**Task1:**

package application;

import javafx.application.Application;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.StringProperty;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

public class AddressBookApp extends Application {

private TableView<Contact> tableView = new TableView<>();

private TextField nameField = new TextField();

private TextField phoneField = new TextField();

private TextField addressField = new TextField();

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Simple Address Book");

TableColumn<Contact, String> nameCol = new TableColumn<>("Name");

nameCol.setCellValueFactory(cellData -> cellData.getValue().nameProperty());

TableColumn<Contact, String> phoneCol = new TableColumn<>("Phone");

phoneCol.setCellValueFactory(cellData -> cellData.getValue().phoneProperty());

TableColumn<Contact, String> addressCol = new TableColumn<>("Address");

addressCol.setCellValueFactory(cellData -> cellData.getValue().addressProperty());

tableView.getColumns().addAll(nameCol, phoneCol, addressCol);

Button addButton = new Button("Add");

addButton.setOnAction(e -> addContact());

Button editButton = new Button("Edit");

editButton.setOnAction(e -> editContact());

HBox inputBox = new HBox(10);

inputBox.getChildren().addAll(nameField, phoneField, addressField, addButton, editButton);

VBox layout = new VBox(10);

layout.getChildren().addAll(tableView, inputBox);

Scene scene = new Scene(layout, 400, 300);

primaryStage.setScene(scene);

primaryStage.show();

}

private void addContact() {

String name = nameField.getText();

String phone = phoneField.getText();

String address = addressField.getText();

Contact newContact = new Contact(name, phone, address);

tableView.getItems().add(newContact);

nameField.clear();

phoneField.clear();

addressField.clear();

}

private void editContact() {

Contact selectedContact = tableView.getSelectionModel().getSelectedItem();

if (selectedContact != null) {

String name = nameField.getText();

String phone = phoneField.getText();

String address = addressField.getText();

selectedContact.setName(name);

selectedContact.setPhone(phone);

selectedContact.setAddress(address);

tableView.refresh();

}

}

public class Contact {

private final StringProperty name = new SimpleStringProperty();

private final StringProperty phone = new SimpleStringProperty();

private final StringProperty address = new SimpleStringProperty();

public Contact(String name, String phone, String address) {

setName(name);

setPhone(phone);

setAddress(address);

}

public String getName() {

return name.get();

}

public void setName(String name) {

this.name.set(name);

}

public StringProperty nameProperty() {

return name;

}

public String getPhone() {

return phone.get();

}

public void setPhone(String phone) {

this.phone.set(phone);

}

public StringProperty phoneProperty() {

return phone;

}

public String getAddress() {

return address.get();

}

public void setAddress(String address) {

this.address.set(address);

}

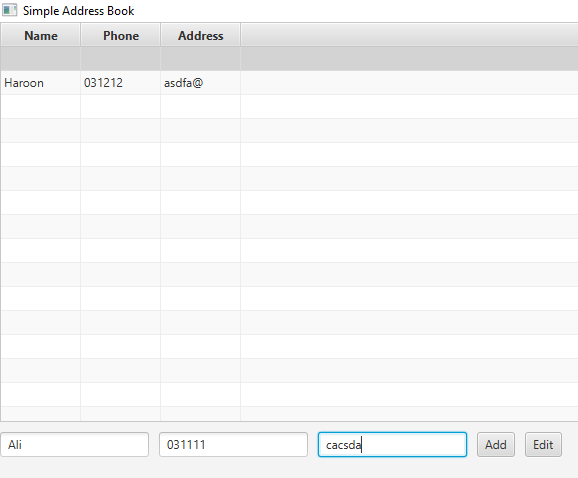
public StringProperty addressProperty() {

return address;

}

}

}



**SWING:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class AddressBookApp extends JFrame {

private DefaultTableModel tableModel;

private JTable tableView;

private JTextField nameField;

private JTextField phoneField;

private JTextField addressField;

public AddressBookApp() {

setTitle("Address Book");

setSize(400, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

tableModel = new DefaultTableModel(new Object[]{"Name", "Phone", "Address"}, 0);

tableView = new JTable(tableModel);

JScrollPane tableScrollPane = new JScrollPane(tableView);

nameField = new JTextField(5);

phoneField = new JTextField(5);

addressField = new JTextField(5);

JButton addButton = new JButton("Add");

addButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addContact();

}

});

JButton editButton = new JButton("Edit");

editButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

editContact();

}

});

JPanel inputPanel = new JPanel();

inputPanel.add(new JLabel("Name:"));

inputPanel.add(nameField);

inputPanel.add(new JLabel("Phone:"));

inputPanel.add(phoneField);

inputPanel.add(new JLabel("Address:"));

inputPanel.add(addressField);

inputPanel.add(addButton);

inputPanel.add(editButton);

setLayout(new BorderLayout());

add(new JLabel("Address Book", SwingConstants.CENTER), BorderLayout.NORTH);

add(tableScrollPane, BorderLayout.CENTER);

add(inputPanel, BorderLayout.SOUTH);

}

private void addContact() {

String name = nameField.getText();

String phone = phoneField.getText();

String address = addressField.getText();

Object[] rowData = {name, phone, address};

tableModel.addRow(rowData);

nameField.setText("");

phoneField.setText("");

addressField.setText("");

}

private void editContact() {

int selectedRow = tableView.getSelectedRow();

if (selectedRow >= 0) {

String name = nameField.getText();

String phone = phoneField.getText();

String address = addressField.getText();

tableModel.setValueAt(name, selectedRow, 0);

tableModel.setValueAt(phone, selectedRow, 1);

tableModel.setValueAt(address, selectedRow, 2);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

AddressBookApp app = new AddressBookApp();

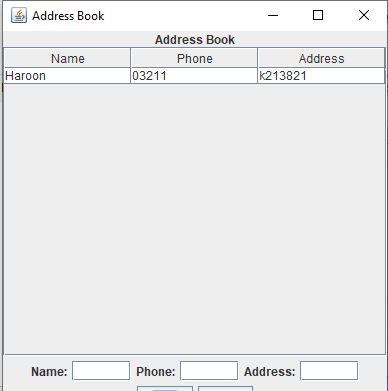
app.setVisible(true);

}

});

}

}



**Task2:**

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.stage.Stage;

public class StudentRegistrationForm extends Application {

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Student Registration Form");

GridPane grid = new GridPane();

grid.setHgap(10);

grid.setVgap(10);

grid.setPadding(new Insets(20, 20, 20, 20));

Label titleLabel = new Label("Registration Form");

titleLabel.setStyle("-fx-font-size: 18px; -fx-font-weight: bold;");

Label nameLabel = new Label("Name:");

TextField nameField = new TextField();

Label emailLabel = new Label("Email:");

TextField emailField = new TextField();

Label ageLabel = new Label("Age:");

TextField ageField = new TextField();

Button submitButton = new Button("Submit");

submitButton.setStyle("-fx-background-color: green; -fx-text-fill: white; -fx-font-size: 14px;");

submitButton.setMaxWidth(Double.MAX\_VALUE);

submitButton.setOnAction(e -> {

// Check if all fields are filled out

if (isValidInput(nameField.getText(), emailField.getText(), ageField.getText())) {

String name = nameField.getText();

String email = emailField.getText();

int age = Integer.parseInt(ageField.getText());

// PRinting on cmmnd

System.out.println("Name: " + name);

System.out.println("Email: " + email);

System.out.println("Age: " + age);

// to clear the fields after submission

nameField.clear();

emailField.clear();

ageField.clear();

} else {

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

alert.setTitle("Error");

alert.setHeaderText(null);

alert.setContentText("All fields are required.");

alert.showAndWait();

}

});

grid.add(titleLabel, 0, 0, 2, 1);

grid.add(nameLabel, 0, 1);

grid.add(nameField, 1, 1);

grid.add(emailLabel, 0, 2);

grid.add(emailField, 1, 2);

grid.add(ageLabel, 0, 3);

grid.add(ageField, 1, 3);

grid.add(submitButton, 0, 4, 2, 1); // Span the submit button across two columns

GridPane.setHalignment(submitButton, javafx.geometry.HPos.CENTER);

Scene scene = new Scene(grid, 300, 220); // Increased height to accommodate the title

primaryStage.setScene(scene);

primaryStage.show();

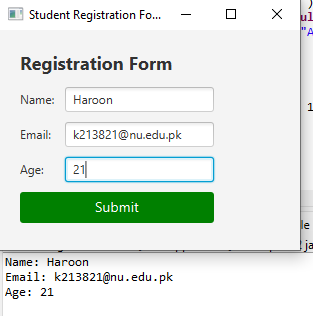
}

private boolean isValidInput(String name, String email, String age) {

return !name.isEmpty() && !email.isEmpty() && !age.isEmpty();

}

}



**SWING:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class StudentRegistrationFormSwing {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("Student Registration Form");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setPreferredSize(new Dimension(300, 220));

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(5, 2, 10, 10));

panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

JLabel titleLabel = new JLabel("Registration Form");

titleLabel.setFont(new Font("Arial", Font.BOLD, 18));

JLabel nameLabel = new JLabel("Name:");

JTextField nameField = new JTextField();

JLabel emailLabel = new JLabel("Email:");

JTextField emailField = new JTextField();

JLabel ageLabel = new JLabel("Age:");

JTextField ageField = new JTextField();

JButton submitButton = new JButton("Submit");

submitButton.setBackground(Color.GREEN);

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.PLAIN, 14));

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String name = nameField.getText();

String email = emailField.getText();

String ageText = ageField.getText();

if (isValidInput(name, email, ageText)) {

int age = Integer.parseInt(ageText);

System.out.println("Name: " + name);

System.out.println("Email: " + email);

System.out.println("Age: " + age);

nameField.setText("");

emailField.setText("");

ageField.setText("");

} else {

JOptionPane.showMessageDialog(frame, "All fields are required.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

});

panel.add(titleLabel);

panel.add(new JLabel()); // Empty cell for spacing

panel.add(nameLabel);

panel.add(nameField);

panel.add(emailLabel);

panel.add(emailField);

panel.add(ageLabel);

panel.add(ageField);

panel.add(new JLabel()); // Empty cell for spacing

panel.add(submitButton);

frame.add(panel);

frame.pack();

frame.setVisible(true);

});

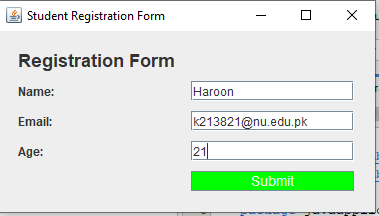
}

private static boolean isValidInput(String name, String email, String age) {

return !name.isEmpty() && !email.isEmpty() && !age.isEmpty();

}

}



**Task3:**

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.stage.Stage;

public class BasicCalculator extends Application {

private StringBuilder input = new StringBuilder();

private Label display = new Label("0");

public static void main(String[] args) {

launch(args);

}

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Basic Calculator");

GridPane grid = new GridPane();

grid.setHgap(10);

grid.setVgap(10);

grid.setPadding(new Insets(20, 20, 20, 20));

display.setStyle("-fx-font-size: 18px;");

GridPane.setHalignment(display, javafx.geometry.HPos.RIGHT);

GridPane.setColumnSpan(display, 4);

grid.add(display, 0, 0, 4, 1);

String[][] buttonLabels = {

{"7", "8", "9", "/"},

{"4", "5", "6", "\*"},

{"1", "2", "3", "-"},

{"0", "C", "+", "="}

};

for (int row = 0; row < buttonLabels.length; row++) {

for (int col = 0; col < buttonLabels[row].length; col++) {

String label = buttonLabels[row][col];

Button button = new Button(label);

button.setMinSize(50, 50);

button.setOnAction(e -> handleButtonClick(label));

grid.add(button, col, row + 1);

}

}

Scene scene = new Scene(grid, 300, 350);

primaryStage.setScene(scene);

primaryStage.show();

}

private void handleButtonClick(String label) {

if (label.equals("C")) {

input.setLength(0); // Clear input

display.setText("0");

} else if (label.equals("=")) {

try {

double result = evaluateExpression(input.toString());

display.setText(Double.toString(result));

input.setLength(0);

input.append(result);

} catch (Exception e) {

display.setText("Error");

input.setLength(0);

}

} else {

if (input.length() == 0 && "+-\*/".contains(label)) {

input.append("0"); // Ensure that input starts with a digit

}

input.append(label);

display.setText(input.toString());

}

}

private double evaluateExpression(String expression) {

return new DoubleEvaluator().evaluate(expression);

}

public static class DoubleEvaluator {

public double evaluate(String expression) {

return new Object() {

int pos = -1, ch;

void nextChar() {

ch = (++pos < expression.length()) ? expression.charAt(pos) : -1;

}

boolean eat(int charToEat) {

while (ch == ' ') nextChar();

if (ch == charToEat) {

nextChar();

return true;

}

return false;

}

double parse() {

nextChar();

double x = parseExpression();

if (pos < expression.length()) throw new RuntimeException("Unexpected: " + (char) ch);

return x;

}

double parseExpression() {

double x = parseTerm();

for (; ; ) {

if (eat('+')) x += parseTerm(); // addition

else if (eat('-')) x -= parseTerm(); // subtraction

else return x;

}

}

double parseTerm() {

double x = parseFactor();

for (; ; ) {

if (eat('\*')) x \*= parseFactor(); // multiplication

else if (eat('/')) x /= parseFactor(); // division

else return x;

}

}

double parseFactor() {

if (eat('+')) return parseFactor(); // unary plus

if (eat('-')) return -parseFactor(); // unary minus

double x;

int startPos = this.pos;

if (eat('(')) { // parentheses

x = parseExpression();

eat(')');

} else if ((ch >= '0' && ch <= '9') || ch == '.') { // numbers

while ((ch >= '0' && ch <= '9') || ch == '.') nextChar();

x = Double.parseDouble(expression.substring(startPos, this.pos));

} else {

throw new RuntimeException("Unexpected: " + (char) ch);

}

return x;

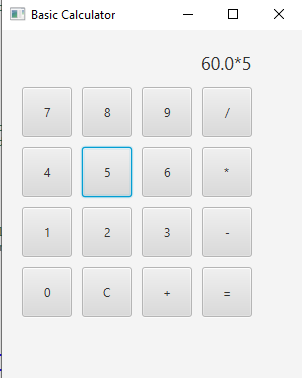
}

}.parse();

}

}

}



**SWING:**

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template

\*/

package javaapplication1;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class BasicCalculatorSwing {

private StringBuilder input = new StringBuilder();

private JLabel display = new JLabel("0");

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

BasicCalculatorSwing calculator = new BasicCalculatorSwing();

calculator.createAndShowGUI();

});

}

private void createAndShowGUI() {

JFrame frame = new JFrame("Basic Calculator");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setPreferredSize(new Dimension(300, 350));

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

display.setFont(new Font("Arial", Font.PLAIN, 18));

gbc.gridx = 0;

gbc.gridy = 0;

gbc.gridwidth = 4;

gbc.anchor = GridBagConstraints.LINE\_END;

panel.add(display, gbc);

String[][] buttonLabels = {

{"7", "8", "9", "/"},

{"4", "5", "6", "\*"},

{"1", "2", "3", "-"},

{"0", "C", "+", "="}

};

gbc.gridwidth = 1;

for (int row = 0; row < buttonLabels.length; row++) {

for (int col = 0; col < buttonLabels[row].length; col++) {

String label = buttonLabels[row][col];

JButton button = new JButton(label);

button.setPreferredSize(new Dimension(50, 50));

button.addActionListener(e -> handleButtonClick(label));

gbc.gridx = col;

gbc.gridy = row + 1;

panel.add(button, gbc);

}

}

frame.add(panel);

frame.pack();

frame.setVisible(true);

}

private void handleButtonClick(String label) {

if (label.equals("C")) {

input.setLength(0);

display.setText("0");

} else if (label.equals("=")) {

try {

double result = evaluateExpression(input.toString());

display.setText(Double.toString(result));

input.setLength(0);

input.append(result);

} catch (Exception e) {

display.setText("Error");

input.setLength(0);

}

} else {

if (input.length() == 0 && "+-\*/".contains(label)) {

input.append("0");

}

input.append(label);

display.setText(input.toString());

}

}

private double evaluateExpression(String expression) {

return new DoubleEvaluator().evaluate(expression);

}

public static class DoubleEvaluator {

public double evaluate(String expression) {

return new Object() {

int pos = -1, ch;

void nextChar() {

ch = (++pos < expression.length()) ? expression.charAt(pos) : -1;

}

boolean eat(int charToEat) {

while (ch == ' ') nextChar();

if (ch == charToEat) {

nextChar();

return true;

}

return false;

}

double parse() {

nextChar();

double x = parseExpression();

if (pos < expression.length()) throw new RuntimeException("Unexpected: " + (char) ch);

return x;

}

double parseExpression() {

double x = parseTerm();

for (; ; ) {

if (eat('+')) x += parseTerm();

else if (eat('-')) x -= parseTerm();

else return x;

}

}

double parseTerm() {

double x = parseFactor();

for (; ; ) {

if (eat('\*')) x \*= parseFactor();

else if (eat('/')) x /= parseFactor();

else return x;

}

}

double parseFactor() {

if (eat('+')) return parseFactor();

if (eat('-')) return -parseFactor();

double x;

int startPos = this.pos;

if (eat('(')) {

x = parseExpression();

eat(')');

} else if ((ch >= '0' && ch <= '9') || ch == '.') {

while ((ch >= '0' && ch <= '9') || ch == '.') nextChar();

x = Double.parseDouble(expression.substring(startPos, this.pos));

} else {

throw new RuntimeException("Unexpected: " + (char) ch);

}

return x;

}

}.parse();

}

}

}

