available over the weekend.

Please note, you may not use any code found on the web to complete this assignment.

You're working for Mathworks (a software company in Boston that makes scientific computing applications) and they are designing a new user interface (UI) that requires a rectangular grid to be displayed. The UI team has already developed the UI code, specifically the `GridPanel`, `GridUI`, and `GridConstants` classes. These three classes are provided to you, and <u>**must not**</u> be modified under any circumstances. Include these files in your HW 4 source code directory.

Your job is to develop a `Point` and `Line` class whose API specifications are provided below, then use Lines to draw grids. Make sure the names of your methods match those listed above <u>**exactly**</u>, including capitalization. The number of parameters and their order must also match. Otherwise, your implementations will not work correctly and your test cases will fail.

(1) Implement the Point class.
Point Class
`Point` has two attributes used to describe it, as well as methods that operate on them. Here are the attributes for `Point` you must implement as private instance variables. Note: You may change the name of each instance variable[1].

- `x` (int)
- `y` (int)

More information about the Point instance variables:

- Initialize the x and y instances variables to 0.

[1] <u>Note:</u> value in parentheses indicate instance variable data type

- The value of the x instance variable must be >=0 and <= GridConstants.MAX_PANEL_WIDTH
- The value of the y instance variable must be >=0 and <= GridConstants.MAX_PANEL_HEIGHT

Constructor	**Description and Preconditions (if any)**
`Point()`	No parameters. In this no argument constructor initialize the x and y instance variables to 0. Precondition: None.
`Point( Point point )` xp	Parameter Point object. In this overloaded constructor use the provided point object's x and y instance variable values to initialize this new point object's x and y instances variable values. Precondition: • point != null • if x >=0 and x <= GridConstants.MAX_PANEL_WIDTH set x instance variable to value in parameter, otherwise set x instance variable to 0 • if y >=0 and y<= GridConstants.MAX_PANEL_HEIGHT set y instance variable to value in parameter, otherwise set y instance variable to 0
`Point( int x, int y )`	Parameters x and y. In this overloaded constructor use the provided x and y primitive values to initialize the x and y instance variable values. Precondition: • if x >=0 and x <= GridConstants.MAX_PANEL_WIDTH set x instance variable to value in parameter, otherwise set x instance variable to 0 • if y >=0 and y<= GridConstants.MAX_PANEL_HEIGHT set y instance variable to value in parameter, otherwise set y instance variable to 0

Get Methods	**Description and Preconditions (if any)**
`int getX()`	Get the x coordinate instance variable. Parameters none. Precondition: None. Returns int.

`int getY()`	Get the y coordinate instance variable. Parameters none. Precondition: None. Returns int.

Set Methods	Description and Preconditions (if any)
`void setX( int x )`	Set the x coordinate value. Parameter x that meets the precondition: • if x >=0 and x <= GridConstants.MAX_PANEL_WIDTH set x instance variable to value in parameter, otherwise set x instance variable to 0
`void setY( int y )`	Set the y coordinate value. Parameter y that meets the precondition: • if y >=0 and y<= GridConstants.MAX_PANEL_HEIGHT set y instance variable to value in parameter, otherwise set y instance variable to 0
`void setPoint( Point p )`	Set to an existing point object. Parameter p that meets the precondition: • p != null, if p == null do nothing. • Additionally, the x and y values stored in the provided point object must meet preconditions outlined in the setX() and setY() methods.

Before going any further, test your Point class and make sure that all constructors and method work as they should.

(2) Implement the line class.

Line Class

`Line` has two attributes used to describe it, as well as methods that operate on them. Here are the attributes for `Line` you must implement as private instance variables. Note: You may change the name of each instance variable.

- `start` (Point)
- `end` (Point)

More information about the `Line` instance variables:

- initialize the start and end instances variables to null.
- start may not be set to a null value by the user (i.e. using the set method)
- end may not be set to a null value by the user (i.e. using the set method)

Constructor	Description and Preconditions (if any)

`Line( Point start, Point end )`	Parameters two Point objects. In this constructor initialize the start and end point instance variables with the point objects passed into the constructor. Precondition: • start != null and end != null, if either is null do nothing

Get Methods	Description and Preconditions (if any)
`Point getStart()`	Get the start point instance variable. Parameters none. Precondition: None. Returns Point object.
`Point getEnd()`	Get the end point instance variable. Parameters none. Precondition: None. Returns Point object.

Set Methods	Description and Preconditions (if any)
`void setStart( Point start )`	Set start point instance variable. Parameter start that meets the precondition: • if start != null, set start instance variable to start object parameter, otherwise do nothing
`void setEnd( Point end )`	Set end point instance variable. Parameter end that meets the precondition: • if end != null, set end instance variable to end object parameter, otherwise do nothing

Other Methods	Description and Preconditions (if any)
`boolean isValid()`	Evaluates if the start and end instances variables are valid (i.e. not null). Returns a Boolean. Precondition none. If start != null and end != null return true, otherwise return false.

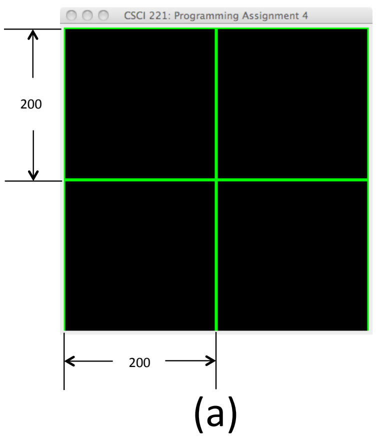
Before going any further, test your Point class and make sure that all constructors and method work as they should.

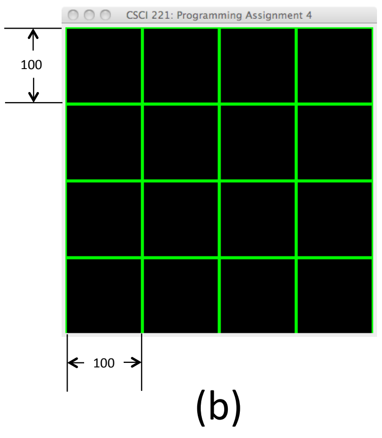
NOTE: Additional specifications:

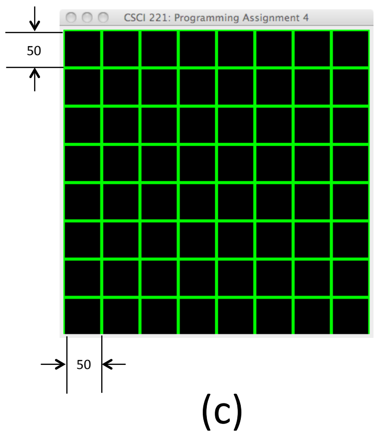
- There should be no package header in your code.
- Exception handling (i.e. try/catch blocks) should not be included in this assignment.

(3) Complete the TestGridUI Class
TestGridUI Class

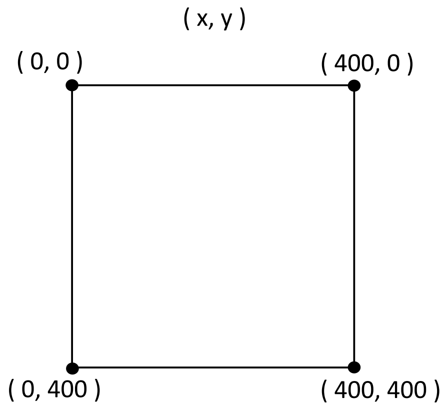
The shell of a `TestGridUI` class that only has a main method is provided. In the main method, write three test cases that generate the three UI grids provided below. (Note, the grid is made up of verticle and horizontal lines. If you Point and Line classes work properly, you can write loops to draw these grids, as all of the drawing code is provided.) As illustrated in this figure, the spacing between the grid lines in (a) is 200, the spacing of the grid lines in (b) is 100, and the spacing between the grid lines in (c) is 50. To execute these test cases you will need to instantiate a GridUI object in the main and use its `addLine` method. For more information about the `GridUI` API shown at the end of this document.







Note: For reference purposes the *(x , y)* coordinate system of the UI is provided below.



Program Documentation

Refer to the documentation standard posted in the content section on OAKS. Each method MUST be preceded by an appropriate Java doc comment. Use Eclipse short cuts (e.g. /** - return key) to generate comment stubs above class definition and methods. Follow the example given in class, or setup an appointment with instructor or TA if require more direction.

For each method in the `Point` and Line classes (in the comment block above the method) state the pre- and post-conditions, i.e. what is expected before and after the method is called (see page 300 in the course textbook).

Program Submission

Create a ZIP file that only contains the completed `Point`, `Line`, and `TestGridUI` java files, that is, no byte-code files (i.e. .class files) are to be submitted. Furthermore, please do not include a file hierarchy (i.e. the package structure). For the ZIP file you **must** use the naming convention:

<LastnameFirstinitial>.zip
e.g., McCauleyR.zip

<u>If the assignment is not submitted in the correct format – it may not be!</u> Submit the ZIP file via OAKS in the Dropbox that corresponds to the assignment. Resubmit, as many times as you like, the newest submission will be the graded submission.

Grading Rubric

10 Points	Style: Comments and Indentation
10 Points	Your test cases (in TestGridUI class)
80 Points	Functionality: • Program compiles (10) • Program runs (10) • For given inputs, program produces correct output (60)

If the submitted program does not compile: 0 of 80 points
If the submitted program compiles but does not run: 10 of 80 points
If the submitted program compiles and runs: 20 of 80 points
If the submitted program compiles, runs, and produces correct output: 80 of 80 points

The correctness of your program will be evaluated using test cases developed by the instructor. Late assignments will not be accepted – no exceptions (please do not email me your assignment after the due date, I will not accept it).

Please feel free to setup an appointment to discuss the assigned problem. I'll be more than happy to listen to your approach and make suggestions. However, I cannot tell you how to code the solution. Furthermore, code debugging is your job; please do not come to my office asking me to fix your code.

Class GridUI

Constructor Detail

GridUI

```
public GridUI()
```

Method Detail

addLine

```
public void addLine(Line line)
```

Parameters:

`line` –

The given line is drawn on the grid

clear

```
public void clear()
```

The grid is cleared.