**Assign2Controller.java**

package patkunja;

import javafx.application.Platform;

import javafx.fxml.FXML;

import javafx.scene.control.\*;

import javafx.scene.paint.Color;

import javax.swing.\*;

public class Assign2Controller {

// inventory list class object to perform operations

private InventoryList \_ivList = new InventoryList();

@FXML

public TextField txtItemId; // item id text field reference

@FXML

public TextField txtItemName; // item name text field reference

@FXML

public TextField txtQOH; // item quantity-on-hand text field reference

@FXML

public TextField txtROP; // item re-order-point text field reference

@FXML

public TextField txtSellingPrice; // item selling price text field reference

@FXML

public TextArea txtAreaResult; // result area field reference

@FXML

public Label lblError; // error message label reference

@FXML

public Button btnAdd; // add button reference

@FXML

public Button btnSave; // save button reference

@FXML

public Button btnOrders; // orders button reference

@FXML

public Button btnExit; // application exit button reference

/\*\*

\* Main initialisation method for this controller

\* It set all action for javafx application controls

\*/

@FXML

private void initialize() {

btnAdd.setOnAction(e -> onbtnAddClicked()); // add button event binder

btnSave.setOnAction(e -> onbtnSaveClicked()); // save item button event binder

btnOrders.setOnAction(e -> onbtnOrdersClicked()); // orders item button event binder

btnExit.setOnAction(e -> onbtnExitClicked()); // exit application button event binder

}

/\*\*

\* clear all form fields and reset for new item to insert

\*/

private void onbtnAddClicked() {

clearForm();

}

/\*\*

\* Add new item to inventory list if the data is valid

\*/

private void onbtnSaveClicked() {

try {

Inventory \_iv = new Inventory(); // create new inventory object to insert data

\_iv.setId(txtItemId.getText()); // set item id to inventory object

\_iv.setName(txtItemName.getText()); // set item name to inventory object

String qoh = txtQOH.getText(); // read QOH value from form

if (qoh.isEmpty() || hasInteger(qoh)) { // Check and throw error if QOH is invalid

throw new IllegalArgumentException("Error: Invalid Q-O-H value");

}

\_iv.setQoh(Integer.parseInt(qoh)); // set QOH to inventory object

String rop = txtROP.getText(); // read ROP value from form

if (rop.isEmpty() || hasInteger(rop)) { // Check and throw error if ROP is invalid

throw new IllegalArgumentException("Error: Invalid R-O-P value");

}

\_iv.setRop(Integer.parseInt(rop)); // set ROP to inventory object

String sellingPrice = txtSellingPrice.getText(); // read selling price value from form

if (sellingPrice.isEmpty() || !hasDouble(sellingPrice)) { // Check and throw error if selling price is invalid

throw new IllegalArgumentException("Error: Invalid Selling price");

}

\_iv.setSellPrice(Double.parseDouble(sellingPrice)); // set selling price to inventory object

\_ivList.add(\_iv); // Save new inventory to inventory list

clearForm(); // reset form data

// Display success message to user

lblError.setTextFill(Color.web("#28a745"));

lblError.setText("Item added successfully");

} catch (Exception ex) {

// clear label message

lblError.setText(null);

String message = "ERROR: Invalid form details"; // default error message

if (ex instanceof IllegalArgumentException) { // check if error is coming from inventory class and set message

message = ex.getMessage();

}

// Display error dialog for user

JOptionPane.showMessageDialog(null, message, "Error", JOptionPane.ERROR\_MESSAGE);

}

}

/\*\*

\* check list of items which are needed to reorder

\*/

private void onbtnOrdersClicked() {

clearForm();

int size = \_ivList.length(); // get size of current inventory list

if (size <= 0) { // check if there is no item in list and display error message

lblError.setTextFill(Color.web("#eb0000"));

lblError.setText("No item to list. Add some!");

return;

}

Inventory \_iv; // inventory object reference

StringBuilder items = new StringBuilder(); // create string to display list of items in text area

for (int i = 0; i < size; i++) {

\_iv = \_ivList.get(i); // get each inventory object from list

if (\_iv.getQoh() <= \_iv.getRop()) { // check if item needs to be re-order and add in list

items.append(\_iv.toString());

}

}

if (items.toString().equals("")) { // display message if no item needs to be reorder

txtAreaResult.setText("No items to re-order.");

return;

}

txtAreaResult.setText(items.toString()); // display list of re-order items in text area

}

/\*\*

\* exit application button handler

\*/

private void onbtnExitClicked() {

// ask user if they wants to quit or not

int choice = JOptionPane.showConfirmDialog(null, "Are you sure you want to Quit?", "Error", JOptionPane.OK\_CANCEL\_OPTION);

if (choice == 0) { // close application if user confirm

Platform.exit();

}

}

/\*\*

\* Clear all form detail and label messages

\*/

private void clearForm() {

txtItemId.setText(null); // clear item id field

txtItemName.setText(null); // clear item name field

txtQOH.setText(null); // clear item QOH field

txtROP.setText(null); // clear item ROP field

txtSellingPrice.setText(null); // clear item selling price field

txtAreaResult.setText(null); // clear result area

lblError.setText(null); // clear label message

}

/\*\*

\* Function check if input is valid string to convert into number or not and return boolean value

\*

\* @param input input string to check if its number(integer) or not

\* @return boolean

\*/

private boolean hasInteger(String input) {

try {

Integer.parseInt(input);

} catch (NumberFormatException ex) {

return true;

}

return false;

}

/\*\*

\* Function check if input is valid string to convert into number or not and return boolean value

\*

\* @param input input string to check if its number(double) or not

\* @return boolean

\*/

private boolean hasDouble(String input) {

try {

Double.parseDouble(input);

} catch (NumberFormatException ex) {

return false;

}

return true;

}

}

**Assign2Main.java**

package patkunja;

import javafx.application.Application;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

public class Assign2Main extends Application {

/\*\*

\* @param primaryStage set new stage window for application window

\* @throws Exception return any exception occur in application launch

\*/

@Override

public void start(Stage primaryStage) throws Exception {

// get fxml file for main window

Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));

// Set title for window

primaryStage.setTitle("Inventory Tracker");

// set window size and stage scene

primaryStage.setScene(new Scene(root, 360, 600));

// display application window

primaryStage.show();

}

/\*\*

\* @param args application input arguments

\*/

public static void main(String[] args) {

launch(args);

}

}

**Inventory.java**

package patkunja;

import java.util.regex.Pattern;

public class Inventory {

// Class variables

private String id;

private String name;

private int qoh; // Quantity On Hand

private int rop; // Re-Order Point

private double sellPrice;

// Default constructor

// Set all default values to inventory object

Inventory() {

this.setId("ABC-1234");

this.setName("New Item");

this.setQoh(0);

this.setRop(25);

this.setSellPrice(0);

}

/\*\*

\* Parameterised constructor

\*

\* @param id inventory if

\* @param name inventory name

\* @param sellPrice inventory selling price

\*/

Inventory(String id, String name, double sellPrice) {

this.setId(id);

this.setName(name);

this.setSellPrice(sellPrice);

}

/\*\*

\* Parameterised constructor

\*

\* @param id inventory id

\* @param name inventory name

\* @param qoh inventory quantity-on-hand

\* @param rop inventory re-order-point

\* @param sellPrice inventory selling price

\*/

Inventory(String id, String name, int qoh, int rop, double sellPrice) {

this.setId(id);

this.setName(name);

this.setQoh(qoh);

this.setRop(rop);

this.setSellPrice(sellPrice);

}

/\*\*

\* Get current item id

\*

\* @return id inventory id

\*/

public String getId() {

return id;

}

/\*\*

\* Set new item id

\*

\* @param id new inventory id

\*/

public void setId(String id) {

Pattern pattern = Pattern.compile("[a-zA-Z]{3}[-][0-9]{4}");

if (!pattern.matcher(id).matches()) {

throw new IllegalArgumentException("ERROR: Invalid Id, expected format : ABC-1234");

}

this.id = id;

}

/\*\*

\* Get item name

\*

\* @return name

\*/

public String getName() {

return name;

}

/\*\*

\* Set new item name

\*

\* @param name inventory name

\*/

public void setName(String name) {

if (name == null || name.trim().isEmpty()) {

throw new IllegalArgumentException("ERROR: You must enter an item name.");

}

this.name = name;

}

/\*\*

\* get quantity on hand

\*

\* @return qoh

\*/

public int getQoh() {

return qoh;

}

/\*\*

\* Set new quantity on hand value

\*

\* @param qoh inventory quantity-on-hand

\*/

public void setQoh(int qoh) {

if (qoh < 0) {

throw new IllegalArgumentException("ERROR: Quantity On Hand must be 0 or more.");

}

this.qoh = qoh;

}

/\*\*

\* Get re-order point value

\*

\* @return rop

\*/

public int getRop() {

return rop;

}

/\*\*

\* Set new re-order point

\*

\* @param rop inventory re-order-point

\*/

public void setRop(int rop) {

if (rop < 1) {

throw new IllegalArgumentException("ERROR: Re-Order Point must be greater then 0.");

}

this.rop = rop;

}

/\*\*

\* get current item selling price

\*

\* @return sellPrice

\*/

public double getSellPrice() {

return sellPrice;

}

/\*\*

\* Set current item new selling price

\*

\* @param sellPrice inventory selling price

\*/

public void setSellPrice(double sellPrice) {

if (sellPrice < 0) {

throw new IllegalArgumentException("ERROR: Selling price nust be greater then 0.");

}

this.sellPrice = sellPrice;

}

/\*\*

\* print formatted item information

\*/

@Override

public String toString() {

return String.format("\n%s (%s), QOH: %d Price: $%.2f\n", this.id, this.name, this.qoh, this.sellPrice);

}

}

**InventoryList.java**

package patkunja;

import java.util.ArrayList;

public class InventoryList {

private ArrayList<Inventory> invList = new ArrayList<>(0); // Inventory list

/\*\*

\* Add new inventory into list

\*

\* @param inventory object of inventory item

\*/

public void add(Inventory inventory) {

invList.add(inventory);

}

/\*\*

\* get inventory item from specific index

\*

\* @param index position in inventory list

\* @return Inventory

\*/

public Inventory get(int index) {

if (index >= length()) {

return null;

}

return invList.get(index);

}

/\*\*

\* get inventory list size

\*

\* @return size

\*/

public int length() {

return invList.size();

}

}

**Sample.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.ColumnConstraints?>

<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.RowConstraints?>

<?import javafx.scene.text.Font?>

<GridPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0"

prefWidth="360.0" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1"

fx:controller="patkunja.Assign2Controller">

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"/>

</columnConstraints>

<rowConstraints>

<RowConstraints maxHeight="361.0" minHeight="0.0" prefHeight="56.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="439.0" minHeight="0.0" prefHeight="47.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="425.0" minHeight="10.0" prefHeight="44.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="440.0" minHeight="0.0" prefHeight="41.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="440.0" minHeight="0.0" prefHeight="46.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="440.0" minHeight="0.0" prefHeight="68.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="440.0" minHeight="10.0" prefHeight="212.0" vgrow="SOMETIMES"/>

<RowConstraints maxHeight="440.0" minHeight="10.0" prefHeight="48.0" vgrow="SOMETIMES"/>

</rowConstraints>

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</padding>

<children>

<!-- START title -->

<HBox fx:id="hbox" alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"

minWidth="-Infinity" prefHeight="50.0" prefWidth="343.0" GridPane.halignment="CENTER"

GridPane.hgrow="ALWAYS" GridPane.valignment="CENTER" GridPane.vgrow="ALWAYS">

<children>

<Label alignment="CENTER" contentDisplay="CENTER" prefHeight="26.0" prefWidth="261.0"

text="Inventory Tracker" textFill="LIME">

<font>

<Font name="Courier New" size="20.0"/>

</font>

</Label>

</children>

</HBox>

<!-- END title -->

<!-- START item id -->

<HBox minHeight="-Infinity" minWidth="-Infinity" prefHeight="37.0" prefWidth="343.0" spacing="10.0"

GridPane.rowIndex="1">

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</padding>

<children>

<Label prefHeight="28.0" prefWidth="70.0" text=" Item ID:" textFill="#004dff"/>

<TextField fx:id="txtItemId" prefHeight="27.0" prefWidth="132.0" promptText="Item ID">

<font>

<Font size="11.0"/>

</font>

</TextField>

</children>

</HBox>

<!-- END item id -->

<!-- START item name -->

<HBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="53.0"

prefWidth="343.0" spacing="10.0" GridPane.rowIndex="2">

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</padding>

<children>

<Label prefHeight="33.0" prefWidth="71.0" text="Item Name:" textAlignment="RIGHT" textFill="#004dff"/>

<TextField fx:id="txtItemName" prefHeight="27.0" prefWidth="240.0" promptText="Item Name">

<font>

<Font size="11.0"/>

</font>

</TextField>

</children>

</HBox>

<!-- END item name -->

<!-- START item QOH and ROP -->

<HBox alignment="CENTER" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"

prefHeight="48.0" prefWidth="343.0" spacing="10.0" GridPane.rowIndex="3">

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</padding>

<children>

<!-- START item QOH -->

<Label prefHeight="17.0" prefWidth="54.0" text=" Q-O-H:" textFill="#004dff"/>

<TextField fx:id="txtQOH" prefHeight="27.0" prefWidth="92.0" promptText="Qnty Hand">

<font>

<Font size="10.0"/>

</font>

</TextField>

<!-- END item QOH -->

<!-- START item ROP -->

<Label text="R-O-P:" textFill="#004dff"/>

<TextField fx:id="txtROP" prefHeight="27.0" prefWidth="89.0" promptText="Re-Order Point">

<font>

<Font size="11.0"/>

</font>

</TextField>

<!-- END item ROP -->

</children>

</HBox>

<!-- END item QOH and ROP -->

<!-- START item selling price -->

<HBox prefHeight="59.0" prefWidth="342.0" spacing="10.0" GridPane.rowIndex="4">

<padding>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</padding>

<children>

<Label prefHeight="27.0" prefWidth="71.0" text=" Sell Price:" textFill="#004dff"/>

<TextField fx:id="txtSellingPrice" prefHeight="27.0" prefWidth="240.0" promptText="Sell Price"/>

</children>

</HBox>

<!-- END item selling price -->

<!-- START function buttons -->

<HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0" spacing="30.0" GridPane.rowIndex="5">

<GridPane.margin>

<Insets/>

</GridPane.margin>

<padding>

<Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>

</padding>

<children>

<Button fx:id="btnAdd" text="\_Add"/> <!-- clear from button -->

<Button fx:id="btnSave" text="\_Save"/> <!-- save item button -->

<Button fx:id="btnOrders" text="\_Orders"/> <!-- show orders button -->

<Button fx:id="btnExit" text="E\_xit"/> <!-- exit application button -->

</children>

</HBox>

<!-- END function buttons -->

<!-- START result area -->

<TextArea fx:id="txtAreaResult" editable="false" prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="6">

<GridPane.margin>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</GridPane.margin>

</TextArea>

<!-- END item id -->

<!-- START message label -->

<Label fx:id="lblError" textFill="#eb0000" GridPane.rowIndex="7">

<GridPane.margin>

<Insets bottom="10.0" left="10.0" right="10.0" top="10.0"/>

</GridPane.margin>

</Label>

<!-- END message label -->

</children>

</GridPane>