Lab 6: Event Handling and Thread

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_Lab6_2020_2_{ID}_{FIRSTNAME}**
    - Example: **2110215_Lab6_2020_2_6131234521_Baba**.
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 one folder:** `src` **make sure to add it into your project.**
    - 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.
5. After finishing the program, create a UML diagram using **ObjectAid** and put the result image (**UML.png**) at the root of your project folder.
6. Export your project into a jar file called **Lab6_2020_2_{ID}** and place it at the root directory of your project.
    - Example: **Lab6_2020_2_6131234521.jar**
7. Push all other commits to your GitHub repository

1. Problem Statement : Deserted Island Resident Service

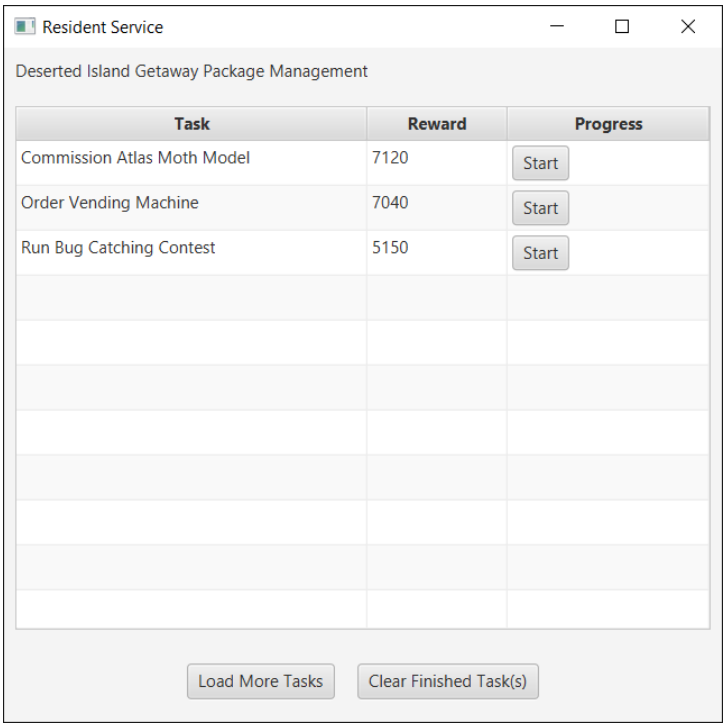
You are a very famous freelance programmer now. One day, you have been contacted by a mysterious client who claims that he works for a large world-class megacompany.

The company itself just started the project called “Deserted Island Getaway Package”. It is a travel package that helps people relieve the stress from the everyday life. The client will get to live the full life on the Deserted Island, surrounded with nature.

However, there is a problem. As the numbers of the resident grows, the number of the task increases and harder to keep track of. So, they decided to build a database program to help assist with the process. Unfortunately, their last programmer went missing after a midnight stroll in a jungle last week and never came back, therefore you are tasked to finish the program.

Do note that the internet and server at the deserted island is very slow, so there will be a noticeable loading time for each data transmission.

1.1 Overview

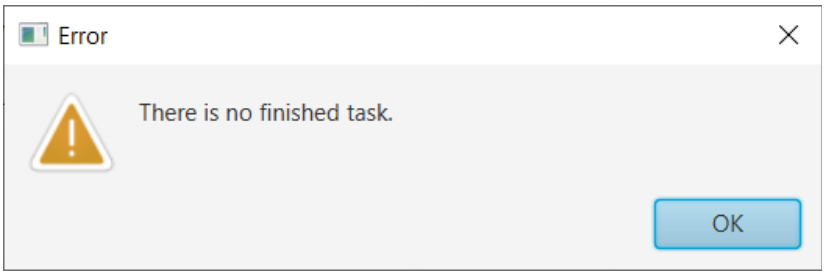


This application contains two main parts, TableView that displays the data and the pane with two control buttons below. Initially, there will be no data loaded in the table, so you need to press “Load More Tasks” first, where the program will gradually load up 3 more tasks.

For each task, there will be a “Start” button that you can click to start on a task. Once clicked, the button will turn into a progress bar and gradually filled up. It will turn into a Label once it finished. When the task is in progress, you can press another “Start” button from

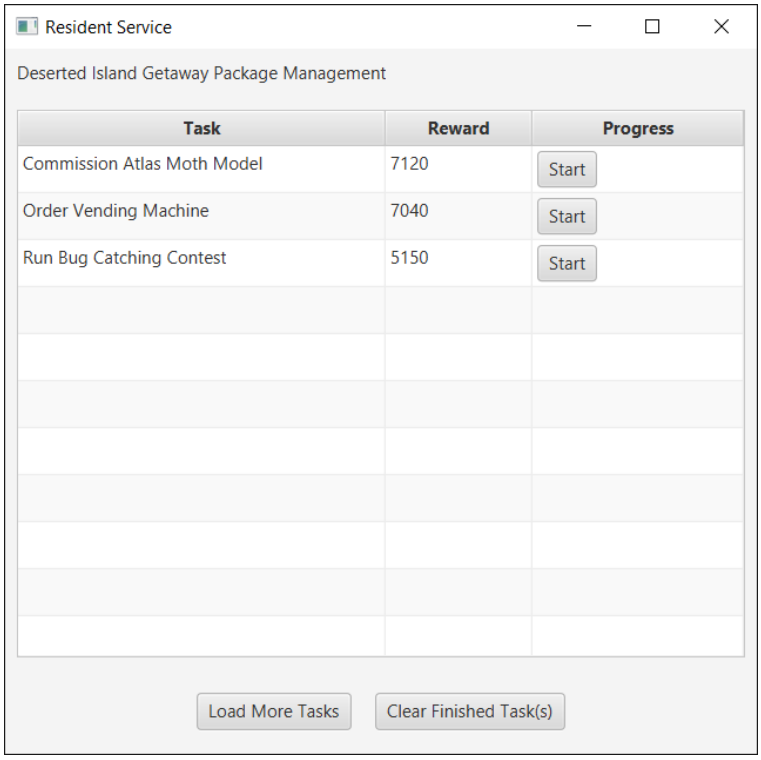
another task and it can be filled concurrently. However, if there is any task in progress, the control buttons at the bottom of the program will be disabled. Similarly, when you load in more tasks, the “Start” button of each task will be disabled until it finished loading.

If the “Clear Finished Tasks” button is pressed, it will clear out all the Finish tasks from the table. However, if there is no finished task, an error will be displayed instead.



Watch the Demo Footage Here: https://streamable.com/gnmgg6

2. Implementation Details:



Most of the Interface structures and logics such as data loading and progression have been provided. You only need to implement the functionality of the Program.

NOTE:

1. The interface of the program **must be responsive all the time**. You can still click the table during loading, for the example. Even when the button is disabled.

2. Thread.sleep commands have been used to simulate the real-life loading delay. **Do not remove it or you will get 0 for this assignment.**

3. Some parts of the program can be implemented without using thread, **but you will not get any point from thread part.**

4. Only important methods/fields will be shown below.

5. Feel free to create your own private methods/fields

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

+ (public), # (protected), - (private), underlined (static), ALL_CAPS (final)

2.1 package `application`

2.1.1 Class `Main` `extends Application`

This class is the main application interface.

2.1.2.1 Fields

-TaskTable taskTable	The Class that contains main TableView of the application
-Button btnLoad	Load Button
-Button btnClear	Clear Button
-int progressingTaskCount	The number of currently progressing tasks.
-int loadingTaskCount	The number of currently loading tasks.

2.1.2.2 Methods

+void start(Stage primaryStage)	Starting the main application, put in Interfaces element. **/*FILL CODE*/**: Loading the Tasks when click Load Button and clearing the task when click Clear Button. The error is shown if there is no

	finished task to be cleared.
+void showWarning()	/*FILL CODE*/ Show the error when there is no finished task to be clear.
+void disableButtonPane()	Disable control buttons (Load/Clear)
+enableButtonPane()	Enable control buttons (Load/Clear)
getter/setter/adder for progressingTaskCount and loadingTaskCount	

2.2 package `data`

2.2.1 Class `TaskTable`

This class contains the main TableView of the application. As well as many API to interact with it.

You do NOT need to modify this class to complete the assignment.

2.2.1.1 Methods

+ TableView<Task> getTableView()	Get the main TableView object of the class
+ void populateData()	Gradually load the data into the table.
+ void removeAllCompletedTask()	Remove all completed task from the table.
+ int getFinishedTaskCount()	Get the number of the current completed task in the table.

2.2.2 Class `ProgressTableCell` `extends TableCell`

This class is the component at the third column of the table that contains the Button, Progress Bar and Label.

2.3.1.1 Fields

- Button button;	The button with the text “Start”

- ProgressBar prog;	The ProgressBar that will be shown during in-progress task.
- Label donetext	The label with the text “Finished” that will be shown when the Task is done.

2.3.1.2 Constructor

+ ProgressTableCell()	**/*FILL CODE*/** Initialize all remaining fields Make Clicking the "Start" Button calling the proper method.

2.3.1.3 Methods

+ void startProgress()	Toggle the UI element to show Progress Bar, as well as calling proper method to fill the Progress Bar. **/*FILL CODE*/** Alter the code here to make UI still responsive while the task is in progress.
+ void simulateProgressIncrement(Double start)	Simulate the Progress Bar filling, with the specified starting value. **/*FILL CODE*/** In this method, there is some JavaFX-related commands. Which will error if called in another thread. Alter the code here to make it works.