Lab 5: Graphical User Interface (GUI) & Basic Listener

Instruction

1. Click the provided link on CourseVille to create your own repository.
2. Open Eclipse and then "File > new > Java Project" and set project name in this format **2110215_Lab5_2019_2_{ID}_{FIRSTNAME}**
    - Example: **2110215_Lab5_2019_2_6137492021_Jirapash**.
3. Initialize git in your project directory
    - **Add .gitignore.**
    - Commit and push initial codes to your GitHub repository.
4. Implement all the classes and methods following the details given in the problem statement file which you can download from CourseVille.
    - **The provided files contain two folders:** `src` **and** `res`**. make sure to add both into your project. (And use both of them as Source Folder)**
    - You should create commits with meaningful messages when you finish each part of your program.
    - You don't need to wait until you finish all features to create a commit.
5. Test your codes with the provided JUnit test cases, they are inside package **test.grader**
    - If you want to create your own test cases, please put them inside package **test.student**
    - Aside from passing all test cases, your program must be able to run properly without any runtime errors.
    - There will be additional test cases to test your code after you submit the final version, **make sure you follow the specifications in this document**.
6. After finishing the program, create a UML diagram using **ObjectAid** and put the result image (**UML.png**) at the root of your project folder.
7. Export your project into a jar file called **Lab5_2019_2_{ID}** and place it at the root directory of your project.
    - Example: **Lab5_2019_2_6137492021.jar**
8. Push all other commits to your GitHub repository

1. Problem Statement: Harvest Simulator

With the economic crisis, CPShop had to be closed. With a little of money left, the manager of the shop chooses to return to his hometown in a countryside and back to the good old way of life of his parents -- agriculture. He knows you who is a programmer and he want to run a simulation on income gain from harvest crops he planted. Also, he takes very good care of his crops so they never wither.

Harvest Simulator

This is a simulation program for calculating income from each harvest. The manager has some programming knowledge so the program is partially done. He asks you to complete it for him. You need to understand JAVAFX and EVENT-HANDLER in order to complete this assignment.

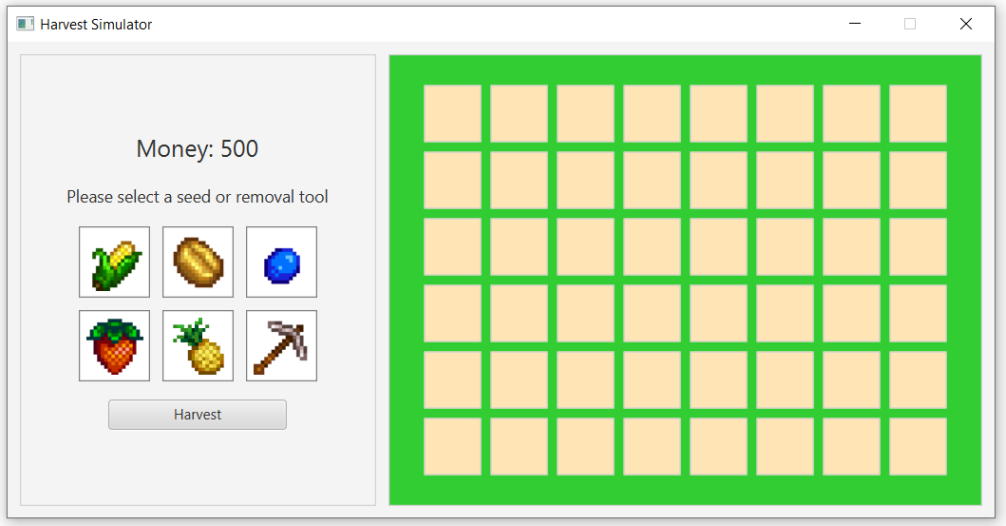


Figure 1: The initial GUI of the application

The GUI can be divided into TWO parts Control part on the left and Field part on the right.

Control part

This is a part to select an item, be it a seed or the removal tool, show current money and button for simulating each harvest. For a seed, it is used to plant in the field. For the removal tool, it is used to remove a plant in the field.

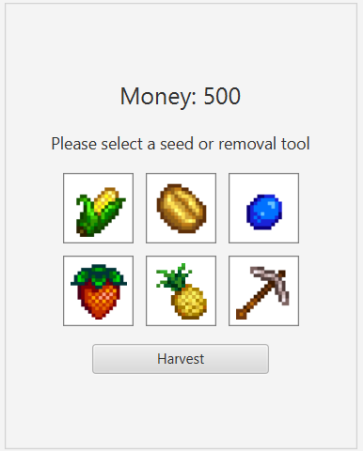


Figure 2: The GUI of the Control part

The selected item will be highlighted if clicked when you have enough money for it. If you choose another item that you can afford, the newly clicked item will be highlighted instead. NO MORE THAN ONE ITEM WILL BE HIGHLIGHTED AT THE SAME TIME.

After planting a crop, if money reduced from buying a seed is less than selected item price, the item will be unselected and unhighlighted. If you hover mouse over an item button, the tooltip of that item will be shown for item information

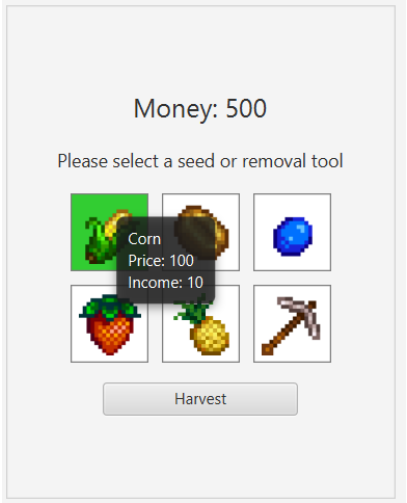


Figure 3: The GUI of the Control part when selected an item and hovering mouse over an item

Field part

This part represents the field with space to plant crops. When clicking a field cell while selecting an item, an action will be done according to item selected.

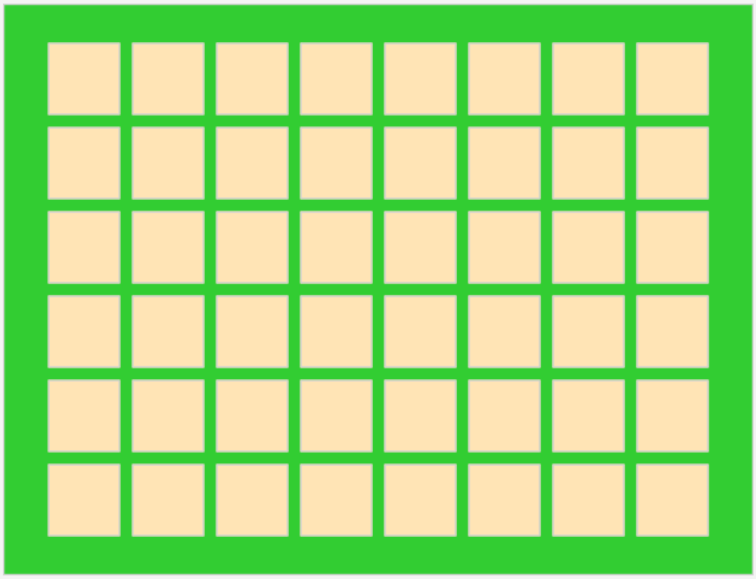


Figure 4: The GUI of the Field part

JavaFX Install and Project Setup

In this assignment, you need to use JavaFX version 11 or more in order to run the project and do this assignment.

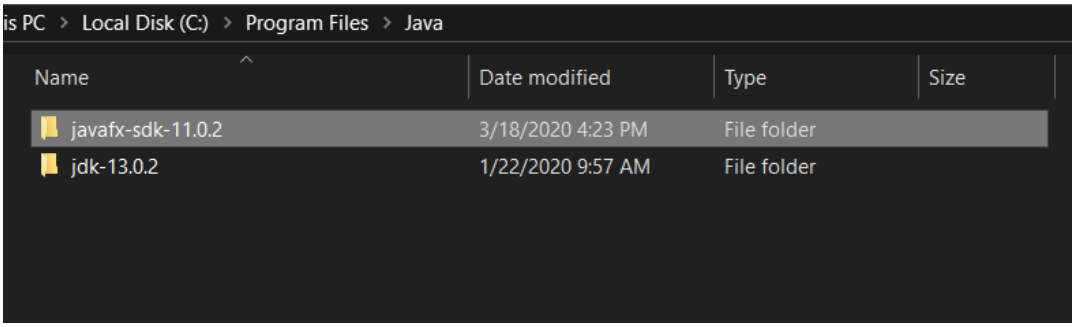
If you don't have any JavaFX, you can get the latest version from

https://gluonhq.com/products/javafx/

Please download the latest release SDK of your OS version.

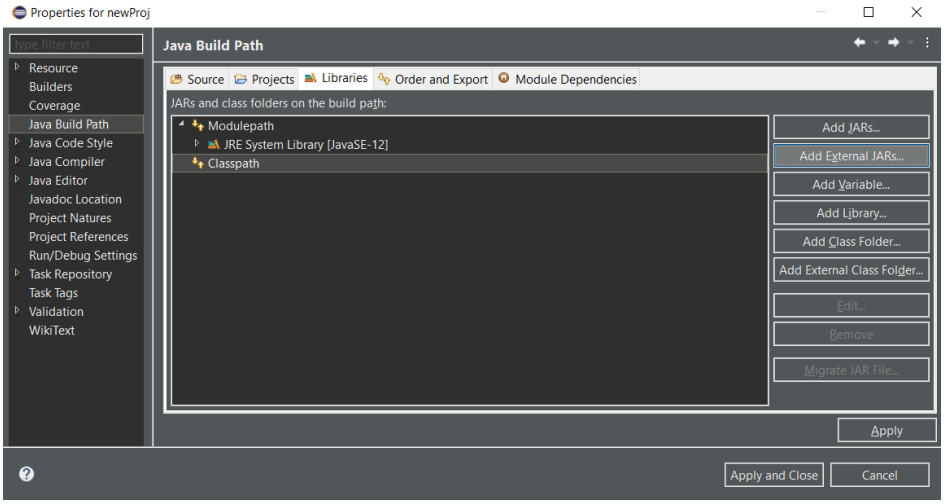
Once the download complete, extract the file then move the extracted folder to somewhere that you think it is easily accessible.

The recommended location is "C:\Program Files\Java"

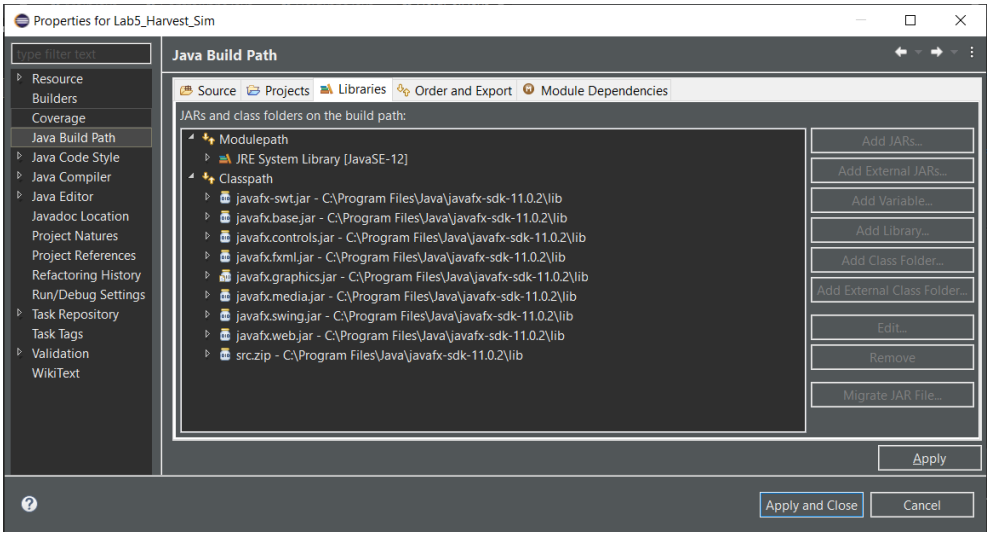


To add JavaFX to the eclipse project, follow these steps.

1. Right-Click your project > Build Path… > Configure Build Path
2. In the Libraries Tab, under Classpath, click **Add External JAR…**



3. Navigate to the previously extracted JavaFX folder, go to the folder **lib**, and select every file in there and click Open.



Your Libraries should look like this.

Then, you need to modify **Run Configurations**.

This can be done by right-clicking Main.java > Run As > Run Configurations…

Under Arguments tab, in the VM arguments section, place this command in

For Windows

--module-path "Path to your JavaFX folder\lib" --add-modules javafx.controls,javafx.graphics,javafx.media,javafx.fxml

For Mac

--module-path "Path to your JavaFX folder\lib" --add-modules javafx.controls,javafx.fxml

Afterwards, click Run and you will be able to run the program from now on.

2. Implementation Details:

Figure 6: Detailed GUI of the manager

There are six classes to be completed or be created if not exist: **Main**, **ControlPane, ItemPane**, **ItemButton, FieldPane and FieldCell**. There are already two classes provided: **Item** and **SimulationManager**. Your program can have more private methods/fields than specified in this instruction.

Tip: Use Ctrl+Space or auto-complete/suggestion feature will help you a lot. You can configure the key by going to Window -> Preference -> General -> Keys and search for content assist.

** Noted that Access Modifier Notations can be listed below*

+ (public)

 # (protected)

- (private)

***static** will be underlined.*

***abstract** will be italic.*

2.1 Class Item

This class represents items or tools in ItemButtons and crops when referred in FieldCells. This class is already provided. There is a destroy tool item for removing crops in FieldCells.

2.1.1 Field

- String itemName	The name of the item.
- String url	The url of the item's image.
- int price	Price of the item.
- int income	Income of the crop when the item is used to plant the crop in `FieldCells`.

2.1.2 Constructor

+ Item (String itemName)	Initializes each field according to the `itemName` parameter.

2.1.3 Method

+ boolean isDestroyTool ( )	Return true if this item is the destroy tool.
+ String getPriceText ( )	Return price String to be used in tooltip
+ String getIncomeText ( )	Return income String to be used in tooltip
+ void getter for each field	

2.2 Class ItemButton extends Button

Figure 8: Left: Unhighlighted ItemButton, Right: Highlight ItemButton

2.2.1 Field

- Item item	Item in this button.

2.2.2 Constructor

+ ItemButton (String itemName)	/* Fill Code */ Initializes the ItemButton with the following specification: - Set the inset padding to 5. - create a new item with itemName and set it to the item field. - Set the `Graphic` (icon) with the `ImageView` which has to create an Image that use String `url` of the item and set `ImageView` fit width and height to 48. - Background and Border are already provided. - Set tooltip by calling provided setTooltip() method.

2.2.3 Method

+ void highlight ( )	/* Fill Code */ Set its **Background** to be **filled** with **LIMEGREEN** color
+ void unhighlight ( )	/* Fill Code */ Set its **Background** to be **filled** with **WHITE** color
+ Item getItem ( )	/* Fill Code */ A getter for item
- void setTooltip( )	Set the tooltip of the button to show information of item in this button when hovering mouse over the button.

* ItemButton's onAction ( ) will be implemented by another class for controlling highlight of many ItemButtons.

2.3 Class ItemPane extends GridPane

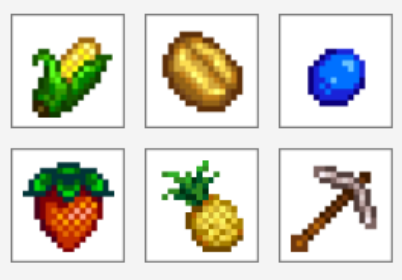


Figure 8: Picture of ItemPane

2.3.1 Field

- ObservableList<ItemButton> itemButtonList	Collects all ItemButton in this pane.

2.3.2 Constructor

+ ItemPane ( )	/* Fill Code */ Initializes the ItemPane. - Sets the alignment of the pane to the **CENTER** - Sets both **vertical gap** and **horizontal gap** to 10 - Instantiates **six** ItemButtons: *"Corn"*, *"Coffee"*, *"Blueberry"*, *"Strawberry"*, *"Pineapple"*, and *"DestroyTool"* - Implement and set setOnAction of each ItemButton to do `setSelectedButton()` if money in SimulationManager is equals to or more than price of Item in ItemButton. **HINT**: See how to write EventHandling at page 56 in GUI lecture slide - Add ItemButtons to each cell. There are two rows with three columns each in GridPane.

2.3.3 Method

+ void setSelectedButton (ItemButton selectedItemButton)	/* Fill Code */ This method sets `selectedItemButton` of SimulationManager to match the parameter, `resetButtonsBackGroundColor()` and then

	`highlight()` the selected one which is the parameter.
+ void resetButtonsBackGroundColor()	/* Fill Code */ This method un`highlight()` each button in itemButtonList

*You can find GridPane's documentation here:
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html
And how to use GridPane here:
http://tutorials.jenkov.com/javafx/gridpane.html

2.4 Class ControlPane extends VBox

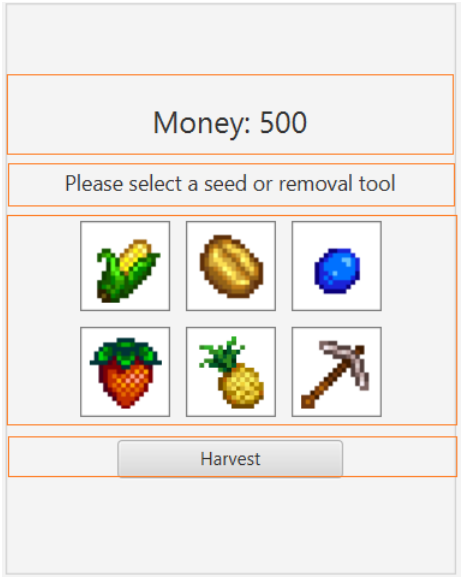


Figure 9: The outline showing how the ControlPane is constructed

2.4.1 Field

- ItemPane itemPane	ItemPane in this ControlPane
- Label moneyLabel	Label to show current money.
- Button harvestButton	Button that triggers each harvest and increase money.

2.4.2 Constructor

+ ControlPane ( )	/* Fill Code */ Initializes the ControlPane. - Set the alignment of the pane to the **CENTER** - Set the preferred width to 300. - Set the spacing to 15. - Set fillWidth to true. - Border is already provided. - Instantiates each node and set to fields: • `moneyLabel` is a label with font size 20 and set the text by setMoneyText(). • A Label with text "Please select a seed or removal tool" and font size 14 • `itemPane` is an ItemPane. • `harvestButton` is a Button with the text "Harvest" and has a width of 150 and setOnAction to

	harvestButtonHandler() - Adds each node to the pane's children in the correct order.

2.4.3 Method

+ void setMoneyLabelText()	Set `moneyLabel` textProperty to "Money: " + SimulationManager.getMoney()
+ Getter method for each field	/* Fill Code */

2.5 Class FieldCell extends Pane

This class represent a cell in FieldPane. It can contain a crop and is clickable to plant a crop while selected an ItemButton. It has tooltip to show the crop information while hovering mouse over it when there is crop in the cell.

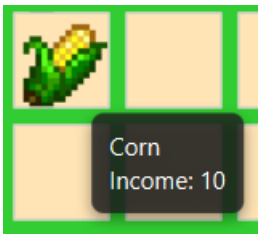


Figure 11: A picture of a FieldCell with a crop while mouse is hovering over it.

2.6.1 Field

- Item crop	Crop in this FieldCell.
- Tooltip tooltip	Tooltip of the FieldCell to show information of the crop in this FieldCell when hovering mouse over the FieldCell.

2.6.2 Constructor

+ FieldCell ( )	/* Fill Code */ Initializes the FieldCell: - Set the preferred width and height to 48. - Set both **min width** and **min height** to 48. - Set the inset padding of 8. - Border is already provided. - Set the **Background** by calling setBackgroundSoilColor( ). - Set tooltip by calling provided setTooltip() method. - Complete the handle method in EventHandler given in the code.

2.6.3 Method

+ void onClickHandler( )	/*Partially Provided*/ This method is the handler when the cell is clicked. If there is a button selected in SimulationManager, do the following. If selectedItem is DestroyTool, set this FieldCell crop to null and call setBackgroundSoilColor( ). If it's not the destroy tool and this Fieldcell doesn't have a crop: - Set crop to selectedItem. - Reduce the money in SimulationManager. - Create new **Image** with selectedItem url and setBackgroundSoilColor(Image) with it - Set tooltip text with selectedItem name and selectedItem income text.
- void setBackgroundSoilColor( )	- Set the **Background** with **BackGroundFill MOCCASIN** Color
- void setBackgroundSoilColor(Image image)	-- Set the **Background** with **BackGroundFill MOCCASIN** Color with image parameter background
- void setUpTooltip( )	Set up the `tooltip` of the FieldCell to show information of the crop in this FieldCell when hovering mouse over the FieldCell.
+ Item getCrop()	/* Fill Code */ Getter for `crop` field.

*You can find Pane's documentation here:
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/Pane.html
And a tutorial here: https://www.geeksforgeeks.org/javafx-pane-class
Pane class is a base class for layout panes like GridPane, Hbox, StackPane and TilePane so it's very plain.

2.6 Class FieldPane extends GridPane

This class represent the field for planting crops and partially provided. It contains FieldCells which is slot for planting crops.

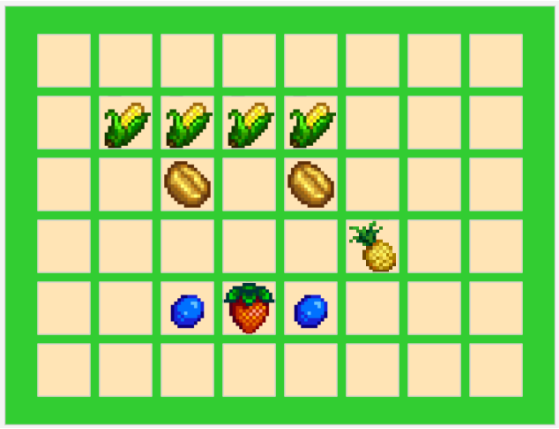


Figure 10: Picture of FieldPane with some crops in FieldCells.

2.5.1 Field

- ObservableList<FieldCell> fieldCells	JavaFX's observable list, which can be used for storing the fieldCells to display on field.

2.5.2 Constructor

+ FieldPane()	/* Fill Code */ Initializes the FieldPane. - Set the preferred width to 500. - Set the alignment of the pane to the **CENTER** - Set both **vertical gap** and **horizontal gap** to 8. - Set the inset padding of 8. - Border is already provided. - Set the **Background** with **BackGroundFill LIMEGREEN** Color - Create and add 6*8 FieldCells to `fieldCells` field and add to this child in 6 rows * 8 columns.

2.5.3 Method

+ int calculateIncome ( )	/* Fill Code */ Calculate total income from each FieldCell in `fieldCells` if that cell contains a crop.

* ObservableList is a list that enables listeners to track changes when they occur.
You might need it in your term project so you can read more about it here:
https://docs.oracle.com/javafx/2/collections/jfxpub-collections.htm

2.7 Class SimulationManager

This class contain reference and control over some part of the simulation. This class is already provided.

2.7.1 Field

- ItemButton selectedItemButton	Selected ItemButton.
- ControlPane controlPane	An instance of ControlPane
- FieldPane fieldPane	An instance of FieldPane
- int money	The money at the start of the program is set 500.

2.7.3 Method

+ void harvestHandler ( )	This is handler function for the harvest button. This method increase `money` by calling ControlPane's calculateIncome().
+ void reduceMoneyBuySeed ( )	This method is called when a crop is planted in a FieldCell to reduce the `money` by that crop price, update moneyLabel's text and unselect ItemButton if `money` left is less than selecting Item price.
+ setters and getter for each field as needed	

2.8 Class Main extends Application

2.9.1 Method

+ void start (Stage primaryStage)	/* Fill Code */ The main entry point of the JavaFX application. This method should: - Creates a root container using HBox with spacing between containing pane of 10 and an inset padding of 10. - Set its preferred height to 400px. - Initializes ControlPane and FieldPane and add them to the root container. - Set SimulationManager's ControlPane and FieldPane. - Create a scene with root container. - Set primaryStage with appropriate scene and title *"Harvest Simulator"*. - Set the stage window size to match the scene. - The stage window must not be resizable. - Show the primaryStage.
+ void main (String [] args)	An entry point of the application.