Applications Programming

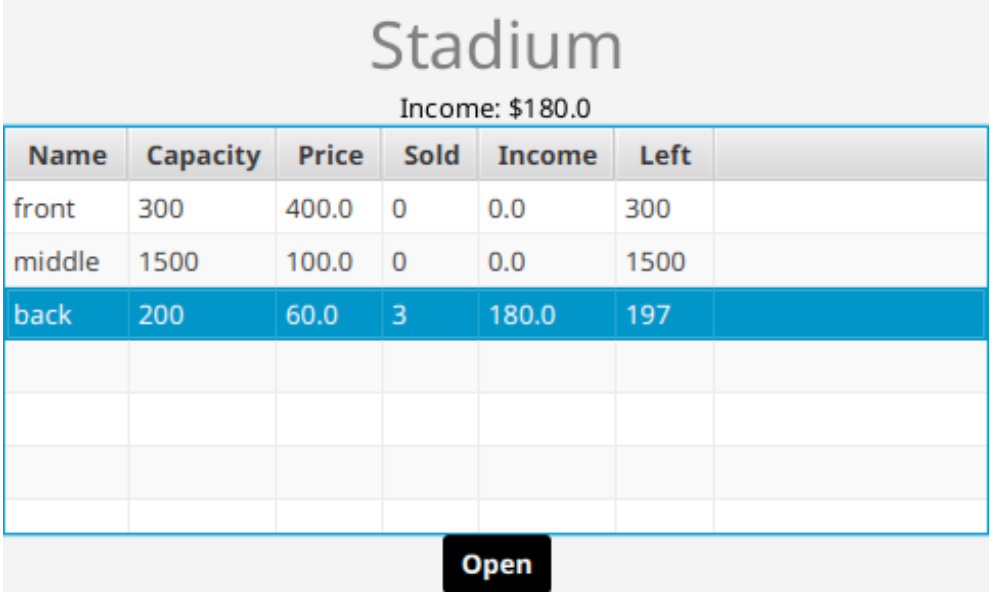
Lab - GUI Tables

REMINDER! Your assignment is due NEXT week! Be sure to ask your tutor for assignment support in this lab.

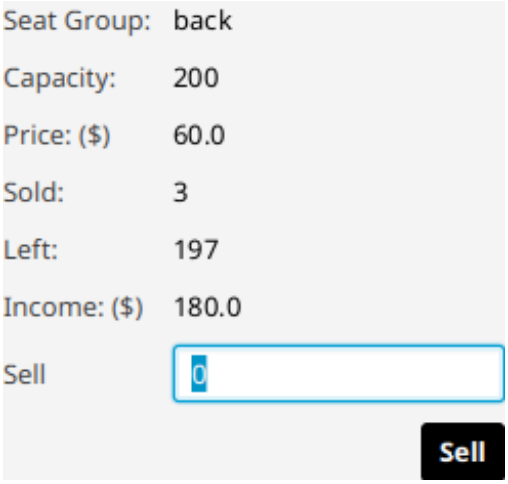
Tutor demo

Main class: `StadiumApplication`

Make an application that sells seats in a stadium. The seats are divided into different groups and each seat group has a different price and capacity. The user interface has two windows as shown below:



Click "Open" →



The user selects a seat group from the table and clicks open. This opens the selected seat group in a new window where the user can view the seat group and sell seats in that group.

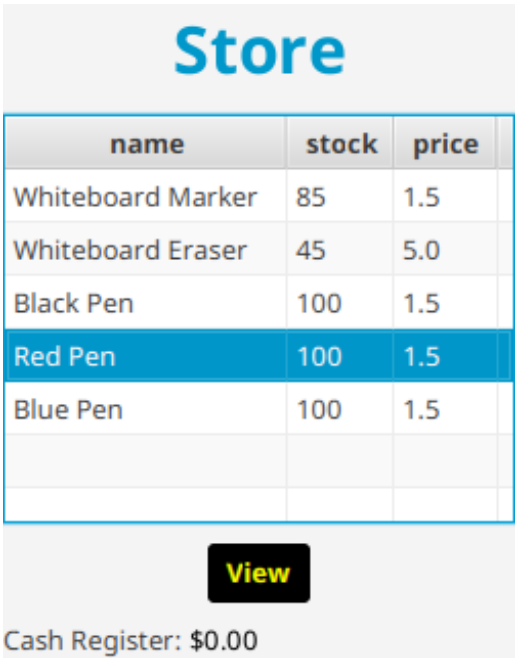
The "Sell" button is disabled except when the TextField is filled with digits.

Your tutor will code the solution.

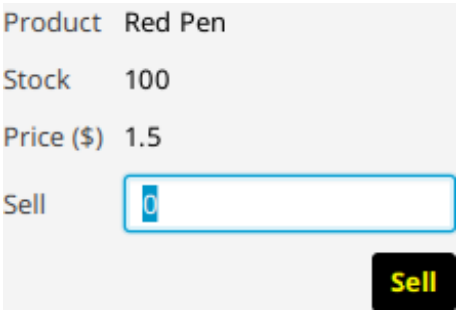
Student Specification

Main class: `StoreApplication`

Make a graphical user interface for a store with a list of products and a cash register. The list is displayed as a TableView. The user interface has two windows as follows:



Click "View" →



In the first window, the user select a product from the table and clicks the "View" button. This opens the second window which shows the details for the selected product and allows the user to sell that product.

Step 1. Open the lecture notes as a reference.

Step 2. Download and open the skeleton code in either NetBeans or your preferred environment. Run the application in its current state before continuing with the following steps.

Step 3. Modify /view/store.fxml to use a TableView instead of a ListView so that it looks like the screenshot above. Inside your TableView FXML element, set the cell value factory for each column using code like this:

```
<TableView fx:id="productsTv" ...etc... >
    <columns>
        <TableColumn text="name">
            <cellValueFactory><PropertyValueFactory property="name"/></cellValueFactory>
        </TableColumn>

        ...etc...

    </columns>
</TableView>
```

Also modify class StoreController so that this TableView is injected into a field of type TableView<Product>. Run your application to make sure it works.

Step 4. Register a ChangeListener with the selectedItem property of the TableView's selection model so that the "View" button is automatically enabled/disabled whenever a product is selected/unselected. You should use code like this:

```
productsTv.getSelectionModel().selectedItemProperty().addListener(
        (observable, oldProduct, newProduct) -> viewBtn.setDisable(newProduct == nul
        );
```

Step 5. *You may move on to your assignment now and come back to this step later*. Register a ChangeListener with text property of the amountTf TextField (beside the Sell label) so that the "Sell" button is enabled only if the user has entered an amount of stock that is less than or equal to the amount of stock available. **Requirement**: Your program must handle exceptions gracefully and not simply print out the strack trace to the terminal.

Assignment Support

Use the remaining lab time to work on your assignment and seek help from your tutor.