**Experiment – 9**

Q1) Write a program that can dynamically change the font of a text in a label displayed on a stack pane. The text can be displayed in bold and italic at the same time. You can select the font name or font size from combo boxes, as shown in Figure. The available font names can be obtained using Font.getFamilies(). The combo box for the font size is initialized with numbers from 1 to 100.



Ans:

**Program:**

package Exp\_9;

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.CheckBox;

import javafx.scene.control.ComboBox;

import javafx.scene.control.Label;

import javafx.scene.layout.HBox;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.scene.text.FontPosture;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

import java.util.List;

import java.util.Arrays;

public class First extends Application {

Label textLabel = new Label("Programming is fun");

ComboBox<String> fontNameComboBox = new ComboBox<>();

ComboBox<Integer> fontSizeComboBox = new ComboBox<>();

CheckBox boldCheckBox = new CheckBox("Bold");

CheckBox italicCheckBox = new CheckBox("Italic");

public void start(Stage primaryStage) {

// Get font families

List<String> fontFamiliesList = Font.getFamilies();

int n = fontFamiliesList.size();

String[] fontFamilies = new String[n];

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

fontFamilies[i] = fontFamiliesList.get(i);

}

Arrays.sort(fontFamilies);

fontNameComboBox.getItems().addAll(fontFamilies);

fontNameComboBox.setValue("Book Antiqua"); // Set default font as in the image

// Initialize font sizes

Integer[] fontSizes = new Integer[100];

for (int i = 0; i < 100; i++) {

fontSizes[i] = i + 1;

}

fontSizeComboBox.getItems().addAll(fontSizes);

fontSizeComboBox.setValue(48); // Set default font size as in the image

// Set initial state of checkboxes

boldCheckBox.setSelected(true);

italicCheckBox.setSelected(true);

fontNameComboBox.setOnAction(event -> applyFontChanges());

fontSizeComboBox.setOnAction(event -> applyFontChanges());

boldCheckBox.setOnAction(event -> applyFontChanges());

italicCheckBox.setOnAction(event -> applyFontChanges());

// Controls layout

HBox fontControls = new HBox(10);

fontControls.getChildren().addAll(new Label("Font Name"), fontNameComboBox,

new Label("Font Size"), fontSizeComboBox);

HBox styleControls = new HBox(10);

styleControls.getChildren().addAll(boldCheckBox, italicCheckBox);

VBox controls = new VBox(10);

controls.setPadding(new Insets(10));

controls.getChildren().addAll(fontControls, styleControls);

StackPane centerPane = new StackPane(textLabel);

VBox root = new VBox(10);

root.getChildren().addAll(controls, centerPane);

Scene scene = new Scene(root, 600, 400);

primaryStage.setTitle("Exercise16\_14");

primaryStage.setScene(scene);

primaryStage.show();

applyFontChanges();

}

void applyFontChanges() {

String selectedFontName = fontNameComboBox.getValue();

Integer selectedFontSize = fontSizeComboBox.getValue();

boolean isBold = boldCheckBox.isSelected();

boolean isItalic = italicCheckBox.isSelected();

if (selectedFontName != null && selectedFontSize != null) {

FontWeight fontWeight = isBold ? FontWeight.BOLD : FontWeight.NORMAL;

FontPosture fontPosture = isItalic ? FontPosture.ITALIC : FontPosture.REGULAR;

Font font = Font.font(selectedFontName, fontWeight, fontPosture, selectedFontSize);

textLabel.setFont(font);

}

}

public static void main(String[] args) {

launch(args);

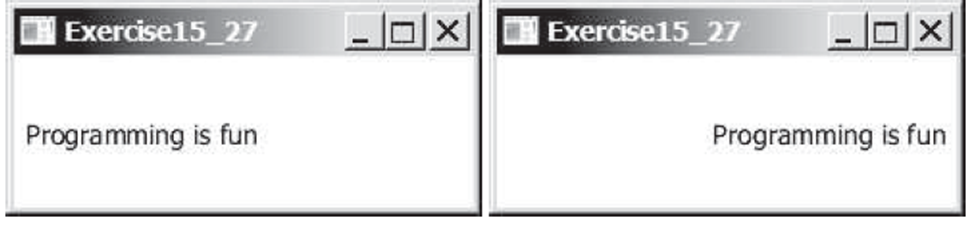
}

}

**Output:**



Q2) Write a program that displays a moving text, as shown in Figure. The text moves from left to right circularly. When it disappears in the right, it reappears from the left. The text freezes when the mouse is pressed and moves again when the button is released.



Ans:

**Program:**

**package** Exp\_9;

**import** javafx.animation.AnimationTimer;

**import** javafx.application.Application;

**import** javafx.scene.Scene;

**import** javafx.scene.input.MouseButton;

**import** javafx.scene.layout.Pane;

**import** javafx.scene.paint.Color;

**import** javafx.scene.text.Font;

**import** javafx.scene.text.Text;

**import** javafx.stage.Stage;

**public** **class** Second **extends** Application {

**static** **final** String ***MOVING\_TEXT*** = "Programming is fun";

**static** **final** **double** ***TEXT\_SIZE*** = 30;

**static** **final** **double** ***MOVE\_SPEED*** = 2;

Text text;

**double** textX = 0;

**boolean** isPaused = **false**;

**public** **void** start(Stage primaryStage) {

Pane pane = **new** Pane();

text = **new** Text(***MOVING\_TEXT***);

text.setFont(Font.*font*("Arial", ***TEXT\_SIZE***));

text.setFill(Color.***BLACK***);

pane.getChildren().add(text);

AnimationTimer timer = **new** AnimationTimer() {

**public** **void** handle(**long** now) {

**if** (!isPaused) {

textX += ***MOVE\_SPEED***;

**double** textWidth = text.getLayoutBounds().getWidth();

**double** paneWidth = pane.getWidth();

**if** (textX > paneWidth) {

textX = -textWidth;

}

text.setX(textX);

}

}

};

timer.start();

pane.setOnMousePressed(event -> {

**if** (event.getButton() == MouseButton.***PRIMARY***) {

isPaused = **true**;

}

});

pane.setOnMouseReleased(event -> {

**if** (event.getButton() == MouseButton.***PRIMARY***) {

isPaused = **false**;

}

});

Scene scene = **new** Scene(pane, 600, 100); // Adjust width and height as needed

primaryStage.setTitle("Exercise15\_27");

primaryStage.setScene(scene);

primaryStage.show();

// Initial positioning of the text

text.setY(pane.getHeight() / 2 + ***TEXT\_SIZE*** / 3); // Center vertically

pane.widthProperty().addListener((observable, oldValue, newValue) -> {

text.setY(newValue.doubleValue() / 2 + ***TEXT\_SIZE*** / 3);

});

}

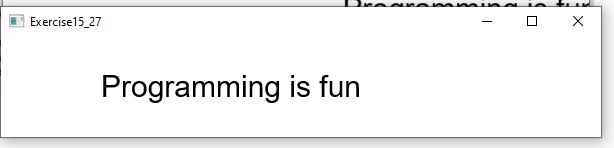
**public** **static** **void** main(String[] args) {

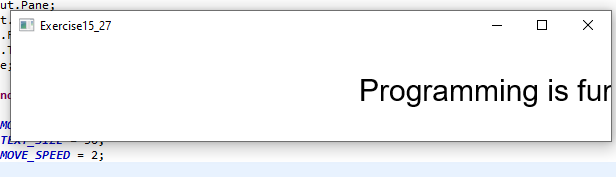
*launch*(args);

}

}

**Output:**





Q3) Create animation in Figure to meet the following requirements:

■ Allow the user to specify the animation speed in a text field.

■ Get the number of iamges and image’s file-name prefix from the user. For example, if the user enters n for the number of images and L for the image prefix, then the files are L1.gif, L2.gif, and so on, to Ln.gif. Assume that the images are stored in the image directory, a subdirectory of the program’s class directory. The animation displays the images one after the other.

■ Allow the user to specify an audio file URL. The audio is played while the animation runs.



Ans:

**Program:**

**package** Exp\_9;

**import** javafx.animation.Animation;

**import** javafx.animation.KeyFrame;

**import** javafx.animation.Timeline;

**import** javafx.application.Application;

**import** javafx.geometry.Insets;

**import** javafx.scene.Scene;

**import** javafx.scene.control.\*;

**import** javafx.scene.image.Image;

**import** javafx.scene.image.ImageView;

**import** javafx.scene.layout.VBox;

**import** javafx.stage.Stage;

**import** javafx.util.Duration;

**import** javax.sound.sampled.\*;

**import** java.io.File;

**import** java.io.IOException;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Third **extends** Application {

**private** ImageView imageView;

**private** List<Image> images;

**private** **int** currentImageIndex = 0;

**private** Timeline timeline;

**private** Clip clip;

**private** TextField speedTextField;

**private** TextField prefixTextField;

**private** TextField numImagesTextField;

**private** TextField audioUrlTextField;

**private** Button startButton;

@Override

**public** **void** start(Stage primaryStage) {

Label speedLabel = **new** Label("Animation speed in milliseconds:");

speedTextField = **new** TextField("200");

Label prefixLabel = **new** Label("Image file prefix:");

prefixTextField = **new** TextField("L");

Label numImagesLabel = **new** Label("Number of images:");

numImagesTextField = **new** TextField("24");

Label audioUrlLabel = **new** Label("Audio file URL:");

audioUrlTextField = **new** TextField("http://www.cs.armstrong.edu/liang/common/audio/anthem/anthem2.mp3");

startButton = **new** Button("Start Animation");

startButton.setOnAction(event -> startAnimation());

Label titleLabel = **new** Label("Learning Java");

imageView = **new** ImageView();

imageView.setFitWidth(400);

imageView.setFitHeight(300);

imageView.setPreserveRatio(**true**);

VBox mainLayout = **new** VBox(10);

mainLayout.setPadding(**new** Insets(10));

mainLayout.getChildren().addAll(titleLabel, speedLabel, speedTextField, prefixLabel, prefixTextField,

numImagesLabel, numImagesTextField, audioUrlLabel, audioUrlTextField, startButton, imageView);

Scene scene = **new** Scene(mainLayout, 600, 500);

primaryStage.setTitle("Exercise16\_23");

primaryStage.setScene(scene);

primaryStage.show();

}

**private** **void** startAnimation() {

**try** {

**int** speed = Integer.*parseInt*(speedTextField.getText());

String prefix = prefixTextField.getText();

**int** numImages = Integer.*parseInt*(numImagesTextField.getText());

String audioUrl = audioUrlTextField.getText();

loadImages(prefix, numImages);

**if** (images.isEmpty()) {

showAlert("Error", "No images loaded. Check prefix and number of images.");

**return**;

}

imageView.setImage(images.get(0));

currentImageIndex = 0;

startTimeline(speed);

playAudio(audioUrl);

} **catch** (NumberFormatException e) {

showAlert("Error", "Invalid input for speed or number of images.");

} **catch** (MalformedURLException e) {

showAlert("Error", "Invalid audio URL: " + audioUrlTextField.getText());

} **catch** (Exception e) {

showAlert("Error", e.getMessage());

}

}

**private** **void** loadImages(String prefix, **int** numImages) {

images = **new** ArrayList<>();

**for** (**int** i = 1; i <= numImages; i++) {

// Path relative to the project root (PracticalS)

String imagePath = "image/" + prefix + i + ".gif";

**try** {

// Use File to load the image from the PracticalS/image/ directory

File file = **new** File(imagePath);

**if** (file.exists()) {

Image image = **new** Image(file.toURI().toString());

images.add(image);

} **else** {

System.***err***.println("Image not found: " + imagePath);

}

} **catch** (Exception e) {

System.***err***.println("Error loading image: " + imagePath + " - " + e.getMessage());

}

}

}

**private** **void** startTimeline(**int** speed) {

**if** (timeline != **null**) {

timeline.stop();

}

timeline = **new** Timeline(

**new** KeyFrame(Duration.*millis*(speed), event -> {

**if** (!images.isEmpty()) {

imageView.setImage(images.get(currentImageIndex));

currentImageIndex = (currentImageIndex + 1) % images.size();

}

})

);

timeline.setCycleCount(Animation.***INDEFINITE***);

timeline.play();

}

**private** **void** playAudio(String audioUrlString) **throws** MalformedURLException, IOException, UnsupportedAudioFileException, LineUnavailableException {

**if** (clip != **null** && clip.isRunning()) {

clip.stop();

clip.close();

}

URL audioUrl = **new** URL(audioUrlString);

**try** (AudioInputStream audioInputStream = AudioSystem.*getAudioInputStream*(audioUrl)) {

clip = AudioSystem.*getClip*();

clip.open(audioInputStream);

clip.loop(Clip.***LOOP\_CONTINUOUSLY***);

clip.start();

}

}

**private** **void** showAlert(String title, String message) {

Alert alert = **new** Alert(Alert.AlertType.***ERROR***);

alert.setTitle(title);

alert.setHeaderText(**null**);

alert.setContentText(message);

alert.showAndWait();

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

**Output:**

