CSC468/568 JavaFX, Observer

DUE: as stated on D2L

The purpose of this assignment is to give you practice with the JavaFX GUI with SDK 17 using IntelliJ, the MVC pattern, and the observer pattern.

Overview

You will be coding the core of a simple cat café simulation program. Cat café are combo tea\coffee houses and cats. Most in the USA also pair with humane societies to help get cats adopted (example: https://treehouseanimals.org/cafe/). The core will give you 170 points out of 200 total for the project. The remaining points will be extensions that you choose. You will be required to follow the OOP diagram given to you for the core functionality.

The simulator has a store info area, a tile info area, a command area, and a display area, as shown below. The main pieces of functionality are

1. Add/remove a cat/furniture type
2. Display the information for a tile area
3. Update to a new week
4. Display basic store information
5. Resize the display area

Program Creation Requirements

The project will only be tested with IntelliJ JDK 17. You must name your package *lastName_firstName* with the same names as shown in D2L. You must name your project *CatCafe.* In IntelliJ, this means setting your "group" to *lastName_firstName*. This is to allow my script to find your code properly and build the module. The "opens" in module-info.java should be *lastName_firstName.catcafe*. if everything is set properly. ***This does mean you may NOT have nested packages to allow the grading script to work.***

If you use resource folders, you must place your resources in the same package name as your java code following the default project structure. Do *not* put anything in the root as this may overwrite someone else's resource.

Layout

Preliminary note: the first tier allows for a simple HBox with all elements to test the button events. This in intentional to encourage the MVC pattern.

While I do expect minor variations, the initial core layout must be very close to the following (main areas marked in orange):

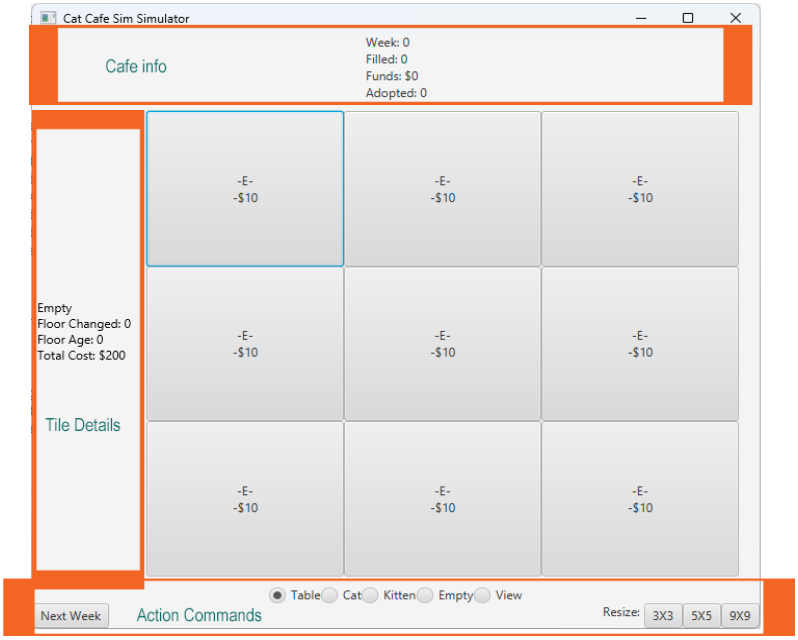


Figure 1

I will be specifically checking the layout components. The window layout begins with a size of 600 X 300 pixels, but is resizable with the following still true:

1. The cafe information bar is on the top of the application, and its text is centered.
2. The tile area information is at the left of the simulation area.
3. The tile area information is centered vertically and horizontally.
4. The action command tile is on the bottom of the application.
5. The action command "Next Week" is at the left in the action command tile.
6. The tile type radio button set must be centered horizontally between "Next Week" and the resize buttons. While not required, you may stack these, have them above the next week button, etc.
7. The resize buttons are labeled, and stacked horizontally, on the right of the action command tile.
8. The simulation area buttons fill the ***remaining area*** in the right area of the screen.
9. The tiles are even rectangles in a grid.

Here is an example for the full screen:

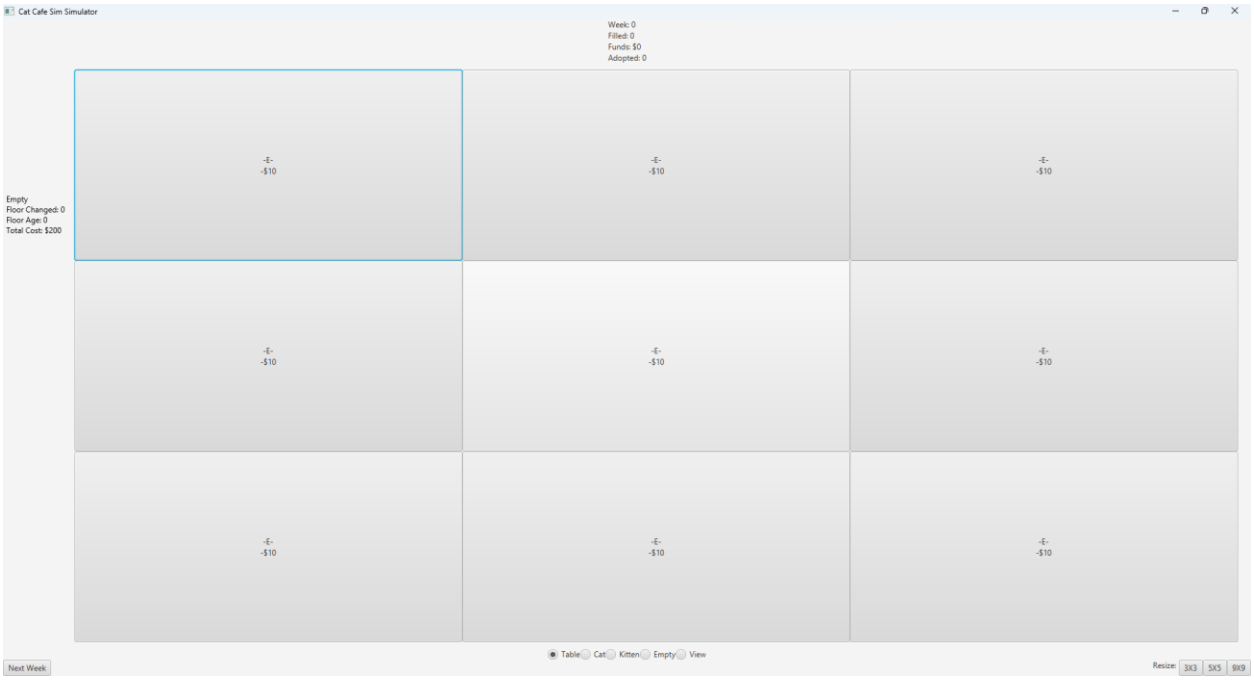


Figure 2

I will not test sizes smaller than the initial size, but some of my example images may be smaller to fit on the paper.

You may do this in code or with the WYSIWYG editor (**Reminder**, I will not answer questions about the WYSIWYG. However, keep in mind that the WYSIWYG is Gluon's SceneBuilder).

Tip: Use a Region with its grow priority set to "always" to push other views away from each other.

Add or Remove a Tile Type

For clarity, I use tile to refer to the button "tile", and floor area to refer to what that tile area represents.

Select the floor area type from a set of radio buttons, along with one extra option to "view" a tile. See Floor Area Stats below for the type. You can continue to add the type repeatedly. A tile area displays the floor area type's first letter, a new line, and then supply\food cost that week. If a furniture floor area, append a new line, and then the revenue that month. If a cat\kitten floor area, append a new line, and then add a "count down" (for adoption). It follows the following format:

```
-<letter>-
-$<cost>
<blank if empty OR weekly revenue OR time until adoption>
```

A tile area must be a subject (PropertyChangeSupport member variable) in the observer pattern, and there must be a custom view named TileView derived from some Node (I used a Button) to act as the observer (PropertyChangeListener interface). This will allow for any changes in the tile to result in an

immediate update of the tile area button. Notify the view after any and all changes in order to display the changes.

For grading purposes, tag the following with inline comments:

- The addPropertyChangeListener(…) call with ***GRADING: SUBJECT***.
- The propertyChange() function member variable with ***GRADING: OBSERVE***.
- The firePropertyChange(…) call(s) with ***GRADING: TRIGGER***.

If desired, you can also make the observer pattern yourself if you do not wish to use the built ins, but you will still need all three tags.

Code help: This is how you can get the radio button choice string anywhere in code:

```
((RadioButton)(group.getSelectedToggle())).getText();
```

Floor Area Stats

A table area

- Costs $300 (initial purchase, do not display this in the tile)
- Costs additional $50 a week 5 (ingredients costs)
- Adds $150 in revenue a week

A cat area

- Costs $200 (initial purchase of toys/ carriers, etc., do not display this in the tile)
- Costs additional $30 a week (food costs)
- Age starts at 52 weeks.
- Countdown starts at 8. When it hits 0, the cat is adopted, and a new cat arrives.

A kitten area

- Costs $200 (initial purchase of toys/ carriers, etc., do not display this in the tile)
- Costs additional $20 a week (food costs)
- Age starts at 10 weeks.
- Countdown starts at 4. When it hits 0, the kitten is adopted, and a new kitten arrives.

A empty area

- Costs $200 if there is something there (disposal costs)
- Costs $10 a week (rental costs)
- Adds $0 in revenue a week (so it does not show in the tiles)

The view radio button is “special”. Rether than changing the floor area on clicking, it display additional information about the floor area in the tile info area.

For example, starting with the state in Figure 2, I put in table areas in the left column, and cats and kittens in the right column. I have selected kitten tile.

Figure 3

The next few weeks (with the kitten selected) are as follows:


Cat Cafe Sim Simulator
Week: 2
Filled: 6
Funds: $-1120
Adopted: 0
-T-
-$50
$150
-E-
-$10
-K-
-$20
2
-T-
-$50
$150
-E-
-$10
-C-
-$30
6
-T-
-$50
$150
-E-
-$10
-C-
-$30
6
Kitten
Floor Changed: 0
Floor Age: 2
Total Cost: $240
Cat age: 12
Weeks until adoption: 2
Table
Cat
Kitten
Empty
View
Next Week
Resize:
3X3
5X5
9X9


Cat Cafe Sim Simulator
Week: 3
Filled: 6
Funds: $-930
Adopted: 0
-T-
-$50
$150
-E-
-$10
-K-
-$20
1
-T-
-$50
$150
-E-
-$10
-C-
-$30
5
-T-
-$50
$150
-E-
-$10
-C-
-$30
5
Kitten
Floor Changed: 0
Floor Age: 3
Total Cost: $260
Cat age: 13
Weeks until adoption: 1
Table
Cat
Kitten
Empty
View
Next Week
Resize:
3X3
5X5
9X9

Figure 4

Display Basic Information About the Store

The store information panel shows the number of weeks since the last resize, the number of filled tile (not empty), the funds currently available (even if in debt), and the number of cats adopted. This must be displayed in the format shown.

Cat adoption numbers should only change if a cat's countdown hits 0, not if the floor area changes.

Update the store information bar as soon as you adjust a tile area.

New Week

Clicking the next week button should update the stats in the following order:

1. Increment the week
2. Apply any effects of the week (e.g. get costs, revenues, ages, adoption, etc.)

Tile Area Information Text

If the "View" radio button selected, clicking a tile will display more information about the tile in the tile area information tile. This must be formatted as shown below:

For furniture, the format is

```
<name>
Floor Changed: <week>
Floor Age: <week>
```

```
Total Cost: $<totalCost>
Total Revenue: $<totalRevenue>
```

For cat\kittens, the format is

```
Floor Changed: <week>
Floor Age: <week>
Total Cost: $<totalCost>
Cat age: <week>
Weeks until adoption: <week>
```

You do not need to automatically update the information in the tile info area when the tile updates\changes, unless you choose that as an extension.

Resize the Simulation Area

The resize buttons should resize to the dimensions as shown in Figure 6. Clicking any one of these buttons will clear the store, reset the info, and remake the store to be a new grid size, even if the store is already the same size. 3 X 3 will have the p button grid area. 5 X 5 will have 25 buttons. 9 X 9 will make 81 buttons. You can take the sizes mentioned as a hint as to how I made my simulation area. For example, clicking 9 X 9 at the state from Figure 4 results in the following:

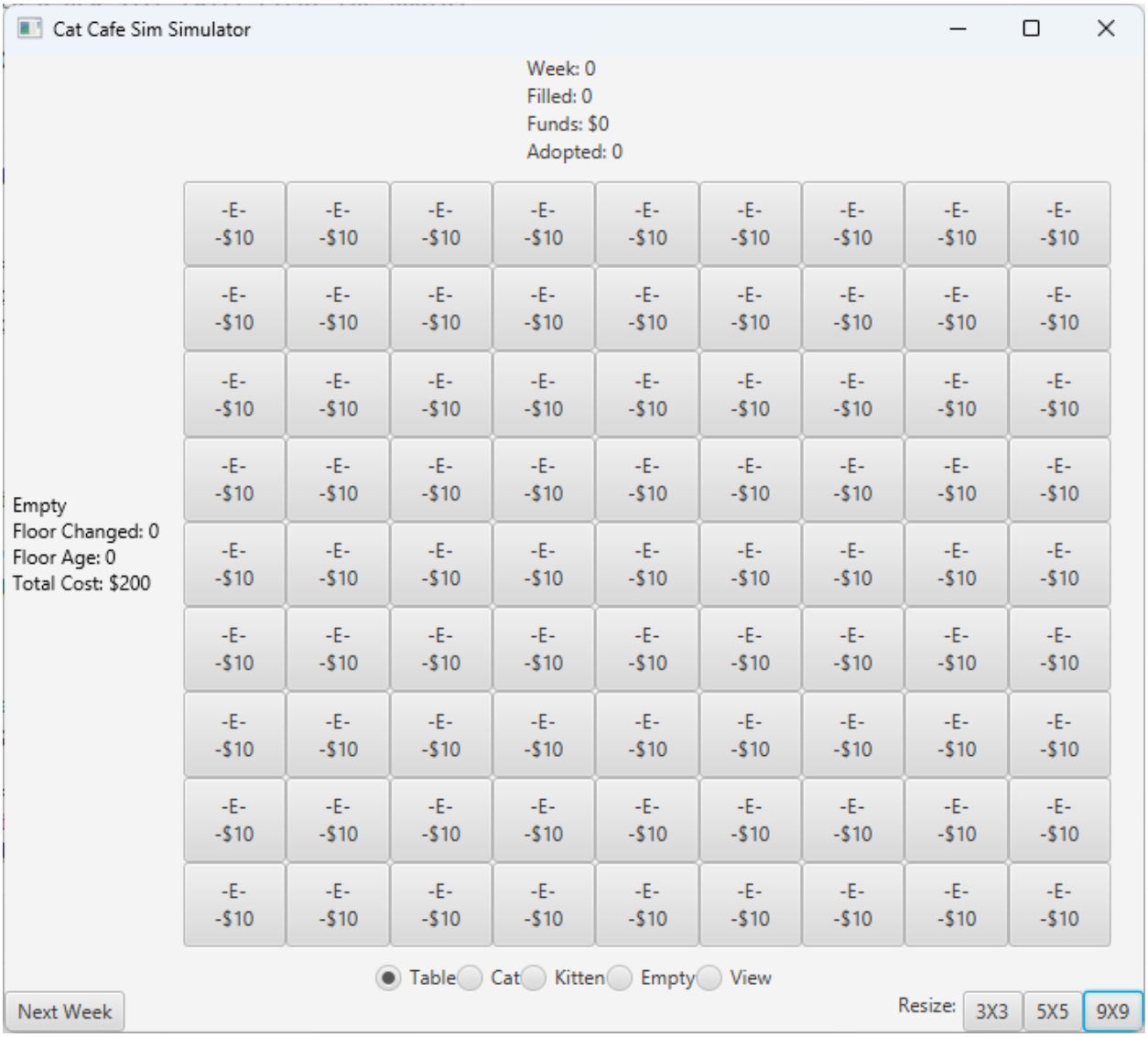


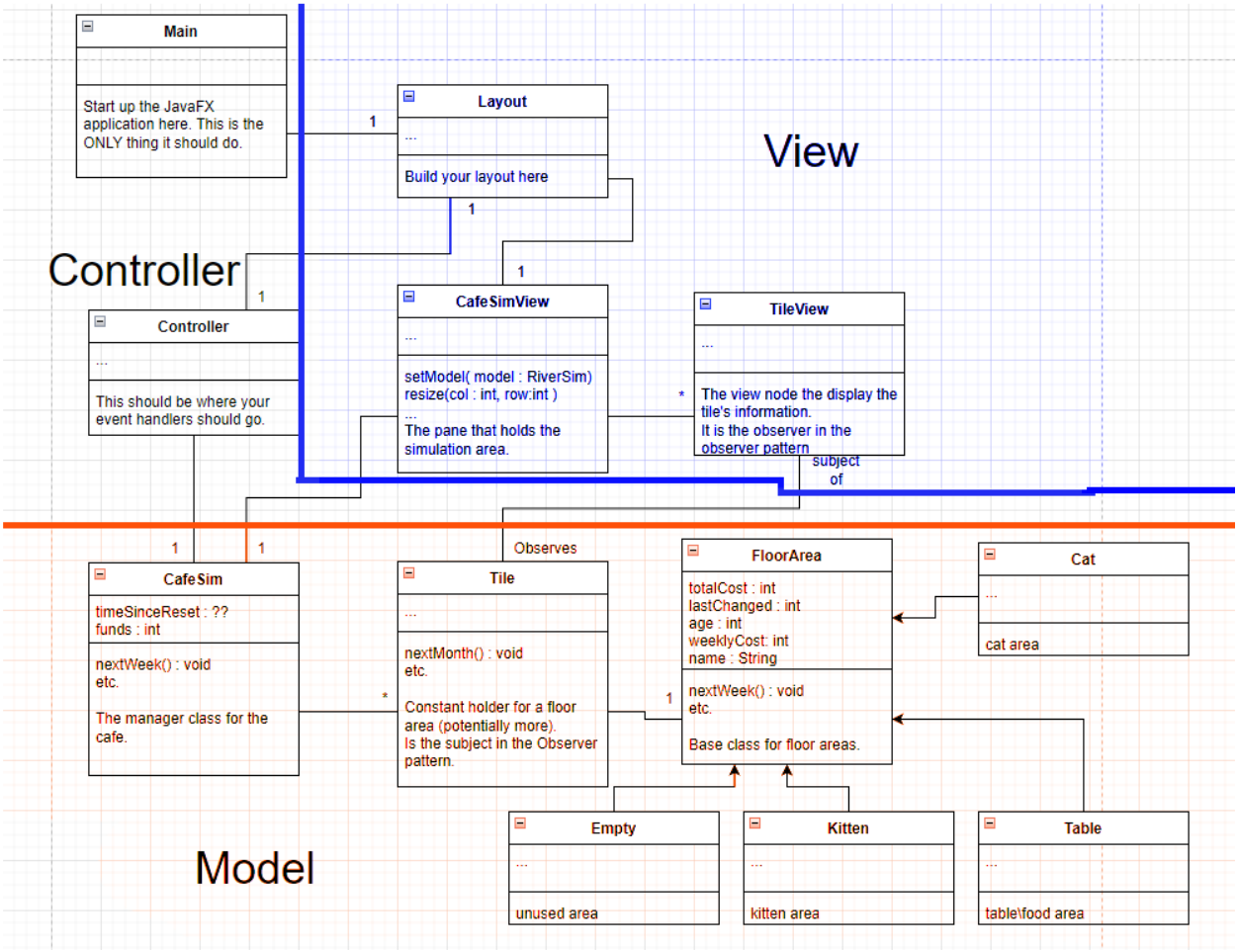
Figure 5

Tip: Assuming you use a GridPane layout for the display area, you can clear its children, and row and column constraints with

```
getChildren().clear();
getRowConstraints().clear();
getColumnConstraints().clear();
```

Required OOP

You MUST use the following OOP diagram for the core portion of your project. You MUST use the Java built interfaces for the observer pattern. Only the PropertyChangeListener/PropteryChangeSupport forms are acceptable implementations of the pattern for this project.



Depending on how you initially build, your Main class may have a connection to Store instead of Layout, and likely Controller and View. This is acceptable, and *the more common case*. The above is the minimum to get it built. CafeView also may be combined with Layout, but it makes several later extensions easier if it is separated.

For compactness, I didn't list the Parent classes for several classes in the diagram. Here is what wasn't listed:

- StoreView needs to be derived from some sort of Pane (I used GridPane).
- TileView needs to be derived from some node (I used Button). It also needs to implement the observer.
- Tile has a member variable of PropertyChangeSupport.
- While optional, you may want to consider a Pet parent class.

Tile mostly just forwards the calls onto Tile area. However, without it, changing the tile area would also destroy the observable! ***Do not remove the Tile class***. Essentially of the Tile class using the strategy pattern, and the floor area is the strategy (at this size it is more commonly called dependency injection). There are still quite a few functions not included in the above that you will need to figure out on your own.

TIP: Think of this like a doubly linked list. You can make your node (model/view/controller) first, then set your links. The lack of link between Main and the model only means a member variable is not needed. It does not forbid using or making a model. In fact, making the model first is good practice.

Extensions

You may choose from the following list for extensions for the project. You may also ask for a different extension, and I'll decline or accept with a set point value. The point values are listed. You MUST list the extensions up to the first extension that reaches 100% (no more!) in the main file header and **how to test/find** for credit.

Record your extension with their line number, name, and number in the checklist.

1. Observer
    a. 15pt: Implement the observer pattern such that the store information tile observes ALL of the tiles, and the Store updates. There should be no other update-type calls (resizing redraws everything automatically, so that never needs the observer pattern). You must mark the same three places as the original observer pattern with ***GRADING: 1.A OBSERVER-GRID, GRADING: 1.A SUBJECT-GRID, GRADING: 1.A TRIGGER-GRID***.
    b. 15pt if not done with above, and 10pt if it is: Implement the observer pattern such that the tile area information tile automatically updates with tile area data changes. At present, the tile info are only updates on tile selection. You must mark the same three places as the original observer pattern with ***GRADING: 1.B OBSERVER-TILE AREA, GRADING: 1.B SUBJECT-TILE AREA, GRADING: 1.B TRIGGER-TILE AREA***
    c. Another 5pt to 1.b changing the area on a tile should break the automatic updates as the area is no longer there. Done naively, the area view will simply watch the last tile regardless of what occurs at the tile. This extension fixes that.
2. Better Visuals
    a. 5pt: Somehow mark what rectangle is being shown in the tile area information tile (e.g. (x,y), outline).
    b. 10pt: Add `Images` instead of using text only (you must make your own). The non-name text must remain.
    c. 10pt: Add an `Image` to the tile area information tile.
    d. 10pt: Add `Colors` instead of using text only to indicate a tile area. You selection must be noted somewhere. The base text must remain.
    e. 5pt: Edit the appearance of the buttons/tiles with styles.
    f. 5pt: Edit the appearance of the tile area/none tiles to reflect what is currently active (pairs with extension 5 for a total of 10 points).
    g. 5pt: Disable the tile selection when "add" is not checked.
3. Better Interactions

a. 10pt: Add hot-keys (mnemonic, accelerator, or scene key listener) to 3+ buttons. You must list what they are for us to test them.
b. 15pt: Add a menu bar to do the same thing as 3+ buttons (Building a menu bar works very similarly to a Scene tree, but has a different root parent).

4. Improve the simulator.
    a. 5pt: Add 2+ more tile area types, each with their own parameters. Add this new tile area under the current tile area. You must list what their needs are in the extension comments. Yhis can include element like “cat areas must be more than a tile away from tables.”
    b. 15pt for basic, but more with fancier proposals: Implement improved simulation using the visitor pattern. The simplest is improving the tile production immediately next to the store using the visitor pattern. You MUST give me a proposal for points on this to ensure it doesn’t break other requirements. You must mark the core of the visitor pattern and the start of the visitor update with GRADING: EXTENSION VISITOR_#, and GRADING: VISITOR_# where # is the visitor count if more than one is implemented.

568 Students Only

You **must do** extensions 1.b and 1.c.

Your scores are rescaled to 240 points, so you must have an additional 40 points in extensions to reach 100%.

Submission Instructions

1. Check the coding conventions before submission.
2. Make sure your project name and package name are correct.
3. **Complete the program checklist, with your extensions if reached, and paste it into your main() file.**
4. Delete the out and target folders (if applicable), then zip your **ENTIRE PROJECT** into ***ONE*** zip folder.

If you are on Linux, make sure you see “hidden folders” and you grab the “.idea” folder. That folder is what actually lists the files included in the project!

5. Submit to D2L. The dropbox will be under the associated topic's content page.
6. *Check* that your submission uploaded properly. No receipt, no submission.

You may upload as many times as you please, but only the last submission will be graded and stored.

If you have any trouble with D2L, please email me (with your submission if necessary).

Rubric: Grading Tiers

These tiers start with the simplest tasks, and go to the most involved. Please refer to the rubric for the tiers. You must "reasonably" complete the lower tiers before gaining points for the later tiers. By "reasonably," I can reasonably assume you honestly thought it was complete.

You may not get points for the extensions until that tier is completed. I need to know your base skills first.

Item	**Points**
Followed the class OOP diagram (-50% for each moderate unapproved change)	14
Observer pattern (ignores tiers, -5 for minor error)	20
1. Tier: Views and tile area	54
a. All objects (ignoring the sim area) (-1 for each missing)	10
b. Have a starting number of tiles in sim area	4
c. Able to add/remove a tile area properly (-33% for each error)	15
d. Info bar listed correctly with all the required elements (-25% for each error)	8
f. Tile Text correct in tile area (-25% per error)	8
g. Radio buttons update properly	3
h. Selecting a rectangle with "view" updates the tile area info (-50% per error)	6
2a Tier: Advanced functionality	34
a. Next week button has some noticeable effect*	4
b. Tile areas updated properly on "next" (-33% per error)*	18
c. Sim info bar updated properly (-25% per error)	8
d. Selecting a tile after an update shows the new information	4
2b: Layout	48
a. Location of all items in correct spot (-20% per error)	12
b. Layout still correct on window resize (-20% for minor error)	12
c. Resize grid at minimum resets the grid and info (-50% if minor error)	12
d. Everything still working that is listed above with resize (-50% if minor error)	12
Final Tier: Extensions Extension 1: <number> <points> <name>: <how to test/find if applicable> Extension 2: <number> <points> <name>: <how to test/find if applicable> Etc.	30
Total	200

* Has required grading tags