Lab 5: GUI

1. Instruction

1. Unzip the file using password “heavendoor”. Code files are not empty. If you get empty files it means you extract the file incorrectly.
2. Click the provided link on myCourseVille to create your own repository.
3. Open Eclipse and then “File > new > Java Project” and set project name in this format **2110215_Lab5_2022_2_{ID}_{FIRSTNAME}**
    - Example: **2110215_Lab5_2022_2_6531234521_Kevin**.
4. Don’t forget to set up JavaFx.
5. Initialize git in your project directory
    - **Add .gitignore.**
    - Commit and push initial codes to your GitHub repository.
6. Implement all the classes and methods following the details given in the problem statement file which you can download from myCourseVille.
    - You should create commits with meaningful messages when you finish each part of your program.
    - Don’t wait until you finish all features to create a commit.
7. After finishing the program, create a UML diagram and put the result image (**UML.png**) at the root of your project folder.
8. Export your project into a **runnable** jar file called **Lab5_2022_2_{ID}** and place it at the root directory of your project.
    - Example: **Lab5_2022_2_6531234521.jar**
9. Push all other commits to your GitHub repository.

2. Problem Statement: Simple(?) To Do List

Now, you have been learning a lot about OOP with Java, and working on JavaFX, which is the one of flexible tool to create cool desktop applications. In this problem, you are assigned to create “Simple(?) To Do List” to keep track of all the work you must do in this crazy year. Our to-do lists are organized into “pages”, which contains a list of to-do items. You can add more to-do items using the text field input at the bottom. You can select the page of to-do list (or create one) with the buttons on the left of the window. You can delete to-do items with its delete button.

Please see the example of the complete application below. Your program might not look the exact same way (especially if you’re not using macOS). You will not be deducted points as long as all instructions are followed. Lastly, please please please PLEASE read the entire document before you start to write. There are methods that are already provided (or noted) so you don’t get lost.

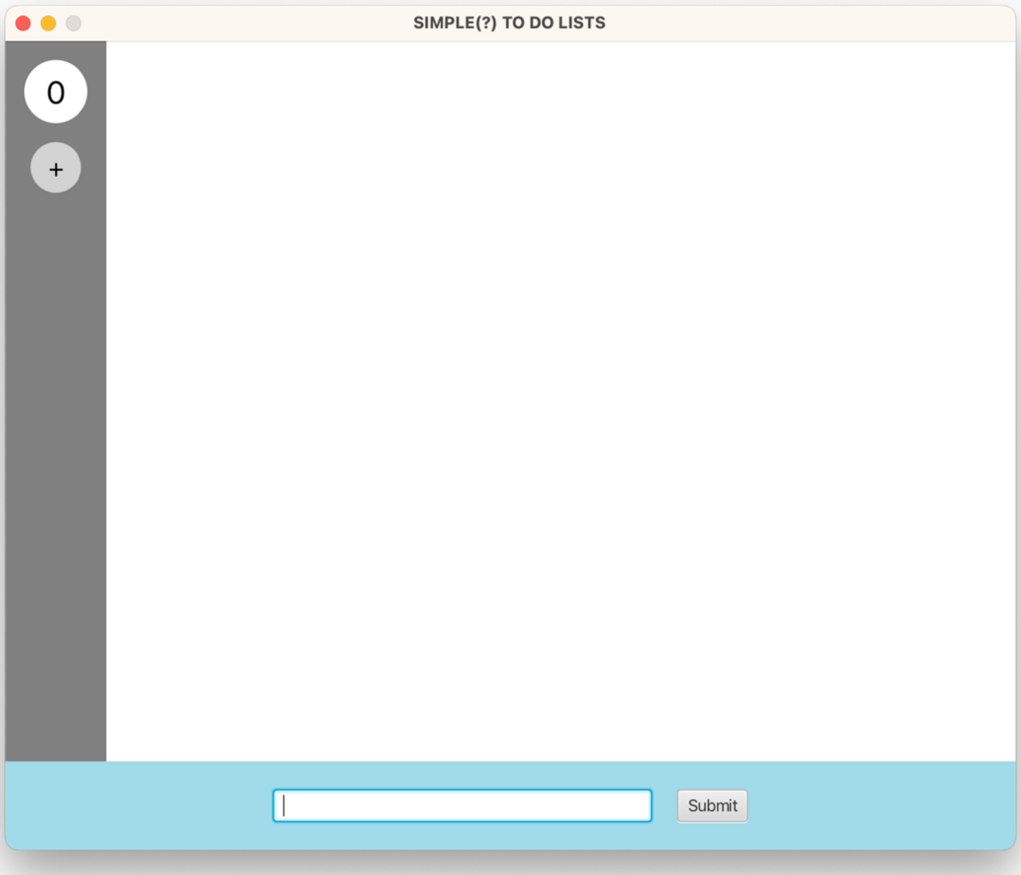


Figure 1. Sample screenshot of when the application starts up. ***Note that the page number starts at 0.***

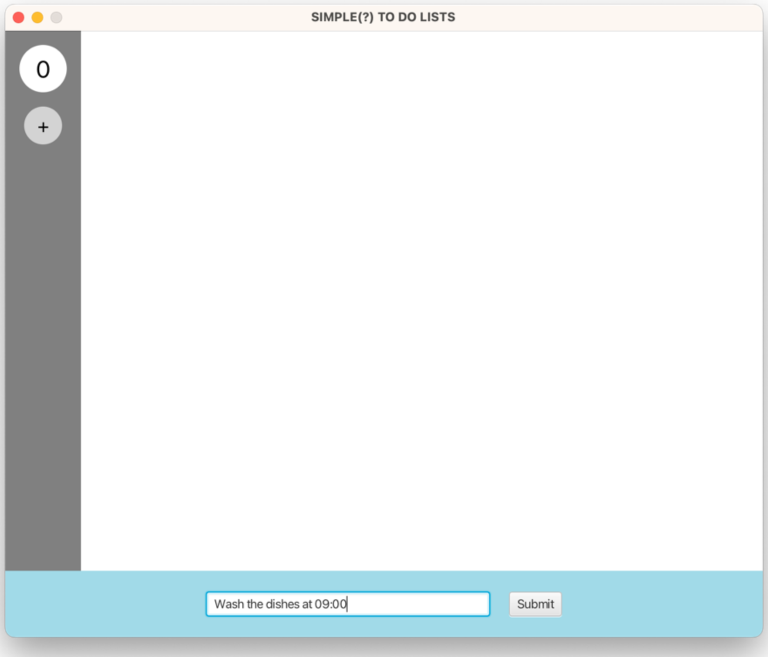


Figure 2. Typing a new to do item before submission

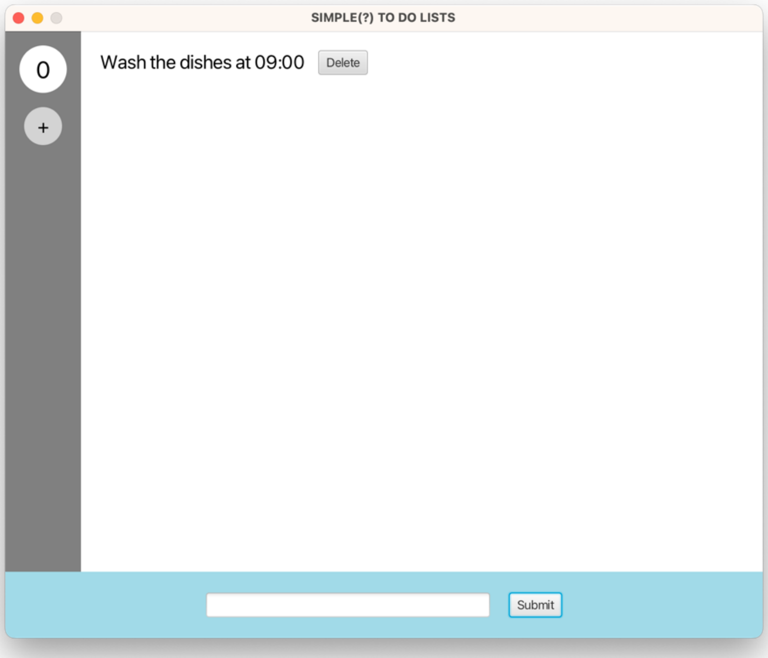
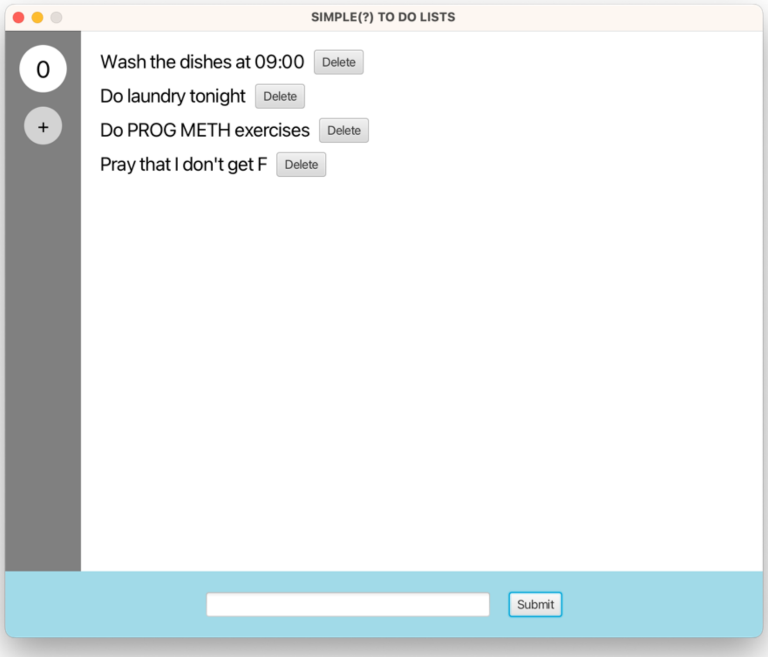
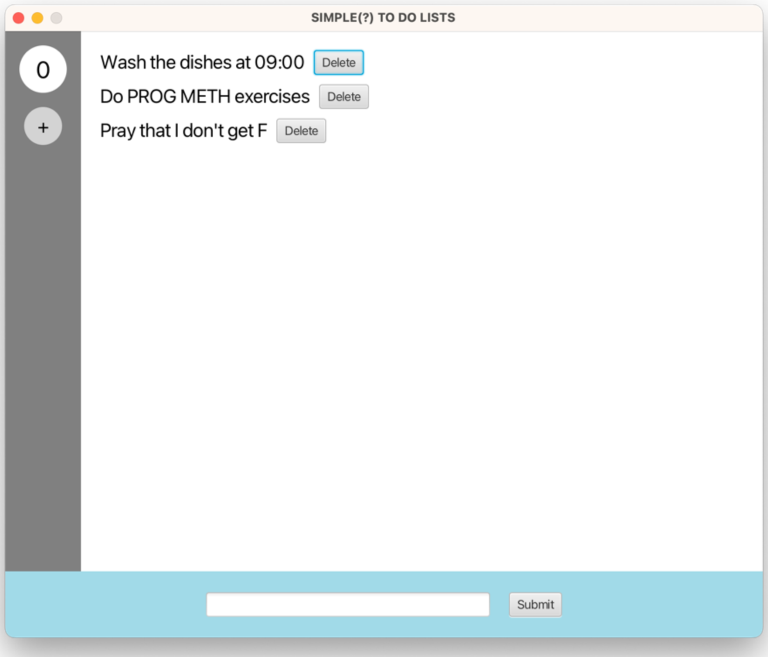


Figure 3. After submitting a new to do item.



Wash the dishes at 09:00
Do laundry tonight
Do PROG METH exercises
Pray that I don't get F

Figure 4. Having multiple items in a list.



Wash the dishes at 09:00
Do PROG METH exercises
Pray that I don't get F

Figure 5. After clicking "Delete" after the "Do laundry" item.

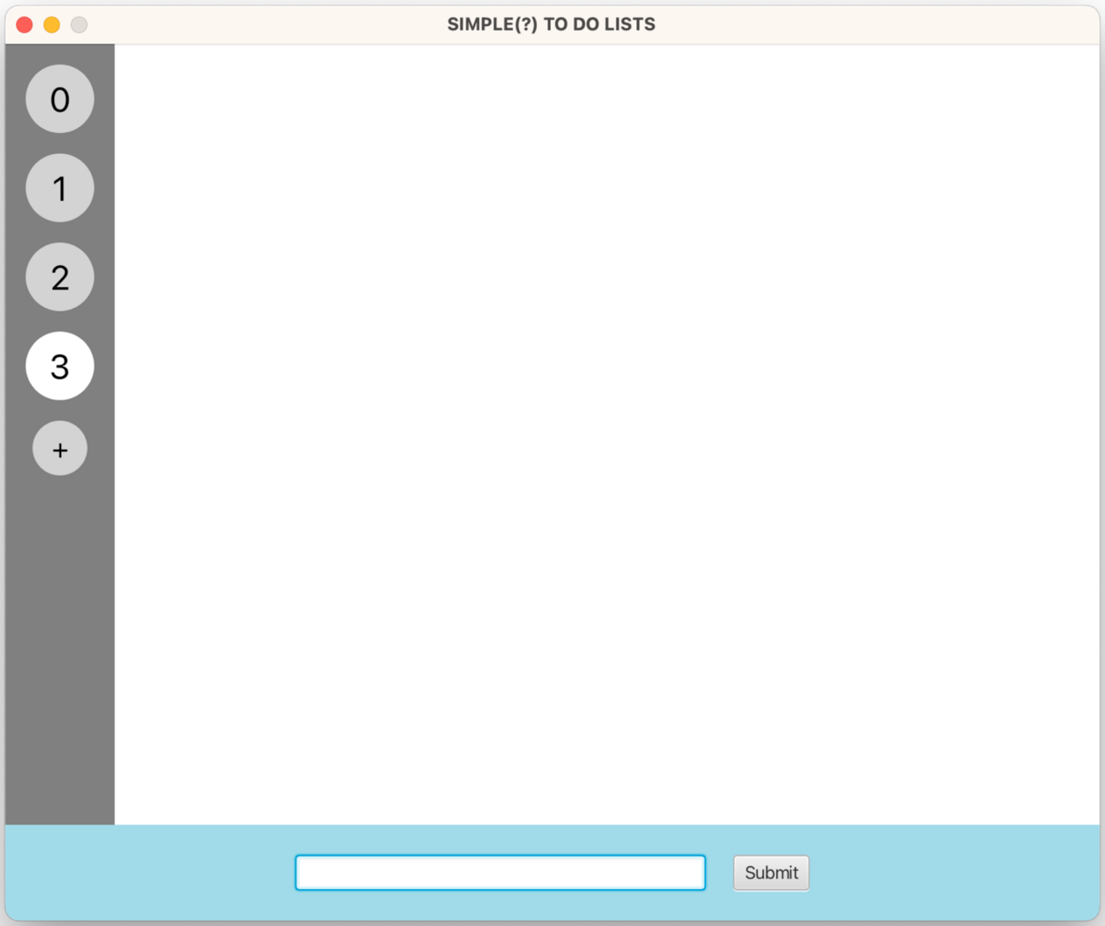


Figure 6. After creating a new page 3 times. The currently active page is highlighted as white, while other pages are light gray. ***Note that all pages' lists are kept separate and do not interfere with one another.***

3. Implementation Detail:

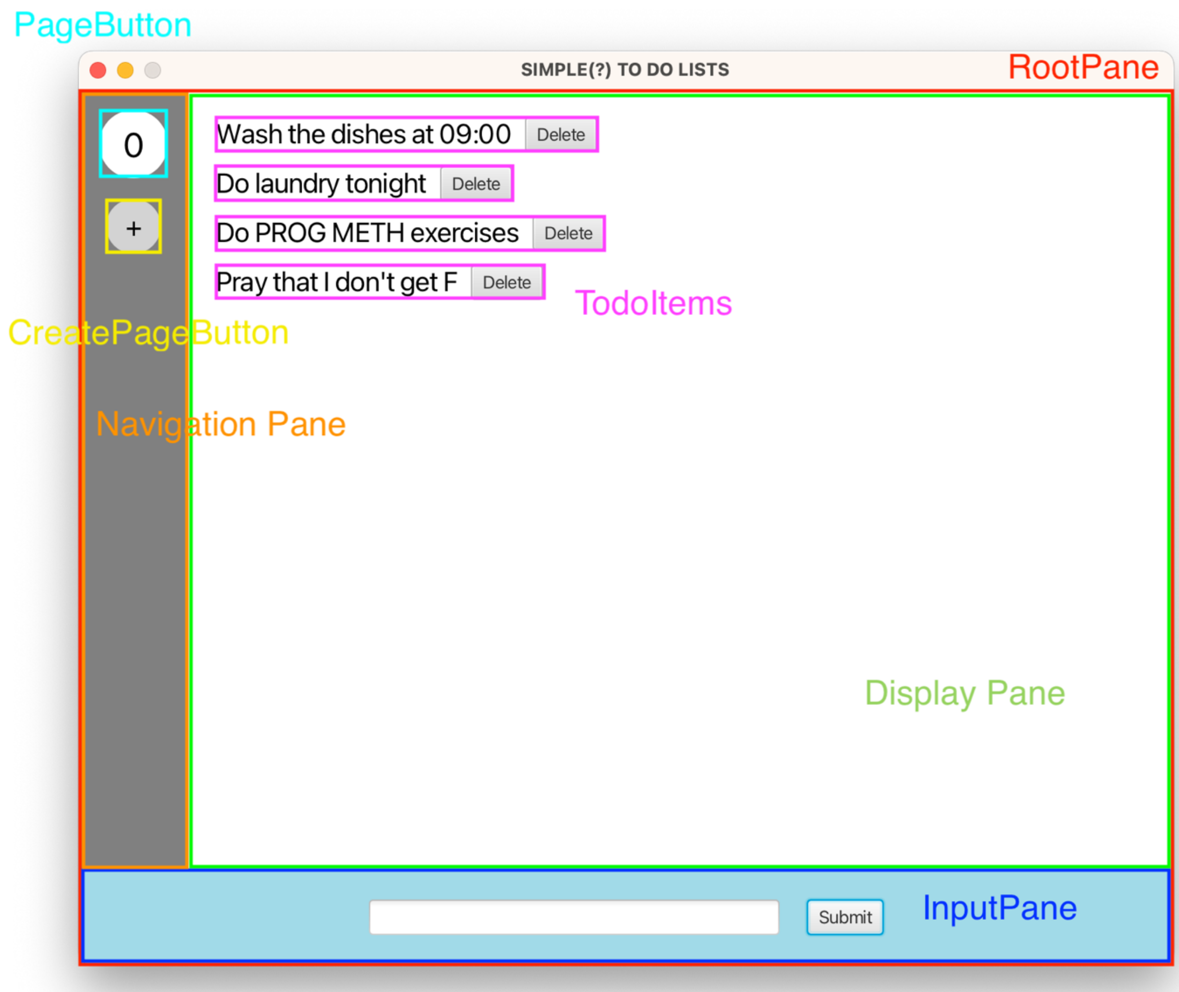


Figure 7: Detailed GUI of the application.

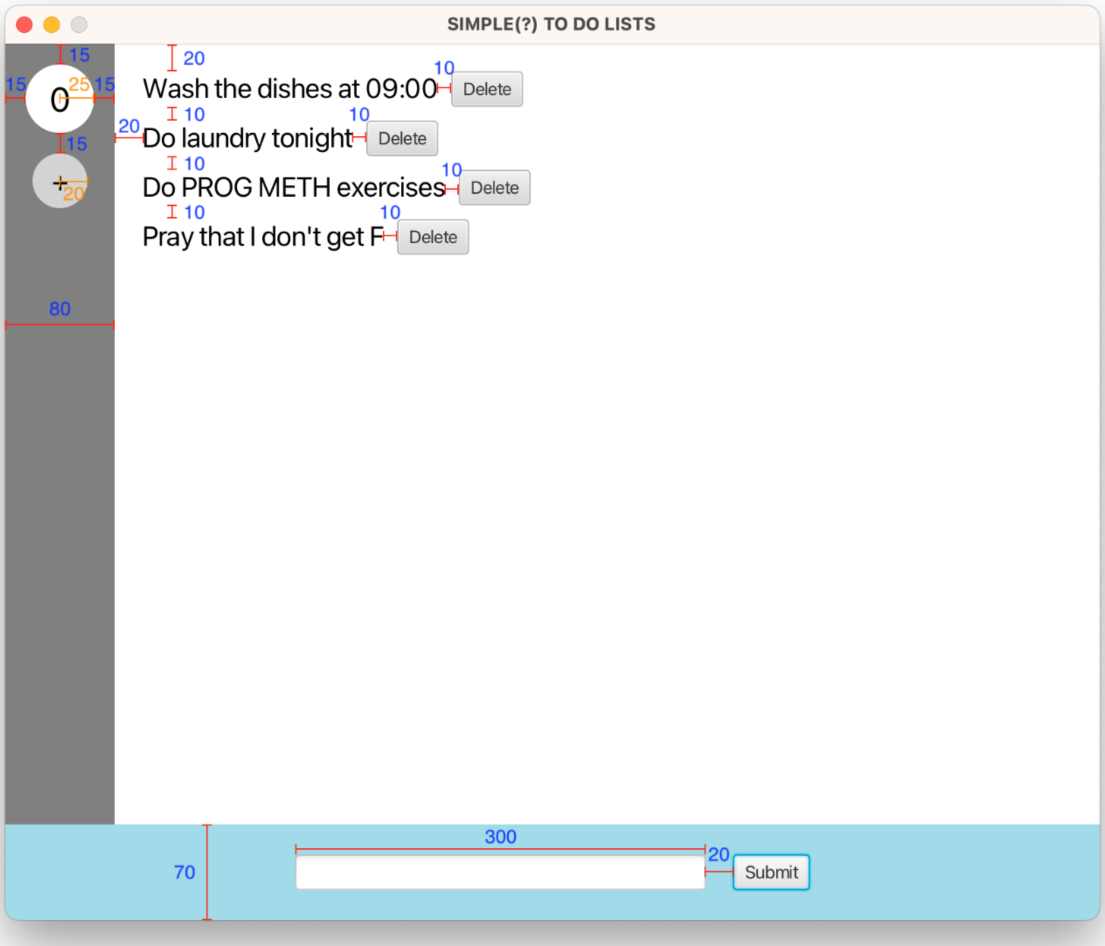


Figure 8: Detailed Spacing of the application.

The class package is summarized below. * In the following class descriptions, only details of IMPORTANT fields and methods are given. *

3.1 **Package pane** /* Partially PROVIDED */

3.1.1 public class **RootPane** /* Partially PROVIDED */

This class, which extends from **BorderPane**, is provided to group the NavigationPane, DisplayPane, and InputPane together. This class contains static methods for getting all three panes easily.

Field

Name	Description
- NavigationPane navigationPane	Represents the NavigationPane. Already provided.
- DisplayPane displayPane	Represents the DisplayPane. Already provided.
- InputPane inputPane	Represents the InputPane. Already provided.

Constructor

Name	Description
+ RootPane()	/* FILL CODE */ Initialize the RootPane instance with the following specifications: - create **NavigationPane**, **DisplayPane**, and **InputPane** instances - add first page with NavigationPane's **addPage()** - set NavigationPane to the **left border** - set DisplayPane to the **center** - set InputPane to the **bottom border** **Note:** Do not use this.getChildren().add() to add children to BorderPane. That will mess up the your entire UI. Please research on how to add children to BorderPane.

Method

Name	Description
+ NavigationPane getNavigationPane()	Static method for returning the navigationPane. Already provided.
+ DisplayPane getDisplayPane()	Static method for returning the displayPane. Already provided.

+ InputPane getInputPane()	Static method for returning the inputPane. Already provided.

3.1.2 public class **NavigationPane** /* Partially PROVIDED */

This class, which extends **VBox,** represents a Pane that contains the page buttons.

Field

Name	Description
- ArrayList<PageButton> pageButtons	An ArrayList of PageButtons. Already provided.
- int currentPage	The currently active page. Already provided.

Constructor

Name	Description
+ NavigationPane()	Initializes the NavigationPane. Already provided.

Method

Name	Description
+ void addPage()	/* FILL CODE */ Create a new page and its **PageButton** with the following specifications: - create a new **PageButton** with its text as one value more than the last page number. - add the new pageButton to pageButtons list. - add the new pageButton as a child of the NavigationPane, **JUST BEFORE** the last index in the list (as that would be the **CreatePageButton**). - add a new todo list in **DisplayPane**.

	- set currentPage to the newly created page. **Hint:** Use static method RootPane.getDisplayPane() to access the DisplayPane.
+ void getCurrentPage()	/* FILL CODE */ Returns currentPage.
+ void setCurrentPage(int pageNumber)	/* FILL CODE */ If pageNumber is in valid range (0 <= pageNumber < number of pages), set the currently active page using the following instructions: - set current pageButton's active status to **false.** - set currentPage to pageNumber. - set the new current pageButton's active status to **true.** - set **DisplayPane**'s active to-do list number to pageNumber.

3.1.3 public class **DisplayPane** /* Partially PROVIDED */

This class, which extends **VBox,** represents a Pane that displays the todo lists selected by the NavigationPane.

Field

Name	Description
- ArrayList<ArrayList<TodoItem>> todoLists	The list of todo lists, containing list data in different pages. Already provided.

Constructor

Name	Description
+ DisplayPane()	/* FILL CODE */ Initialize the DisplayPane instance with the following specifications: - set background color to **Color.WHITE**. - set padding to **20**. - set spacing to **10**.

Method

Name	Description
+ void addTodoList()	/* FILL CODE */ Creates an empty list of TodoItem and add it to **todoLists**.
+ void setActiveTodoList(int index)	/* FILL CODE */ If index is in valid range (0 <= index < length of **todoLists**), clears the **DisplayPane** and adds all **TodoItem** of the **todoList** at the given index. Hint: ArrayList has method clear()
+ void addTodoItem(TodoItem todoItem)	Adds a new todo item to the active todo list. Also updates the DisplayPane to show the new todo item. Already provided.
+ void removeTodoItem(TodoItem todoItem)	Removes a todo item from the active todo list. Also updates the DisplayPane to remove the todo item. Already provided.

3.1.4 public class **InputPane** /* Partially PROVIDED */

This class, which extends **HBox,** represents a Pane that displays a text input and a button. When the button is clicked, it will create a new **TodoItem** and add it to the current page.

Constructor

Name	Description
+ InputPane()	/* FILL CODE */ Initialize the InputPane instance with the following specifications: - set preferred height to **70**. - set background color to **Color.LIGHTBLUE.** - set spacing to **20.** - set alignment to **CENTER**. - initialize a new TextField with preferred width set to **300**. - initialize a new Button with text set to **"Submit"**. - set button's action to do the following: 1. Check if textField's value is not empty. **If it's not empty**, do 2-4. 2. create a new **TodoItem** with textField's value. 3. add the new **TodoItem** to currentPage's todoList. 4. clear textField's value. - add the textField and button as this object's children.

3.2 **Package component** /* You must implement this ENTIRE package from scratch */

3.2.1 public class **PageButton** /* You must implement this class from scratch */

This class, which extends from **StackPane**, represents the page button in the **NavigationPane** that selects its page when clicked.

Field

Name	Description

- Circle circle	The Circle object in this StackPane. See the constructor for more info.

Constructor

Name	Description
+ PageButton(int pageNumber)	/* FILL CODE */ Initialize the PageButton instance with the following specifications: - initialize a new **Circle** with radius set to **25** and fill color set to **Color.LIGHTGRAY**. - initialize a new Text with text set to **pageNumber** and font size set to **25**. - set circle field to the newly created **Circle**. - set cursor to **Cursor.HAND**. - when this object is clicked, **set NavigationPane's currentPage to this object's pageNumber**. - add circle and text as this object's children. **Hint**: See **NavigationPane** for useful methods.

Method

Name	Description
+ void setActive(boolean value)	/* FILL CODE */ If value is true, set circle's fill color to **Color.WHITE**. Otherwise, set it to **Color.LIGHTGRAY**.

3.2.2 public class **CreatePageButton** /* You must implement this class from scratch */

This class, which extends from **StackPane**, represents the CreatePageButton in the **NavigationPane** that creates a new page when clicked.

Constructor

Name	Description
+ CreatePageButton()	/* FILL CODE */ Initialize the CreatePageButton instance with the following specifications: - initialize a new Circle with radius set to **20** and fill color set to **Color.LIGHTGRAY**. - initialize a new Text with text set to "**+**" and font size set to **20**. - set cursor to **Cursor.HAND**. - when this object is clicked, add a new page in the **NavigationPane**. - add circle and text as this object's children. **Hint**: See **NavigationPane** for useful methods.

3.2.3 public class **TodoItem** /* You must implement this class from scratch */

This class, which extends from **HBox**, represents a single todo item**.** It contains a Text object and a "Delete" button for deleting itself.

Constructor

Name	Description
+ TodoItem(String value)	/* FILL CODE */ Initialize the TodoItem instance with the following specifications:

	- set spacing to **10**. - initialize a new Text with text set to the given **value** and font size set to **20**. - initialize a new Button with text set to **“Delete”**. - set button’s action to do the following: 1. remove this object from currentPage’s todoList. - add the text and button as this object’s children. **Hint**: See **DisplayPane** for useful methods.

3.3 **Package application**

3.3.1 public class **Main**: This class, which extends from **Application**, is provided to launch the JavaFX application. /* Partially PROVIDED */

Method

Name	Description
+ void start(Stage stage)	/* FILL CODE */ Override start method of Application with the following - initialize **RootPane**. - create Scene with following properties: 1. parent is **RootPane** 2. width is **800**. 3. height is **640**. - set stage’s scene to the newly created scene. - set title by **“SIMPLE(?) TO DO LISTS”.** Please set the title EXACTLY as stated here for good fortune. Thanks. - set resizable to **false**. - show stage.

+ void *main*()	Already provided.

Scoring Criteria (22 points, will be scaled to 2.5)

Main

- (1 mark) Main stage is size 800 (w) x 640 (h) and unresizable.
- (1 mark) Title of the main stage is “**SIMPLE(?) TO DO LISTS**”.

RootPane

- (1 mark) RootPane contains NavigationPane, DisplayPane, and InputPane in correct locations.
- (1 mark) RootPane adds page 0 on startup. If this succeeds, the application should start with page 0 already created and is active.

NavigationPane

- (1 mark) NavigationPane displays PageButtons equal to the number of pages.
- (1 mark) NavigationPane contains a single CreatePageButton, which is **always** the last button in the NavigationPane.

DisplayPane

- (2 mark) DisplayPane’s style, spacing and children are as stated in instruction.

InputPane

- (2 mark) InputPane’s style, spacing and children are as stated in instruction.
- (1 mark) When you submit a new todo item, a new TodoItem is created and displayed in current DisplayPane.

PageButton

- (2 mark) PageButton’s style and spacing is as stated in instruction (including cursor).
- (1 mark) When clicked, the todoList in the DisplayPane is replaced with the selected page’s todoList.

- (1 mark) When clicked, the button’s active status is set to TRUE, while other buttons’ active status is set to FALSE.

CreatePageButton

- (2 mark) CreatePageButton’s style and spacing is as stated in instruction (including cursor).
- (1 mark) When clicked, a new page is created.
- (1 mark) When clicked, set current page to the new page (with correct active status).

TodoItem

- (2 mark) TodoItem’s style and spacing is as stated in instruction. The todo string value is displayed correctly with a Button next to it with text “Delete”.
- (1 mark) When delete button clicked, that TodoItem is removed from the DisplayPane.