import java.util.ArrayList;  
*/\*\* Author: Bryon Taylor  
 \* PID: A11895348  
 \*   
 \* This application simulates a slot machine. Good luck!  
 \*/*import java.util.List;  
import java.util.Optional;  
import java.io.File;  
import javafx.animation.PauseTransition;  
import javafx.application.Application;  
import javafx.geometry.HPos;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Group;  
import javafx.scene.Scene;  
import javafx.scene.control.Alert;  
import javafx.scene.control.Button;  
import javafx.scene.control.ChoiceDialog;  
import javafx.scene.control.TextField;  
import javafx.scene.control.Alert.AlertType;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.layout.GridPane;  
import javafx.scene.layout.HBox;  
import javafx.scene.layout.Pane;  
import javafx.scene.media.Media;  
import javafx.scene.media.MediaPlayer;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.scene.text.Text;  
import javafx.stage.Stage;  
import javafx.stage.StageStyle;  
import javafx.util.Duration;  
  
public class SlotMachine extends Application {  
   
 // scales the interface for different size screens *todo: scale entire application* private static final double *SCALAR* = .7;  
  
 // default width and height for game window  
 private static final int *WINDOW\_WIDTH* = 1100;  
 private static final int *WINDOW\_HEIGHT* = 1000;  
  
 // default sizes for fields and their fonts  
 private static final int *FONT\_SIZE\_FIELDS* = 30;  
 private static final int *FONT\_SIZE\_FIELD\_DESCRIPTION* = 35;  
 private static final int *FIELD\_WIDTH* = 300;  
 private static final int *FIELD\_HEIGHT* = 50;  
  
 // variables to manage bets and credits  
 private int bet = 1;  
 private int initialCredits = 0;  
 private int creditsPaid = 0;  
 private int creditsWon = 0;  
 private int updatedCredits = 0;  
 private int credits = 0;  
 private int maxBet = 5;  
 private int minCredits = 1;  
   
 // multipliers for winnings  
 private int multiplier1 = 1,  
 multiplier2 = 3,  
 multiplier3 = 5,  
 multiplier4 = 10,  
 multiplier5 = 15,  
 multiplier6 = 25,  
 multiplier7 = 50,  
 multiplier8 = 75,  
 jackpotMultiplier = 1000;  
  
 // media player to play sound *todo: only create singleton media player?* private MediaPlayer player; // *todo: make all vars private* // Array to hold dollar amounts. Amounts are displayed in the "add cash" dialog  
 private static final Integer[] *dollarAmounts* = {1, 5, 10, 20, 50, 100};  
  
 // create three reel objects // *todo: initialize reels in method* private Reel reel1 = new Reel();  
 private Reel reel2 = new Reel();  
 private Reel reel3 = new Reel();  
  
 // String representation of reel's value, i.e. "cherries" or "triple 7's"  
 private String reel1Value;  
 private String reel2Value;  
 private String reel3Value;  
  
 // UI components  
 private TextField creditsField,  
 betField,  
 paidField;  
  
 private Text creditsText,  
 betText,  
 paidText;  
  
 private Image imgStardust;  
 private ImageView stardustImgView;  
 private HBox hBoxTop;  
  
 // find user directory to create path for files  
 private String absolutePath = System.*getProperty*("user.dir");  
  
 public void start(Stage mainStage) {  
  
 initTopImage();  
 initTextFields();  
 initFieldDescriptions();  
  
 // initializes reels to all triple 7's image *TODO: add method* Pane firstReel = reel1.getDisplayReel();  
 Pane secondReel = reel2.getDisplayReel();  
 Pane thirdReel = reel3.getDisplayReel();  
  
 // 3 panes get animation of spinning reels // *TODO: add method* Pane firstAnimation = reel1.createAnimatedPane();  
 Pane secondAnimation = reel2.createAnimatedPane();  
 Pane thirdAnimation = reel3.createAnimatedPane();  
  
 */\*\*\*\*\*\*\* bottomButtons \*\*\*\*\*\*\*\*\*/* // create "ADD BET" button and set behavior  
 int doubleScale = (int) Math.*round*(2 \* *SCALAR*); // *todo: doubleScale vs quadScale for buttons* Button addBetButn = new Button("ADD BET");  
 addBetButn.setScaleX(doubleScale);  
 addBetButn.setScaleY(doubleScale);  
 addBetButn.setStyle("-fx-base: #14425a;");  
  
 addBetButn.setOnAction(e -> {  
 if(bet < maxBet){  
 bet++;  
 betField.setText(Integer.*toString*(bet));  
 String addSubCoins = "SMB\_Coin.mp3";  
 Media subAddCoins = new Media(new File(addSubCoins).toURI().toString());  
 player = new MediaPlayer(subAddCoins);  
 player.play();  
 }  
 });  
  
 // create "MINUS BET" button and set behavior  
 Button minusBetButn = new Button("MINUS BET");  
 minusBetButn.setScaleX(doubleScale);  
 minusBetButn.setScaleY(doubleScale);  
 minusBetButn.setStyle("-fx-base: #14425a");  
  
 minusBetButn.setOnAction(e -> { // *todo: no lambda* if(bet > 1){  
 bet--;  
 betField.setText(Integer.*toString*(bet));  
 String addSubCoins = "SMB\_Coin.mp3";  
 Media subAddCoins = new Media(new File(addSubCoins).toURI().toString());  
 player = new MediaPlayer(subAddCoins);  
 player.play();  
 }  
 });  
  
 // create "ADD CASH" button and set behavior  
 Button addCashButn = new Button("ADD CASH");  
 addCashButn.setScaleX(doubleScale);  
 addCashButn.setScaleY(doubleScale);  
 addCashButn.setStyle("-fx-base: #14425a");  
  
 // adds cash amount to existing credits  
 addCashButn.setOnAction(e -> { // *todo: lambda* int addedAmt = addCash();  
 credits = Integer.*parseInt*(creditsField.getText());  
 creditsField.setText(String.*valueOf*(credits + addedAmt));  
 });  
  
 // create "CASHOUT" button and set behavior  
 Button cashOutButn = new Button("CASHOUT");  
 cashOutButn.setScaleX(doubleScale);  
 cashOutButn.setScaleY(doubleScale);  
 cashOutButn.setStyle("-fx-base: #14425a");  
  
 cashOutButn.setOnAction(e -> {  
 Alert cashOutAlert = new Alert(AlertType.*INFORMATION*);  
 cashOutAlert.setTitle("Winner!");  
 String coinsDropping = "coinsDroppingShort.mp3";  
 Media coinsDrop = new Media(new File(coinsDropping).toURI().toString());  
 player = new MediaPlayer(coinsDrop);  
 player.play();  
 cashOutAlert.setHeaderText("Thanks for playing!");  
 int currentCredits = Integer.*parseInt*(creditsField.getText());  
 cashOutAlert.setHeaderText("Feel free to donate your winnings to the author.");  
 cashOutAlert.setContentText("Your winnings: $" + currentCredits);  
 cashOutAlert.showAndWait();  
 creditsField.setText("0");  
 });  
  
 // creates "SPIN" button  
 Button spinButton = new Button("SPIN");  
 int scale = (int) Math.*round*(4 \* *SCALAR*); // *todo: magic num* spinButton.setScaleX(scale);  
 spinButton.setScaleY(scale);  
 spinButton.setStyle("-fx-base: #14425a;");  
  
 // create HBox's and add elements  
 HBox hBoxBottom = new HBox();  
 int hBoxInsetTop = (int) Math.*round*(40 \* *SCALAR*),  
 hBoxInsetRight = (int) Math.*round*(16 \* *SCALAR*),  
 hBoxInsetBottom = (int) Math.*round*(500 \* *SCALAR*),  
 hBoxInsetLeft = (int) Math.*round*(230 \* *SCALAR*),  
 hBoxInsetTop2 = (int) Math.*round*(50 \* *SCALAR*),  
 hBoxInsetRight2 = (int) Math.*round*(20 \* *SCALAR*),  
 hBoxInsetLeft2 = (int) Math.*round*(20 \* *SCALAR*);  
  
 hBoxTop.setAlignment(Pos.*CENTER*);  
 hBoxTop.setPadding(new Insets(hBoxInsetTop, hBoxInsetRight,  
 hBoxInsetBottom, hBoxInsetLeft));  
 hBoxBottom.setPadding(new Insets(hBoxInsetTop2, hBoxInsetRight2,  
 0, hBoxInsetLeft2));  
 hBoxBottom.setAlignment(Pos.*BASELINE\_CENTER*);  
 //hBoxBottom.setAlignment(Pos.CENTER);  
 hBoxBottom.getChildren().add(spinButton);  
  
 int hBoxSpacing = (int) Math.*round*(85 \* *SCALAR*); // *todo: magic numbers  
 /\*\*\*FOR ORIGINAL SIZE SPACING USING "hBoxSpacing" as argument instead of 10\*\*\*/* HBox playOrQuitBox = new HBox(10);  
 playOrQuitBox.setPadding(new Insets(hBoxInsetLeft2, 0, 0, 0));  
 playOrQuitBox.setAlignment(Pos.*CENTER*);  
 playOrQuitBox.getChildren().add(addCashButn);  
 playOrQuitBox.getChildren().add(cashOutButn);  
  
 */\*\*\*FOR ORIGINAL SIZE SPACING USING "hBoxSpacing" as argument instead of 10\*\*\*/* HBox adjustBetBox = new HBox(10);  
 adjustBetBox.setPadding(new Insets(hBoxInsetLeft2, 0, 0, 0));  
 adjustBetBox.setAlignment(Pos.*CENTER*);  
 adjustBetBox.getChildren().add(addBetButn);  
 adjustBetBox.getChildren().add(minusBetButn);  
  
 // column and row numbers for gridpane  
 int col0 = 0,  
 col1 = 1,  
 col2 = 2;  
  
 int row1 = 1,  
 row3 = 3, // *todo: row2?* row4 = 4,  
 row5 = 5;  
  
 // lines 254 - 285 add elements to gridpane  
 int verticalGap = (int) Math.*round*(15 \* *SCALAR*); // *todo: magic num* int horizontalGap = (int) Math.*round*(25 \* *SCALAR*);  
 GridPane gridPane = new GridPane();  
 gridPane.setHgap(horizontalGap);  
 gridPane.setVgap(verticalGap);  
  
 gridPane.add(adjustBetBox, col0, row5);  
 gridPane.add(playOrQuitBox, col2, row5);  
  
 gridPane.add(firstReel, col0, row1);  
 gridPane.add(secondReel, col1, row1);  
 gridPane.add(thirdReel, col2, row1);  
  
 gridPane.add(creditsText, col0, row4);  
 gridPane.add(betText, col1, row4);  
 gridPane.add(paidText, col2, row4);  
  
 gridPane.add(creditsField, col0, row3);  
 gridPane.add(betField, col1, row3);  
 gridPane.add(paidField, col2, row3);  
  
 int gPanePadTop = (int) Math.*round*(350 \* *SCALAR*); // *todo: magic numbers* int gPanePadRight = (int) Math.*round*(75 \* *SCALAR*);  
 int gPanePadBottom = (int) Math.*round*(50 \* *SCALAR*);  
 int gPanePadLeft = (int) Math.*round*(75 \* *SCALAR*);  
 gridPane.add(hBoxBottom, col1, row5);  
 gridPane.setPadding(new Insets(gPanePadTop, gPanePadRight,  
 gPanePadBottom , gPanePadLeft));  
 gridPane.*setHalignment*(creditsText, HPos.*CENTER*);  
 gridPane.*setHalignment*(betText, HPos.*CENTER*);  
 gridPane.*setHalignment*(paidText, HPos.*CENTER*);  
 gridPane.*setHalignment*(spinButton, HPos.*CENTER*);  
  
 // add components to stage  
 Group root = new Group();  
  
 root.getChildren().add(gridPane);  
 root.getChildren().add(hBoxTop);  
  
  
 double widthScaled = *WINDOW\_WIDTH* \* *SCALAR*;  
 int width = (int) Math.*round*(widthScaled);  
 double heightScaled = *WINDOW\_HEIGHT* \* *SCALAR*;  
 int height = (int) Math.*round*(heightScaled);  
  
 Scene scene = new Scene(root, width, height, Color.*BLACK*);  
 mainStage.setScene(scene);  
  
 mainStage.setTitle("Stardust - Lucky Sevens");  
 mainStage.initStyle(StageStyle.*UTILITY*); // was UTILITY  
 mainStage.show();  
  
 // when user clicks SPIN button  
 spinButton.setOnAction(e -> {  
 paidField.setText("0");  
  
 // get current credits  
 credits = Integer.*parseInt*(creditsField.getText());  
  
 // if current credits > zero and credits minus bet is >= 0,  
 // update credits field minus bet  
 if(credits > 0 && (credits - bet) >= 0) {  
 credits -= bet;  
 creditsField.setText(String.*valueOf*(credits));  
 // else prompt user to add cash to play  
 } else {  
  
 int addedAmt = addCash();  
 credits += addedAmt;  
  
 // loop while the bet exceeds the added amounts plus current credits  
 while((credits - bet) < 0) {  
 System.*out*.println("in while loop");  
 creditsField.setText(String.*valueOf*(credits));  
 addedAmt = addCash();  
 credits += addedAmt;  
 }  
 creditsField.setText(String.*valueOf*(credits - bet));  
 }  
  
 // plays audio while reels spin  
 String spinningReels = "spinningReels.wav";  
 Media reelsSound = new Media(new File(spinningReels).  
 toURI().toString());  
 player = new MediaPlayer(reelsSound);  
 player.play();  
  
 String reelStop = "reelStop.wav";  
 Media reelStopSound = new Media(new File(reelStop).  
 toURI().toString());  
  
 // replaces static images with animated reels  
 gridPane.getChildren().remove(firstReel);  
 gridPane.add(firstAnimation, 0 , 1);  
 gridPane.getChildren().remove(secondReel);  
 gridPane.add(secondAnimation, 1, 1);  
 gridPane.getChildren().remove(thirdReel);  
 gridPane.add(thirdAnimation, 2, 1);  
  
  
 // lines 369 - 383 replace animations with static images  
 // after defined pause time // 2300 millis  
 int pauseTime3 = 2300;  
 int pauseTime2 = 1900;  
 int pauseTime1 = 1500;  
  
 //*TODO: change when reels stop spinning -- NEED 2 new pauses* PauseTransition pauseFirstReel = new PauseTransition(Duration.*millis* (pauseTime1));  
 PauseTransition pauseSecondReel = new PauseTransition(Duration.*millis* (pauseTime2));  
 PauseTransition pauseThirdReel = new PauseTransition(Duration.*millis* (pauseTime3));  
  
 pauseFirstReel.setOnFinished(event -> {  
 gridPane.getChildren().remove(firstAnimation);  
 gridPane.add(firstReel, 0, 1);  
 reel1Value = reel1.spinReel();  
 player = new MediaPlayer(reelStopSound);  
 player.play();});  
  
 pauseFirstReel.play();  
  
 pauseSecondReel.setOnFinished(event -> {  
 gridPane.getChildren().remove(secondAnimation);  
 gridPane.add(secondReel, 1, 1);  
 reel2Value = reel2.spinReel();  
 player = new MediaPlayer(reelStopSound);  
 player.play();});  
  
 pauseSecondReel.play();  
  
 pauseThirdReel.setOnFinished(event -> {  
 /\*gridPane.getChildren().remove(firstAnimation);  
 gridPane.add(firstReel, 0, 1);  
  
 gridPane.getChildren().remove(secondAnimation);  
 gridPane.add(secondReel, 1, 1);\*/  
  
 gridPane.getChildren().remove(thirdAnimation);  
 gridPane.add(thirdReel, 2, 1);});  
  
 pauseThirdReel.play();  
  
  
  
 // following code executes after animations stop (code is paused)  
 PauseTransition pause2First = new PauseTransition  
 (Duration.*millis*(pauseTime1));  
  
 PauseTransition pause2Second = new PauseTransition  
 (Duration.*millis*(pauseTime2));  
  
 PauseTransition pause2Third = new PauseTransition  
 (Duration.*millis*(pauseTime3));  
  
  
  
 /\*pause2First.setOnFinished(event -> {  
 reel1Value = reel1.spinReel();  
 player = new MediaPlayer(reelStopSound);  
 player.play();  
 });\*/  
  
  
 pause2Third.setOnFinished(event -> {  
  
 // reelValues store the String result of images  
 // selected by the random generator  
 //reel1Value = reel1.spinReel();  
 //reel2Value = reel2.spinReel();  
 reel3Value = reel3.spinReel();  
  
 // stop sound for reels  
 //String reelStop = "reelStop.wav";  
 //Media reelStopSound = new Media(new File(reelStop).  
 // toURI().toString());  
 player = new MediaPlayer(reelStopSound);  
 player.play();  
  
 // if all three reels are equal  
 if(reel1Value.equals(reel2Value) && reel2Value.equals(reel3Value)){  
 // play audio to indicate winner  
 String winSoundShort = "casinoWinShort.mp3";  
 Media winShort = new Media(new File(winSoundShort)  
 .toURI().toString());  
 player = new MediaPlayer(winShort);  
 player.play();  
 // gets proper pay out and updates credit field  
 creditsPaid = reelsMatch(reel1Value);  
 paidField.setText(String.*valueOf*(creditsPaid));  
 updatedCredits = credits + creditsPaid;  
 creditsField.setText(String.*valueOf*(updatedCredits));  
  
 // if all three reels contain bars but are not equal *todo: below condition is always true* } else if((!reel1Value.equals(reel2Value) || !reel2Value.equals  
 (reel3Value)) && reel1Value.contains("Bar") && reel2Value.  
 contains("Bar") && reel3Value.contains("Bar")) {  
 //play sound to indicate winner  
 String winSoundShort = "casinoWinShort.mp3";  
 Media winShort = new Media(new File(winSoundShort)  
 .toURI().toString());  
 player = new MediaPlayer(winShort);  
 player.play();  
 // gets proper pay out and updates credit field  
 creditsPaid = nonMatchedBars();  
 paidField.setText(String.*valueOf*(creditsPaid));  
 updatedCredits = credits + creditsPaid;  
 creditsField.setText(String.*valueOf*(updatedCredits));  
  
 // if all reels contain 7's but are not equal  
 } else if(reel1Value.contains("7") && reel2Value.contains("7")  
 && reel3Value.contains("7")) {  
 // play sound to indicate winner  
 String winSoundShort = "casinoWinShort.mp3";  
 Media winShort = new Media(new File(winSoundShort)  
 .toURI().toString());  
 player = new MediaPlayer(winShort);  
 player.play();  
 // determines proper pay out and updates credit field  
 creditsPaid = nonMatchedSevens();  
 paidField.setText(String.*valueOf*(creditsPaid));  
 updatedCredits = credits + creditsPaid;  
 creditsField.setText(String.*valueOf*(updatedCredits));  
 }  
 });  
 pause2Third.play();  
 });  
 }  
  
 */\*\* determines proper pay out based on all three reels matching \*/* public int reelsMatch(String reel1) {  
 switch(reel1){  
 case "cherries":  
 creditsWon = multiplier2 \* bet;  
 break;  
 case "singleBar":  
 creditsWon = multiplier3 \* bet;  
 break;  
 case "doubleBar":  
 creditsWon = multiplier4 \* bet;  
 break;  
 case "tripleBar":  
 creditsWon = multiplier5 \* bet;  
 break;  
 case "single7":  
 creditsWon = multiplier6 \* bet;  
 break;  
 case "double7s":  
 creditsWon = multiplier7 \* bet;  
 break;  
 case "triple7s":  
 creditsWon = multiplier8 \* bet;  
 break;  
 case "trip7sWinner":  
 creditsWon = jackpotMultiplier \* bet;  
 break;  
 }  
 return creditsWon;  
 }  
  
 */\*\* creates a dialog box that warns user that they are out of money  
 and allow them to add money, returns dollar amount chosen \*/* public int addCash() {  
 int dollarAmt = 0;  
 ChoiceDialog<Integer> addCashDialog = new ChoiceDialog<> (10, *dollarAmounts*);  
 addCashDialog.setTitle("You are out of money!");  
 addCashDialog.setHeaderText("Add cash to play.");  
 addCashDialog.setContentText("Select a dollar amount to add. $");  
  
 // "Optional" will only have a value if "ok" button clicked, else will return Optional.empty  
 Optional<Integer> btnClickResult = addCashDialog.showAndWait();  
  
 // condition will only be true if user clicks "ok", false if "cancel" button clicked  
 if(btnClickResult.isPresent()) {  
 dollarAmt = addCashDialog.getSelectedItem();  
 }  
  
 System.*out*.println(dollarAmt);  
 return dollarAmt;  
 }  
  
 */\*\* pay out for all three reels that contain bars, but don't all match \*/* public int nonMatchedBars() {  
 creditsWon = multiplier1 \* bet;  
 return creditsWon;  
 }  
  
 */\*\* pay out for all three reels that contain 7's, but don't all match \*/* public int nonMatchedSevens() {  
 creditsWon = multiplier3 \* bet;  
 return creditsWon;  
 }  
  
 private void initTopImage() {  
 // stores animated banner at top in HBox  
 imgStardust = new Image("file:" + absolutePath + "/src/StardustLarger.gif");  
 stardustImgView = new ImageView(imgStardust);  
 stardustImgView.setFitWidth(650 \* *SCALAR*); // *todo* stardustImgView.setFitHeight(309 \* *SCALAR*);  
  
 hBoxTop = new HBox();  
 hBoxTop.getChildren().add(stardustImgView);  
 }  
 // dollar amounts that appear in the "add money" dialog prompt  
 private void initTextFields() {  
 // font style for fields *todo: add method for setup* int font1Size = (int) Math.*round*(*FONT\_SIZE\_FIELDS* \* *SCALAR*);  
 Font font1 = new Font("SansSerif", font1Size);  
  
 // width and height for CREDITS, BET, PAID fields  
 int fieldWidth = (int) Math.*round*(*FIELD\_WIDTH* \* *SCALAR*);  
 int fieldHeight = (int) Math.*round*(*FIELD\_HEIGHT* \* *SCALAR*);  
 // text field showing available credits  
 creditsField = new TextField(String.*valueOf*(initialCredits)); // *todo: make private global vars* creditsField.setAlignment(Pos.*CENTER\_RIGHT*);  
 creditsField.setFont(font1);  
 creditsField.setEditable(false);  
 creditsField.setPrefSize(fieldWidth, fieldHeight);  
 creditsField.setMaxWidth(fieldWidth);  
  
 // text field for bet amount  
 betField = new TextField(String.*valueOf*(bet));  
 betField.setAlignment(Pos.*CENTER\_RIGHT*);  
 betField.setFont(font1);  
 betField.setEditable(false);  
 betField.setPrefSize(fieldWidth, fieldHeight);  
 betField.setMaxWidth(fieldWidth);  
  
 // text field to show the pay out for a spin  
 paidField = new TextField(String.*valueOf*(creditsPaid));  
 paidField.setAlignment(Pos.*CENTER\_RIGHT*);  
 paidField.setFont(font1);  
 paidField.setEditable(false);  
 paidField.setPrefSize(fieldWidth, fieldHeight);  
 paidField.setMaxWidth(fieldWidth);  
 }  
  
 private void initFieldDescriptions() {  
 // text descriptions for fields  
 int fontSize = (int) Math.*round*(*FONT\_SIZE\_FIELD\_DESCRIPTION* \* *SCALAR*);  
  
 // CREDITS  
 creditsText = new Text("CREDITS");  
 creditsText.setFont(Font.*font*("Arial", FontWeight.*BOLD*, fontSize));  
 creditsText.setFill(Color.*RED*);  
  
 // BET  
 betText = new Text("BET");  
 betText.setFont(Font.*font*("Arial", FontWeight.*BOLD*, fontSize));  
 betText.setFill(Color.*RED*);  
  
 // WINNER PAID  
 paidText = new Text("WINNER PAID");  
 paidText.setFont(Font.*font*("Arial", FontWeight.*BOLD*, fontSize));  
 paidText.setFill(Color.*RED*);  
 }  
   
 */\*\* to launch application \*/* public static void main(String[] args) {  
 *launch*(args);  
 }  
}