Q3: (75 minutes, 10.30-11.45)

1. Instruction

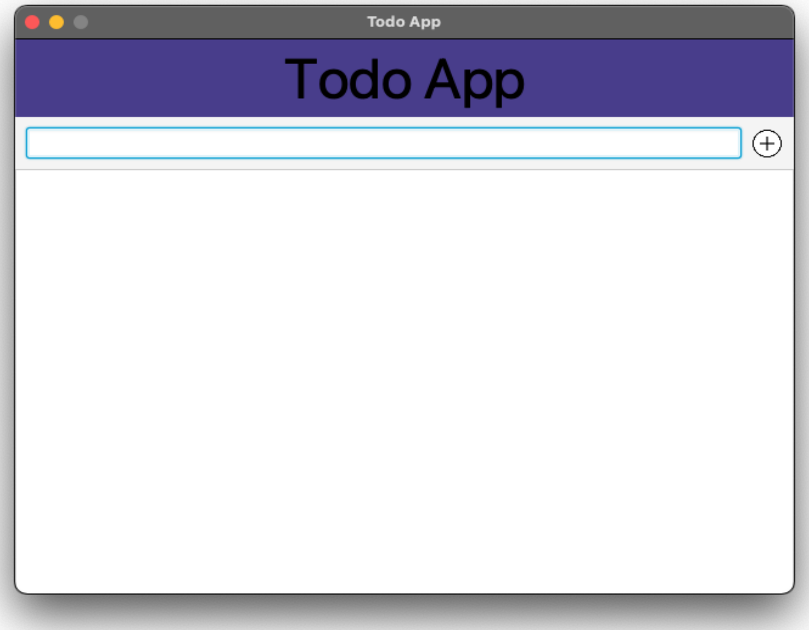
1. Create a Java Project named “**2110215_Final_Q3**”.
2. Copy folders inside “**toStudentQ3**” to your project directory **src** folder.
3. Example command line parameter is given in file “**vmArgument.txt**”
4. You are to implement the following classes (detail for each class is given in section 3 and 4)

a)	ImageButton	(component package)
b)	TitlePane	(component package)
c)	Todo	(component package)
d)	AdderBar	(component package)
e)	Main	(application package)

5. To submit:
    a) go to src folder that you actually do the coding for this question.
    b) Zip this question’s src folder. Name it YOUR-ID_Q3.zip (for example, 6332112121_Q3.zip)
    c) Submit the zipped file as an assignment on MyCourseville.

2. Problem Statement: Jotaro Kujo's note

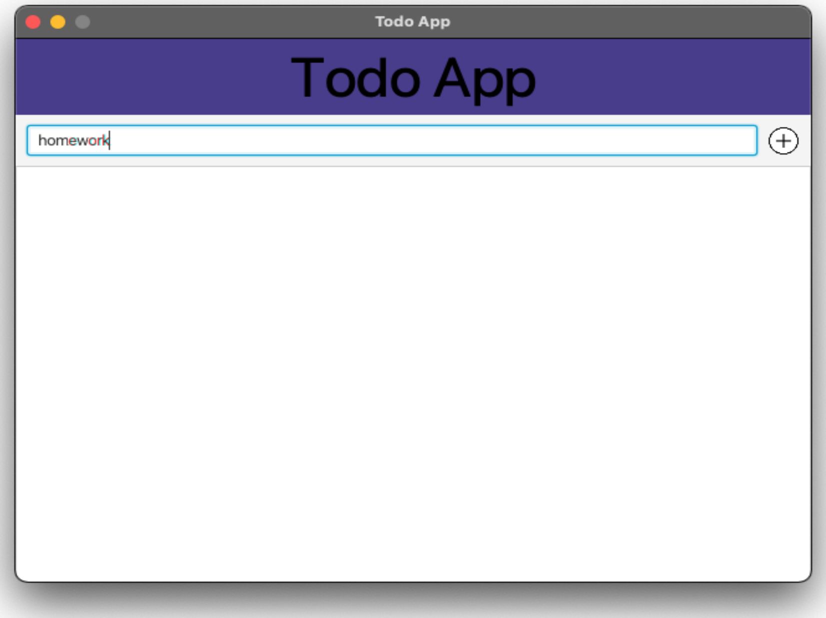
Jotaro Kujo is an ordinary student (the above picture was taken at his hip-hop dance club). Jotaro loves to write a to-do list. He usually writes it on paper. But after an outdoor shopping with his friend, Steely Dan, he had to write so many things and got so tired of writing the list by hand. In this problem, you are assigned to create a Java application to help Jotaro write his to-do list.



Sample screenshot of the application when starts.

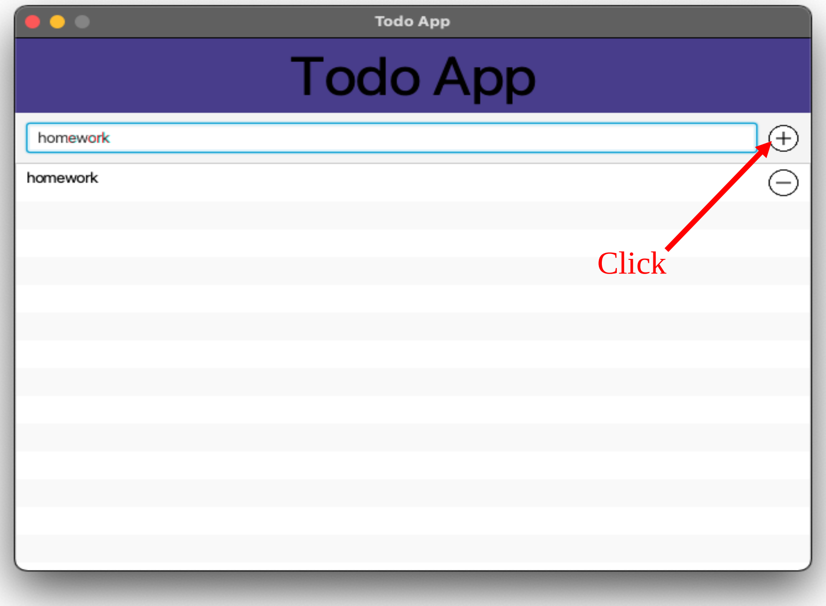
How the application works:

Step 1: input to-do task to text field.



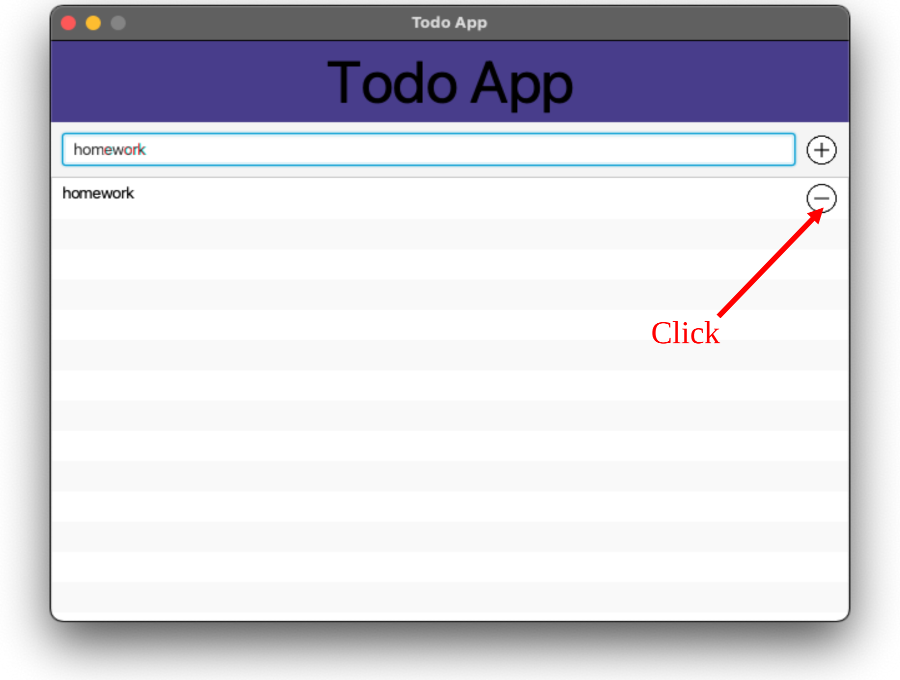
Step 1

Step 2: Click add button to add to-do task to list view.



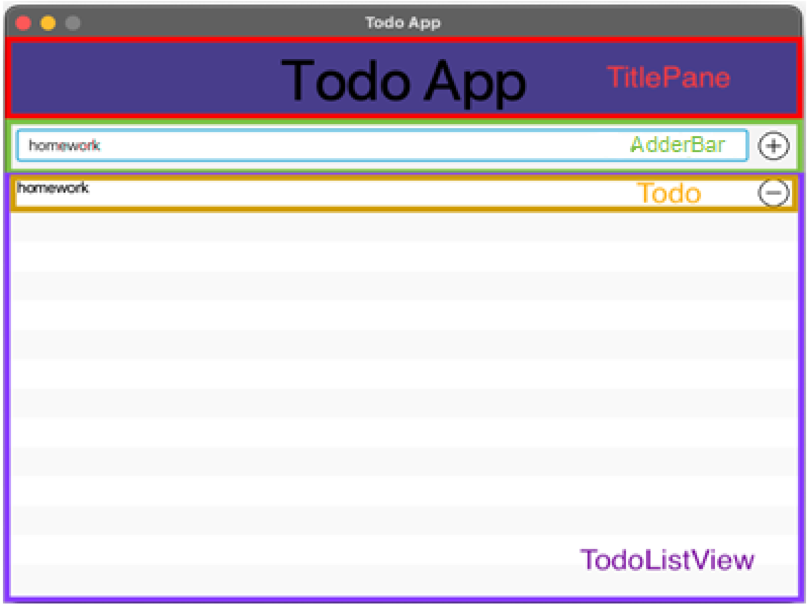
Step 2

Step 3: Click remove button to remove its corresponding to-do task from list view. (You can remove in any row if there are more than one)



Step 3

3. Implementation Detail:



Detailed GUI of the Application.

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

3.1 Package Logic

3.1.1 public class **TodoLogic**: This class contains a useful button method to complete some parts of the application. /* ALREADY PROVIDED */

Method

Name	Description
+ void *addTodo* (String todoText)	This method should be called by ImageButton with ADD mode, it will add a to-do task to TodoListView.
+ void *removeTodo* (ImageButton imageButton)	This method should be called by ImageButton with REMOVE mode, it will remove a to-do task from TodoListView.

3.2 Package component

3.2.1 public class **RootPane**: This class represents a Pane that groups the essential components. /* ALREADY PROVIDED */

Field

Name	Description
- TitlePane title	Representing the TitlePane.
- AdderBar adderBar	Representing the AdderBar.
- TodoListView todoListView	Representing the TodoListView.

Constructor

Name	Description
+ RootPane()	Constructor method, which is already provided.

Method

Name	Description
+ getter/setter for each field.	Already provided.

3.2.2 public class **ImageButton**: This class represents an ImageView to be an either ADD button or REMOVE button. You must implement 2 methods for this class.

Field

Name	Description
- String *addButtonUrl*	Url for the add button image.
- String *removeButtonUrl*	Url for the remove button image.

Constructor

Name	Description
+ ImageButton()	This method is already provided.

Method

Name	Description
- void initImageButton(ImageButtonType imageButtonType)	/* FILL CODE */ Initialize image button with the following specifications: - set fit width to 26. - set fit height to 26. - set image to match the image button type. Suggestion: you may need to change the path for your picture files in the code.

- void initEventHandler(ImageButtonType imageButtonType)	/* FILL CODE */ Initialize image button event handler with the following specifications: - set cursor with "Cursor.HAND". - set mouse clicked event to call correct method in TodoLogic class. #HINT1 Please see the TodoLogic class for the necessary method. #HINT2 Please see the Main class to use some methods to get the information that you want. #HINT3 "<OUTER-CLASS>.this" can be used to invoke "this" of outer class in anonymous class.
+ getter/setter for each field.	These methods are already provided.

3.2.3 public class **TitlePane**: This class represents a BorderPane that can group the essential components. **You must implement 2 methods for this class.**

Field

Name	Description
- Text title	Representing the title text.

Constructor

Name	Description
+ TitlePane()	/* FILL CODE */ Constructor method. Initialize with the following specifications: - Initialize super class.

	- Initialize title text with initTitle() method. - set title pane background with "BackgroundFill(Color.DARKSLATEBLUE, null, null)". - set title pane preference height to 100. - add the title text to the center of TitlePane, using method setCenter.

Method

Name	Description
- void initTitle()	/* FILL CODE */ Initialize title with the following specifications: - set title font with name="Roboto" and size=48 - set title text value="Todo App"
+ getter/setter for each field.	These methods are already provided.

3.2.4 public class **Todo**: This class represents a BorderPane that can group the essential components. You must implement 1 method for this class.

Field

Name	Description
- ImageButton imageButton	Representing the image button.
- Text text	Representing the text value in Todo.

Constructor

Name	Description
+ Todo(Text text, ImageButton imageButton)	/* FILL CODE */ Constructor method. Initialize with the following specifications: - Initialize super class.

	- set imageButton. - set text. - add text to the left of BorderPane, using method setLeft. - add imageButton to the right of BorderPane, using method setRight.

Method

Name	Description
+ ImageButton getImageButton()	Getter of imageButton.
+ getter/setter for each field.	These methods are already provided.

3.2.5 public class **AdderBar**: This class represents a HBox that can group the essential components. You must implement 3 methods for this class.

Field

Name	Description
- TextField textField	Representing the text field.
- ImageButton imageButton	Representing the ADD image button.

Constructor

Name	Description
+ AdderBar()	/* FILL CODE */ Constructor method. Initialize with the following specifications: - Initialize super class. - Initialize textField with initTextField() method. - Initialize imageButton with initImageButton() method. - set preference height to 50. - set padding with "Insets(9)" - set spacing to 8

	- set alignment with “Pos.CENTER” - add textField and imageButton to “this” with correct order.

Method

Name	Description
- void initTextField()	/* FILL CODE */ Initialize textField with the following specifications: - set preference width to 640. - set preference height to 35.
- void initImageButton()	/* FILL CODE */ Initialize imageButton with the following specifications: - set imageButton with “ImageButtonType.ADD”. #HINT Please see the ImageButton implementation.
+ TextField getTextField()	Getter of textField.
+ getter/setter for each field.	These methods are already provided.

3.2.6 public class **TodoListView**: This class represents a ListView<Todo>. /* ALREADY PROVIDED */

3.3 Package Application

3.3.1 public enum **ImageButtonType**: This enum contains image button type. /* ALREADY PROVIDED */

3.3.2 public class **Main:** This class contains the main method. It is an entry point of the application. You must implement 1 method for this class.

Field

Name	Description
- RootPane *rootPane*	Representing the root pane.

Method

Name	Description
+ void start(Stage stage)	/* FILL CODE */ At the TODO section, you must fill the code to complete the start method with the following: (1) set the scene with the root pane as a child with width=640 and height=480. (2) set resizable with false. (3) set title with "Todo App". (4) show the stage
+ String *getAdderBarCurrentText*()	Getter of AdderBar current text value.
+ getter/setter for each field.	These methods are already provided.

Scoring Criteria (15 points, will be scaled to 5)

TitlePane

- (1 mark) TitlePane is visible at its correct size and location.
- (1 mark) TitlePanne has the correct color.
- (1 mark) TiplePane has text “Todo App”.
- (1 mark) The text is at the center.

AdderBar

- (1 mark) AdderBar appears below, next to TitlePane.
- (1 mark) text field is to its left
- (1 mark) “+” button is to its right.
- (1 mark) text field can fill one line of character.
- (1 mark) text field is long until it reaches the “+” button.
- (2 mark) When a “+” button is pressed, a new to-do item is added as the last item (the lowest)
- (1 mark) New to-do item has “-“ appearing at its end.

To-do Item

- (2 mark) When a “-“ button is pressed, the corresponding to-do item disappears.
- (1 mark) Other items must shift to take its place.