|  |
| --- |
| Exp: 13 Date: 01/11/2023  **JAVA PROGRAMMING**  **CS6308** |

GUI

Name: VIJAI SURIA M

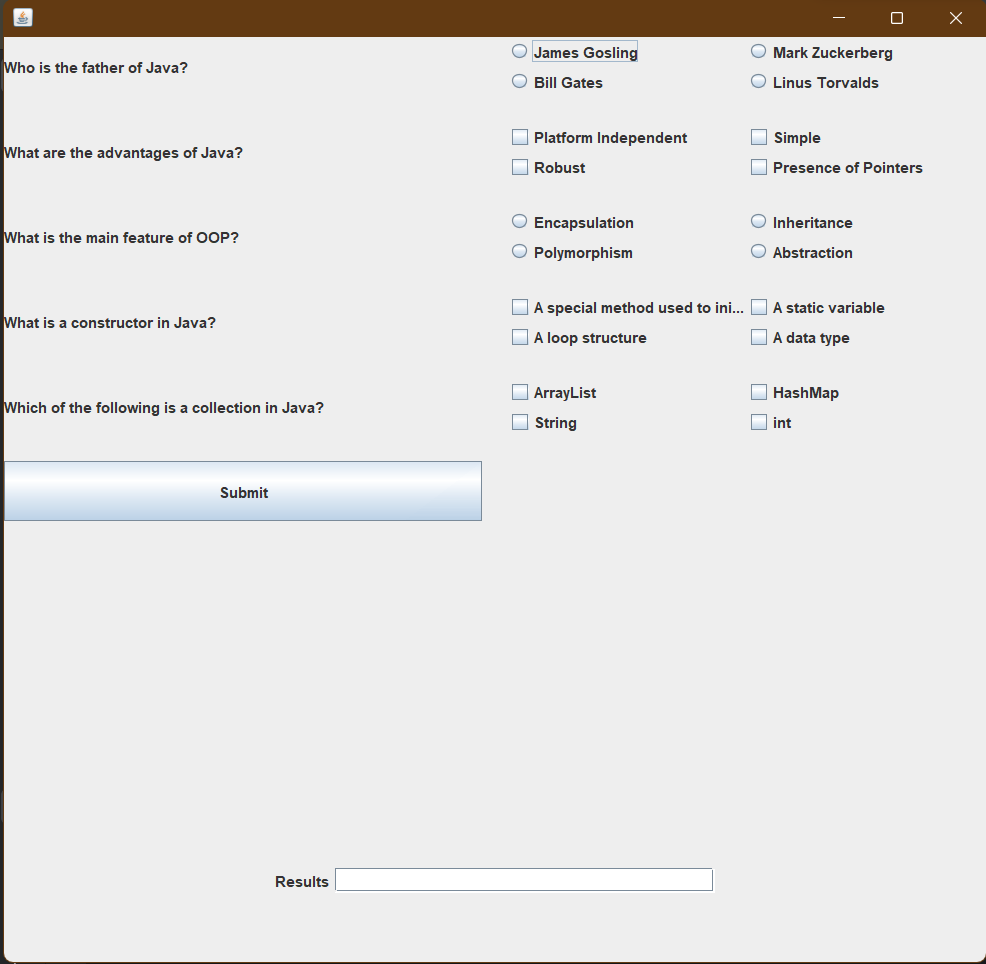
Reg No.: 2021503568

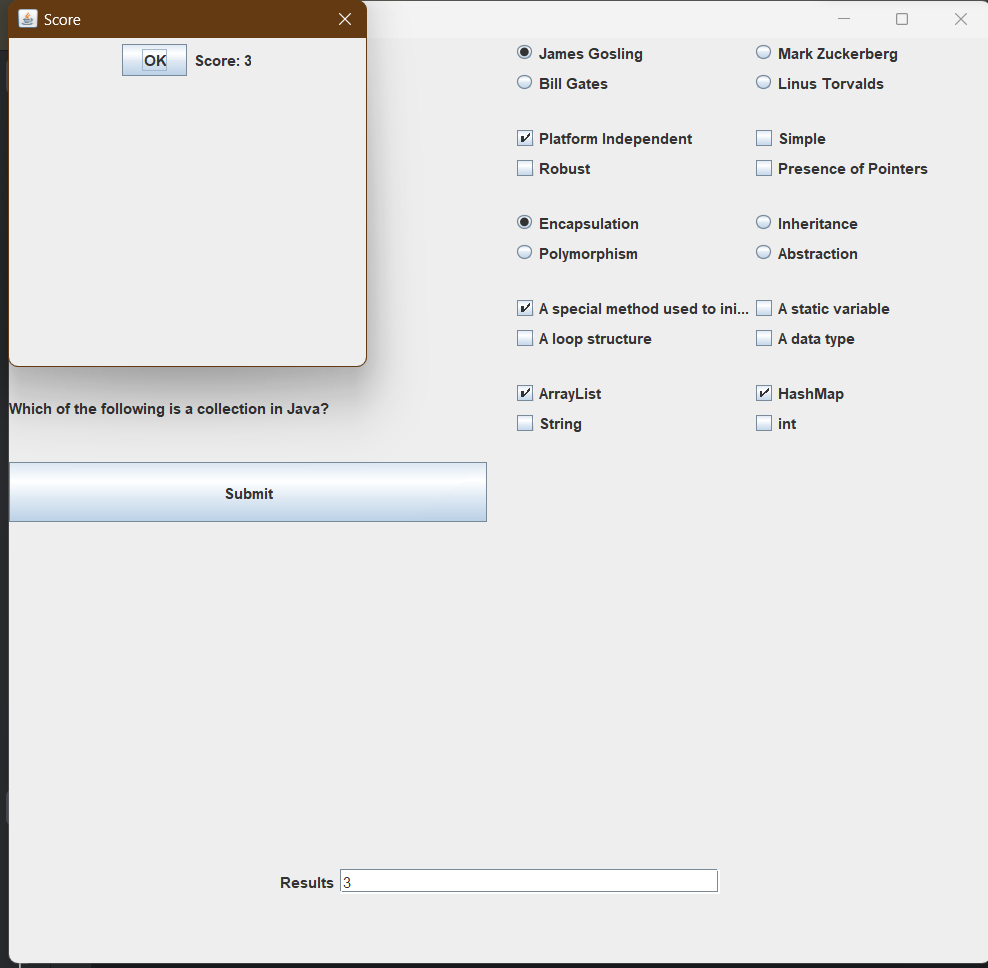
1. Create an interactive Quiz form using radio buttons, check box, text box, text area. Write a method to find the total score of the user. Display the right answer to the user at end.

**CODE**

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  class QuizApp\_3568 extends JFrame implements ActionListener {  JPanel qPanel, resPanel;  JLabel ql1, ql2, ql3, ql4, ql5, res;  JTextField score;  private JRadioButton[] radioButtons;  private JCheckBox[] checkBoxes;  ButtonGroup radioGroup;  JButton submit, ok;  JDialog result;  QuizApp() {  radioButtons = new JRadioButton[4];  checkBoxes = new JCheckBox[4];  result = new JDialog(this, "Score", false);  JPanel q1, q2, q3, q4, q5;  ql1 = new JLabel("Who is the father of Java?");  q1 = new JPanel(new GridLayout(2, 4));  radioButtons[0] = new JRadioButton("James Gosling");  radioButtons[1] = new JRadioButton("Mark Zuckerberg");  radioButtons[2] = new JRadioButton("Bill Gates");  radioButtons[3] = new JRadioButton("Linus Torvalds");  radioGroup = new ButtonGroup();  for (JRadioButton radioButton : radioButtons) {  radioGroup.add(radioButton);  }  q1.add(radioButtons[0]);  q1.add(radioButtons[1]);  q1.add(radioButtons[2]);  q1.add(radioButtons[3]);  ql2 = new JLabel("What are the advantages of Java?");  q2 = new JPanel(new GridLayout(2, 4));  checkBoxes[0] = new JCheckBox("Platform Independent");  checkBoxes[1] = new JCheckBox("Simple");  checkBoxes[2] = new JCheckBox("Robust");  checkBoxes[3] = new JCheckBox("Presence of Pointers");  q2.add(checkBoxes[0]);  q2.add(checkBoxes[1]);  q2.add(checkBoxes[2]);  q2.add(checkBoxes[3]);  ql3 = new JLabel("What is the main feature of OOP?");  q3 = new JPanel(new GridLayout(2, 4));  radioButtons[0] = new JRadioButton("Encapsulation");  radioButtons[1] = new JRadioButton("Inheritance");  radioButtons[2] = new JRadioButton("Polymorphism");  radioButtons[3] = new JRadioButton("Abstraction");  radioGroup = new ButtonGroup();  for (JRadioButton radioButton : radioButtons) {  radioGroup.add(radioButton);  }  q3.add(radioButtons[0]);  q3.add(radioButtons[1]);  q3.add(radioButtons[2]);  q3.add(radioButtons[3]);  ql4 = new JLabel("What is a constructor in Java?");  q4 = new JPanel(new GridLayout(2, 4));  checkBoxes[0] = new JCheckBox("A special method used to initialize objects");  checkBoxes[1] = new JCheckBox("A static variable");  checkBoxes[2] = new JCheckBox("A loop structure");  checkBoxes[3] = new JCheckBox("A data type");  q4.add(checkBoxes[0]);  q4.add(checkBoxes[1]);  q4.add(checkBoxes[2]);  q4.add(checkBoxes[3]);  ql5 = new JLabel("Which of the following is a collection in Java?");  q5 = new JPanel(new GridLayout(2, 4));  checkBoxes[0] = new JCheckBox("ArrayList");  checkBoxes[1] = new JCheckBox("HashMap");  checkBoxes[2] = new JCheckBox("String");  checkBoxes[3] = new JCheckBox("int");  q5.add(checkBoxes[0]);  q5.add(checkBoxes[1]);  q5.add(checkBoxes[2]);  q5.add(checkBoxes[3]);  submit = new JButton("Submit");  submit.setBounds(30, 30, 120, 50);  submit.addActionListener(this);  qPanel = new JPanel(new GridLayout(10, 1, 20, 20));  qPanel.add(ql1);  qPanel.add(q1);  qPanel.add(ql2);  qPanel.add(q2);  qPanel.add(ql3);  qPanel.add(q3);  qPanel.add(ql4);  qPanel.add(q4);  qPanel.add(ql5);  qPanel.add(q5);  qPanel.add(submit);  resPanel = new JPanel(new FlowLayout());  res = new JLabel("Results");  score = new JTextField(30);  resPanel.add(res);  resPanel.add(score);  ok = new JButton("OK");  ok.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  result.setVisible(false);  }  });  result.add(ok);  setVisible(true);  setLayout(new BorderLayout());  setSize(800, 500);  add(qPanel, BorderLayout.NORTH);  add(resPanel, BorderLayout.CENTER);  }  public void actionPerformed(ActionEvent e) {  int c = 0;  if (radioButtons[0].isSelected()) {  c++;  }  if (checkBoxes[0].isSelected() && checkBoxes[1].isSelected() && checkBoxes[2].isSelected()) {  c++;  }  if (radioButtons[3].isSelected()) {  c++;  }  if (checkBoxes[0].isSelected()) {  c++;  }  if (checkBoxes[0].isSelected() && checkBoxes[1].isSelected()) {  c++;  }  score.setText(String.valueOf(c));  result.add(new JLabel("Score: " + c));  result.setLayout(new FlowLayout());  result.setSize(300, 300);  result.setVisible(true);  }  public static void main(String[] args) {  new QuizApp();  }  } |

**OUTPUT**





2. Create a shopping cart program using Java Swing. The program should allow users to add items to the cart, display the cart's contents (including the items and their prices), and calculate the total amount to be paid. Create a Java Swing application that fulfills these requirements.

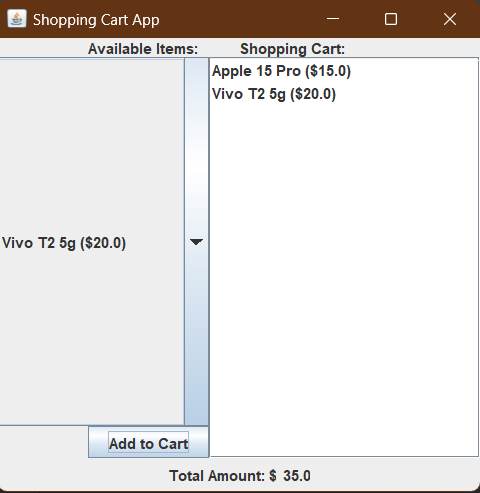
**CODE**

|  |
| --- |
| import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.util.ArrayList; class Item {  String name;  double price;   public Item(String name, double price) {  this.name = name;  this.price = price;  }   @Override  public String toString() {  return name + " ($" + price + ")";  } }  public class ShoppingCartApp\_3568 {  private JFrame frame;  private DefaultListModel<Item> cartModel;  private JList<Item> cartList;  private double totalAmount;   JLabel totalLabel;   public ShoppingCartApp() {  frame = new JFrame("Shopping Cart App");  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setSize(400, 400);   cartModel = new DefaultListModel<>();  cartList = new JList<>(cartModel);   JPanel itemPanel = createItemPanel();  JPanel cartPanel = createCartPanel();  JPanel totalPanel = new JPanel(new FlowLayout());   JLabel totalLabelT = new JLabel("Total Amount: $");  totalLabel = new JLabel("00");  totalPanel.add(totalLabelT);  totalPanel.add(totalLabel);   frame.setLayout(new BorderLayout());  frame.add(itemPanel, BorderLayout.*WEST*);  frame.add(cartPanel, BorderLayout.*CENTER*);  frame.add(totalPanel, BorderLayout.*SOUTH*);   frame.setVisible(true);  }   private JPanel createItemPanel() {  JPanel panel = new JPanel();  panel.setLayout(new BoxLayout(panel, BoxLayout.*Y\_AXIS*));   JLabel label = new JLabel("Available Items:");  panel.add(label);   JComboBox<Item> itemComboBox = new JComboBox<>();  itemComboBox.addItem(new Item("Samsung S23", 10.0));  itemComboBox.addItem(new Item("Apple 15 Pro", 15.0));  itemComboBox.addItem(new Item("Vivo T2 5g", 20.0));  itemComboBox.addItem(new Item("Nokio 6a", 25.0));  panel.add(itemComboBox);   JButton addButton = new JButton("Add to Cart");  addButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  Item selected = (Item) itemComboBox.getSelectedItem();  cartModel.addElement(selected);  assert selected != null;  totalAmount += selected.price;  totalLabel.setText(String.*valueOf*(totalAmount));  }  });  panel.add(addButton);   return panel;  }   private JPanel createCartPanel() {  JPanel panel = new JPanel();  panel.setLayout(new BoxLayout(panel, BoxLayout.*Y\_AXIS*));   JLabel label = new JLabel("Shopping Cart:");  panel.add(label);   JScrollPane scrollPane = new JScrollPane(cartList);  panel.add(scrollPane);   return panel;  }   public static void main(String[] args) {  SwingUtilities.*invokeLater*(() -> {  new ShoppingCartApp();  });  } } |

**OUTPUT**

A screenshot of a computer

Description automatically generated



3. Write a program to create google account with details as shown in figure.



**CODE**

|  |
| --- |
| import javax.swing.\*; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.util.regex.Matcher; import java.util.regex.Pattern; public class GoogleForm\_3568 {  public static void main(String[] args) {  JFrame frame = new JFrame("Google Account Registration");  frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setSize(400, 400);  JPanel panel = new JPanel(new GridBagLayout());  GridBagConstraints gbc = new GridBagConstraints();  gbc.fill = GridBagConstraints.*HORIZONTAL*;  gbc.insets = new Insets(5, 5, 5, 5);  JLabel firstNameLabel = new JLabel("First Name:");  JTextField firstNameField = new JTextField(20);  JLabel lastNameLabel = new JLabel("Last Name:");  JTextField lastNameField = new JTextField(20);  JLabel emailLabel = new JLabel("Email (Username):");  JTextField emailField = new JTextField(20);  JLabel passwordLabel = new JLabel("Password:");  JPasswordField passwordField = new JPasswordField(20);  JLabel confirmPasswordLabel = new JLabel("Confirm Password:");  JPasswordField confirmPasswordField = new JPasswordField(20);  JLabel countryLabel = new JLabel("Country:");  String[] countries = { "Select", "India", "US", "UK" };  JComboBox<String> countryComboBox = new JComboBox<>(countries);  JLabel phoneLabel = new JLabel("Phone number:");  JTextField phoneField = new JTextField(20);  JLabel recoveryEmailLabel = new JLabel("Recovery Email:");  JTextField recoveryEmailField = new JTextField(20);  JLabel dobLabel = new JLabel("Date of Birth:");  JComboBox<String> monthComboBox = new JComboBox<>(new String[] { "Select", "January", "February", "March",  "April", "May", "June", "July", "August", "September", "October", "November", "December" });  JComboBox<String> dayComboBox = new JComboBox<>(new String[] { "Select" });  JComboBox<String> yearComboBox = new JComboBox<>(new String[] { "Select" });  // Populate dayComboBox with days 1 to 31 initially  String[] daysInitial = new String[32];  for (int i = 0; i <= 31; i++) {  daysInitial[i] = Integer.*toString*(i);  }  dayComboBox.setModel(new DefaultComboBoxModel<>(daysInitial));  // Now, add the ActionListener for monthComboBox  monthComboBox.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String selectedMonth = (String) monthComboBox.getSelectedItem();  int selectedYear = Integer.*parseInt*((String) yearComboBox.getSelectedItem());  if (selectedMonth.equals("Select")) {  dayComboBox.setModel(new DefaultComboBoxModel<>(new String[] { "Select" }));  } else {  int maxDays = *getMaxDaysForMonth*(selectedMonth, selectedYear);  String[] days = new String[maxDays + 1];  days[0] = "Select";  for (int i = 1; i <= maxDays; i++) {  days[i] = String.*valueOf*(i);  }  dayComboBox.setModel(new DefaultComboBoxModel<>(days));  }  }  });  // Populate dayComboBox and yearComboBox based on the selected month  monthComboBox.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  String selectedMonth = (String) monthComboBox.getSelectedItem();  int selectedYear = Integer.*parseInt*((String) yearComboBox.getSelectedItem());  if (selectedMonth.equals("Select")) {  dayComboBox.setModel(new DefaultComboBoxModel<>(new String[] { "Select" }));  } else {  int maxDays = *getMaxDaysForMonth*(selectedMonth, selectedYear);  String[] days = new String[maxDays + 1];  days[0] = "Select";  for (int i = 1; i <= maxDays; i++) {  days[i] = String.*valueOf*(i);  }  dayComboBox.setModel(new DefaultComboBoxModel<>(days));  }  }  });  // Populate yearComboBox with a range of years  String[] years = new String[200];  for (int i = 1900; i <= 2018; i++) {  years[i - 1900] = Integer.*toString*(i);  }  years[0] = "Select";  yearComboBox.setModel(new DefaultComboBoxModel<>(years));  JLabel genderLabel = new JLabel("Gender:");  String[] genders = { "Select", "Male", "Female", "Other" };  JComboBox<String> genderComboBox = new JComboBox<>(genders);  JButton resetButton = new JButton("Reset");  JButton submitButton = new JButton("Submit");  gbc.gridx = 0;  gbc.gridy = 0;  panel.add(firstNameLabel, gbc);  gbc.gridx = 1;  panel.add(firstNameField, gbc);  gbc.gridx = 0;  gbc.gridy = 1;  panel.add(lastNameLabel, gbc);  gbc.gridx = 1;  panel.add(lastNameField, gbc);  gbc.gridx = 0;  gbc.gridy = 2;  panel.add(emailLabel, gbc);  gbc.gridx = 1;  panel.add(emailField, gbc);  gbc.gridx = 0;  gbc.gridy = 3;  panel.add(passwordLabel, gbc);  gbc.gridx = 1;  panel.add(passwordField, gbc);  gbc.gridx = 0;  gbc.gridy = 4;  panel.add(confirmPasswordLabel, gbc);  gbc.gridx = 1;  panel.add(confirmPasswordField, gbc);  gbc.gridx = 0;  gbc.gridy = 5;  panel.add(countryLabel, gbc);  gbc.gridx = 1;  panel.add(countryComboBox, gbc);  gbc.gridx = 0;  gbc.gridy = 6;  panel.add(phoneLabel, gbc);  gbc.gridx = 1;  panel.add(phoneField, gbc);  gbc.gridx = 0;  gbc.gridy = 7;  panel.add(recoveryEmailLabel, gbc);  gbc.gridx = 1;  panel.add(recoveryEmailField, gbc);  gbc.gridx = 0;  gbc.gridy = 8;  panel.add(dobLabel, gbc);  gbc.gridx = 1;  panel.add(monthComboBox, gbc);  gbc.gridx = 2;  panel.add(dayComboBox, gbc);  gbc.gridx = 3;  panel.add(yearComboBox, gbc);  gbc.gridx = 0;  gbc.gridy = 9;  panel.add(genderLabel, gbc);  gbc.gridx = 1;  panel.add(genderComboBox, gbc);  gbc.gridx = 0;  gbc.gridy = 10;  gbc.gridwidth = 2;  panel.add(resetButton, gbc);  gbc.gridx = 2;  panel.add(submitButton, gbc);  frame.add(panel, BorderLayout.*CENTER*);  resetButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  *resetForm*(firstNameField, lastNameField, emailField, passwordField, confirmPasswordField,  countryComboBox, phoneField, recoveryEmailField, yearComboBox, monthComboBox, dayComboBox,  genderComboBox);  }  });  submitButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  try {  *validateForm*(firstNameField, lastNameField, emailField, passwordField, confirmPasswordField,  countryComboBox, phoneField, recoveryEmailField, yearComboBox, monthComboBox, dayComboBox,  genderComboBox);  JOptionPane.*showMessageDialog*(frame, "Account successfully created!");  } catch (Exception ex) {  JOptionPane.*showMessageDialog*(frame, ex.getMessage(), "Error", JOptionPane.*ERROR\_MESSAGE*);  }  }  });  frame.setVisible(true);  }  private static void resetForm(JTextField firstName, JTextField lastName, JTextField email, JPasswordField password,  JPasswordField confirmPassword,  JComboBox<String> country, JTextField phone, JTextField recoveryEmail, JComboBox<String> year,  JComboBox<String> month,  JComboBox<String> day, JComboBox<String> gender) {  firstName.setText("");  lastName.setText("");  email.setText("");  password.setText("");  confirmPassword.setText("");  country.setSelectedIndex(0);  phone.setText("");  recoveryEmail.setText("");  year.setSelectedIndex(0);  month.setSelectedIndex(0);  day.setSelectedIndex(0);  gender.setSelectedIndex(0);  }  private static void validateForm(JTextField firstName, JTextField lastName, JTextField email,  JPasswordField password, JPasswordField confirmPassword,  JComboBox<String> country, JTextField phone, JTextField recoveryEmail, JComboBox<String> year,  JComboBox<String> month,  JComboBox<String> day, JComboBox<String> gender) throws Exception {  if (firstName.getText().isEmpty() || lastName.getText().isEmpty() || email.getText().isEmpty()  || password.getPassword().length == 0 || confirmPassword.getPassword().length == 0  || country.getSelectedIndex() == 0 || phone.getText().isEmpty() || year.getSelectedIndex() == 0  || month.getSelectedIndex() == 0 || day.getSelectedIndex() == 0 || gender.getSelectedIndex() == 0) {  throw new Exception("All fields are required.");  }  if (!*isValidEmail*(email.getText())) {  throw new Exception("Invalid email address.");  }  if (!*isValidEmail*(recoveryEmail.getText())) {  throw new Exception("Invalid recovery email address.");  }  String passwordStr = new String(password.getPassword());  String confirmPasswordStr = new String(confirmPassword.getPassword());  if (passwordStr.length() < 8) {  throw new Exception("Password must be at least 8 characters long.");  }  if (!passwordStr.equals(confirmPasswordStr)) {  throw new Exception("Passwords do not match.");  }  }  private static boolean isValidEmail(String email) {  String regex = "^(.+)@(.+)\\.com$";  Pattern pattern = Pattern.*compile*(regex);  Matcher matcher = pattern.matcher(email);  return matcher.matches();  }  private static int getMaxDaysForMonth(String month, int year) {  switch (month) {  case "April":  case "June":  case "September":  case "November":  return 30;  case "February":  if (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0)) {  return 29; // Leap year  } else {  return 28;  }  default:  return 31;  }  } } |

**OUTPUT**

