Lab 5

Objectives:

- Gain familiarity with JavaFX (MVC) , Exception Handling and Collections
- Gain understand about aesthetics related to GUI development (titles, size of windows, position of buttons, position of images, and such)

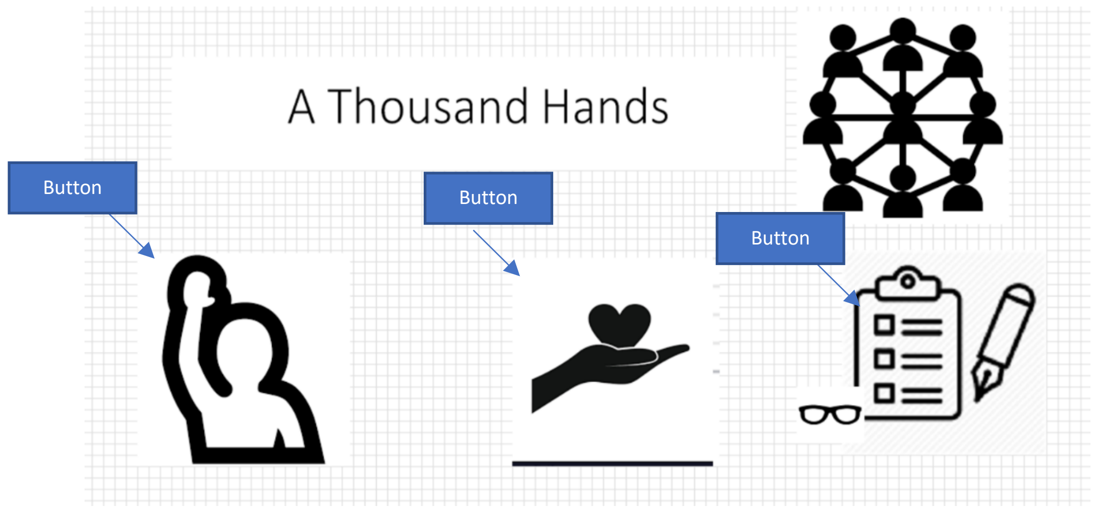
Task: It's all about UX/UI

UX refers to the user experience, which focuses on how something works and how people interact with it. UI, or user interface, focuses on the look and layout. A UX designer ensures a product makes sense to the user by creating a path that logically flows from one step to the next. Many companies are looking for UX/UI engineers and the salary can easily be in the $60K-$120K range. In this lab, you will learn how to use the JavaFX concepts to design and build a User Interface (UI) application, called **AThousandHands**. Your goal is to design and build a simple app that allows novice users (think folks who may not be tech-savvy) to both request and donate items. The goal is to create a repository of items available in the community that we can use/share. We will call this app is called "A Thousand Hands".

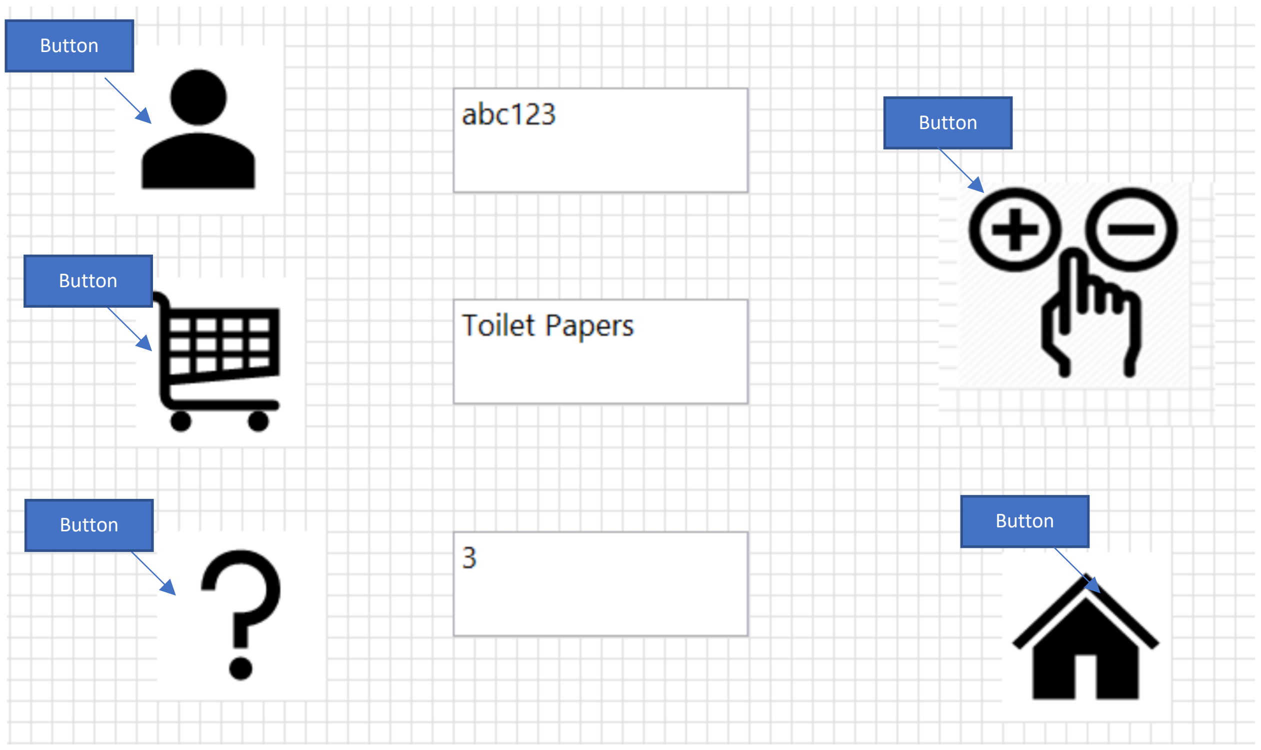
Task1:
In this task, you are required to design a user interface which will show 3 views like the one shown below. Please note the use of images (graphics) is to denote actions. The graphics shown are examples. You can pick/choose other graphics if you feel that they are universally understood.

- **Main.fxml:** In this scene the user can select between three options namely Need, Give and Lookup Inventory.
- **NeedGive.fxml:** This scene allows the user to donate or take items from the repository. The string control next to the Person button allows user to enter userid. The userid should follow a particular format, such as abc123. The string control/Textfield next to the Cart button allows user to enter the item they wish to donate/need. The string control/Textfield next to the Question button allows user to enter the quantity that they wish to donate/need. The "Add/Subtract" button (toggle switch) adds or deletes the items (based on whether User clicked on the Need or Donate button to come to this screen). The "Home" button returns the user to Main.fxml.
- **Inventory.fxml:** This scene would display a user interface that shows all the items available in the inventory with their quantities. The user can search the item presented in the listview and check the quantity by clicking on the Lookup button. The "Home" button returns the user to Main.fxml.

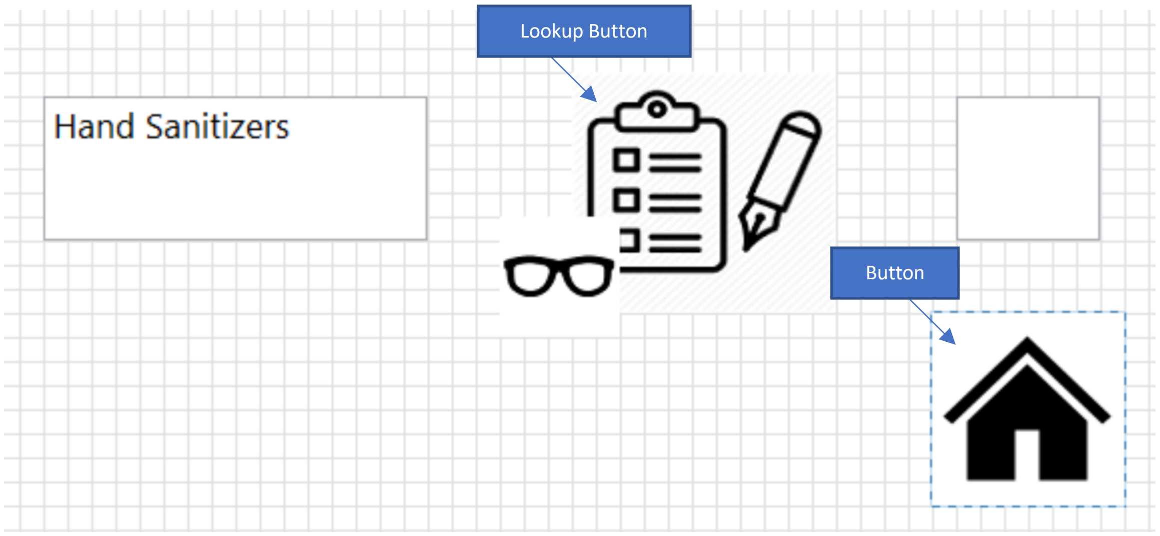
Please note that the user interface shown below is only an example. As seen in the Rubric below, about 25% of the points for this task are assigned for Creativity. Please be creative in your design approach to convey and gather the information as shown in the UI below.



This view will be the **Main.fxml**.



This view will be the **NeedGive.fxml**.



This view will be the **Inventory.fxml**.

It's officially time to get creative! You may choose the images, fonts, colors, app size, and any other style features. Use this opportunity to learn about background colors, good user design, or any aspects that might be useful to your team project. The above views are minimal examples of the requirements for the lab - your submission must reflect all GUI components shown. This includes labels, textfields, buttons, and a list view.

Task 2:
Making it Work
Main.fxml
Main.java will launch the application and show Main.fxml.
NeedGive.fxml.
When this view the user may add a new item to this view by typing in user id, item, and number of items. When the user clicks the "add" button, the following should happen:

- The user-id and type fields should be cleared.
  Use clear() method to clear the textfield.
- A message should display under the add area indicating whether the add was successful using alert pop ups as explained in the class for confirming or error.

Inventory.fxml The user may look up for a item in the inventory view by typing in the name of the item.
MainController.java will be the *EventHandler* for this fxml, and should be connected to all buttons in this view.
NeedGiveController.java class will be the event handler for NeedGive.fxml. When this view is loaded, the username, item name and quantity will be displayed. The user may add a new item to this view by typing in user id, item, and number of items. When the user clicks the "add" button, the following should happen:

- The user-id and type fields should be cleared.
- A message should display under the add area indicating whether the add was successful.
- The number of items should be updated.

InventoryController.java: The user may look up for an item in the inventory view by typing in the name of the item or using a listview (using initialize method). When the user clicks on lookup button, it should populate the quantity available for the selected item.

NeedGiveModel.java: The model class should have the following methods:

- **addItem()** : Implement what happens when the user clicks on the icon to donate an item. If the item already exists in the inventory, its quantity should be incremented by the value indicating the number of items that the user is donating. If the item does not exist in the inventory, it should first be added with qty 0 and then the quantity value should be incremented accordingly.
- **getNumberOfItemsInInventory()**: Implement what happens when the user clicks on the icon to lookup an inventory. It should check if the item requested is in the inventory and if it is, get the value indicating the quantity that is available.
- **subtractItem()** : Implement what happens when the user clicks on the icon to request an item (needs an item). If the item already exists in the inventory, its quantity should be decremented by the value indicating the number of items that the user is requesting.
- **addUserName()** : Implement what happens when an username is added in the app. If the username already exists, then do nothing. If the user name does not exist, then add the user name to the list of users. Also, if the userID is not in the format of abc123, it should give an error saying the ID is not in proper format.

Task 3:

Saving

The user of our app will expect that changes made will persist after they close and reopen the app. In our given example, if the user closes the app after adding the items, then when they reopen the app he should see the updates in the inventory view. This means that upon adding, you will need to call upon the save method previously implemented. As always, the app should maintain one stage only. The app must use FXML for the views and follow MVC design pattern. Model classes must be responsible for the data of this application, and controller classes will call upon methods in the model classes in order to complete the tasks identified above. More model classes can be added based on the requirements of the lab.

The Concept of Java Collection

In this lab, you will use the concept of Java Collection. The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion. Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Map) and classes (ArrayList, LinkedList, PriorityQueue, HashSet, HashMap). You can pick the appropriate classes (both type and number) to store the userid, items and quantities.

Rubric:

Comments & Formatting - All code is properly formatted and commented. This includes your full name and UTSA ID (abc123) at the top of each class. Each class and method should have a

description of what it does, and methods should additionally have a description of any parameters and returned data, as applicable.
Submission - The Eclipse project is correctly submitted to Blackboard, as abc123-lab5.zip.
Correctness - All the programs are written correctly using proper syntax and the logic is correct.
Tests - The grader will test your code by running your program and testing if the output matches the sample output.

Task	Comments and formatting	Class Design	Correctness	Test
1	5	4	8	8
2	5	4	8	8
3	5	13	16	16

What to submit:

Submit a zipped JavaFX project named <abc123>-lab5, and create the following packages, classes, and FXML files:

- application.Main
- application.controller.MainController
- application.controller.NeedGiveController
- application.controller.InventoryController
- application.Model.NeedGiveModel
- application.view.Main.fxml
- application.view.NeedGive.fxml
- application.view.Inventory.fxml