Project Burpien

Team: NoLikie - Wael Alghamdi, Alan Ayala

Date: May 1, 2020

Class: CSIS 3750 – Software Engineering

Instructor: Ajoy Kumar

Nova Southeastern University

Abstract

On the professional food business, the cashiers are the ones that are in charge of making a business trade their service for money. We teamed up to provide a better experience for such cashiers, as they role is crucial for operations. After investigating local food stores we came to the conclusion that the current cashier system are too rigid and confusing to operate, so we designed a fully customizable menu systems for cashiers based on our research on current trends and investigation on the impact of such a system on the current world.

Table of Contents

Abstract ---------------------------------------- Page 1

Introduction ---------------------------------------- Page 3

Discussion --------------------------------------- Page 3

Limitations --------------------------------------- Page 5

Conclusion --------------------------------------- Page 6

List of Figures --------------------------------------- Page 7

Appendix A: Controller Package ------------ Page 12

Appendix B: Modules Package ------------ Page 16

Appendix C: Views Package ------------ Page 21

Introduction

The purpose of this project was to provide a better system for cashiers on most industries by creating a simple yet extremely customizable system to sell products on stores. In order to do this, we perfected our craft on JavaFX and Java in order to provide the best possible program to the best of our abilities. The problem that kickstarted this project is the fact that current store cashiers’ systems are too rigid and not too user friendly, so we worked together to create Burpien, a project aimed to fix such issue. Our main limitation overall was time. In order to come up with this design, we visited various stores and examined their cashier systems in order to come up with something similar yet fresh that could be applied anywhere.

Discussion

For the logging system, we built our own as an ID-based system. This ID is based on each user that are, for now, built into the system. In our version 1 of this program, the users are built into the system and those users include: a RootUser with id 0000, a Supervisor with id 0001, and two employee users with ids 1234 and 1235, respectively. More on the types of users later. Figure 1 displays the login page with entrée 0000 for RootUser.

In figure 2 we can see the program’s main page. Here is where everything is done. Here is a list of what can be done here, new order creation, order traversal, order canceling, showcasing the categories of and the items on sale, the order basket, which means the list of items in the order, the payment, and in the case of having enough privileges, one can switch to the settings page, more on that later. Figure 3 show cases a creation of a new order and a selection of some items into the basket.

New order creation is done with clicking on the “New Order” button on the top bar on the right. Traversing the orders created, can be done using the arrows to the left and to the right of the “New Order” button. Creating a new order or traversing the order list will not affects the current order and the order will stay in an “open” condition. An “open” condition means the order was payed for and cannot be modified any longer which in turn also means that it cannot be canceled. If an order was made by mistake or that the customer changed their minds, the order could be canceled using the “Cancel” button on the top right.

When a new order is created, an empty order is added to the list of orders. To add items(the products to be sold) to the basket, basically, just click on which item wanted from the big area on the left side of the main page, as shown in figure 2 and 3.

The items are included under categories and cannot exist without being under a category. By clicking on the category buttons, the buttons on the top of the left side, the list of items under it will change to be populated with that category’s items.

The basket section displays the current selected order from the order list. The items names and prices will show up in the white box and the totals will be calculated underneath it. If an item was added by mistake it can be deleted from the basket using the “del” button. The item to be deleted must be selected first then it will be deleted, otherwise nothing happens. When clicking the “Pay” button, the order is now considered complete and will not be modifiable anymore. A new order must be created to be able to do any further action in the app.

In the Settings page, we can modify the list of categories and items in the app. Category creation is done by clicking the button “new” then selecting the type “Category”. Selection must be made for all things to be created, be it either an Item or a Category. In figure 1, we can see that we are creating a new category called “extra”. In figure 5 we see the category created. While also, in figure 5, we see that we are creating a new item and adding it to the “extra” category we made beforehand.

For the time being, all Users, Items and Categories are hard-coded, meaning that when the programs shuts down, anything that was done on that session would be lost.

Limitations

* Currently the app cannot store the orders for further use as no database exists for the app.

This is a problem as the app will require to store values to be accessed by the many functions of the app. This also includes the database for the categories and items.

* Currently the app cannot create new users through the program, they have to be hard coded in order to be accessible. This is a problem as currently the app is limited to two users and a root user, and its not able to create new users through the app, which would be hard for not computer savvy employees.
* There exist formatting issues on the basket. This is a problem as the basket is what shows employees the items that are to be sold, something that could cause issues when interacting with customers.
* The create button on the settings page cannot modify or delete existing items, it can only create new ones. This is a problem as one of the main features of this app is to be fully customizable, and not being able to edit or delete created buttons is a huge problem for such a project.
* If too many items are present on one category, the app keeps creating new buttons but they will be displayed out of bounds, not being accessible even if the app is on full screen. This is a problem as the app itself doesn’t have a limit on how many buttons can be created per category, making the experience not optimal as an user could be trying to create new buttons but wouldn’t know the buttons exist but can be accessed.
* Further polish of the app’s UI is required. This is a problem as the UI of the final product should be polished as much as possible before release.
* General bug fixes and quality of life improvements are needed. This is a problem as bugs on the program can cause issues for companies.

Conclusion

In its current version, the program is but a concept program. It can be used but is not practical. With the future improvement it can be used as it was intended. But for now, without having the ability to create users, store order history in a database, or modify/delete items and categories, it is not useable in a professional environment.

List of Figures

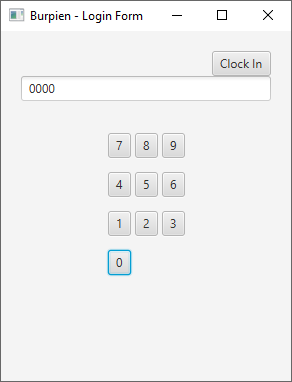


Figure 1 – Login Page

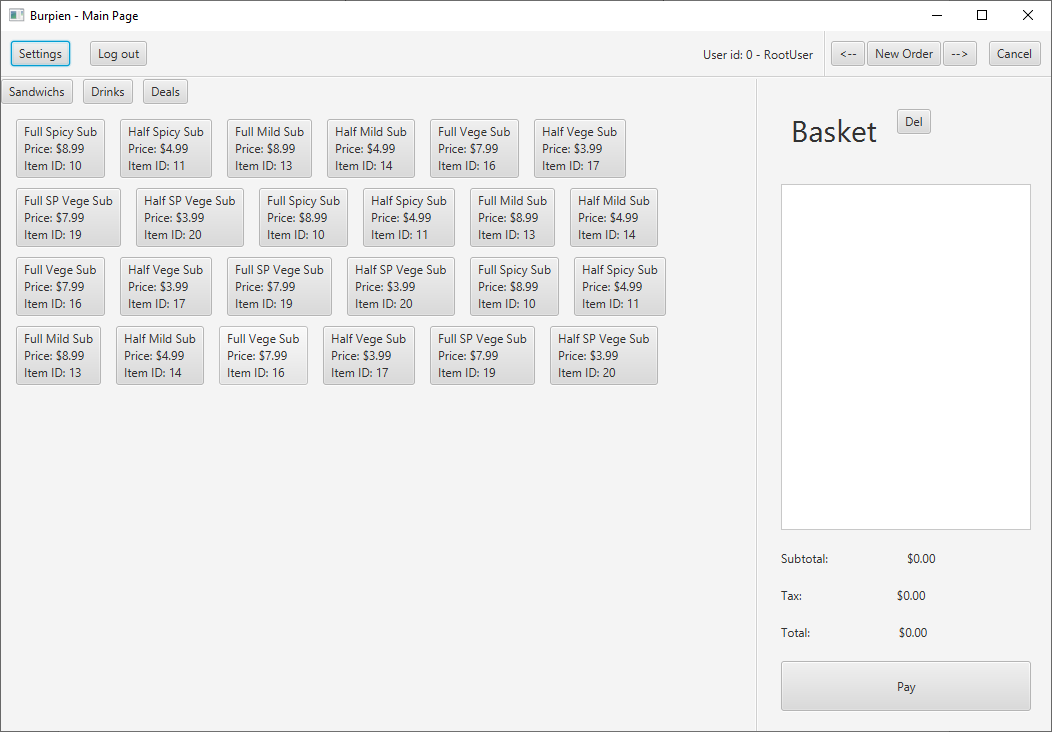


Figure 2 – Main program page

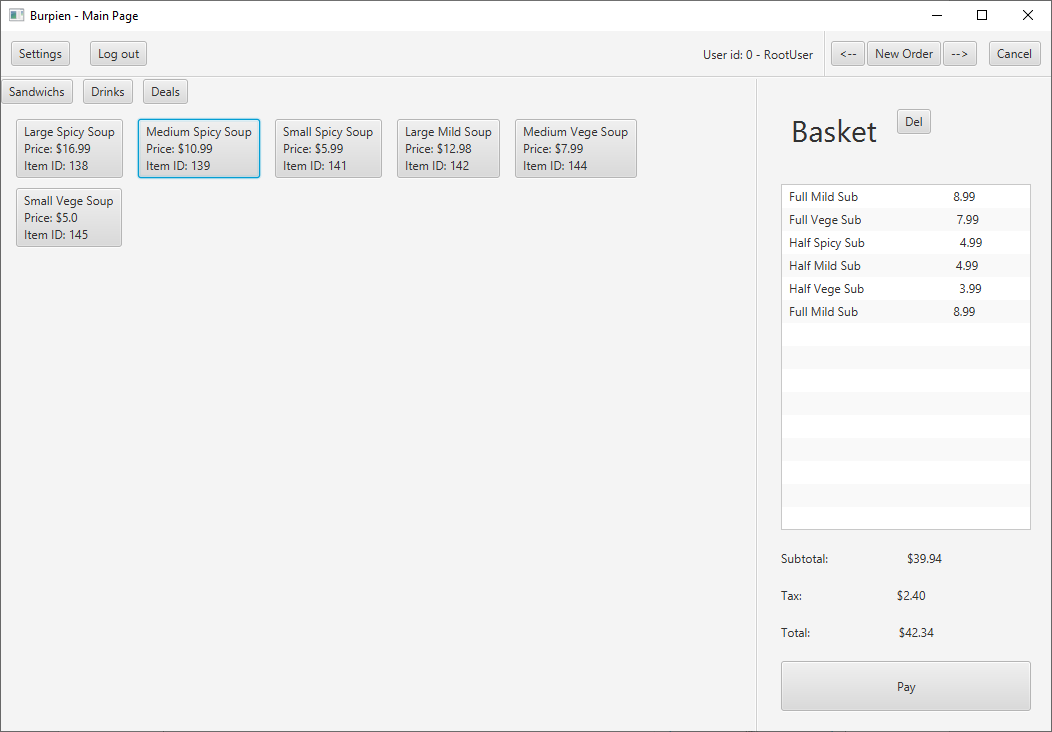


Figure 3 – Main page basket with items

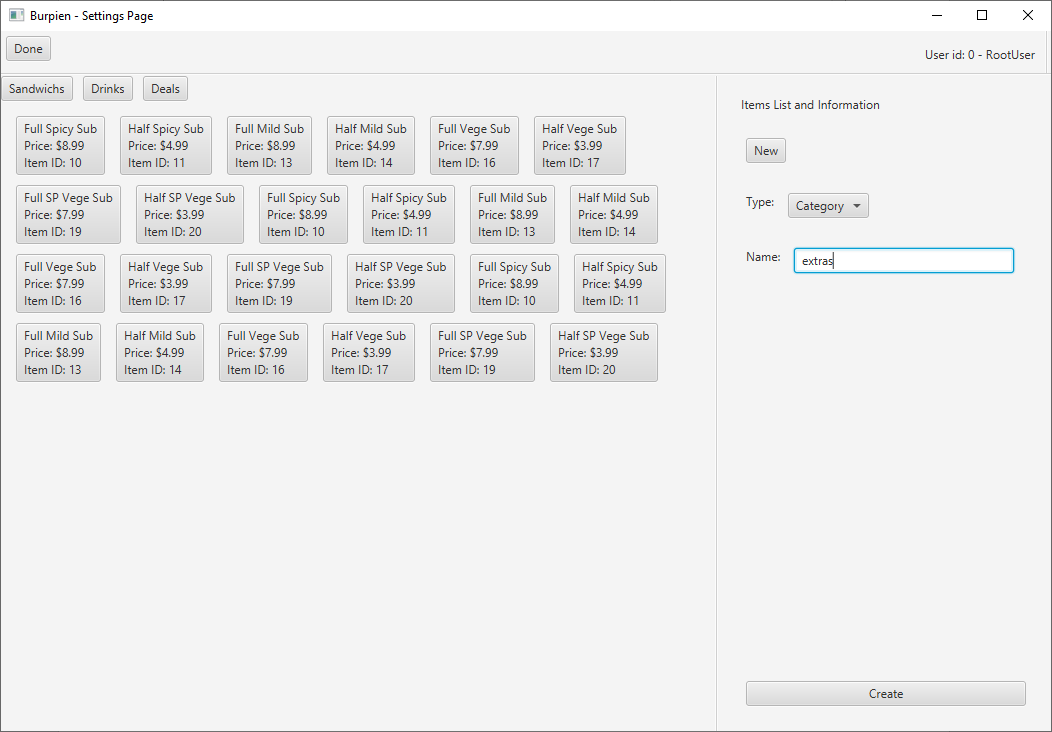


Figure 4 – Settings page – new category selection

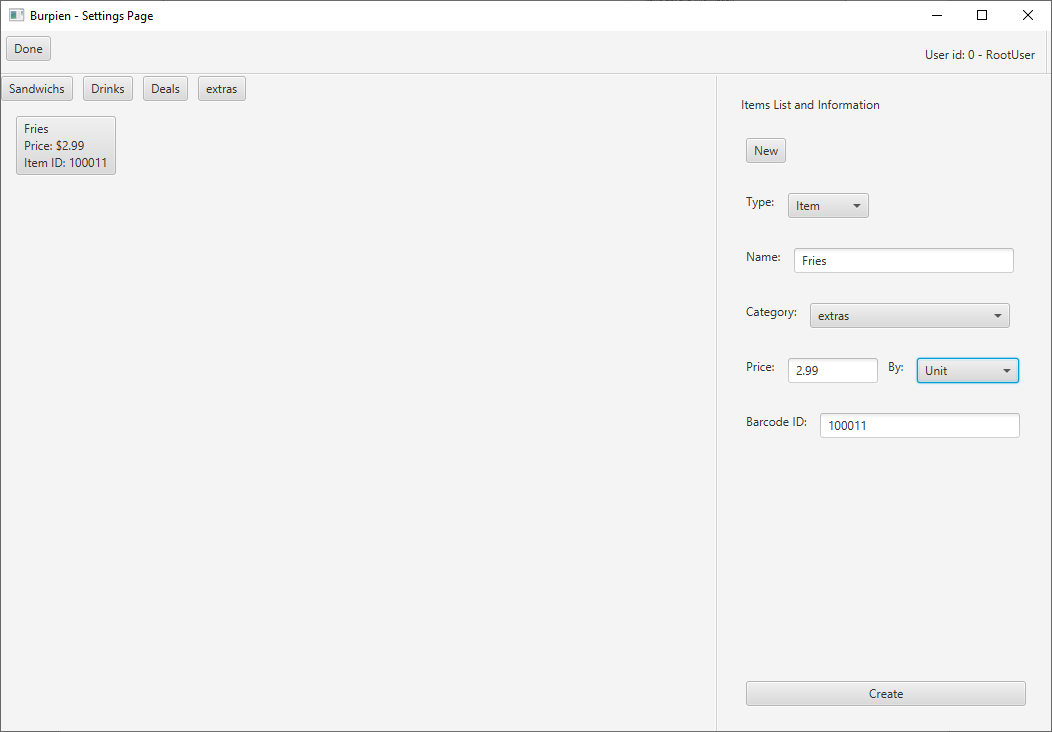


Figure 5 – Settings page – new Item selection

**Appendix A: Controller Package**

Controller.java

package Application.controller;  
  
import java.util.ArrayList;  
  
import Application.modules.\*;  
import Application.views.\*;  
  
import javafx.application.Application;  
import javafx.event.ActionEvent;  
import javafx.stage.Stage;  
  
public class Controller extends Application {  
  
 public final static double *TAX\_RATE* = 0.06;  
 private static ArrayList<User> *users*;  
 private static ArrayList<Category> *categories*;  
  
 private static ArrayList<Basket> *baskets*;  
 private static User *currentUser*;  
  
 private static Stage *stage*;  
   
 //start up the application  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
 @Override  
 public void start(Stage primaryStage) {  
 specialSetup();  
 *stage* = primaryStage;  
 *newStart*();  
 }  
   
 //setup - temporary - until database is made  
 private void specialSetup() {  
 //initialization of needed variables  
 *baskets* = new ArrayList<Basket>();  
  
 //add users  
 *users* = new ArrayList<User>();  
 *users*.add(new User("RootUser", 0000, Privilage.*Root*));  
 *users*.add(new User("Supervisor", 0001, Privilage.*SuperVisor*));  
 *users*.add(new User("Wael A.", 1234, Privilage.*Employee*));  
 *users*.add(new User("Alan A.", 1235, Privilage.*Employee*));  
   
 //add some dummy categories  
 *categories* = new ArrayList<Category>();  
 *categories*.add(new Category("Sandwichs"));  
 *categories*.add(new Category("Drinks"));  
 *categories*.add(new Category("Deals"));  
   
 //add some dummy items   
 *categories*.get(0).additem("Full Spicy Sub", 8.99, PriceType.*Unit*, 00012);  
 *categories*.get(0).additem("Half Spicy Sub", 4.99, PriceType.*Unit*, 00013);  
 *categories*.get(0).additem("Full Mild Sub", 8.99, PriceType.*Unit*, 00015);  
 *categories*.get(0).additem("Half Mild Sub", 4.99, PriceType.*Unit*, 00016);  
 *categories*.get(0).additem("Full Vege Sub", 7.99, PriceType.*Unit*, 00020);  
 *categories*.get(0).additem("Half Vege Sub", 3.99, PriceType.*Unit*, 00021);  
 *categories*.get(0).additem("Full SP Vege Sub", 7.99, PriceType.*Unit*, 00023);  
 *categories*.get(0).additem("Half SP Vege Sub", 3.99, PriceType.*Unit*, 00024);  
  
 *categories*.get(0).additem("Full Spicy Sub", 8.99, PriceType.*Unit*, 00012);  
 *categories*.get(0).additem("Half Spicy Sub", 4.99, PriceType.*Unit*, 00013);  
 *categories*.get(0).additem("Full Mild Sub", 8.99, PriceType.*Unit*, 00015);  
 *categories*.get(0).additem("Half Mild Sub", 4.99, PriceType.*Unit*, 00016);  
 *categories*.get(0).additem("Full Vege Sub", 7.99, PriceType.*Unit*, 00020);  
 *categories*.get(0).additem("Half Vege Sub", 3.99, PriceType.*Unit*, 00021);  
 *categories*.get(0).additem("Full SP Vege Sub", 7.99, PriceType.*Unit*, 00023);  
 *categories*.get(0).additem("Half SP Vege Sub", 3.99, PriceType.*Unit*, 00024);  
  
 *categories*.get(0).additem("Full Spicy Sub", 8.99, PriceType.*Unit*, 00012);  
 *categories*.get(0).additem("Full Vege Sub", 7.99, PriceType.*Unit*, 00020);  
 *categories*.get(0).additem("Half Vege Sub", 3.99, PriceType.*Unit*, 00021);  
 *categories*.get(0).additem("Full SP Vege Sub", 7.99, PriceType.*Unit*, 00023);  
 *categories*.get(0).additem("Half SP Vege Sub", 3.99, PriceType.*Unit*, 00024);  
   
 *categories*.get(1).additem("Large Spicy Soup", 16.99, PriceType.*Unit*, 00212);  
 *categories*.get(1).additem("Medium Spicy Soup", 10.99, PriceType.*Unit*, 00213);  
 *categories*.get(1).additem("Small Spicy Soup", 5.99, PriceType.*Unit*, 00215);  
 *categories*.get(1).additem("Large Mild Soup", 12.98, PriceType.*Unit*, 00216);  
 *categories*.get(1).additem("Medium Vege Soup", 7.99, PriceType.*Unit*, 00220);  
 *categories*.get(1).additem("Small Vege Soup", 5.00, PriceType.*Unit*, 00221);  
  
 *categories*.get(2).additem("Large Spicy Soup\nHalf Spicy Sub", 18.99, PriceType.*Unit*, 00312);   
 }  
   
 //handlers  
 public static void handleEventsFromMainPage(ActionEvent e) {  
 //handle events  
 } // for later versions  
 public static void switchToSettingsPage(){  
 SettingsPageView.*start*(*stage*);  
 }  
 public static void switchToMainPage(){  
 MainPageView.*start*(*stage*);  
 }  
 public static void logout() {  
 Controller.*setCurrentUser*(null);  
 *newStart*();  
 }  
 private static void newStart() {  
 new LoginPage(*stage*);  
 }  
  
 //getters  
 public static ArrayList<Category> getCategories() {  
 return *categories*;  
 }  
 public static ArrayList<User> getUsers() {  
 return *users*;  
 }  
 public static User getCurrentUser() {  
 return *currentUser*;  
 }  
 public static ArrayList<Basket> getBaskets() {  
 return *baskets*;  
 }  
 //setters   
 public static void setCurrentUser(User currentUser) {  
 Controller.*currentUser* = currentUser;  
 }  
}

**Appendix B: Modules Package**

Basket.java

package Application.modules;  
  
import Application.controller.Controller;  
  
import java.util.ArrayList;  
  
public class Basket {  
 private ArrayList<Item> itemsInOrder;  
 private User user;  
 private boolean isEditable;  
 private double subTotal, tax, total;  
  
 public Basket(User user) {  
 this.user = user;  
 isEditable = true;  
 itemsInOrder = new ArrayList<Item>();  
  
 subTotal = 0.0;  
 tax = 0.0;  
 total = 0.0;  
 }  
  
 public void addItemToOrder(Item item) {   
 itemsInOrder.add(item);  
 calculation();  
 }  
 public void removeItemInOrder(int itemIndex) {  
 //itemsInOrder.remove(item);  
 itemsInOrder.remove(itemIndex);  
 calculation();  
 }  
 public ArrayList<Item> getItemsInOrder() {  
 return itemsInOrder;  
 }  
  
 private void calculation() {  
 subTotal = 0.0;  
 for (Item item: itemsInOrder)  
 subTotal += item.getPrice();  
 tax = subTotal \* Controller.*TAX\_RATE*;  
 total = subTotal + tax;  
 }  
  
 public double getSubTotal() {  
 return subTotal;  
 }  
 public double getTax() {  
 return tax;  
 }  
 public double getTotal() {  
 return total;  
 }  
  
 public boolean isOrderEditable() {  
 return isEditable;  
 }  
 public void finalizeOrder() {  
 isEditable = false;  
 }  
}

Category.java

package Application.modules;  
  
import java.util.ArrayList;  
  
public class Category {  
 private String name;  
 private ArrayList<Item> items;  
   
 public Category(String name) {  
 this.name = name;  
 items = new ArrayList<Item>();  
 }  
   
 //setters  
 public void additem(String nameOfItem, double price, PriceType unit, int id) {  
 items.add(new Item(nameOfItem, price, unit, id));  
 }  
  
 //getters  
 public ArrayList<Item> getItems() {  
 return items;  
 }  
   
 public String getName() {  
 return name;  
 }  
}

Item.java

package Application.modules;  
  
public class Item {  
 private String nameOfItem;  
 private double price;  
 private PriceType unit;  
 private int id;  
  
 public Item(String nameOfItem, double price, PriceType unit, int id) {  
 this.nameOfItem = nameOfItem;  
 this.price = price;  
 this.unit = unit;  
 this.id = id;  
 }  
  
 public String getNameOfItem() {  
 return nameOfItem;  
 }  
  
 public void setNameOfItem(String nameOfItem) {  
 this.nameOfItem = nameOfItem;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 public PriceType getUnit() {  
 return unit;  
 }  
  
 public void setUnit(PriceType unit) {  
 this.unit = unit;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String toString() {// Half Spicy Sub\nPrice: $4.99  
 return nameOfItem + "\nPrice: $" + price + "\nItem ID: " + id;  
 }  
}

PriceType.java

package Application.modules;  
  
public enum PriceType {  
 *Unit*,  
 *Weight*}

Privilege.java

package Application.modules;  
  
public enum Privilage {  
 *Root*,  
 *SuperVisor*,  
 *Employee*,  
}

User.java

package Application.modules;  
  
public class User {  
 private String userName;  
 private int userId;  
 private Privilage privilage;  
  
 public User(String userName, int userId, Privilage privilage) {  
 this.privilage = privilage;  
 this.userId = userId;  
 this.userName = userName;  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public int getUserId() {  
 return userId;  
 }  
  
 public Privilage getPrivilage() {  
 return privilage;  
 }  
  
}

**Appendix C: Views Package**

LoginPage.java

package Application.views;  
  
  
import Application.controller.Controller;  
import Application.modules.User;  
import javafx.event.ActionEvent;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Priority;  
import javafx.scene.layout.Region;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
public class LoginPage {  
  
 private static Button *clockinbtn*;  
 private static Button *btn1*, *btn2*, *btn3*, *btn4*, *btn5*, *btn6*, *btn7*, *btn8*, *btn9*, *btn0*;  
 private static TextField *textfield1*;  
 private Stage stage;  
  
 public LoginPage(Stage stage) {  
 // *TODO Auto-generated constructor stub* this.stage = stage;  
 start(stage);  
 }  
  
 public void start(Stage primaryStage) {  
 //prep for Page  
 initializeButtons();  
 setOnActionsForButtons();  
  
 //page setup  
 Region spacer = new Region();  
 HBox.*setHgrow*(spacer, Priority.*ALWAYS*);  
  
 HBox hboxtitles = new HBox(spacer, *clockinbtn*);  
  
 GridPane gridpane1 = new GridPane();  
 gridpane1.setPadding(new Insets(10));  
 gridpane1.setHgap(10);  
 gridpane1.setVgap(10);  
 gridpane1.setAlignment(Pos.*CENTER*);  
 gridpane1.setGridLinesVisible(false);  
   
 HBox hbox1 = new HBox();  
 hbox1.getChildren().addAll(*btn7*, *btn8*, *btn9*);  
 HBox.*setMargin*(*btn7*, new Insets(2));  
 HBox.*setMargin*(*btn8*, new Insets(2));  
 HBox.*setMargin*(*btn9*, new Insets(2));  
  
 HBox hbox2 = new HBox();  
 hbox2.getChildren().addAll(*btn4*, *btn5*, *btn6*);  
 HBox.*setMargin*(*btn4*, new Insets(2));  
 HBox.*setMargin*(*btn5*, new Insets(2));  
 HBox.*setMargin*(*btn6*, new Insets(2));  
  
 HBox hbox3 = new HBox();  
 hbox3.getChildren().addAll(*btn1*, *btn2*, *btn3*);  
 HBox.*setMargin*(*btn1*, new Insets(2));  
 HBox.*setMargin*(*btn2*, new Insets(2));  
 HBox.*setMargin*(*btn3*, new Insets(2));  
  
 HBox hbox4 = new HBox();  
 hbox4.getChildren().addAll(*btn0*);  
 HBox.*setMargin*(*btn0*, new Insets(2));  
  
 gridpane1.add(hbox1, 0, 2);  
 gridpane1.add(hbox2, 0, 3);  
 gridpane1.add(hbox3, 0, 4);  
 gridpane1.add(hbox4, 0, 5);  
  
 VBox vbox1 = new VBox(hboxtitles, *textfield1*, gridpane1);  
 vbox1.setPadding(new Insets(10));  
  
 BorderPane bp = new BorderPane();  
 bp.setCenter(vbox1);  
 bp.setPadding(new Insets(10));  
   
   
 //Scene setup  
 Scene scene = new Scene(bp, 290, 350);  
 primaryStage.setMinHeight(350);  
 primaryStage.setMinWidth(290);  
 primaryStage.setScene(scene);  
 primaryStage.sizeToScene();  
 primaryStage.setTitle("Burpien - Login Form");  
 primaryStage.show();  
  
 }  
  
 private void initializeButtons() {  
 // first create labels  
 *clockinbtn* = new Button("Clock In");  
  
 // now create text fields  
 *textfield1* = new TextField();  
 *textfield1*.setEditable(false);  
  
 // now create the buttons  
 *btn1* = new Button("1");  
 *btn2* = new Button("2");  
 *btn3* = new Button("3");  
 *btn4* = new Button("4");  
 *btn5* = new Button("5");  
 *btn6* = new Button("6");  
 *btn7* = new Button("7");  
 *btn8* = new Button("8");  
 *btn9* = new Button("9");  
 *btn0* = new Button("0");  
 }  
  
 private void setOnActionsForButtons() {  
 *clockinbtn*.setOnAction(e -> handleButtonClicks(e));  
  
 *btn1*.setOnAction(e -> handleButtonClicks(e));  
 *btn2*.setOnAction(e -> handleButtonClicks(e));  
 *btn3*.setOnAction(e -> handleButtonClicks(e));  
 *btn4*.setOnAction(e -> handleButtonClicks(e));  
 *btn5*.setOnAction(e -> handleButtonClicks(e));  
 *btn6*.setOnAction(e -> handleButtonClicks(e));  
 *btn7*.setOnAction(e -> handleButtonClicks(e));  
 *btn8*.setOnAction(e -> handleButtonClicks(e));  
 *btn9*.setOnAction(e -> handleButtonClicks(e));  
 *btn0*.setOnAction(e -> handleButtonClicks(e));  
   
 }  
  
 private void handleButtonClicks(ActionEvent e) {  
 //handle clock-in action  
 if (e.getSource() == *clockinbtn*) {  
 if (!*textfield1*.getText().equals("")) {  
 int id = Integer.*parseInt*(*textfield1*.getText()); // this will cause the program to overflow when entering a huge number. -- fix this  
 *textfield1*.setText("");  
   
 for(User user: Controller.*getUsers*())  
 if (user.getUserId() == id) {  
 Controller.*setCurrentUser*(user);  
 Controller.*switchToMainPage*();  
 }  
 }  
 }  
   
 //handling the number pad  
 else if (e.getSource() == *btn1*)  
 *textfield1*.setText(*textfield1*.getText() + "1");  
 else if (e.getSource() == *btn2*)  
 *textfield1*.setText(*textfield1*.getText() + "2");  
 else if (e.getSource() == *btn3*)  
 *textfield1*.setText(*textfield1*.getText() + "3");  
 else if (e.getSource() == *btn4*)  
 *textfield1*.setText(*textfield1*.getText() + "4");  
 else if (e.getSource() == *btn5*)  
 *textfield1*.setText(*textfield1*.getText() + "5");  
 else if (e.getSource() == *btn6*)  
 *textfield1*.setText(*textfield1*.getText() + "6");  
 else if (e.getSource() == *btn7*)  
 *textfield1*.setText(*textfield1*.getText() + "7");  
 else if (e.getSource() == *btn8*)  
 *textfield1*.setText(*textfield1*.getText() + "8");  
 else if (e.getSource() == *btn9*)  
 *textfield1*.setText(*textfield1*.getText() + "9");  
 else if (e.getSource() == *btn0*)  
 *textfield1*.setText(*textfield1*.getText() + "0");  
   
 }  
  
}

MainPageView.java

package Application.views;  
  
  
import javafx.geometry.Orientation;  
import javafx.scene.Scene;  
import javafx.scene.control.Separator;  
import javafx.scene.layout.\*;  
import javafx.stage.Stage;  
  
public class MainPageView {  
 static VBox *topAndBottomOfMain*;  
 static HBox *bottomRightAndLeftSide*;  
  
 public static void start(Stage primaryStage) {  
 // The general layout of the program  
 *topAndBottomOfMain* = new VBox();  
 *bottomRightAndLeftSide* = new HBox();  
 //-----------------------------------  
   
   
 // the elements of the page  
 BorderPane itemsAndCatview = new CatAndItemsView().getItemsAndCatView();  
 VBox basketView = new BasketView().getBasketView();  
 HBox menubar = new MainPageMenubarView().getMenuBar();  
 //-----------------------------------  
   
  
 // add all together  
 Separator separatorTopButton = new Separator();  
 Separator separatorButtomParts = new Separator();  
 separatorButtomParts.setOrientation(Orientation.*VERTICAL*);  
   
 HBox.*setHgrow*(itemsAndCatview, Priority.*ALWAYS*);  
   
 *bottomRightAndLeftSide*.getChildren().addAll(itemsAndCatview, separatorButtomParts, basketView);  
 *topAndBottomOfMain*.getChildren().addAll(menubar, separatorTopButton, *bottomRightAndLeftSide*);  
 //----------------------------------  
  
   
 // Display it all  
 Scene scene = new Scene(*topAndBottomOfMain*, 1050, 700);  
 primaryStage.setMinWidth(720);  
 primaryStage.setMinHeight(500);  
 primaryStage.setScene(scene);  
 primaryStage.sizeToScene();  
 primaryStage.setTitle("Burpien - Main Page");  
 primaryStage.show();  
 //----------------------------------  
 }  
}

MainPageMenubarView.java

package Application.views;  
  
import Application.controller.Controller;  
import Application.modules.Privilage;  
import Application.modules.User;  
import javafx.event.ActionEvent;  
import javafx.geometry.Insets;  
import javafx.geometry.Orientation;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.Separator;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Priority;  
import javafx.scene.layout.Region;  
  
public class MainPageMenubarView {  
  
 private HBox menubar;  
 private Button settingsbtn, logbtn;  
 private Label userlbl;  
 private Button leftarrowbtn, neworderbtn,rightarrowbtn,cancelbtn;  
  
 public MainPageMenubarView() {  
  
 settingsbtn = new Button("Settings");  
 logbtn = new Button("Log out");  
  
 logbtn.setOnAction(e -> setOnActionsForButtons(e));  
 settingsbtn.setOnAction(e -> setOnActionsForButtons(e));  
  
 userlbl = new Label("User id: " + Controller.*getCurrentUser*().getUserId() + " - " + Controller.*getCurrentUser*().getUserName());  
  
 leftarrowbtn = new Button("<--");  
 neworderbtn = new Button("New Order");  
 rightarrowbtn = new Button("-->");  
 cancelbtn = new Button("Cancel");  
  
 leftarrowbtn.setOnAction(e -> setOnActionsForButtons(e));  
 neworderbtn.setOnAction(e -> setOnActionsForButtons(e));  
 rightarrowbtn.setOnAction(e -> setOnActionsForButtons(e));  
 cancelbtn.setOnAction(e -> setOnActionsForButtons(e));  
  
  
 Region spacer = new Region();  
 HBox.*setHgrow*(spacer, Priority.*ALWAYS*);  
  
 Separator separatorMenuBar = new Separator();  
 separatorMenuBar.setOrientation(Orientation.*VERTICAL*);  
  
 menubar = new HBox(settingsbtn, logbtn, spacer, userlbl, separatorMenuBar, leftarrowbtn, neworderbtn, rightarrowbtn, cancelbtn);  
  
 HBox.*setMargin*(settingsbtn, new Insets(10));  
 HBox.*setMargin*(logbtn, new Insets(10));  
 HBox.*setMargin*(userlbl, new Insets(15, 10, 10, 10));  
  
 HBox.*setMargin*(leftarrowbtn, new Insets(10, 2, 10, 2));  
 HBox.*setMargin*(neworderbtn, new Insets(10, 0, 10, 0));  
 HBox.*setMargin*(rightarrowbtn, new Insets(10, 2, 10, 2));  
  
 HBox.*setMargin*(cancelbtn, new Insets(10));  
  
 }  
 private void setOnActionsForButtons(ActionEvent e) {  
 //handle the menubar buttons  
 if (e.getSource() == settingsbtn){  
 //System.out.println(Controller.getCurrentUser().getPrivilage().compareTo(Privilage.SuperVisor));  
 if (Controller.*getCurrentUser*().getPrivilage().compareTo(Privilage.*SuperVisor*) < 1 ) {  
 Controller.*switchToSettingsPage*();  
 }  
 }  
  
 else if (e.getSource() == logbtn)  
 Controller.*logout*();  
  
 //handle order buttons  
 else if (e.getSource() == leftarrowbtn)  
 BasketView.*prevBasket*();  
 else if (e.getSource() == neworderbtn)  
 BasketView.*newOrderToBasket*();  
 else if (e.getSource() == rightarrowbtn)  
 BasketView.*nextBasket*();  
 else if (e.getSource() == cancelbtn)  
 BasketView.*cancelCurrentOrder*();  
 }  
  
 public HBox getMenuBar() {  
 return menubar;  
 }  
  
}

CatAndItemsView.java

package Application.views;  
  
import java.util.ArrayList;  
  
import Application.modules.\*;  
  
import Application.controller.Controller;  
import javafx.event.ActionEvent;  
import javafx.geometry.Insets;  
import javafx.scene.control.Button;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.FlowPane;  
import javafx.scene.layout.HBox;  
  
public class CatAndItemsView {  
   
 private BorderPane itemsAndCatView;  
 private ArrayList<Button> catButtons;  
 private ArrayList<Button> itemButtons;  
 private ArrayList<Category> categories;  
 private int indexOfCurrentCategory;  
 private HBox cats;  
 private FlowPane items;  
   
 public CatAndItemsView() {  
 itemsAndCatView = new BorderPane();  
 cats = new HBox(10);  
 items = new FlowPane();  
 items.setPadding(new Insets(15));  
 items.setHgap(15);  
 items.setVgap(10);  
  
 initializeCategoriesButtons();  
 }  
   
 private void initializeCategoriesButtons() {  
 categories = Controller.*getCategories*();  
 catButtons = new ArrayList<Button>();  
  
 for(Category cat: categories) {  
 catButtons.add(new Button(cat.getName()));  
 catButtons.get(catButtons.size() -1).setOnAction(e -> handleButtonActions(e));  
 cats.getChildren().add(catButtons.get(catButtons.size() - 1));  
 }  
  
 itemsAndCatView.setTop(cats);  
 indexOfCurrentCategory = 0;  
 loadItemButtons(indexOfCurrentCategory);  
  
 }  
 private void loadItemButtons(int indexOfCategory) {  
 itemButtons = new ArrayList<Button>();  
  
 items = new FlowPane();  
 items.setPadding(new Insets(15));  
 items.setHgap(15);  
 items.setVgap(10);  
  
 for(Item item: categories.get(indexOfCategory).getItems()) {  
 itemButtons.add(new Button(item.toString()));  
 itemButtons.get(itemButtons.size() -1).setOnAction(e -> handleButtonActions(e));  
 items.getChildren().add(itemButtons.get(itemButtons.size() -1));  
 }  
  
 itemsAndCatView.setCenter(items);  
 }  
 private void handleButtonActions(ActionEvent e) {  
 if (catButtons.contains(e.getSource())){  
 for(Button catBtn: catButtons){  
 if(catBtn == e.getSource()){  
 indexOfCurrentCategory = catButtons.indexOf(catBtn);  
 loadItemButtons(indexOfCurrentCategory);  
 break;  
 }  
 }  
 }  
  
 else if (itemButtons.contains(e.getSource())){  
 BasketView.*addItem*(categories.get(indexOfCurrentCategory).getItems().get(itemButtons.indexOf(e.getSource())));  
 }  
  
 }  
   
 public BorderPane getItemsAndCatView() {  
 return itemsAndCatView;  
 }  
  
}

BasketView.java

package Application.views;  
  
import Application.controller.Controller;  
import Application.modules.Basket;  
import Application.modules.Item;  
import javafx.event.ActionEvent;  
import javafx.geometry.Insets;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.ListView;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Priority;  
import javafx.scene.layout.Region;  
import javafx.scene.layout.VBox;  
import javafx.scene.text.Font;  
  
import java.util.ArrayList;  
  
public class BasketView {  
   
 private static VBox *basketView*;  
 private static HBox *hbox*;  
 private static ListView<String> *listViewOfItems*;  
 private static ArrayList<Basket> *baskets*;  
 private static Basket *currentBasket*;  
 private static Label *basketLabel*, *subTotalLabel*, *taxLabel*, *totalLabel*;  
 private static Button *deletebtn*, *paybtn*;  
   
 public BasketView() {  
 //initialization of nodes  
 *listViewOfItems* = new ListView<String>();  
 *listViewOfItems*.setPrefHeight(2000); //find a better way to do this.  
  
 *basketLabel* = new Label("Basket");  
 *basketLabel*.setFont(new Font(30));  
 *subTotalLabel* = new Label("Subtotal:‬");  
 *taxLabel* = new Label("Tax:‬");  
 *totalLabel* = new Label("Total:");  
  
 *deletebtn* = new Button("Del");  
 *paybtn* = new Button("Pay");  
 *paybtn*.setMinSize(250, 50);  
  
 *deletebtn*.setOnAction(e -> *setOnActionsForButtons*(e));  
 *paybtn*.setOnAction(e -> *setOnActionsForButtons*(e));  
  
 *baskets* = Controller.*getBaskets*();  
 *currentBasket* = null;  
 *newOrderToBasket*();  
 //--------------------------------  
  
  
 // layout of basket  
 *basketView* = new VBox(20);  
 *basketView*.setPadding(new Insets(20));  
 //basketView.setAlignment(Pos.TOP\_RIGHT);  
 Region spacer = new Region();  
 VBox.*setVgrow*(spacer, Priority.*ALWAYS*);  
  
 *hbox* = new HBox();  
 HBox.*setMargin*(*basketLabel*, new Insets(10));  
 HBox.*setMargin*(*deletebtn*, new Insets(10));  
 *hbox*.getChildren().addAll(*basketLabel*, *deletebtn*);  
 *basketView*.getChildren().addAll(*hbox*,*listViewOfItems*,*subTotalLabel*,*taxLabel*, *totalLabel*, *paybtn*);  
 //------------------------------  
 }  
  
 private static void setOnActionsForButtons(ActionEvent e) {  
 if(e.getSource() == *paybtn*){  
 *currentBasket*.finalizeOrder();  
 } else {  
 *removeItem*();  
 }  
 }  
  
 public static void addItem(Item item) {  
 if(*currentBasket*.isOrderEditable()) {  
 *currentBasket*.addItemToOrder(item);  
 *refreshBasket*();  
 }  
  
 }  
 public static void removeItem(/\*Item item\*/) {  
 if(*currentBasket*.isOrderEditable()) {  
 if (*listViewOfItems*.getSelectionModel().getSelectedIndex() != -1) {  
 *currentBasket*.removeItemInOrder(*listViewOfItems*.getSelectionModel().getSelectedIndex());  
 *listViewOfItems*.getItems().remove(*listViewOfItems*.getSelectionModel().getSelectedItem());  
 *listViewOfItems*.getSelectionModel().clearSelection();  
 }  
 *refreshBasket*();  
 }  
 }  
 public VBox getBasketView() {  
 return *basketView*;  
 }  
  
 public static void nextBasket() {  
 int i = *baskets*.indexOf(*currentBasket*);  
 if (i < *baskets*.size() - 1) {  
 *currentBasket* = *baskets*.get(i + 1);  
 *refreshBasket*();  
 System.*out*.println("next Order");  
 }  
 }  
 public static void prevBasket() {  
 int i = *baskets*.indexOf(*currentBasket*);  
 if (0 < i) {  
 *currentBasket* = *baskets*.get(i - 1);  
 *refreshBasket*();  
 System.*out*.println("Prev Order");  
 }  
 }  
 public static void newOrderToBasket(){  
 *baskets*.add(new Basket(Controller.*getCurrentUser*()));  
 *currentBasket* = *baskets*.get(*baskets*.size() - 1);  
 *refreshBasket*();  
 System.*out*.println("New Order Made");  
 }  
 public static void cancelCurrentOrder(){ // there is a bug here that would couse a "payed for" order to be deleted. -- fix it  
 if (!*baskets*.isEmpty()){  
 if(*currentBasket*.isOrderEditable()) {  
 *baskets*.remove(*currentBasket*);  
 if (!*baskets*.isEmpty()) {  
 *currentBasket* = *baskets*.get(*baskets*.size() - 1);  
 *refreshBasket*();  
 } else  
 *newOrderToBasket*();  
 System.*out*.println("Order Canceled");  
 }  
 }  
 }  
  
 private static void refreshBasket() {  
 *listViewOfItems*.getItems().clear();  
 for(Item item: *currentBasket*.getItemsInOrder())  
 *listViewOfItems*.getItems().add(item.getNameOfItem() + " " + item.getPrice());  
  
 *subTotalLabel*.setText(String.*format*("%-32s $%.2f", "Subtotal:",*currentBasket*.getSubTotal()));  
 *taxLabel*.setText(String.*format*("%-32s $%.2f", "Tax:", *currentBasket*.getTax()));  
 *totalLabel*.setText(String.*format*("%-32s $%.2f", "Total:", *currentBasket*.getTotal()));  
 }  
}

settingsPageView.java

package Application.views;  
  
import javafx.geometry.Orientation;  
import javafx.scene.Scene;  
import javafx.scene.control.Separator;  
import javafx.scene.layout.BorderPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Priority;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
  
public class SettingsPageView {  
 private static VBox *topAndBottomOfMain*;  
 private static HBox *bottomRightAndLeftSide*;  
  
 private static BorderPane *itemsAndCatview*;  
 private static VBox *itemsCreatorView*;  
 private static HBox *menubarView*;  
  
 private static Separator *separatorTopButton*, *separatorButtomParts*;  
  
 public static void start(Stage primaryStage) {  
 // The general layout of the program  
 *topAndBottomOfMain* = new VBox();  
 *bottomRightAndLeftSide* = new HBox();  
 //-----------------------------------  
  
  
 // the elements of the page  
 *itemsAndCatview* = new CatAndItemsView().getItemsAndCatView();  
 *itemsCreatorView* = new ItemsCreatorAndModifierView().*getView*();  
 *menubarView* = new SettingsPageMenuBarView().getView();  
 //-----------------------------------  
  
 // add all together  
 *separatorTopButton* = new Separator();  
 *separatorButtomParts* = new Separator();  
 *separatorButtomParts*.setOrientation(Orientation.*VERTICAL*);  
  
 HBox.*setHgrow*(*itemsAndCatview*, Priority.*ALWAYS*);  
  
 *bottomRightAndLeftSide*.getChildren().addAll(*itemsAndCatview*, *separatorButtomParts*, *itemsCreatorView*);  
 *topAndBottomOfMain*.getChildren().addAll(*menubarView*, *separatorTopButton*, *bottomRightAndLeftSide*);  
 //----------------------------------  
  
  
 // Display it all  
 Scene scene = new Scene(*topAndBottomOfMain*, 1050, 700);  
 primaryStage.setMinWidth(720);  
 primaryStage.setMinHeight(500);  
 primaryStage.setScene(scene);  
 primaryStage.sizeToScene();  
 primaryStage.setTitle("Burpien - Settings Page");  
 primaryStage.show();  
 //----------------------------------  
 }  
 public static void updateUI(){  
 *itemsAndCatview* = new CatAndItemsView().getItemsAndCatView();  
  
 *bottomRightAndLeftSide*.getChildren().clear();  
 *topAndBottomOfMain*.getChildren().clear();  
  
 *bottomRightAndLeftSide*.getChildren().addAll(*itemsAndCatview*, *separatorButtomParts*, *itemsCreatorView*);  
 *topAndBottomOfMain*.getChildren().addAll(*menubarView*, *separatorTopButton*, *bottomRightAndLeftSide*);  
 }  
}

SettingsPageMenuBarView.java

package Application.views;  
  
import Application.controller.Controller;  
import javafx.event.ActionEvent;  
import javafx.geometry.Insets;  
import javafx.geometry.Orientation;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.control.Separator;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Priority;  
import javafx.scene.layout.Region;  
import javafx.application.Application;  
import javafx.stage.Stage;  
  
public class SettingsPageMenuBarView {  
  
 private HBox menubar;  
 private Button donebtn, loadbtn, cancelbtn;  
 private Label userlbl;  
  
 public SettingsPageMenuBarView() {  
  
 donebtn = new Button("Done");  
 donebtn.setOnAction(e -> handleButtonClick(e));  
 //loadbtn = new Button("Load");  
 //cancelbtn = new Button("Cancel");  
  
 userlbl = new Label("User id: " + Controller.*getCurrentUser*().getUserId() + " - "  
 + Controller.*getCurrentUser*().getUserName());  
  
  
 Region spacer = new Region();  
 HBox.*setHgrow*(spacer, Priority.*ALWAYS*);  
  
 Separator separatorMenuBar = new Separator();  
 separatorMenuBar.setOrientation(Orientation.*VERTICAL*);  
  
 menubar = new HBox(donebtn, spacer, userlbl, separatorMenuBar);  
  
 HBox.*setMargin*(donebtn, new Insets(5));  
 //HBox.setMargin(loadbtn, new Insets(5));  
 //HBox.setMargin(cancelbtn, new Insets(5));  
 HBox.*setMargin*(userlbl, new Insets(15, 10, 10, 10));  
 }  
  
  
 public HBox getView() {  
 return menubar;  
 }  
  
 private void handleButtonClick(ActionEvent e) {  
 if (e.getSource() == donebtn)  
 Controller.*switchToMainPage*();  
 }  
  
}

ItemsCreatorAndModifierView.java

package Application.views;  
  
import Application.controller.Controller;  
import Application.modules.Category;  
import Application.modules.PriceType;  
import Application.modules.Privilage;  
import javafx.collections.FXCollections;  
import javafx.event.ActionEvent;  
import javafx.geometry.Insets;  
import javafx.scene.control.\*;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Priority;  
import javafx.scene.layout.Region;  
import javafx.scene.layout.VBox;  
  
import java.util.ArrayList;  
  
public class ItemsCreatorAndModifierView {  
 private static VBox *itemCreatorView*;  
 private static HBox *topButtonsHBox*, *itemTypeHBox*, *categoriesHBox*, *priceHBox*, *barcodeHBox*, *createButtonHBox*, *saveDeleteHBox*, *itemNameHBox*;  
 private static Label *infollbl*,*typelbl*,*namelbl*,*catlbl*,*prclbl*, *bylbl*, *barcodelbl*;  
 private static Button *newbtn*, *infobtn*, *createbtn*, *savebtn*,*delbtn*;  
 private static ChoiceBox *itemTypeChoiceBox*, *categoriesChoiceBox*, *itemPriceTypeChoiceBox*;  
 private static TextField *nameTextField*, *priceTextField*, *barcodeTextField*;  
 private static ArrayList<Category> *categories*;  
  
 public ItemsCreatorAndModifierView() {  
 initializeClassObjectsOnce();  
  
 itemsCreatorPart1();  
 //itemsCreatorPart2();  
 //itemsCreatorButtons();  
 //itemsModifierButtons();  
 }  
  
 private void initializeClassObjectsOnce(){  
 *infollbl* = new Label("Items List and Information");  
 *newbtn* = new Button("New");  
 *infobtn* = new Button("Info");  
 *typelbl* = new Label("Type: ");  
 *namelbl*= new Label("Name: ");  
 *createbtn* = new Button("Create");  
  
 *catlbl* = new Label("Category: ");  
 *prclbl* = new Label("Price: ");  
 *bylbl* = new Label("By: ");  
 *barcodelbl* = new Label("Barcode ID: ");  
  
 *nameTextField* = new TextField();  
 *nameTextField*.setPrefWidth(220);  
 *priceTextField* = new TextField();  
 *priceTextField*.setMaxWidth(90);  
 *barcodeTextField* = new TextField();  
 *barcodeTextField*.setPrefWidth(200);  
  
 *savebtn* = new Button("Save");  
 *delbtn* = new Button("Delete");  
  
 *categories* = Controller.*getCategories*();  
  
 *newbtn*.setOnAction(e -> handleButtonClicks(e));  
 *infobtn*.setOnAction(e -> handleButtonClicks(e));  
 *createbtn*.setOnAction(e -> handleButtonClicks(e));  
 *savebtn*.setOnAction(e -> handleButtonClicks(e));  
 *delbtn*.setOnAction(e -> handleButtonClicks(e));  
  
 *itemCreatorView* = new VBox(20);  
 *itemCreatorView*.setPadding(new Insets(20));  
 //basketView.setAlignment(Pos.TOP\_RIGHT);  
 *itemCreatorView*.setPrefHeight(2000);  
 *itemCreatorView*.setMinWidth(330);  
 *itemCreatorView*.setMaxWidth(330);  
  
 *topButtonsHBox* = new HBox();  
 HBox.*setMargin*(*infollbl*, new Insets(5));  
 HBox.*setMargin*(*newbtn*, new Insets(5));  
 HBox.*setMargin*(*infobtn*, new Insets(5));  
 *topButtonsHBox*.getChildren().addAll(*newbtn*/\*, infobtn\*/);  
  
 *itemTypeChoiceBox* = new ChoiceBox();  
 //itemTypeChoiceBox.setItems(FXCollections.observableArrayList("Category", /\*new Separator(),\*/ "Item"));  
 *itemTypeChoiceBox*.getItems().addAll("Category", /\*new Separator(),\*/ "Item");  
 *itemTypeChoiceBox*.getSelectionModel().selectedIndexProperty().addListener(  
 (observableValue, number, number2) -> {  
 //System.out.println(itemTypeChoiceBox.getItems().get((Integer) number2));  
 handleChangeInCategory((Integer) number2 );  
 }  
 );  
  
 *itemTypeHBox* = new HBox();  
 HBox.*setMargin*(*typelbl*, new Insets(5));  
 HBox.*setMargin*(*itemTypeChoiceBox*, new Insets(5));  
 *itemTypeHBox*.getChildren().addAll(*typelbl*, *itemTypeChoiceBox*);  
  
 *itemNameHBox* = new HBox();  
 HBox.*setMargin*(*namelbl*, new Insets(5));  
 HBox.*setMargin*(*nameTextField*, new Insets(5));  
 *itemNameHBox*.getChildren().addAll(*namelbl*, *nameTextField*);  
  
 *categoriesChoiceBox* = new ChoiceBox();  
 *categoriesChoiceBox*.setPrefWidth(200);  
 //categoriesChoiceBox.setItems(FXCollections.observableArrayList("Cat 1", /\*new Separator(),\*/ "Cat 2", /\*new Separator(),\*/ "Etc"));  
 for (Category cat: *categories*)  
 *categoriesChoiceBox*.getItems().add(cat.getName());  
  
 *categoriesHBox* = new HBox();  
 HBox.*setMargin*(*catlbl*, new Insets(5));  
 HBox.*setMargin*(*categoriesChoiceBox*, new Insets(5));  
 *categoriesHBox*.getChildren().addAll(*catlbl*, *categoriesChoiceBox*);  
  
 *itemPriceTypeChoiceBox* = new ChoiceBox();  
 *itemPriceTypeChoiceBox*.setItems(FXCollections.*observableArrayList*("Unit", /\*new Separator(),\*/ "Weight (1 lb)"));  
  
 *priceHBox* = new HBox();  
 HBox.*setMargin*(*prclbl*, new Insets(5));  
 HBox.*setMargin*(*priceTextField*, new Insets(5));  
 HBox.*setMargin*(*bylbl*, new Insets(5));  
 HBox.*setMargin*(*itemPriceTypeChoiceBox*, new Insets(5));  
 *priceHBox*.getChildren().addAll(*prclbl*, *priceTextField*,*bylbl*, *itemPriceTypeChoiceBox*);  
  
 *barcodeHBox* = new HBox();  
 HBox.*setMargin*(*barcodelbl*, new Insets(5));  
 HBox.*setMargin*(*barcodeTextField*, new Insets(5));  
 *barcodeHBox*.getChildren().addAll(*barcodelbl*, *barcodeTextField*);  
  
 *createButtonHBox* = new HBox();  
 HBox.*setHgrow*(*createbtn*, Priority.*ALWAYS*);  
 *createbtn*.setMaxWidth(Double.*MAX\_VALUE*);  
 HBox.*setMargin*(*createbtn*, new Insets(5));  
 *createButtonHBox*.getChildren().addAll(*createbtn*);  
  
 Region spacer = new Region();  
 HBox.*setHgrow*(spacer, Priority.*ALWAYS*);  
  
 *saveDeleteHBox* = new HBox();  
 HBox.*setMargin*(*savebtn*, new Insets(10));  
 HBox.*setMargin*(*delbtn*, new Insets(10));  
 *saveDeleteHBox*.getChildren().addAll(*savebtn*, spacer, *delbtn*);  
 }  
  
 private void itemsCreatorPart1(){  
 *itemCreatorView*.getChildren().clear();  
 *itemCreatorView*.getChildren().addAll(*infollbl*, *topButtonsHBox*, *itemTypeHBox*);  
 }  
 private void itemsCreatorPart2(){  
 *itemCreatorView*.getChildren().addAll(*itemNameHBox*);  
 }  
 private void itemsCreatorPart3(){  
 *itemCreatorView*.getChildren().addAll(*categoriesHBox*, *priceHBox*, *barcodeHBox*);  
 }  
 private void itemsCreatorButtons(){  
 Region spacer = new Region();  
 VBox.*setVgrow*(spacer, Priority.*ALWAYS*);  
  
 *itemCreatorView*.getChildren().addAll(spacer, *createButtonHBox*);  
 }  
 private void itemsModifierButtons(){  
 Region spacer = new Region();  
 VBox.*setVgrow*(spacer, Priority.*ALWAYS*);  
  
 *itemCreatorView*.getChildren().addAll(spacer, *saveDeleteHBox*);  
 }  
  
 private void handleButtonClicks(ActionEvent e) {  
 if (e.getSource() == *newbtn*){  
 itemsCreatorPart1();  
 //itemsCreatorPart2();  
 //itemsCreatorButtons();  
 }  
 else if (e.getSource() == *infobtn*){  
 itemsCreatorPart1();  
 //itemsCreatorPart2();  
 itemsModifierButtons();  
 }  
 else if (e.getSource() == *createbtn*){  
 createItem();  
 }  
 else if (e.getSource() == *savebtn*){  
  
 }  
 else if (e.getSource() == *delbtn*){  
  
 }  
 }  
 private void handleChangeInCategory(int i) {  
 if (i == 0){  
 itemsCreatorPart1();  
 itemsCreatorPart2();  
 itemsCreatorButtons();  
 }  
 else if (i == 1){  
 itemsCreatorPart1();  
 itemsCreatorPart2();  
 itemsCreatorPart3();  
 itemsCreatorButtons();  
 }  
 }  
  
 private void createItem(){  
 if (*itemTypeChoiceBox*.getSelectionModel().getSelectedIndex() == 0){  
 if (!*nameTextField*.getText().isEmpty()) {  
 *categories*.add( new Category(*nameTextField*.getText()));  
 *nameTextField*.setText("");  
  
 Controller.*switchToSettingsPage*();  
 //SettingsPageView.updateUI();  
 System.*out*.println("new category created");  
 }  
 }  
 else if (*itemTypeChoiceBox*.getSelectionModel().getSelectedIndex() == 1){  
 if (!*nameTextField*.getText().isEmpty() &&  
 !*priceTextField*.getText().isEmpty() &&  
 !*barcodeTextField*.getText().isEmpty() &&  
 *categoriesChoiceBox*.getSelectionModel().getSelectedIndex() != -1 &&  
 *itemPriceTypeChoiceBox*.getSelectionModel().getSelectedIndex() != -1) {  
  
 PriceType pt;  
 if (*itemPriceTypeChoiceBox*.getSelectionModel().getSelectedIndex() == 0)  
 pt = PriceType.*Unit*;  
 else  
 pt = PriceType.*Weight*;  
  
 *categories*.get(*categoriesChoiceBox*.getSelectionModel().getSelectedIndex()).additem(*nameTextField*.getText(), Double.*parseDouble*(*priceTextField*.getText()), pt , Integer.*parseInt*(*barcodeTextField*.getText()));  
  
 Controller.*switchToSettingsPage*();  
 //SettingsPageView.updateUI();  
  
 System.*out*.println("new item created");  
 // reset all  
 *nameTextField*.setText("");  
 *priceTextField*.setText("");  
 *barcodeTextField*.setText("");  
 *categoriesChoiceBox*.getSelectionModel().clearSelection();  
 *itemPriceTypeChoiceBox*.getSelectionModel().clearSelection();  
 }  
 }  
 }  
 private void modifyItem() {}  
 private void deleteItem() {}  
  
 public static VBox getView(){  
 return *itemCreatorView*;  
 }  
  
}