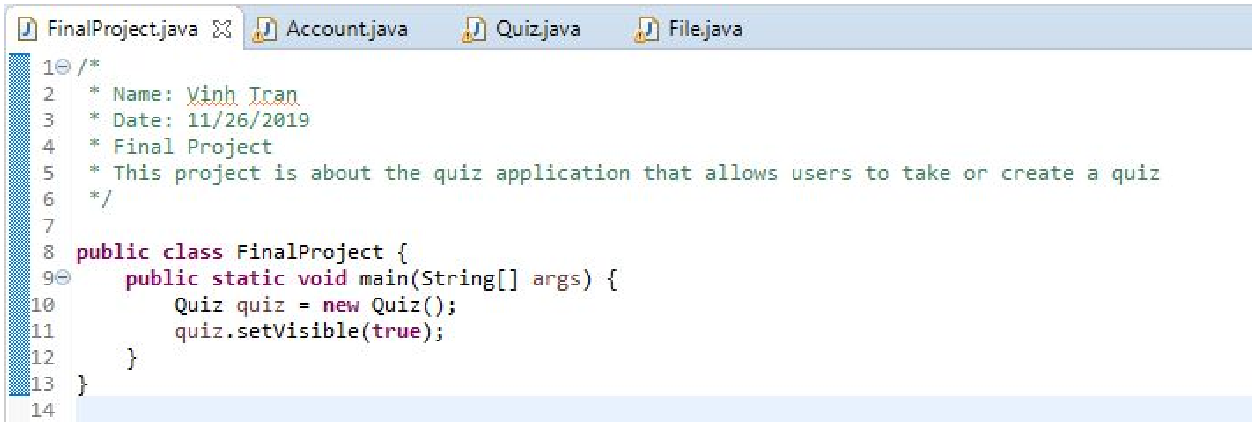
# Final Project - Quiz Application

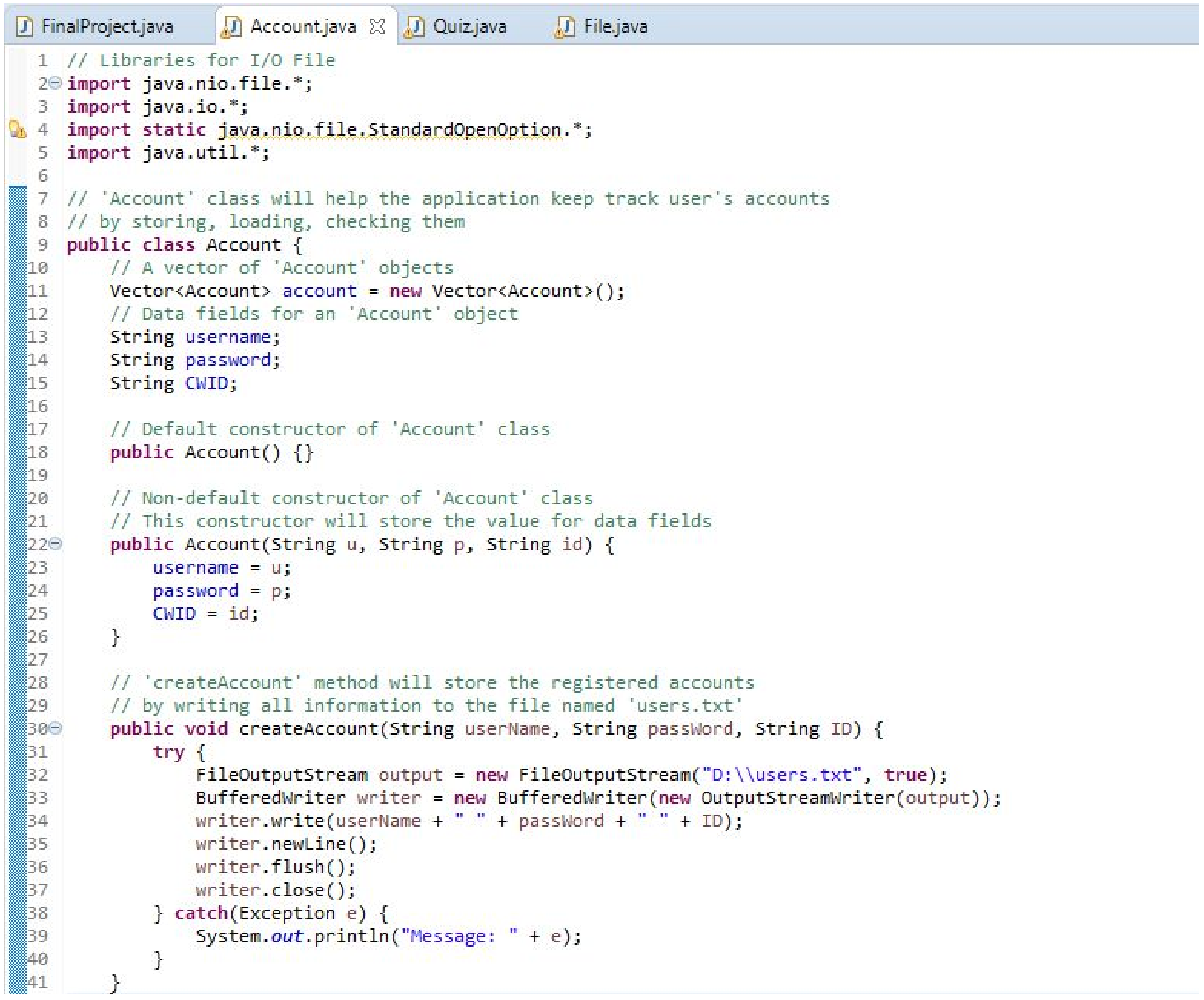
## Code:

## ‘FinalProject.java’

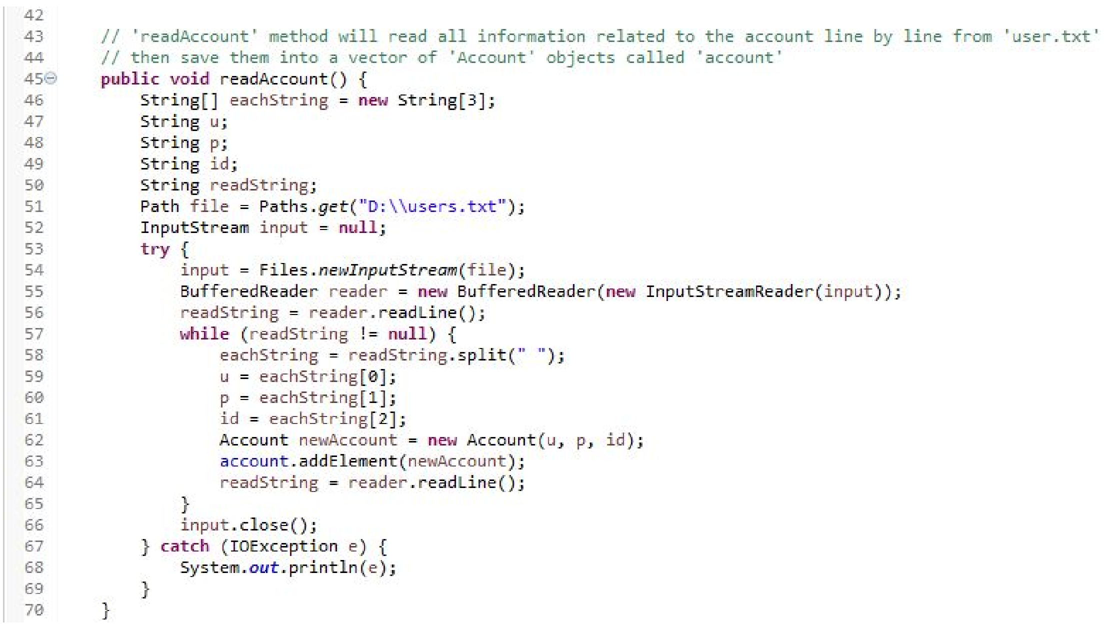

FinalProject.java | Account.java | Quiz.java | File.java

```java
/*
 * Name: Vinh Tran
 * Date: 11/26/2019
 * Final Project
 * This project is about the quiz application that allows users to take or create a quiz
 */

public class FinalProject {
    public static void main(String[] args) {
        Quiz quiz = new Quiz();
        quiz.setVisible(true);
    }
}
```

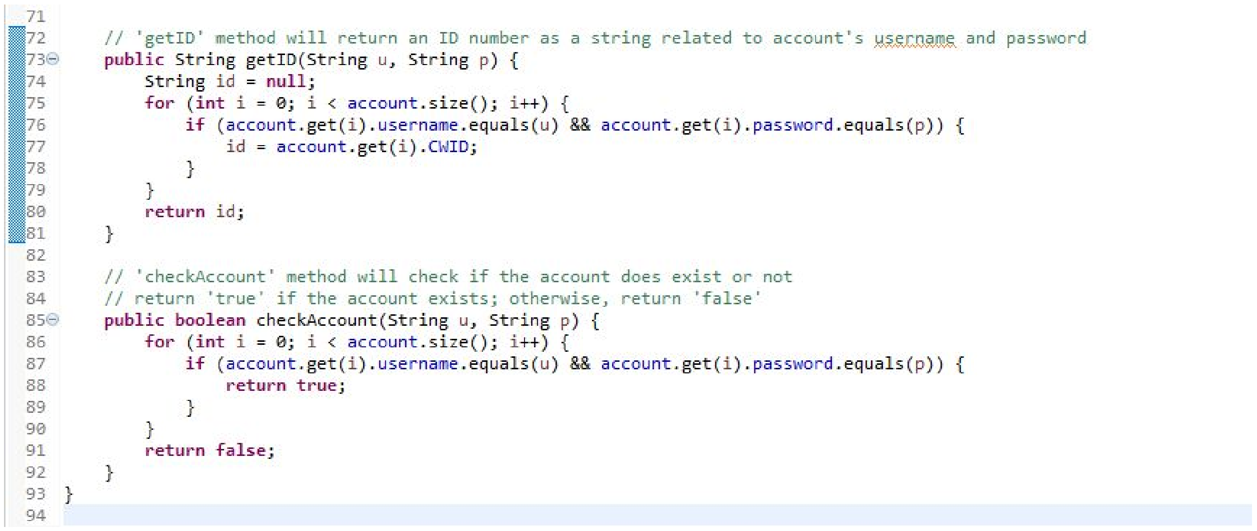
## ‘Account.java’


FinalProject.java | Account.java | Quiz.java | File.java

```java
// Libraries for I/O File
import java.nio.file.*;
import java.io.*;
import static java.nio.file.StandardOpenOption.*;
import java.util.*;

// 'Account' class will help the application keep track user's accounts
// by storing, loading, checking them
public class Account {
    // A vector of 'Account' objects
    Vector<Account> account = new Vector<Account>();
    // Data fields for an 'Account' object
    String username;
    String password;
    String CWID;

    // Default constructor of 'Account' class
    public Account() {}

    // Non-default constructor of 'Account' class
    // This constructor will store the value for data fields
    public Account(String u, String p, String id) {
        username = u;
        password = p;
        CWID = id;
    }

    // 'createAccount' method will store the registered accounts
    // by writing all information to the file named 'users.txt'
    public void createAccount(String userName, String passWord, String ID) {
        try {
            FileOutputStream output = new FileOutputStream("D:\\users.txt", true);
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(output));
            writer.write(userName + " " + passWord + " " + ID);
            writer.newLine();
            writer.flush();
            writer.close();
        } catch(Exception e) {
            System.out.println("Message: " + e);
        }
    }
```

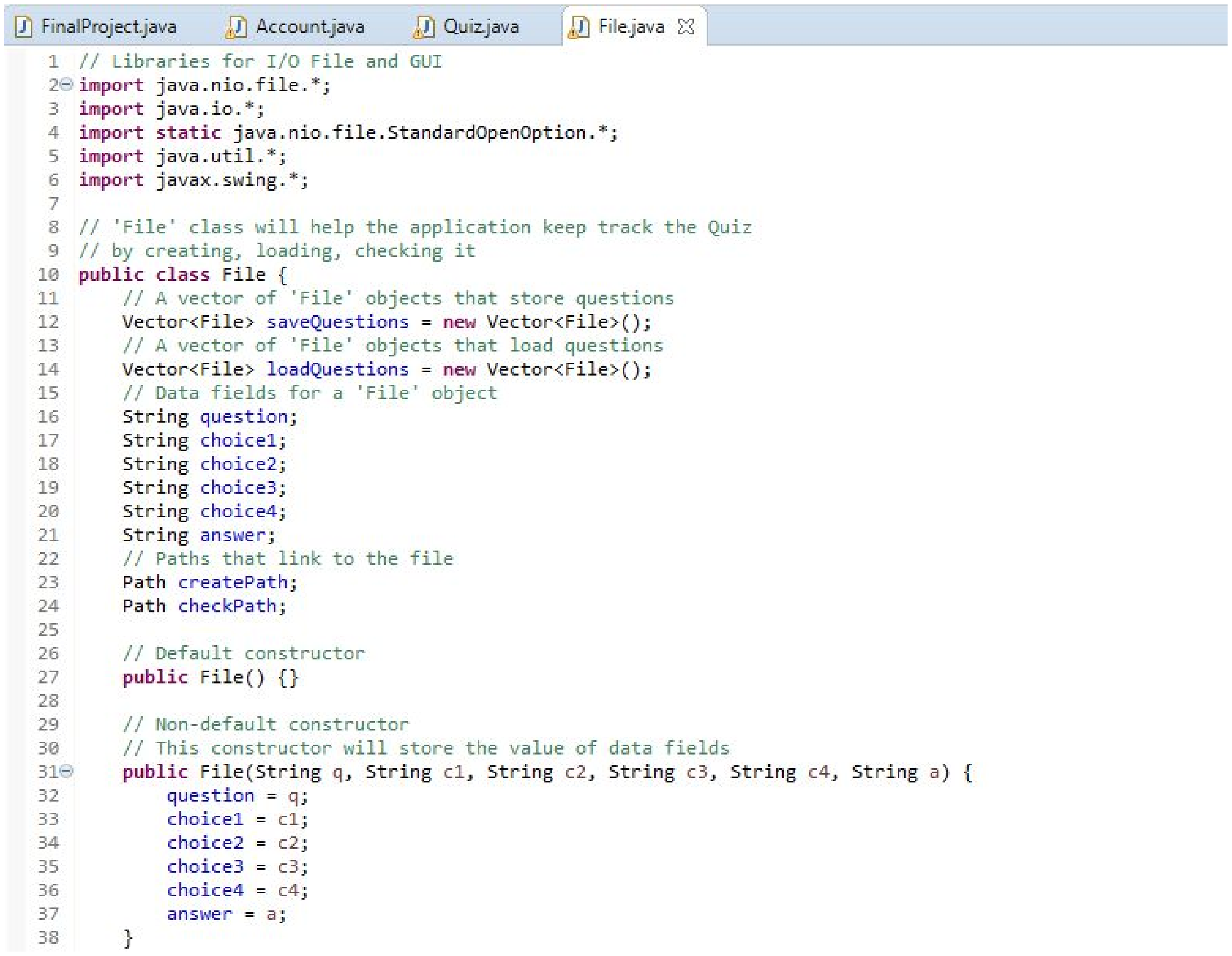


```java

    // 'readAccount' method will read all information related to the account line by line from 'user.txt'
    // then save them into a vector of 'Account' objects called 'account'
    public void readAccount() {
        String[] eachString = new String[3];
        String u;
        String p;
        String id;
        String readString;
        Path file = Paths.get("D:\\users.txt");
        InputStream input = null;
        try {
            input = Files.newInputStream(file);
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));
            readString = reader.readLine();
            while (readString != null) {
                eachString = readString.split(" ");
                u = eachString[0];
                p = eachString[1];
                id = eachString[2];
                Account newAccount = new Account(u, p, id);
                account.addElement(newAccount);
                readString = reader.readLine();
            }
            input.close();
        } catch (IOException e) {
            System.out.println(e);
        }
    }
```

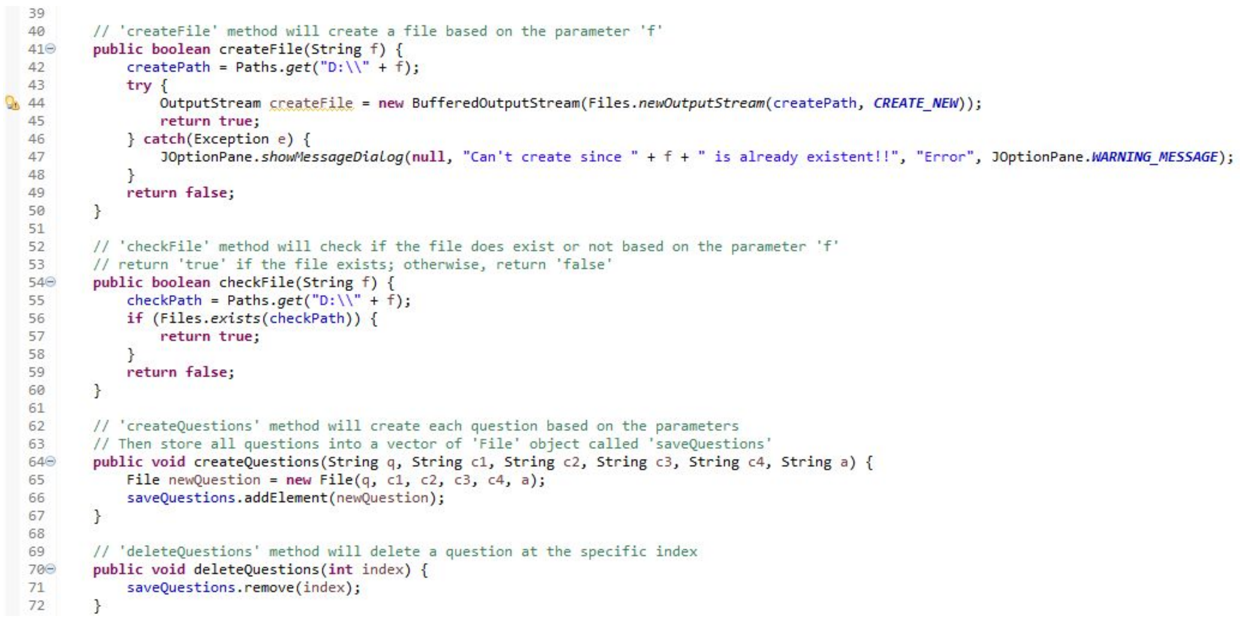


```java

    // 'getID' method will return an ID number as a string related to account's username and password
    public String getID(String u, String p) {
        String id = null;
        for (int i = 0; i < account.size(); i++) {
            if (account.get(i).username.equals(u) && account.get(i).password.equals(p)) {
                id = account.get(i).CWID;
            }
        }
        return id;
    }

    // 'checkAccount' method will check if the account does exist or not
    // return 'true' if the account exists; otherwise, return 'false'
    public boolean checkAccount(String u, String p) {
        for (int i = 0; i < account.size(); i++) {
            if (account.get(i).username.equals(u) && account.get(i).password.equals(p)) {
                return true;
            }
        }
        return false;
    }
}

```

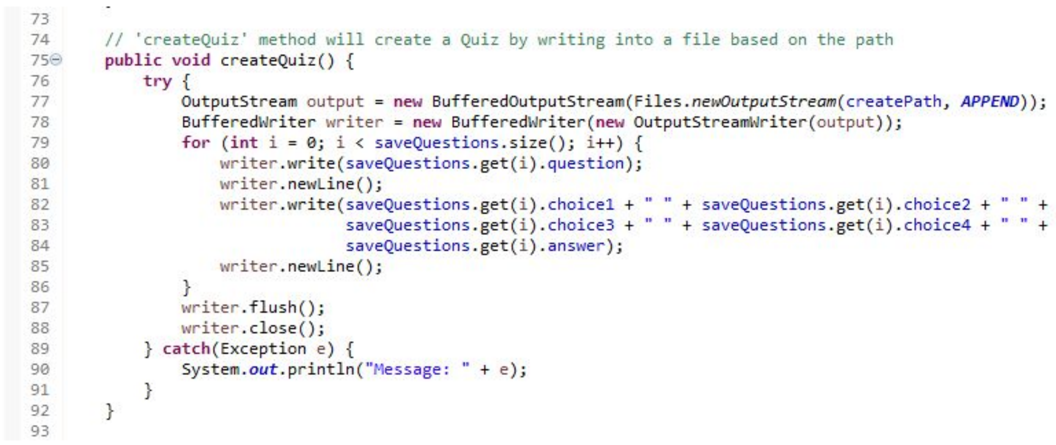
**‘File.java’**



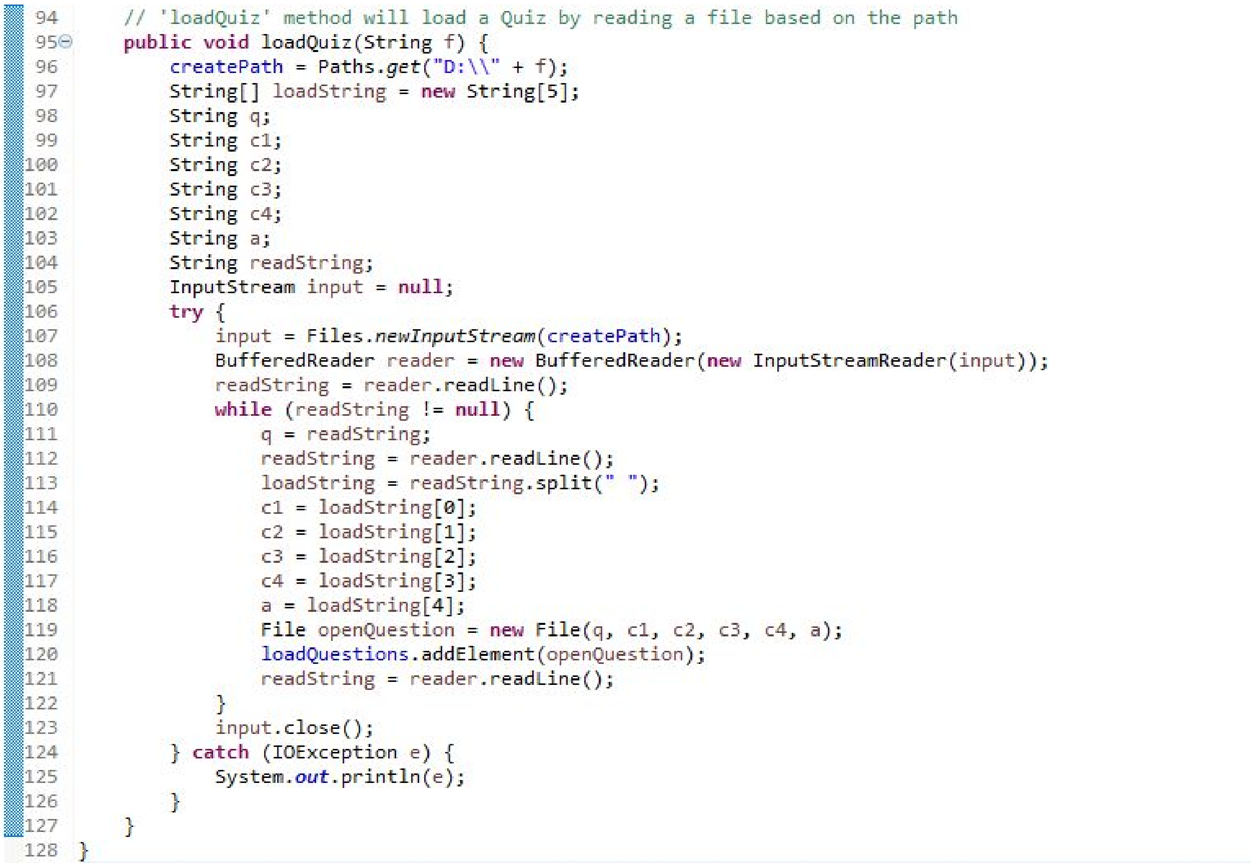
```java
// Libraries for I/O File and GUI
import java.nio.file.*;
import java.io.*;
import static java.nio.file.StandardOpenOption.*;
import java.util.*;
import javax.swing.*;

// 'File' class will help the application keep track the Quiz
// by creating, loading, checking it
public class File {
    // A vector of 'File' objects that store questions
    Vector<File> saveQuestions = new Vector<File>();
    // A vector of 'File' objects that load questions
    Vector<File> loadQuestions = new Vector<File>();
    // Data fields for a 'File' object
    String question;
    String choice1;
    String choice2;
    String choice3;
    String choice4;
    String answer;
    // Paths that link to the file
    Path createPath;
    Path checkPath;

    // Default constructor
    public File() {}

    // Non-default constructor
    // This constructor will store the value of data fields
    public File(String q, String c1, String c2, String c3, String c4, String a) {
        question = q;
        choice1 = c1;
        choice2 = c2;
        choice3 = c3;
        choice4 = c4;
        answer = a;
    }
```



```java

    // 'createFile' method will create a file based on the parameter 'f'
    public boolean createFile(String f) {
        createPath = Paths.get("D:\\" + f);
        try {
            OutputStream createFile = new BufferedOutputStream(Files.newOutputStream(createPath, CREATE_NEW));
            return true;
        } catch(Exception e) {
            JOptionPane.showMessageDialog(null, "Can't create since " + f + " is already existent!!", "Error", JOptionPane.WARNING_MESSAGE);
        }
        return false;
    }

    // 'checkFile' method will check if the file does exist or not based on the parameter 'f'
    // return 'true' if the file exists; otherwise, return 'false'
    public boolean checkFile(String f) {
        checkPath = Paths.get("D:\\" + f);
        if (Files.exists(checkPath)) {
            return true;
        }
        return false;
    }

    // 'createQuestions' method will create each question based on the parameters
    // Then store all questions into a vector of 'File' object called 'saveQuestions'
    public void createQuestions(String q, String c1, String c2, String c3, String c4, String a) {
        File newQuestion = new File(q, c1, c2, c3, c4, a);
        saveQuestions.addElement(newQuestion);
    }

    // 'deleteQuestions' method will delete a question at the specific index
    public void deleteQuestions(int index) {
        saveQuestions.remove(index);
    }
```

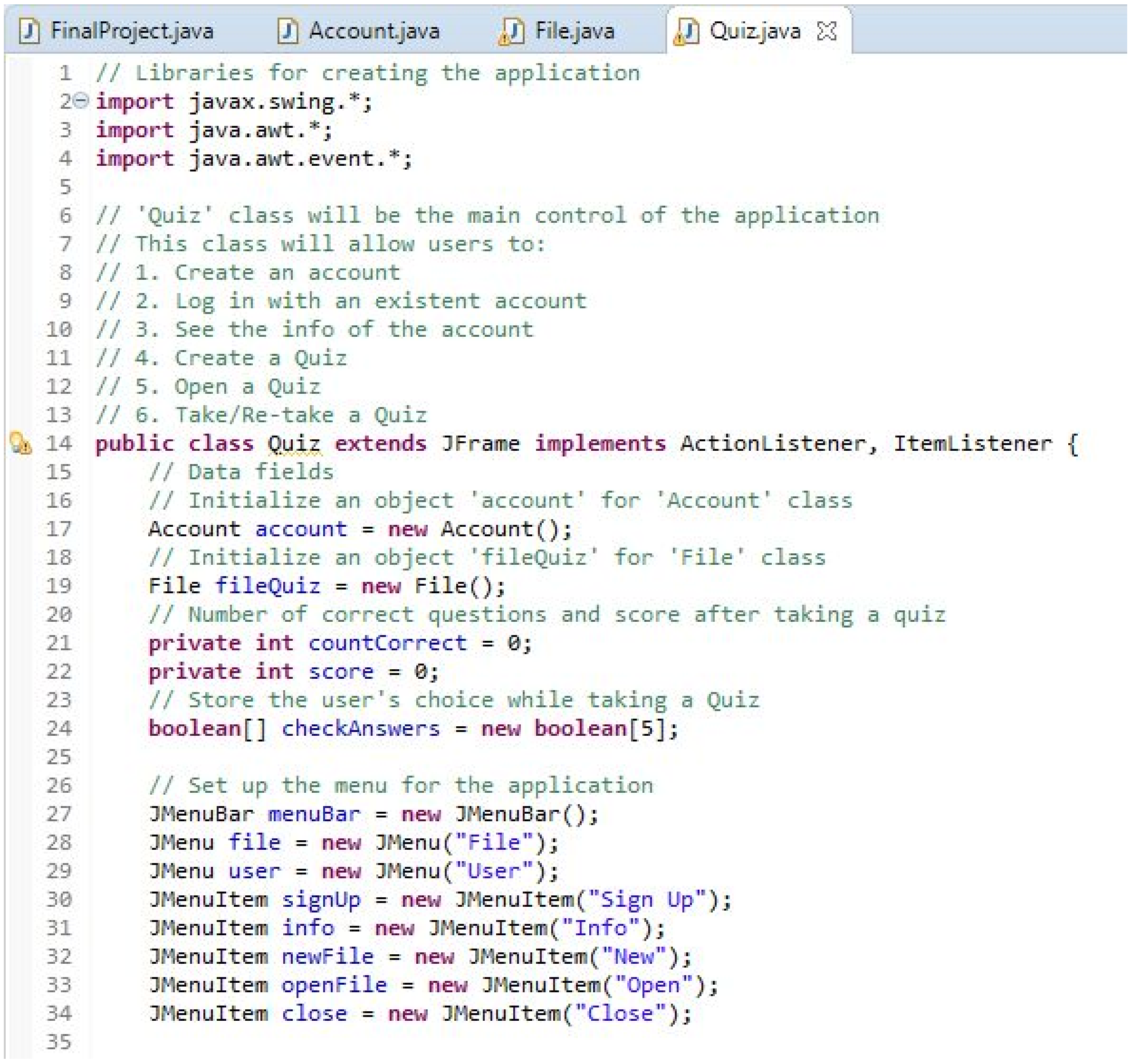


```java
    // 'createQuiz' method will create a Quiz by writing into a file based on the path
    public void createQuiz() {
        try {
            OutputStream output = new BufferedOutputStream(Files.newOutputStream(createPath, APPEND));
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(output));
            for (int i = 0; i < saveQuestions.size(); i++) {
                writer.write(saveQuestions.get(i).question);
                writer.newLine();
                writer.write(saveQuestions.get(i).choice1 + " " + saveQuestions.get(i).choice2 + " " +
                             saveQuestions.get(i).choice3 + " " + saveQuestions.get(i).choice4 + " " +
                             saveQuestions.get(i).answer);
                writer.newLine();
            }
            writer.flush();
            writer.close();
        } catch(Exception e) {
            System.out.println("Message: " + e);
        }
    }

```

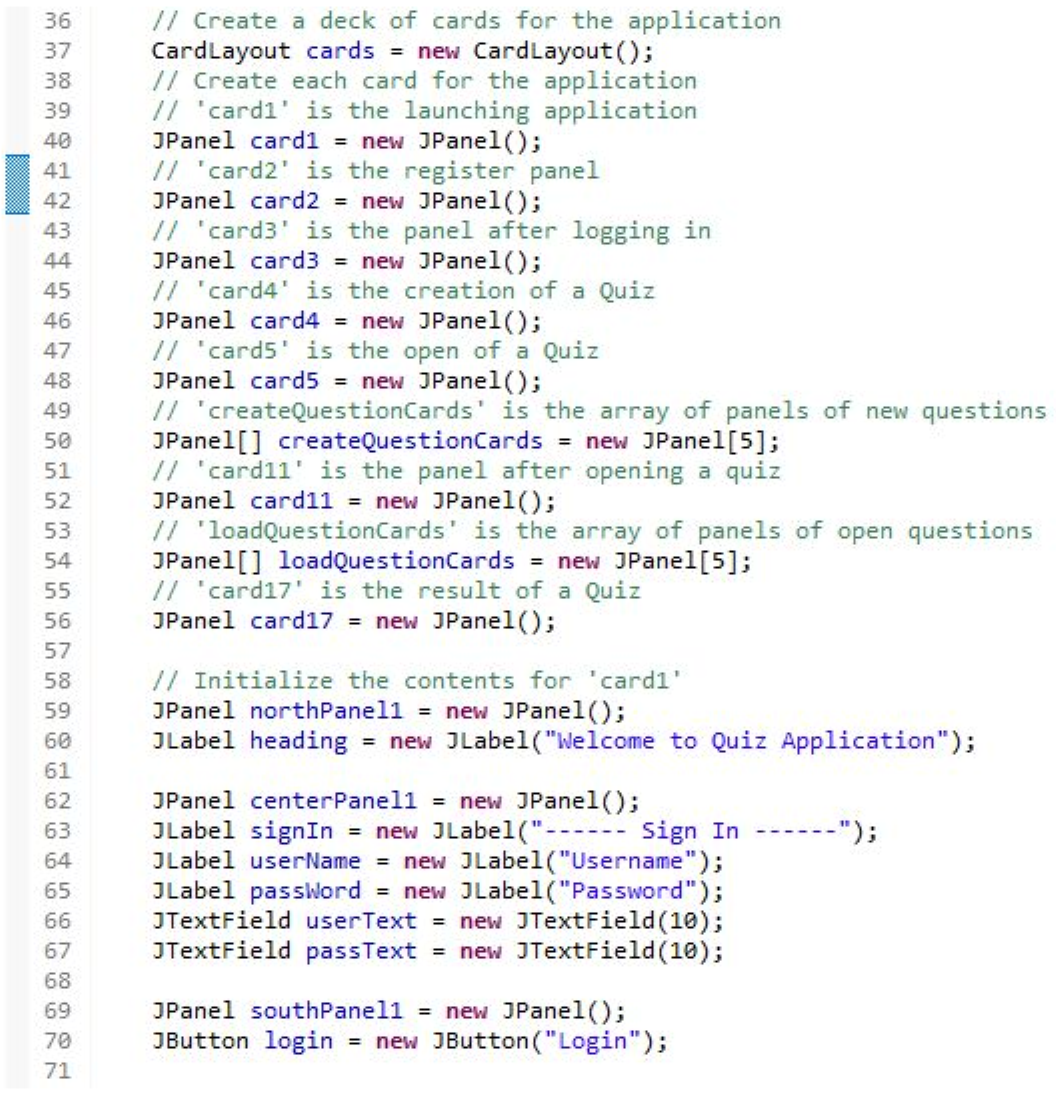


```java
    // 'loadQuiz' method will load a Quiz by reading a file based on the path
    public void loadQuiz(String f) {
        createPath = Paths.get("D:\\" + f);
        String[] loadString = new String[5];
        String q;
        String c1;
        String c2;
        String c3;
        String c4;
        String a;
        String readString;
        InputStream input = null;
        try {
            input = Files.newInputStream(createPath);
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));
            readString = reader.readLine();
            while (readString != null) {
                q = readString;
                readString = reader.readLine();
                loadString = readString.split(" ");
                c1 = loadString[0];
                c2 = loadString[1];
                c3 = loadString[2];
                c4 = loadString[3];
                a = loadString[4];
                File openQuestion = new File(q, c1, c2, c3, c4, a);
                loadQuestions.addElement(openQuestion);
                readString = reader.readLine();
            }
            input.close();
        } catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

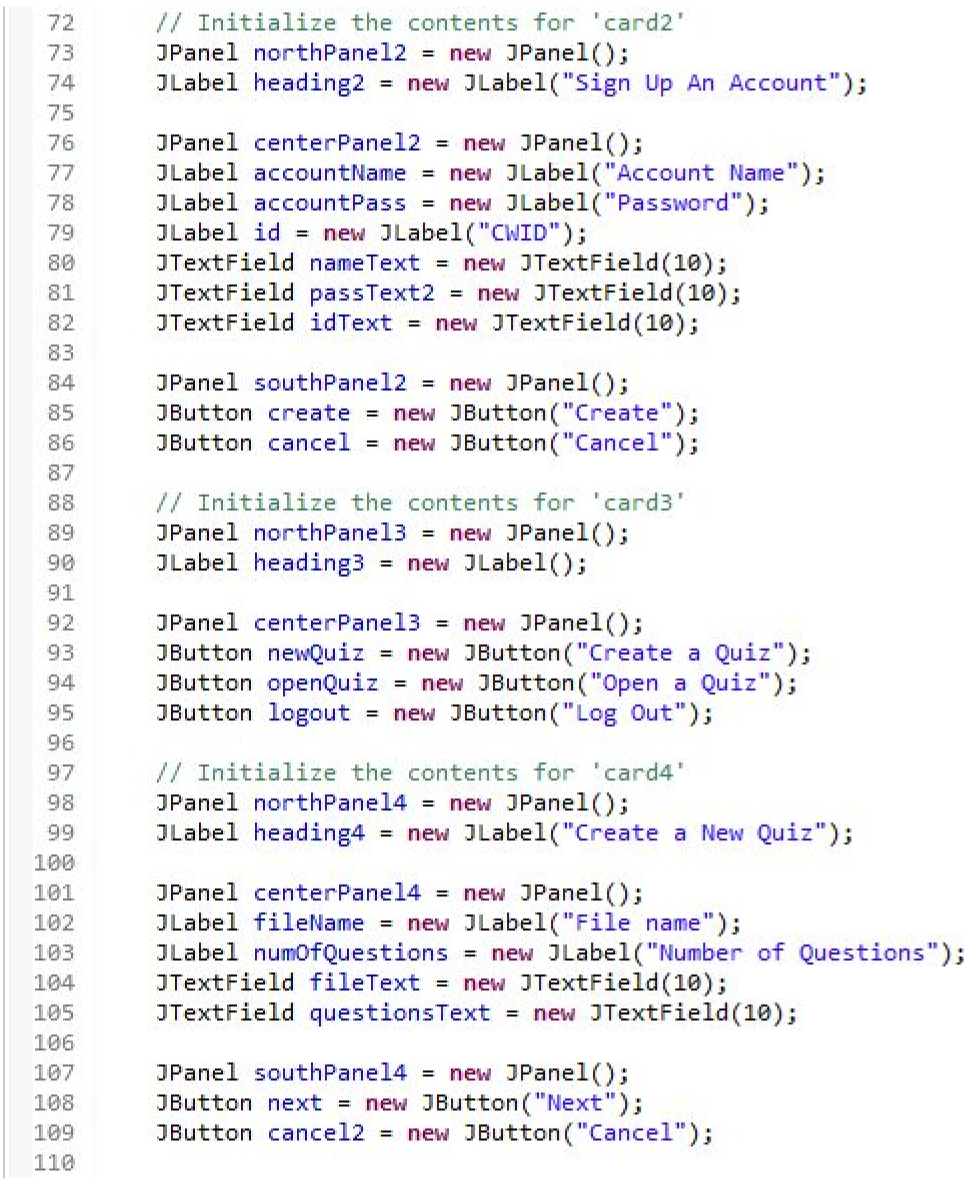
**‘Quiz.java’**



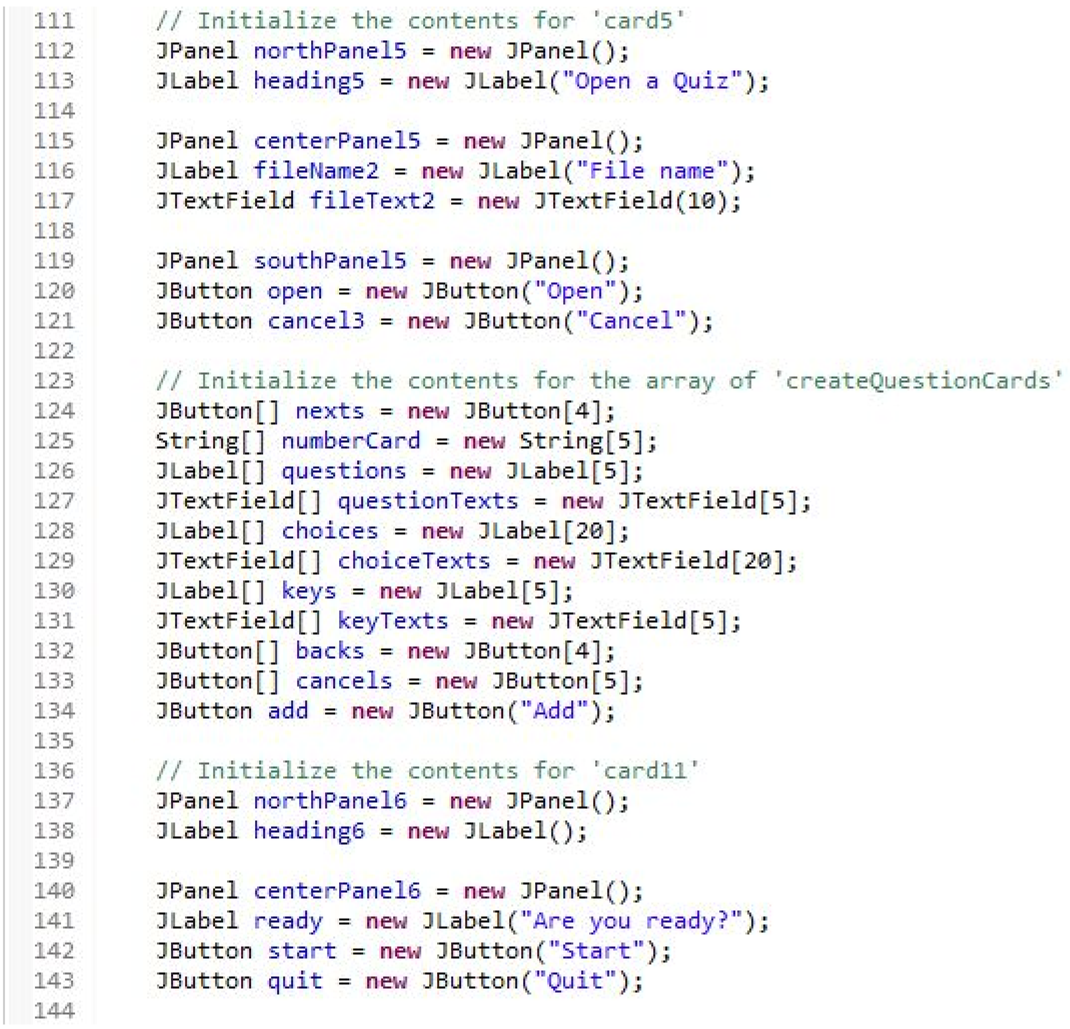
```java
// Libraries for creating the application
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

// 'Quiz' class will be the main control of the application
// This class will allow users to:
// 1. Create an account
// 2. Log in with an existent account
// 3. See the info of the account
// 4. Create a Quiz
// 5. Open a Quiz
// 6. Take/Re-take a Quiz
public class Quiz extends JFrame implements ActionListener, ItemListener {
    // Data fields
    // Initialize an object 'account' for 'Account' class
    Account account = new Account();
    // Initialize an object 'fileQuiz' for 'File' class
    File fileQuiz = new File();
    // Number of correct questions and score after taking a quiz
    private int countCorrect = 0;
    private int score = 0;
    // Store the user's choice while taking a Quiz
    boolean[] checkAnswers = new boolean[5];

    // Set up the menu for the application
    JMenuBar menuBar = new JMenuBar();
    JMenu file = new JMenu("File");
    JMenu user = new JMenu("User");
    JMenuItem signUp = new JMenuItem("Sign Up");
    JMenuItem info = new JMenuItem("Info");
    JMenuItem newFile = new JMenuItem("New");
    JMenuItem openFile = new JMenuItem("Open");
    JMenuItem close = new JMenuItem("Close");

```



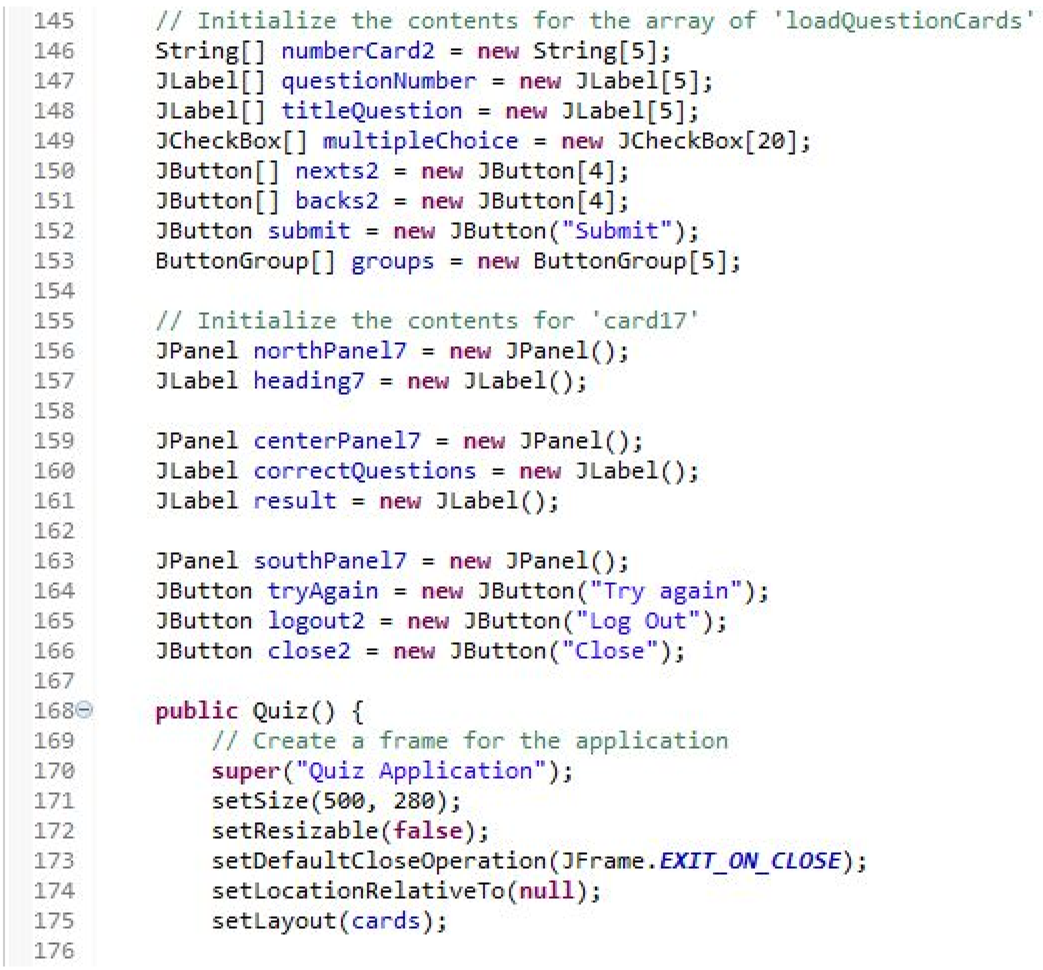
```java
    // Create a deck of cards for the application
    CardLayout cards = new CardLayout();
    // Create each card for the application
    // 'card1' is the launching application
    JPanel card1 = new JPanel();
    // 'card2' is the register panel
    JPanel card2 = new JPanel();
    // 'card3' is the panel after logging in
    JPanel card3 = new JPanel();
    // 'card4' is the creation of a Quiz
    JPanel card4 = new JPanel();
    // 'card5' is the open of a Quiz
    JPanel card5 = new JPanel();
    // 'createQuestionCards' is the array of panels of new questions
    JPanel[] createQuestionCards = new JPanel[5];
    // 'card11' is the panel after opening a quiz
    JPanel card11 = new JPanel();
    // 'loadQuestionCards' is the array of panels of open questions
    JPanel[] loadQuestionCards = new JPanel[5];
    // 'card17' is the result of a Quiz
    JPanel card17 = new JPanel();

    // Initialize the contents for 'card1'
    JPanel northPanel1 = new JPanel();
    JLabel heading = new JLabel("Welcome to Quiz Application");

    JPanel centerPanel1 = new JPanel();
    JLabel signIn = new JLabel("------ Sign In ------");
    JLabel userName = new JLabel("Username");
    JLabel passWord = new JLabel("Password");
    JTextField userText = new JTextField(10);
    JTextField passText = new JTextField(10);

    JPanel southPanel1 = new JPanel();
    JButton login = new JButton("Login");
```



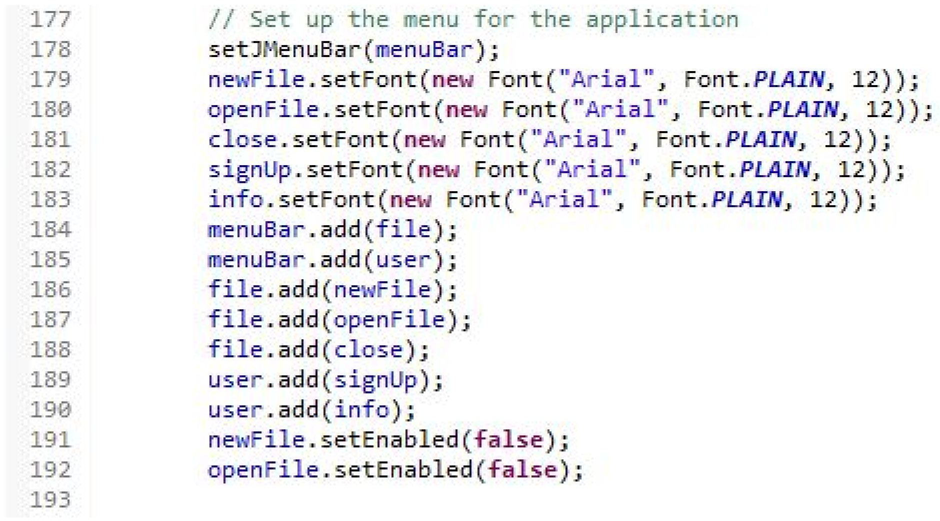
```java
    // Initialize the contents for 'card2'
    JPanel northPanel2 = new JPanel();
    JLabel heading2 = new JLabel("Sign Up An Account");

    JPanel centerPanel2 = new JPanel();
    JLabel accountName = new JLabel("Account Name");
    JLabel accountPass = new JLabel("Password");
    JLabel id = new JLabel("CWID");
    JTextField nameText = new JTextField(10);
    JTextField passText2 = new JTextField(10);
    JTextField idText = new JTextField(10);

    JPanel southPanel2 = new JPanel();
    JButton create = new JButton("Create");
    JButton cancel = new JButton("Cancel");

    // Initialize the contents for 'card3'
    JPanel northPanel3 = new JPanel();
    JLabel heading3 = new JLabel();

    JPanel centerPanel3 = new JPanel();
    JButton newQuiz = new JButton("Create a Quiz");
    JButton openQuiz = new JButton("Open a Quiz");
    JButton logout = new JButton("Log Out");

    // Initialize the contents for 'card4'
    JPanel northPanel4 = new JPanel();
    JLabel heading4 = new JLabel("Create a New Quiz");

    JPanel centerPanel4 = new JPanel();
    JLabel fileName = new JLabel("File name");
    JLabel numOfQuestions = new JLabel("Number of Questions");
    JTextField fileText = new JTextField(10);
    JTextField questionsText = new JTextField(10);

    JPanel southPanel4 = new JPanel();
    JButton next = new JButton("Next");
    JButton cancel2 = new JButton("Cancel");

```



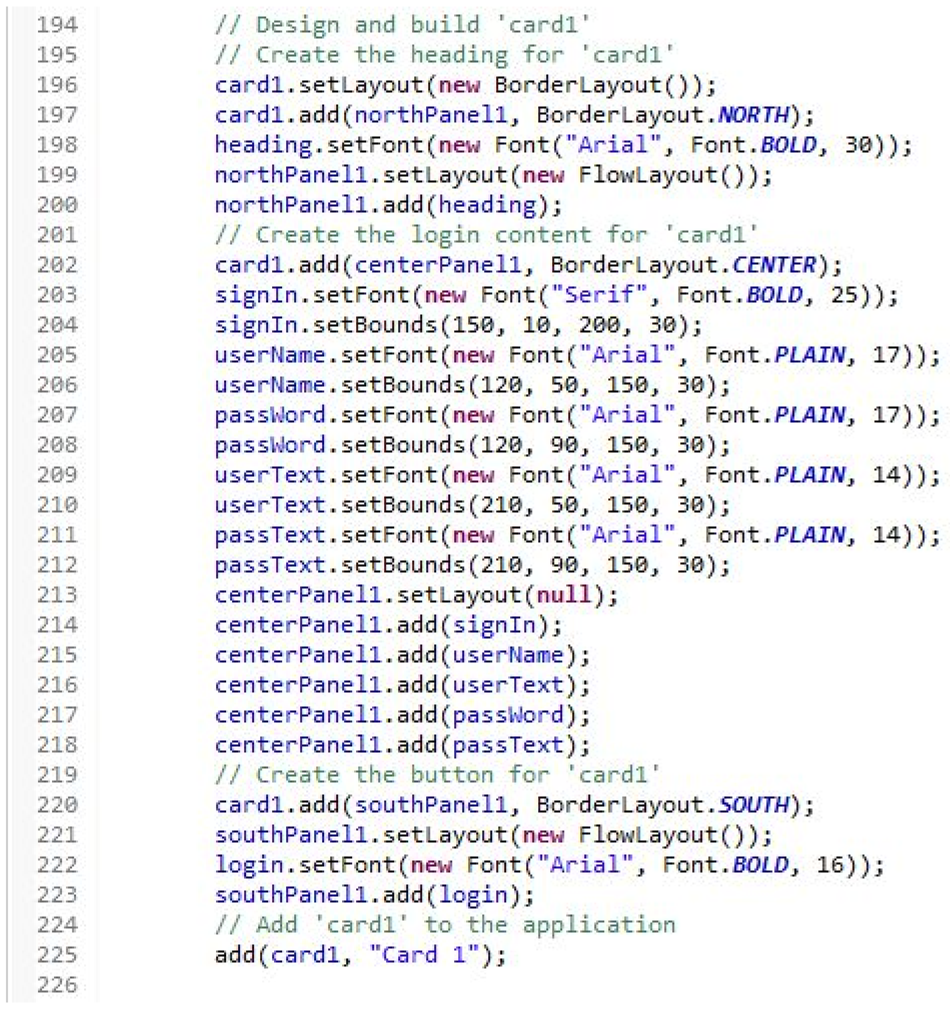
```java
    // Initialize the contents for 'card5'
    JPanel northPanel5 = new JPanel();
    JLabel heading5 = new JLabel("Open a Quiz");

    JPanel centerPanel5 = new JPanel();
    JLabel fileName2 = new JLabel("File name");
    JTextField fileText2 = new JTextField(10);

    JPanel southPanel5 = new JPanel();
    JButton open = new JButton("Open");
    JButton cancel3 = new JButton("Cancel");

    // Initialize the contents for the array of 'createQuestionCards'
    JButton[] nexts = new JButton[4];
    String[] numberCard = new String[5];
    JLabel[] questions = new JLabel[5];
    JTextField[] questionTexts = new JTextField[5];
    JLabel[] choices = new JLabel[20];
    JTextField[] choiceTexts = new JTextField[20];
    JLabel[] keys = new JLabel[5];
    JTextField[] keyTexts = new JTextField[5];
    JButton[] backs = new JButton[4];
    JButton[] cancels = new JButton[5];
    JButton add = new JButton("Add");

    // Initialize the contents for 'card11'
    JPanel northPanel6 = new JPanel();
    JLabel heading6 = new JLabel();

    JPanel centerPanel6 = new JPanel();
    JLabel ready = new JLabel("Are you ready?");
    JButton start = new JButton("Start");
    JButton quit = new JButton("Quit");

```



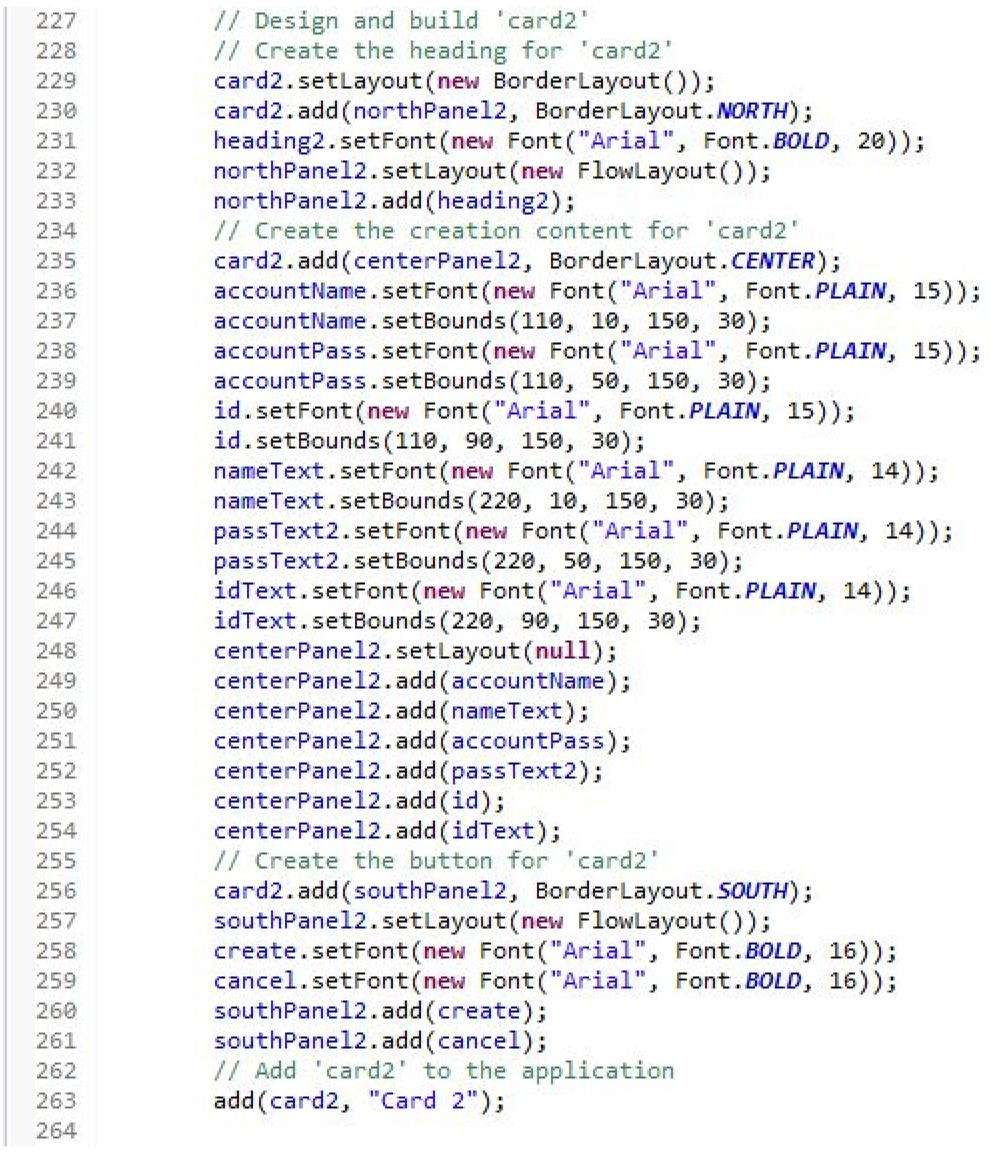
```java
    // Initialize the contents for the array of 'loadQuestionCards'
    String[] numberCard2 = new String[5];
    JLabel[] questionNumber = new JLabel[5];
    JLabel[] titleQuestion = new JLabel[5];
    JCheckBox[] multipleChoice = new JCheckBox[20];
    JButton[] nexts2 = new JButton[4];
    JButton[] backs2 = new JButton[4];
    JButton submit = new JButton("Submit");
    ButtonGroup[] groups = new ButtonGroup[5];

    // Initialize the contents for 'card17'
    JPanel northPanel7 = new JPanel();
    JLabel heading7 = new JLabel();

    JPanel centerPanel7 = new JPanel();
    JLabel correctQuestions = new JLabel();
    JLabel result = new JLabel();

    JPanel southPanel7 = new JPanel();
    JButton tryAgain = new JButton("Try again");
    JButton logout2 = new JButton("Log Out");
    JButton close2 = new JButton("Close");

    public Quiz() {
        // Create a frame for the application
        super("Quiz Application");
        setSize(500, 280);
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(cards);

```



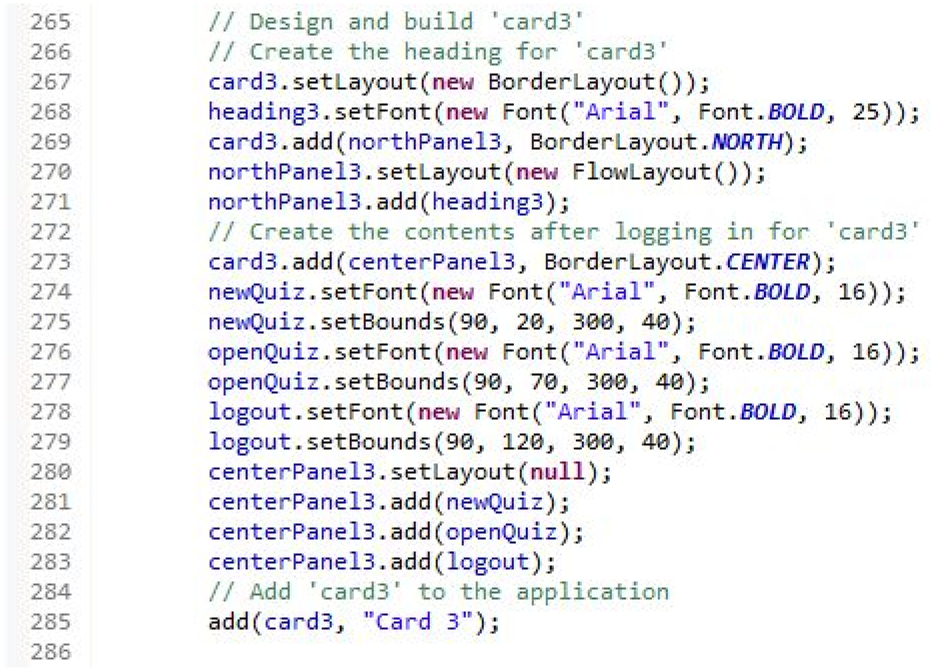
```java
        // Set up the menu for the application
        setJMenuBar(menuBar);
        newFile.setFont(new Font("Arial", Font.PLAIN, 12));
        openFile.setFont(new Font("Arial", Font.PLAIN, 12));
        close.setFont(new Font("Arial", Font.PLAIN, 12));
        signUp.setFont(new Font("Arial", Font.PLAIN, 12));
        info.setFont(new Font("Arial", Font.PLAIN, 12));
        menuBar.add(file);
        menuBar.add(user);
        file.add(newFile);
        file.add(openFile);
        file.add(close);
        user.add(signUp);
        user.add(info);
        newFile.setEnabled(false);
        openFile.setEnabled(false);

```



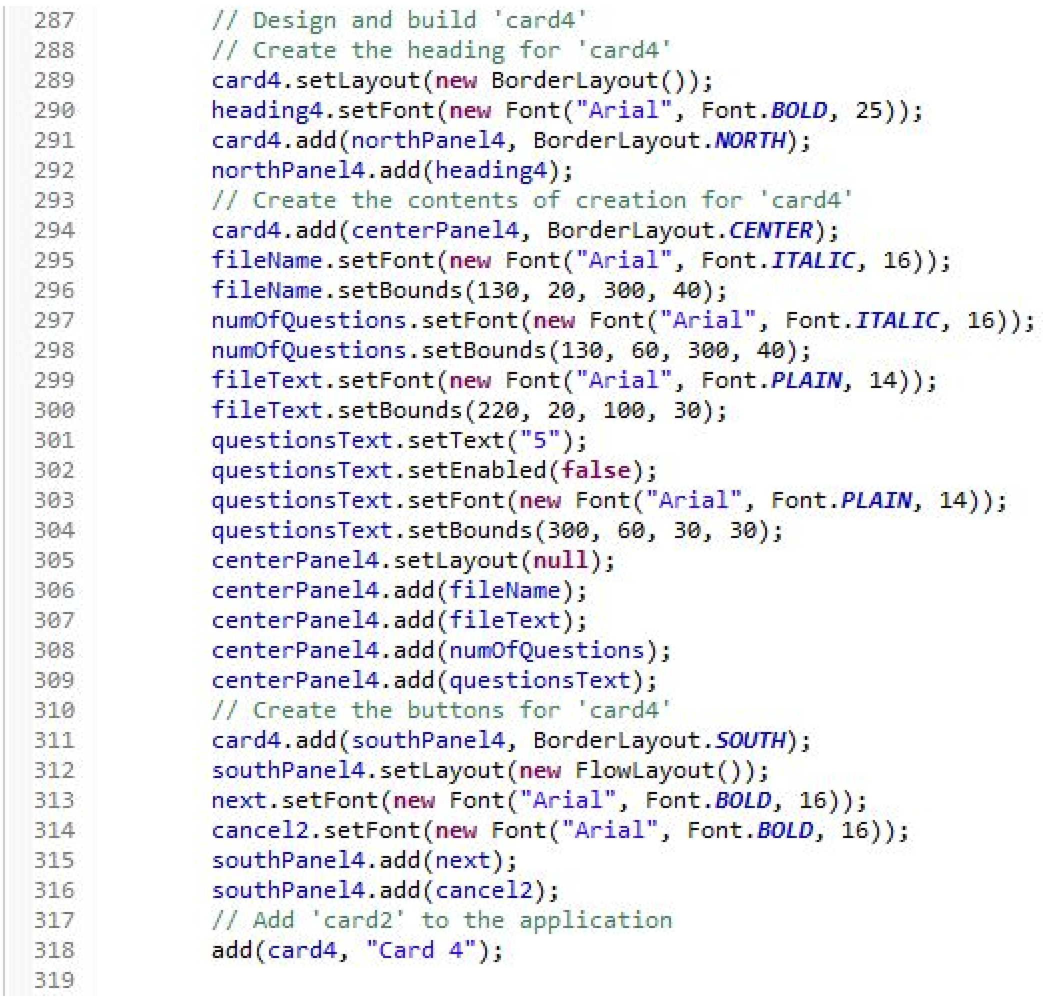
```java
        // Design and build 'card1'
        // Create the heading for 'card1'
        card1.setLayout(new BorderLayout());
        card1.add(northPanel1, BorderLayout.NORTH);
        heading.setFont(new Font("Arial", Font.BOLD, 30));
        northPanel1.setLayout(new FlowLayout());
        northPanel1.add(heading);
        // Create the login content for 'card1'
        card1.add(centerPanel1, BorderLayout.CENTER);
        signIn.setFont(new Font("Serif", Font.BOLD, 25));
        signIn.setBounds(150, 10, 200, 30);
        userName.setFont(new Font("Arial", Font.PLAIN, 17));
        userName.setBounds(120, 50, 150, 30);
        passWord.setFont(new Font("Arial", Font.PLAIN, 17));
        passWord.setBounds(120, 90, 150, 30);
        userText.setFont(new Font("Arial", Font.PLAIN, 14));
        userText.setBounds(210, 50, 150, 30);
        passText.setFont(new Font("Arial", Font.PLAIN, 14));
        passText.setBounds(210, 90, 150, 30);
        centerPanel1.setLayout(null);
        centerPanel1.add(signIn);
        centerPanel1.add(userName);
        centerPanel1.add(userText);
        centerPanel1.add(passWord);
        centerPanel1.add(passText);
        // Create the button for 'card1'
        card1.add(southPanel1, BorderLayout.SOUTH);
        southPanel1.setLayout(new FlowLayout());
        login.setFont(new Font("Arial", Font.BOLD, 16));
        southPanel1.add(login);
        // Add 'card1' to the application
        add(card1, "Card 1");

```



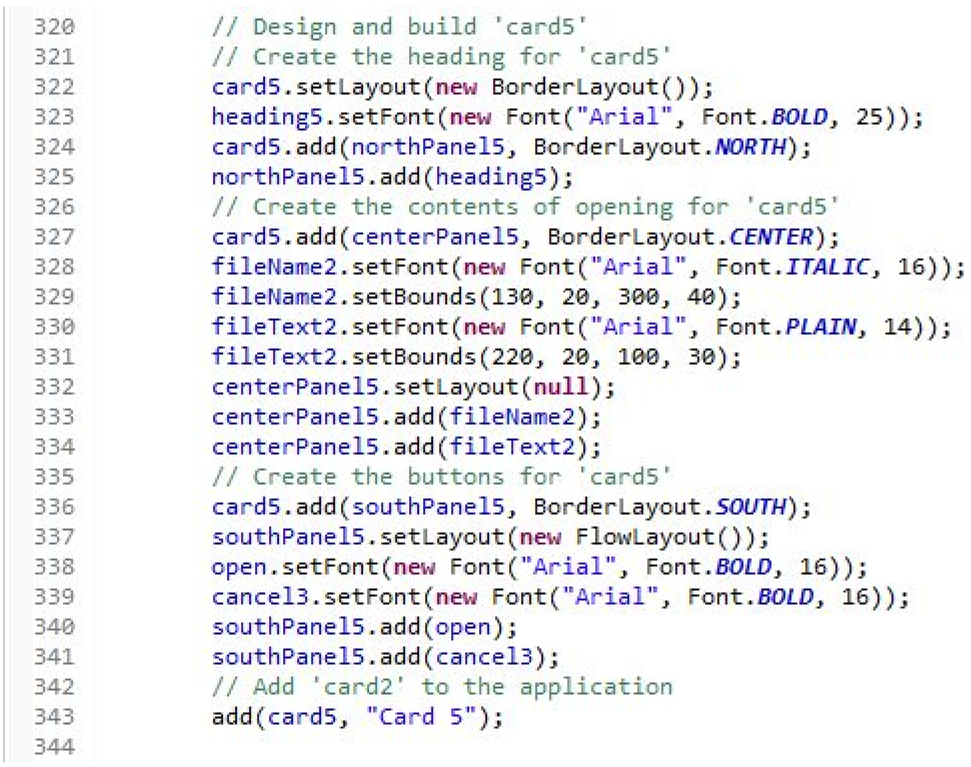
```java
        // Design and build 'card2'
        // Create the heading for 'card2'
        card2.setLayout(new BorderLayout());
        card2.add(northPanel2, BorderLayout.NORTH);
        heading2.setFont(new Font("Arial", Font.BOLD, 20));
        northPanel2.setLayout(new FlowLayout());
        northPanel2.add(heading2);
        // Create the creation content for 'card2'
        card2.add(centerPanel2, BorderLayout.CENTER);
        accountName.setFont(new Font("Arial", Font.PLAIN, 15));
        accountName.setBounds(110, 10, 150, 30);
        accountPass.setFont(new Font("Arial", Font.PLAIN, 15));
        accountPass.setBounds(110, 50, 150, 30);
        id.setFont(new Font("Arial", Font.PLAIN, 15));
        id.setBounds(110, 90, 150, 30);
        nameText.setFont(new Font("Arial", Font.PLAIN, 14));
        nameText.setBounds(220, 10, 150, 30);
        passText2.setFont(new Font("Arial", Font.PLAIN, 14));
        passText2.setBounds(220, 50, 150, 30);
        idText.setFont(new Font("Arial", Font.PLAIN, 14));
        idText.setBounds(220, 90, 150, 30);
        centerPanel2.setLayout(null);
        centerPanel2.add(accountName);
        centerPanel2.add(nameText);
        centerPanel2.add(accountPass);
        centerPanel2.add(passText2);
        centerPanel2.add(id);
        centerPanel2.add(idText);
        // Create the button for 'card2'
        card2.add(southPanel2, BorderLayout.SOUTH);
        southPanel2.setLayout(new FlowLayout());
        create.setFont(new Font("Arial", Font.BOLD, 16));
        cancel.setFont(new Font("Arial", Font.BOLD, 16));
        southPanel2.add(create);
        southPanel2.add(cancel);
        // Add 'card2' to the application
        add(card2, "Card 2");

```



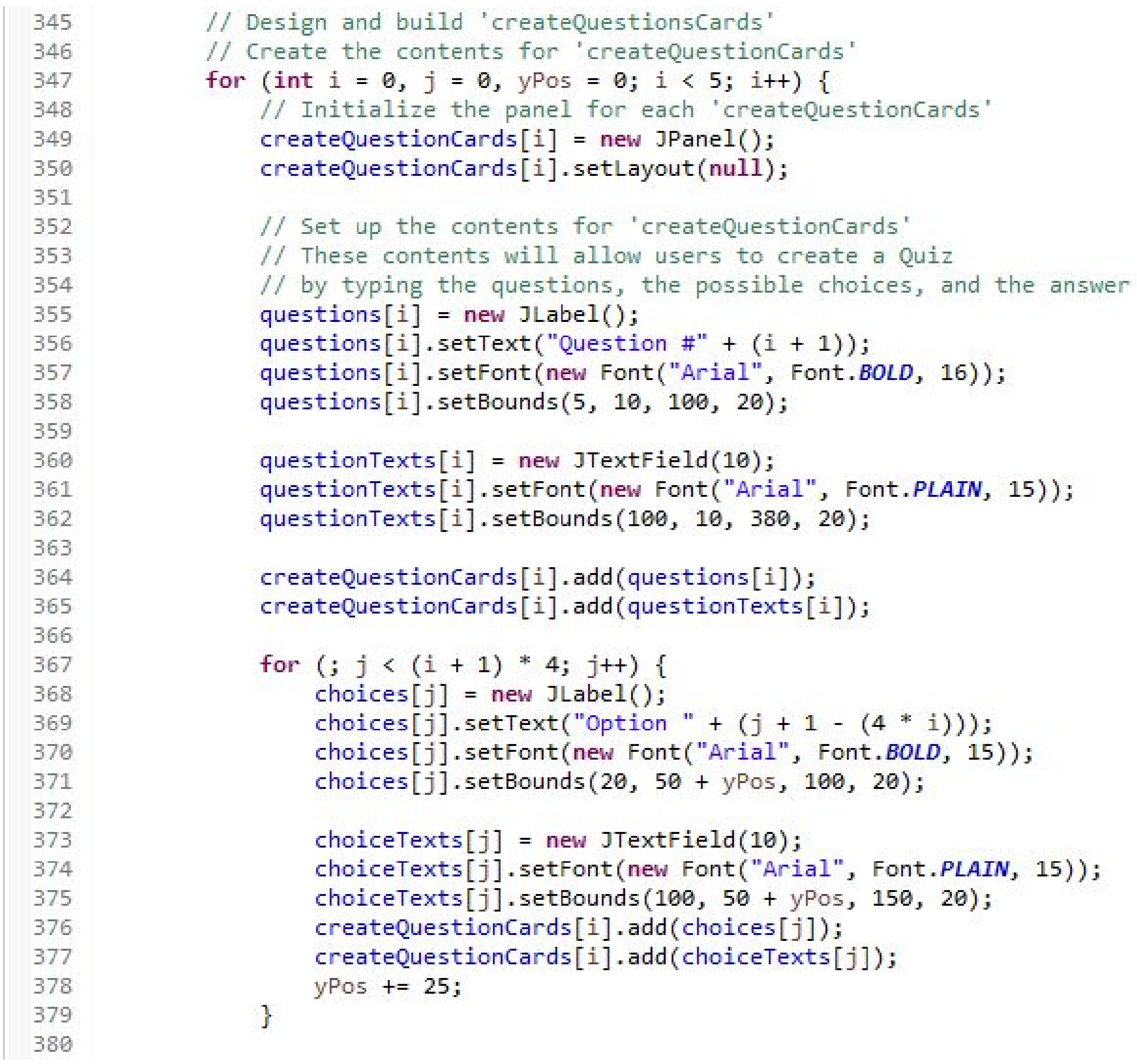
```java
        // Design and build 'card3'
        // Create the heading for 'card3'
        card3.setLayout(new BorderLayout());
        heading3.setFont(new Font("Arial", Font.BOLD, 25));
        card3.add(northPanel3, BorderLayout.NORTH);
        northPanel3.setLayout(new FlowLayout());
        northPanel3.add(heading3);
        // Create the contents after logging in for 'card3'
        card3.add(centerPanel3, BorderLayout.CENTER);
        newQuiz.setFont(new Font("Arial", Font.BOLD, 16));
        newQuiz.setBounds(90, 20, 300, 40);
        openQuiz.setFont(new Font("Arial", Font.BOLD, 16));
        openQuiz.setBounds(90, 70, 300, 40);
        logout.setFont(new Font("Arial", Font.BOLD, 16));
        logout.setBounds(90, 120, 300, 40);
        centerPanel3.setLayout(null);
        centerPanel3.add(newQuiz);
        centerPanel3.add(openQuiz);
        centerPanel3.add(logout);
        // Add 'card3' to the application
        add(card3, "Card 3");

```



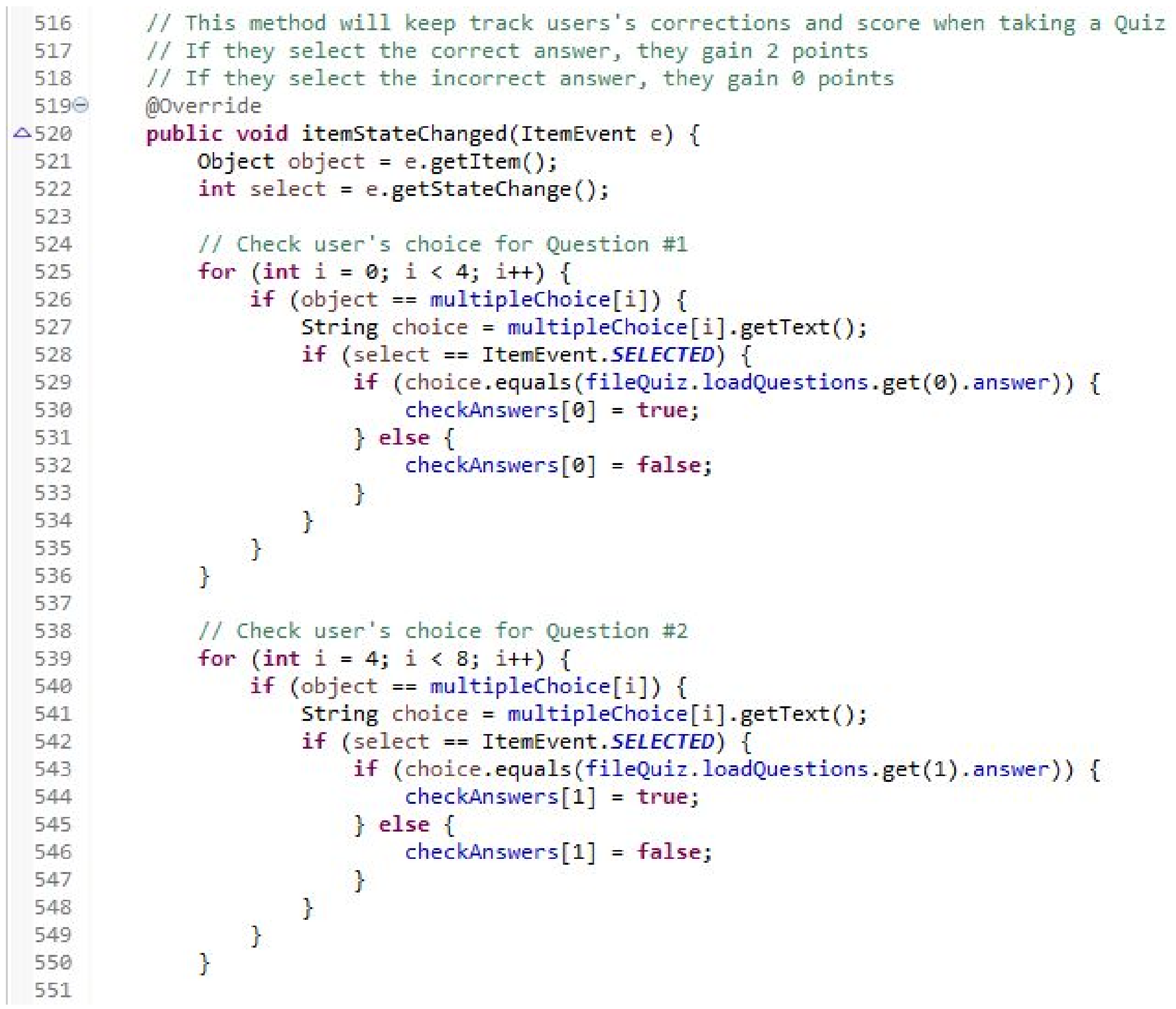
```java
        // Design and build 'card4'
        // Create the heading for 'card4'
        card4.setLayout(new BorderLayout());
        heading4.setFont(new Font("Arial", Font.BOLD, 25));
        card4.add(northPanel4, BorderLayout.NORTH);
        northPanel4.add(heading4);
        // Create the contents of creation for 'card4'
        card4.add(centerPanel4, BorderLayout.CENTER);
        fileName.setFont(new Font("Arial", Font.ITALIC, 16));
        fileName.setBounds(130, 20, 300, 40);
        numOfQuestions.setFont(new Font("Arial", Font.ITALIC, 16));
        numOfQuestions.setBounds(130, 60, 300, 40);
        fileText.setFont(new Font("Arial", Font.PLAIN, 14));
        fileText.setBounds(220, 20, 100, 30);
        questionsText.setText("5");
        questionsText.setEnabled(false);
        questionsText.setFont(new Font("Arial", Font.PLAIN, 14));
        questionsText.setBounds(300, 60, 30, 30);
        centerPanel4.setLayout(null);
        centerPanel4.add(fileName);
        centerPanel4.add(fileText);
        centerPanel4.add(numOfQuestions);
        centerPanel4.add(questionsText);
        // Create the buttons for 'card4'
        card4.add(southPanel4, BorderLayout.SOUTH);
        southPanel4.setLayout(new FlowLayout());
        next.setFont(new Font("Arial", Font.BOLD, 16));
        cancel2.setFont(new Font("Arial", Font.BOLD, 16));
        southPanel4.add(next);
        southPanel4.add(cancel2);
        // Add 'card2' to the application
        add(card4, "Card 4");

```



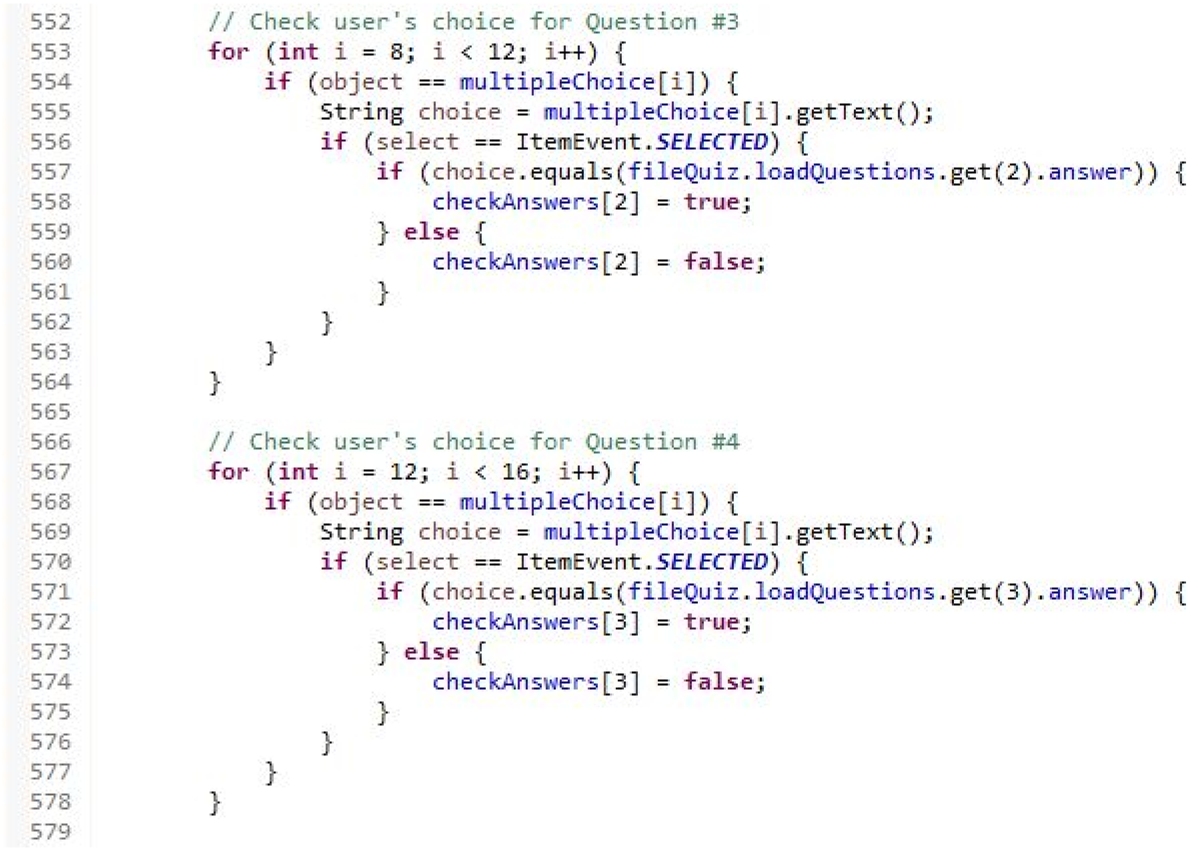
```java
        // Design and build 'card5'
        // Create the heading for 'card5'
        card5.setLayout(new BorderLayout());
        heading5.setFont(new Font("Arial", Font.BOLD, 25));
        card5.add(northPanel5, BorderLayout.NORTH);
        northPanel5.add(heading5);
        // Create the contents of opening for 'card5'
        card5.add(centerPanel5, BorderLayout.CENTER);
        fileName2.setFont(new Font("Arial", Font.ITALIC, 16));
        fileName2.setBounds(130, 20, 300, 40);
        fileText2.setFont(new Font("Arial", Font.PLAIN, 14));
        fileText2.setBounds(220, 20, 100, 30);
        centerPanel5.setLayout(null);
        centerPanel5.add(fileName2);
        centerPanel5.add(fileText2);
        // Create the buttons for 'card5'
        card5.add(southPanel5, BorderLayout.SOUTH);
        southPanel5.setLayout(new FlowLayout());
        open.setFont(new Font("Arial", Font.BOLD, 16));
        cancel3.setFont(new Font("Arial", Font.BOLD, 16));
        southPanel5.add(open);
        southPanel5.add(cancel3);
        // Add 'card2' to the application
        add(card5, "Card 5");

```



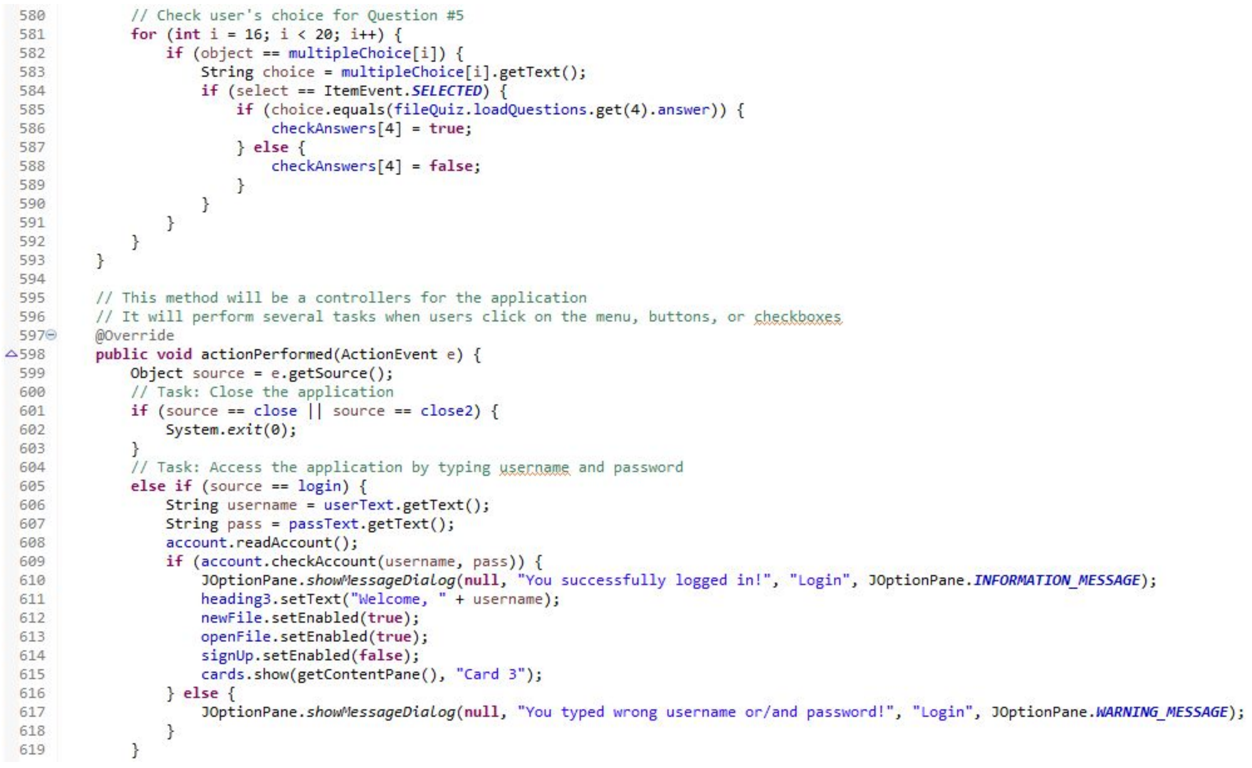
```java
        // Design and build 'createQuestionsCards'
        // Create the contents for 'createQuestionCards'
        for (int i = 0, j = 0, yPos = 0; i < 5; i++) {
            // Initialize the panel for each 'createQuestionCards'
            createQuestionCards[i] = new JPanel();
            createQuestionCards[i].setLayout(null);

            // Set up the contents for 'createQuestionCards'
            // These contents will allow users to create a Quiz
            // by typing the questions, the possible choices, and the answer
            questions[i] = new JLabel();
            questions[i].setText("Question #" + (i + 1));
            questions[i].setFont(new Font("Arial", Font.BOLD, 16));
            questions[i].setBounds(5, 10, 100, 20);

            questionTexts[i] = new JTextField(10);
            questionTexts[i].setFont(new Font("Arial", Font.PLAIN, 15));
            questionTexts[i].setBounds(100, 10, 380, 20);

            createQuestionCards[i].add(questions[i]);
            createQuestionCards[i].add(questionTexts[i]);

            for (; j < (i + 1) * 4; j++) {
                choices[j] = new JLabel();
                choices[j].setText("Option " + (j + 1 - (4 * i)));
                choices[j].setFont(new Font("Arial", Font.BOLD, 15));
                choices[j].setBounds(20, 50 + yPos, 100, 20);

                choiceTexts[j] = new JTextField(10);
                choiceTexts[j].setFont(new Font("Arial", Font.PLAIN, 15));
                choiceTexts[j].setBounds(100, 50 + yPos, 150, 20);
                createQuestionCards[i].add(choices[j]);
                createQuestionCards[i].add(choiceTexts[j]);
                yPos += 25;
            }

            keys[i] = new JLabel("Answer Key");
            keys[i].setFont(new Font("Arial", Font.BOLD, 15));
            keys[i].setBounds(20, 50 + yPos, 100, 20);
            createQuestionCards[i].add(keys[i]);

            keyTexts[i] = new JTextField(10);
            keyTexts[i].setFont(new Font("Arial", Font.PLAIN, 15));
            keyTexts[i].setBounds(120, 50 + yPos, 150, 20);
            createQuestionCards[i].add(keyTexts[i]);
            yPos = 0;

            // Set up buttons for creating a Quiz to flip the cards
            if (i < 4) {
                nexts[i] = new JButton("Next");
                nexts[i].setFont(new Font("Arial", Font.BOLD, 14));
                nexts[i].setBounds(100, 180, 70, 30);
                createQuestionCards[i].add(nexts[i]);
            }
            if (i > 0 && i < 5) {
                backs[i - 1] = new JButton("Back");
                backs[i - 1].setFont(new Font("Arial", Font.BOLD, 14));
                backs[i - 1].setBounds(180, 180, 70, 30);
                createQuestionCards[i].add(backs[i - 1]);
            }
            if (i == 4) {
                add.setFont(new Font("Arial", Font.BOLD, 15));
                add.setBounds(400, 180, 70, 30);
                createQuestionCards[i].add(add);
            }

            cancels[i] = new JButton("Cancel");
            cancels[i].setFont(new Font("Arial", Font.BOLD, 14));
            cancels[i].setBounds(260, 180, 90, 30);
            createQuestionCards[i].add(cancels[i]);
            numberCard[i] = "Card " + (6 + i);
            // Add each card to the application
            add(createQuestionCards[i], numberCard[i]);
        }

        // Design and build 'card11'
        // Create the heading for 'card11'
        card11.setLayout(new BorderLayout());
        heading6.setFont(new Font("Arial", Font.BOLD, 25));
        card11.add(northPanel6, BorderLayout.NORTH);
        northPanel6.add(heading6);
        // Create the contents for 'card11'
        card11.add(centerPanel6, BorderLayout.CENTER);
        ready.setFont(new Font("Arial", Font.ITALIC, 20));
        ready.setBounds(180, 20, 300, 40);
        start.setFont(new Font("Arial", Font.BOLD, 14));
        start.setBounds(160, 60, 70, 30);
        quit.setFont(new Font("Arial", Font.BOLD, 14));
        quit.setBounds(240, 60, 70, 30);
        centerPanel6.setLayout(null);
        centerPanel6.add(ready);
        centerPanel6.add(start);
        centerPanel6.add(quit);
        // Add 'card2' to the application
        add(card11, "Card 11");

        // Create the contents for 'loadQuestionCards'
        // Create the buttons for 'loadQuestionCards'
        for (int i = 0; i < 4; i++) {
            nexts2[i] = new JButton("Next");
            backs2[i] = new JButton("Back");
        }
        // Create CheckBoxes for 'loadQuestionCards'
        for (int i = 0; i < 20; i ++) {
            multipleChoice[i] = new JCheckBox();
        }

        // Design and build 'card17'
        // Create the heading for 'card17'
        card17.setLayout(new BorderLayout());
        heading7.setFont(new Font("Arial", Font.BOLD, 25));
        card17.add(northPanel7, BorderLayout.NORTH);
        northPanel7.add(heading7);
        // Create the contents for the result of 'card17'
        card17.add(centerPanel7, BorderLayout.CENTER);
        correctQuestions.setFont(new Font("Arial", Font.PLAIN, 16));
        correctQuestions.setBounds(130, 20, 300, 40);
        result.setFont(new Font("Arial", Font.PLAIN, 16));
        result.setBounds(130, 60, 300, 40);
        centerPanel7.setLayout(null);
        centerPanel7.add(correctQuestions);
        centerPanel7.add(result);
        // Create the buttons for 'card17'
        card17.add(southPanel7, BorderLayout.SOUTH);
        southPanel7.setLayout(new FlowLayout());
        tryAgain.setFont(new Font("Arial", Font.BOLD, 16));
        logout2.setFont(new Font("Arial", Font.BOLD, 16));
        close2.setFont(new Font("Arial", Font.BOLD, 16));
        southPanel7.add(tryAgain);
        southPanel7.add(logout2);
        southPanel7.add(close2);
        // Add 'card2' to the application
        add(card17, "Card 17");

        // Add the event/logic for buttons and items of the application
        signUp.addActionListener(this);
        login.addActionListener(this);
        close.addActionListener(this);
        create.addActionListener(this);
        cancel.addActionListener(this);
        logout.addActionListener(this);
        info.addActionListener(this);
        next.addActionListener(this);
        cancel2.addActionListener(this);
        newFile.addActionListener(this);
        newQuiz.addActionListener(this);
        openFile.addActionListener(this);
        openQuiz.addActionListener(this);
        cancel3.addActionListener(this);
        add.addActionListener(this);
        open.addActionListener(this);
        quit.addActionListener(this);
        submit.addActionListener(this);
        start.addActionListener(this);
        tryAgain.addActionListener(this);
        logout2.addActionListener(this);
        close2.addActionListener(this);
        for (int i = 0; i < 4; i++) {
            nexts[i].addActionListener(this);
            backs[i].addActionListener(this);
            nexts2[i].addActionListener(this);
            backs2[i].addActionListener(this);
        }
        for (int i = 0; i < 5; i++) {
            cancels[i].addActionListener(this);
        }
        for (int i = 0; i < 20; i++) {
            multipleChoice[i].addItemListener(this);
        }
    }

```



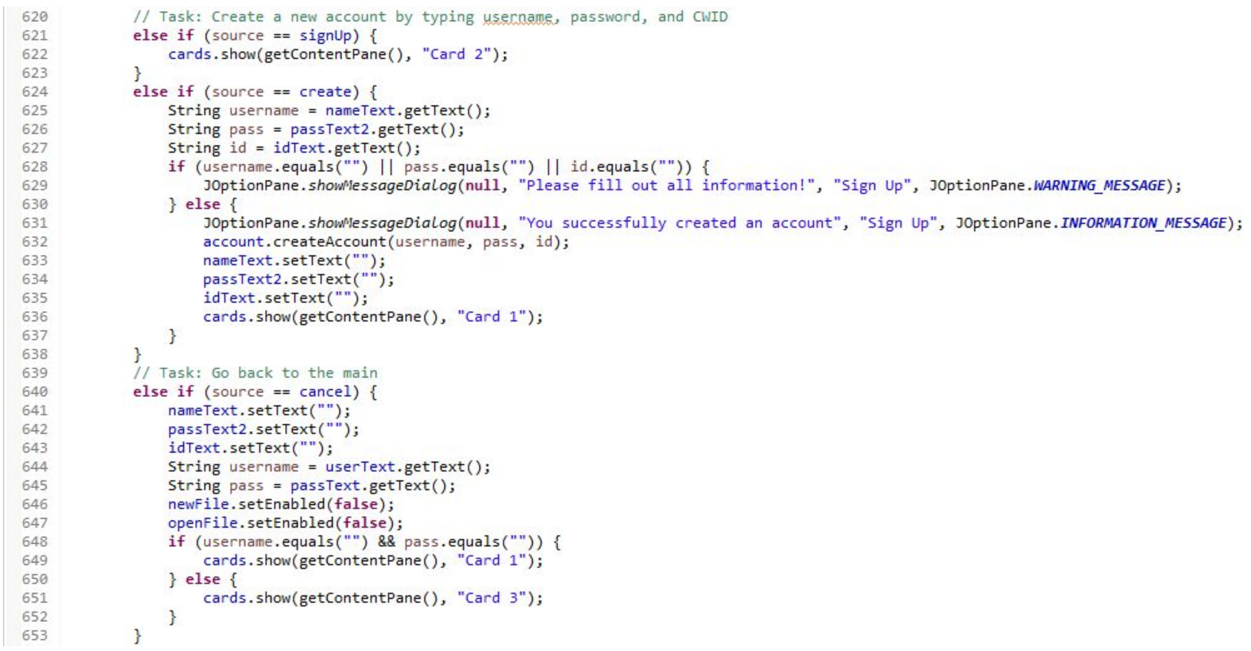
```java
    // This method will keep track users's corrections and score when taking a Quiz
    // If they select the correct answer, they gain 2 points
    // If they select the incorrect answer, they gain 0 points
    @Override
    public void itemStateChanged(ItemEvent e) {
        Object object = e.getItem();
        int select = e.getStateChange();

        // Check user's choice for Question #1
        for (int i = 0; i < 4; i++) {
            if (object == multipleChoice[i]) {
                String choice = multipleChoice[i].getText();
                if (select == ItemEvent.SELECTED) {
                    if (choice.equals(fileQuiz.loadQuestions.get(0).answer)) {
                        checkAnswers[0] = true;
                    } else {
                        checkAnswers[0] = false;
                    }
                }
            }
        }

        // Check user's choice for Question #2
        for (int i = 4; i < 8; i++) {
            if (object == multipleChoice[i]) {
                String choice = multipleChoice[i].getText();
                if (select == ItemEvent.SELECTED) {
                    if (choice.equals(fileQuiz.loadQuestions.get(1).answer)) {
                        checkAnswers[1] = true;
                    } else {
                        checkAnswers[1] = false;
                    }
                }
            }
        }

```



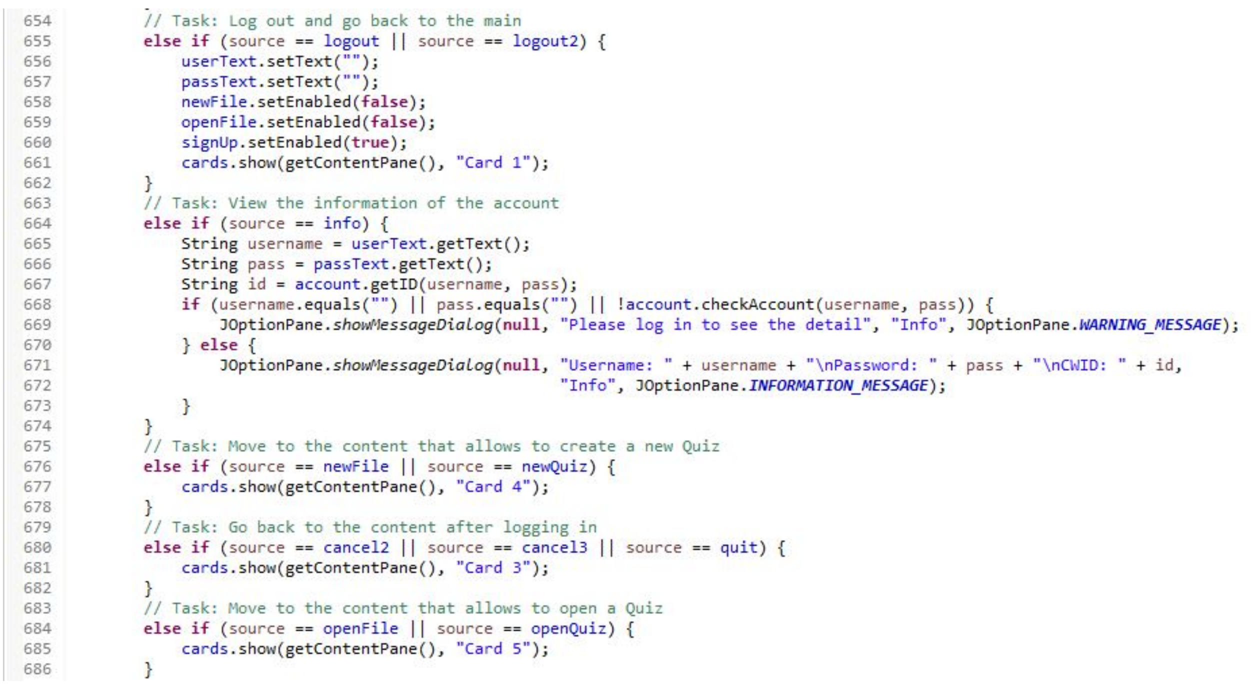
```java
        // Check user's choice for Question #3
        for (int i = 8; i < 12; i++) {
            if (object == multipleChoice[i]) {
                String choice = multipleChoice[i].getText();
                if (select == ItemEvent.SELECTED) {
                    if (choice.equals(fileQuiz.loadQuestions.get(2).answer)) {
                        checkAnswers[2] = true;
                    } else {
                        checkAnswers[2] = false;
                    }
                }
            }
        }

        // Check user's choice for Question #4
        for (int i = 12; i < 16; i++) {
            if (object == multipleChoice[i]) {
                String choice = multipleChoice[i].getText();
                if (select == ItemEvent.SELECTED) {
                    if (choice.equals(fileQuiz.loadQuestions.get(3).answer)) {
                        checkAnswers[3] = true;
                    } else {
                        checkAnswers[3] = false;
                    }
                }
            }
        }

```



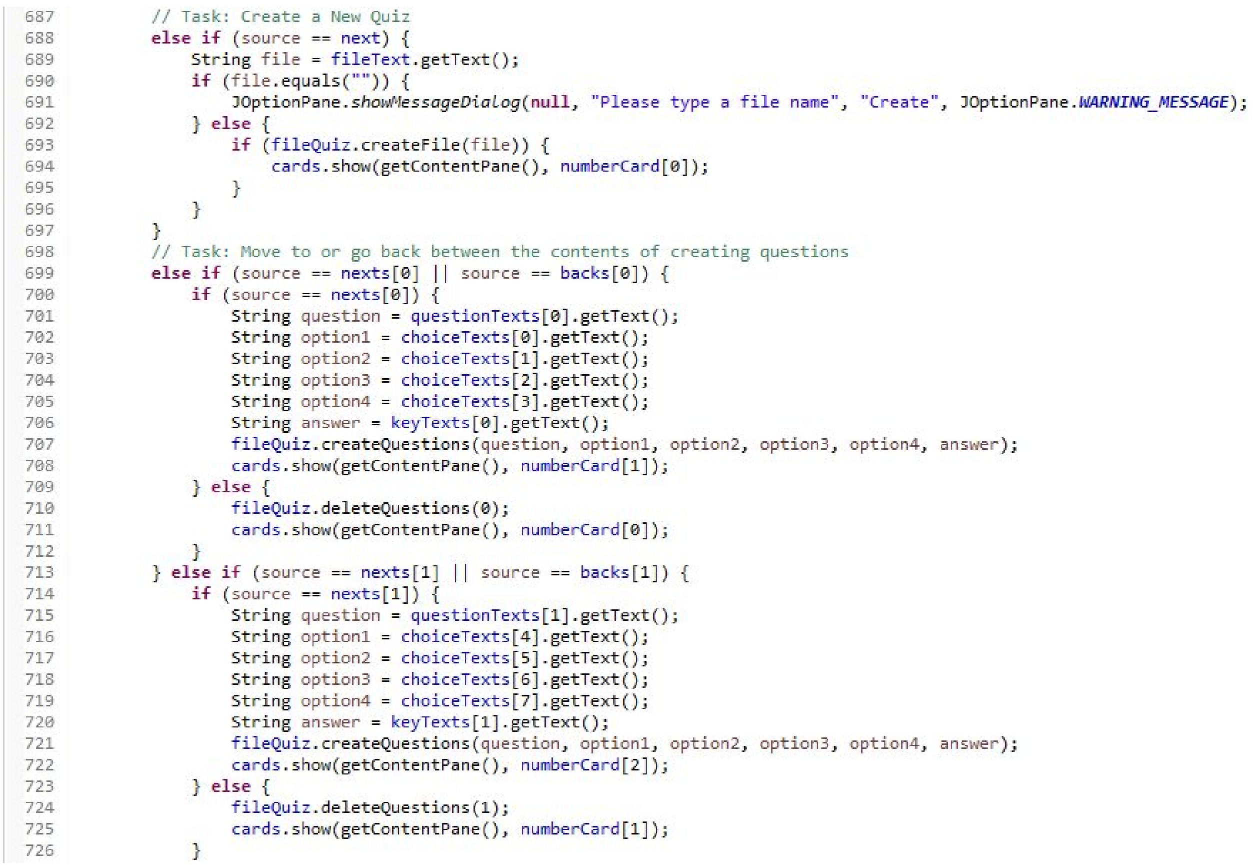
```java
        // Check user's choice for Question #5
        for (int i = 16; i < 20; i++) {
            if (object == multipleChoice[i]) {
                String choice = multipleChoice[i].getText();
                if (select == ItemEvent.SELECTED) {
                    if (choice.equals(fileQuiz.loadQuestions.get(4).answer)) {
                        checkAnswers[4] = true;
                    } else {
                        checkAnswers[4] = false;
                    }
                }
            }
        }
    }

    // This method will be a controllers for the application
    // It will perform several tasks when users click on the menu, buttons, or checkboxes
    @Override
    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        // Task: Close the application
        if (source == close || source == close2) {
            System.exit(0);
        }
        // Task: Access the application by typing username and password
        else if (source == login) {
            String username = userText.getText();
            String pass = passText.getText();
            account.readAccount();
            if (account.checkAccount(username, pass)) {
                JOptionPane.showMessageDialog(null, "You successfully logged in!", "Login", JOptionPane.INFORMATION_MESSAGE);
                heading3.setText("Welcome, " + username);
                newFile.setEnabled(true);
                openFile.setEnabled(true);
                signUp.setEnabled(false);
                cards.show(getContentPane(), "Card 3");
            } else {
                JOptionPane.showMessageDialog(null, "You typed wrong username or/and password!", "Login", JOptionPane.WARNING_MESSAGE);
            }
        }
```



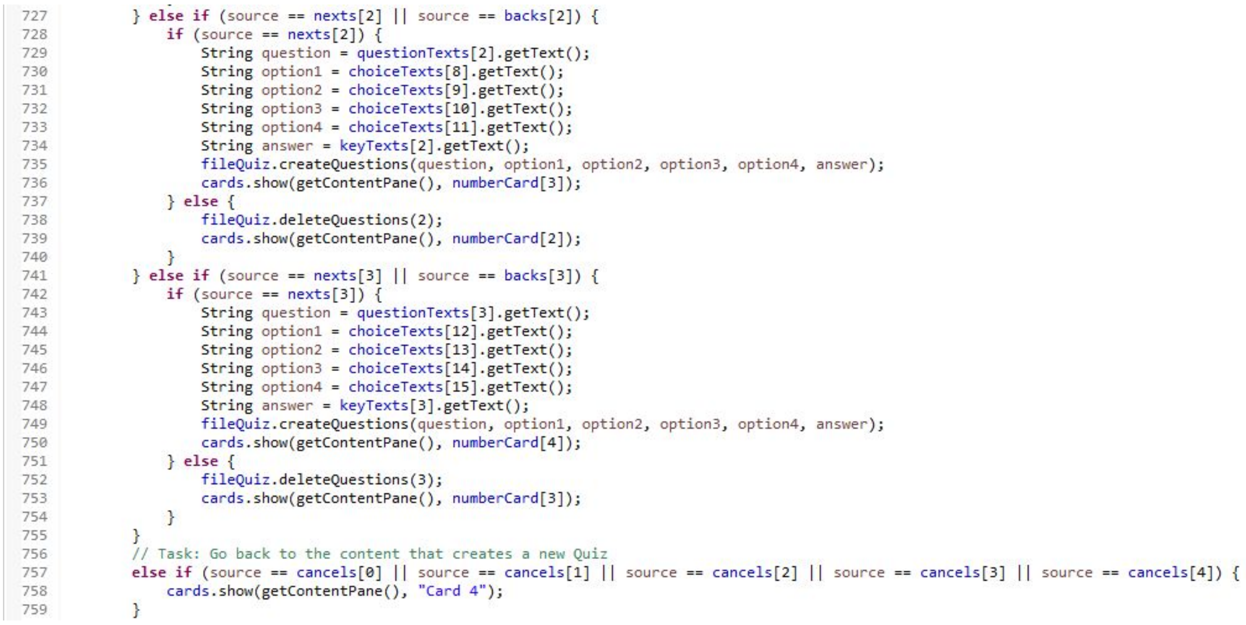
```java
        // Task: Create a new account by typing username, password, and CWID
        else if (source == signUp) {
            cards.show(getContentPane(), "Card 2");
        }
        else if (source == create) {
            String username = nameText.getText();
            String pass = passText2.getText();
            String id = idText.getText();
            if (username.equals("") || pass.equals("") || id.equals("")) {
                JOptionPane.showMessageDialog(null, "Please fill out all information!", "Sign Up", JOptionPane.WARNING_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "You successfully created an account", "Sign Up", JOptionPane.INFORMATION_MESSAGE);
                account.createAccount(username, pass, id);
                nameText.setText("");
                passText2.setText("");
                idText.setText("");
                cards.show(getContentPane(), "Card 1");
            }
        }
        // Task: Go back to the main
        else if (source == cancel) {
            nameText.setText("");
            passText2.setText("");
            idText.setText("");
            String username = userText.getText();
            String pass = passText.getText();
            newFile.setEnabled(false);
            openFile.setEnabled(false);
            if (username.equals("") && pass.equals("")) {
                cards.show(getContentPane(), "Card 1");
            } else {
                cards.show(getContentPane(), "Card 3");
            }
        }
```



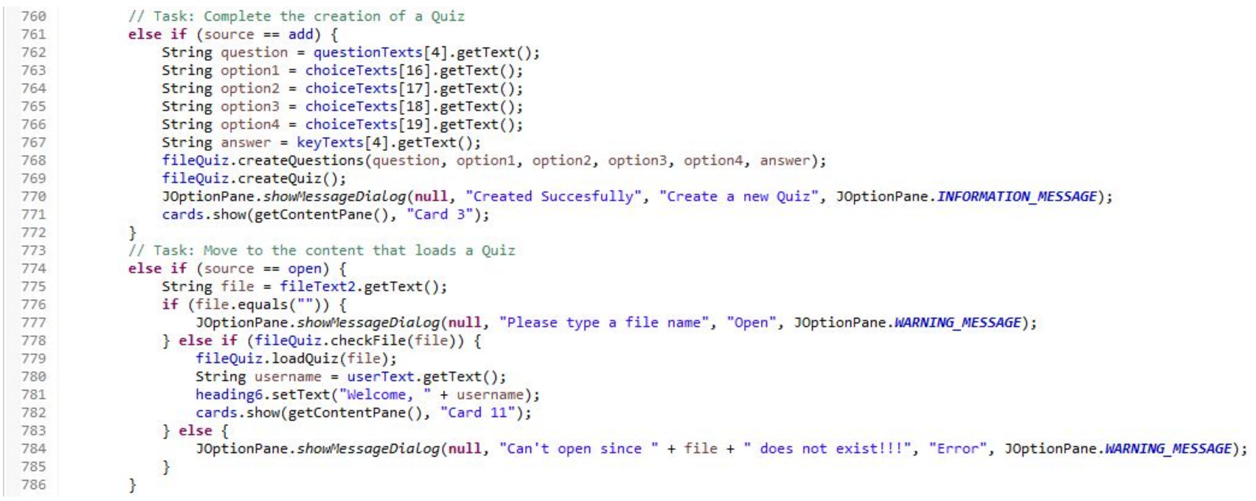
```java
        // Task: Log out and go back to the main
        else if (source == logout || source == logout2) {
            userText.setText("");
            passText.setText("");
            newFile.setEnabled(false);
            openFile.setEnabled(false);
            signUp.setEnabled(true);
            cards.show(getContentPane(), "Card 1");
        }
        // Task: View the information of the account
        else if (source == info) {
            String username = userText.getText();
            String pass = passText.getText();
            String id = account.getID(username, pass);
            if (username.equals("") || pass.equals("") || !account.checkAccount(username, pass)) {
                JOptionPane.showMessageDialog(null, "Please log in to see the detail", "Info", JOptionPane.WARNING_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "Username: " + username + "\nPassword: " + pass + "\nCWID: " + id,
                                                    "Info", JOptionPane.INFORMATION_MESSAGE);
            }
        }
        // Task: Move to the content that allows to create a new Quiz
        else if (source == newFile || source == newQuiz) {
            cards.show(getContentPane(), "Card 4");
        }
        // Task: Go back to the content after logging in
        else if (source == cancel2 || source == cancel3 || source == quit) {
            cards.show(getContentPane(), "Card 3");
        }
        // Task: Move to the content that allows to open a Quiz
        else if (source == openFile || source == openQuiz) {
            cards.show(getContentPane(), "Card 5");
        }
```



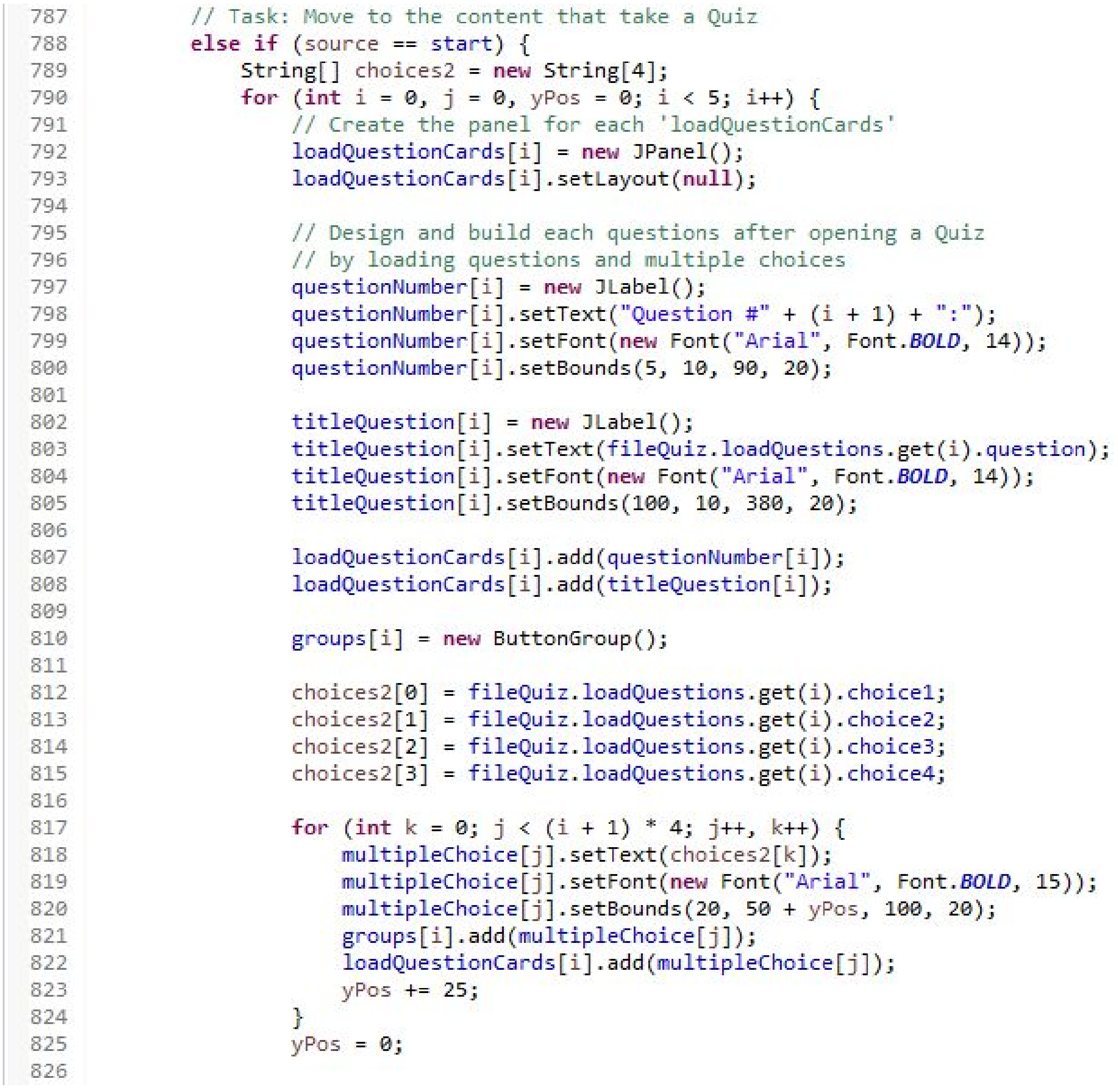
```java
        // Task: Create a New Quiz
        else if (source == next) {
            String file = fileText.getText();
            if (file.equals("")) {
                JOptionPane.showMessageDialog(null, "Please type a file name", "Create", JOptionPane.WARNING_MESSAGE);
            } else {
                if (fileQuiz.createFile(file)) {
                    cards.show(getContentPane(), numberCard[0]);
                }
            }
        }
        // Task: Move to or go back between the contents of creating questions
        else if (source == nexts[0] || source == backs[0]) {
            if (source == nexts[0]) {
                String question = questionTexts[0].getText();
                String option1 = choiceTexts[0].getText();
                String option2 = choiceTexts[1].getText();
                String option3 = choiceTexts[2].getText();
                String option4 = choiceTexts[3].getText();
                String answer = keyTexts[0].getText();
                fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
                cards.show(getContentPane(), numberCard[1]);
            } else {
                fileQuiz.deleteQuestions(0);
                cards.show(getContentPane(), numberCard[0]);
            }
        } else if (source == nexts[1] || source == backs[1]) {
            if (source == nexts[1]) {
                String question = questionTexts[1].getText();
                String option1 = choiceTexts[4].getText();
                String option2 = choiceTexts[5].getText();
                String option3 = choiceTexts[6].getText();
                String option4 = choiceTexts[7].getText();
                String answer = keyTexts[1].getText();
                fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
                cards.show(getContentPane(), numberCard[2]);
            } else {
                fileQuiz.deleteQuestions(1);
                cards.show(getContentPane(), numberCard[1]);
            }
```



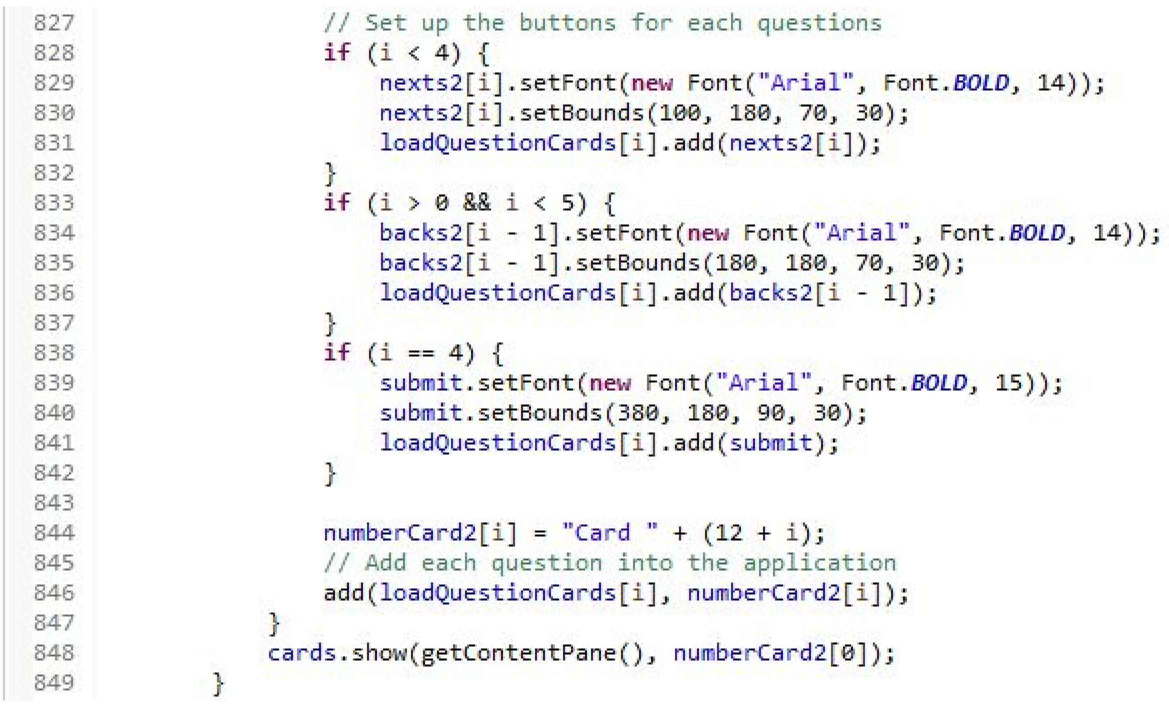
```java
        } else if (source == nexts[2] || source == backs[2]) {
            if (source == nexts[2]) {
                String question = questionTexts[2].getText();
                String option1 = choiceTexts[8].getText();
                String option2 = choiceTexts[9].getText();
                String option3 = choiceTexts[10].getText();
                String option4 = choiceTexts[11].getText();
                String answer = keyTexts[2].getText();
                fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
                cards.show(getContentPane(), numberCard[3]);
            } else {
                fileQuiz.deleteQuestions(2);
                cards.show(getContentPane(), numberCard[2]);
            }
        } else if (source == nexts[3] || source == backs[3]) {
            if (source == nexts[3]) {
                String question = questionTexts[3].getText();
                String option1 = choiceTexts[12].getText();
                String option2 = choiceTexts[13].getText();
                String option3 = choiceTexts[14].getText();
                String option4 = choiceTexts[15].getText();
                String answer = keyTexts[3].getText();
                fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
                cards.show(getContentPane(), numberCard[4]);
            } else {
                fileQuiz.deleteQuestions(3);
                cards.show(getContentPane(), numberCard[3]);
            }
        }
        // Task: Go back to the content that creates a new Quiz
        else if (source == cancels[0] || source == cancels[1] || source == cancels[2] || source == cancels[3] || source == cancels[4]) {
            cards.show(getContentPane(), "Card 4");
        }
```



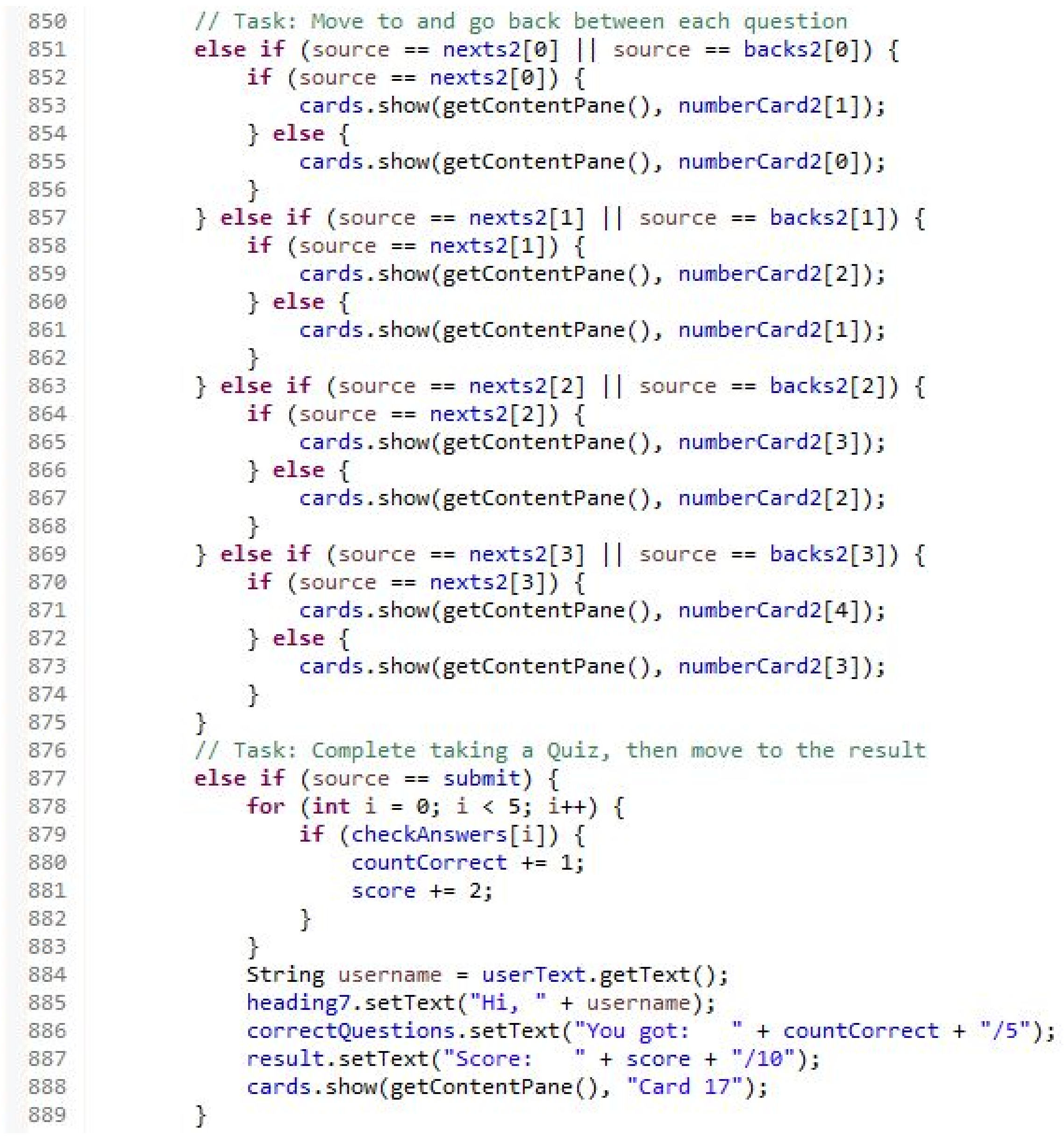
```java
        // Task: Complete the creation of a Quiz
        else if (source == add) {
            String question = questionTexts[4].getText();
            String option1 = choiceTexts[16].getText();
            String option2 = choiceTexts[17].getText();
            String option3 = choiceTexts[18].getText();
            String option4 = choiceTexts[19].getText();
            String answer = keyTexts[4].getText();
            fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
            fileQuiz.createQuiz();
            JOptionPane.showMessageDialog(null, "Created Succesfully", "Create a new Quiz", JOptionPane.INFORMATION_MESSAGE);
            cards.show(getContentPane(), "Card 3");
        }
        // Task: Move to the content that loads a Quiz
        else if (source == open) {
            String file = fileText2.getText();
            if (file.equals("")) {
                JOptionPane.showMessageDialog(null, "Please type a file name", "Open", JOptionPane.WARNING_MESSAGE);
            } else if (fileQuiz.checkFile(file)) {
                fileQuiz.loadQuiz(file);
                String username = userText.getText();
                heading6.setText("Welcome, " + username);
                cards.show(getContentPane(), "Card 11");
            } else {
                JOptionPane.showMessageDialog(null, "Can't open since " + file + " does not exist!!!", "Error", JOptionPane.WARNING_MESSAGE);
            }
        }
```



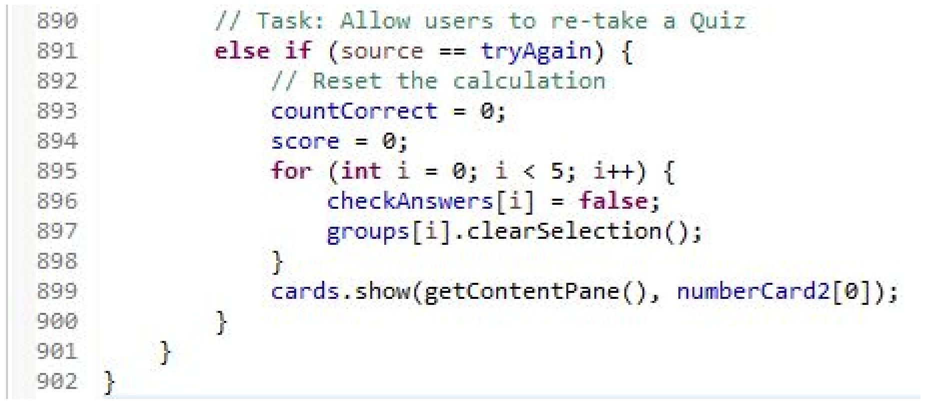
```java
        // Task: Move to the content that take a Quiz
        else if (source == start) {
            String[] choices2 = new String[4];
            for (int i = 0, j = 0, yPos = 0; i < 5; i++) {
                // Create the panel for each 'loadQuestionCards'
                loadQuestionCards[i] = new JPanel();
                loadQuestionCards[i].setLayout(null);

                // Design and build each questions after opening a Quiz
                // by loading questions and multiple choices
                questionNumber[i] = new JLabel();
                questionNumber[i].setText("Question #" + (i + 1) + ":");
                questionNumber[i].setFont(new Font("Arial", Font.BOLD, 14));
                questionNumber[i].setBounds(5, 10, 90, 20);

                titleQuestion[i] = new JLabel();
                titleQuestion[i].setText(fileQuiz.loadQuestions.get(i).question);
                titleQuestion[i].setFont(new Font("Arial", Font.BOLD, 14));
                titleQuestion[i].setBounds(100, 10, 380, 20);

                loadQuestionCards[i].add(questionNumber[i]);
                loadQuestionCards[i].add(titleQuestion[i]);

                groups[i] = new ButtonGroup();

                choices2[0] = fileQuiz.loadQuestions.get(i).choice1;
                choices2[1] = fileQuiz.loadQuestions.get(i).choice2;
                choices2[2] = fileQuiz.loadQuestions.get(i).choice3;
                choices2[3] = fileQuiz.loadQuestions.get(i).choice4;

                for (int k = 0; j < (i + 1) * 4; j++, k++) {
                    multipleChoice[j].setText(choices2[k]);
                    multipleChoice[j].setFont(new Font("Arial", Font.BOLD, 15));
                    multipleChoice[j].setBounds(20, 50 + yPos, 100, 20);
                    groups[i].add(multipleChoice[j]);
                    loadQuestionCards[i].add(multipleChoice[j]);
                    yPos += 25;
                }
                yPos = 0;

```



```java
                // Set up the buttons for each questions
                if (i < 4) {
                    nexts2[i].setFont(new Font("Arial", Font.BOLD, 14));
                    nexts2[i].setBounds(100, 180, 70, 30);
                    loadQuestionCards[i].add(nexts2[i]);
                }
                if (i > 0 && i < 5) {
                    backs2[i - 1].setFont(new Font("Arial", Font.BOLD, 14));
                    backs2[i - 1].setBounds(180, 180, 70, 30);
                    loadQuestionCards[i].add(backs2[i - 1]);
                }
                if (i == 4) {
                    submit.setFont(new Font("Arial", Font.BOLD, 15));
                    submit.setBounds(380, 180, 90, 30);
                    loadQuestionCards[i].add(submit);
                }

                numberCard2[i] = "Card " + (12 + i);
                // Add each question into the application
                add(loadQuestionCards[i], numberCard2[i]);
            }
            cards.show(getContentPane(), numberCard2[0]);
        }
```



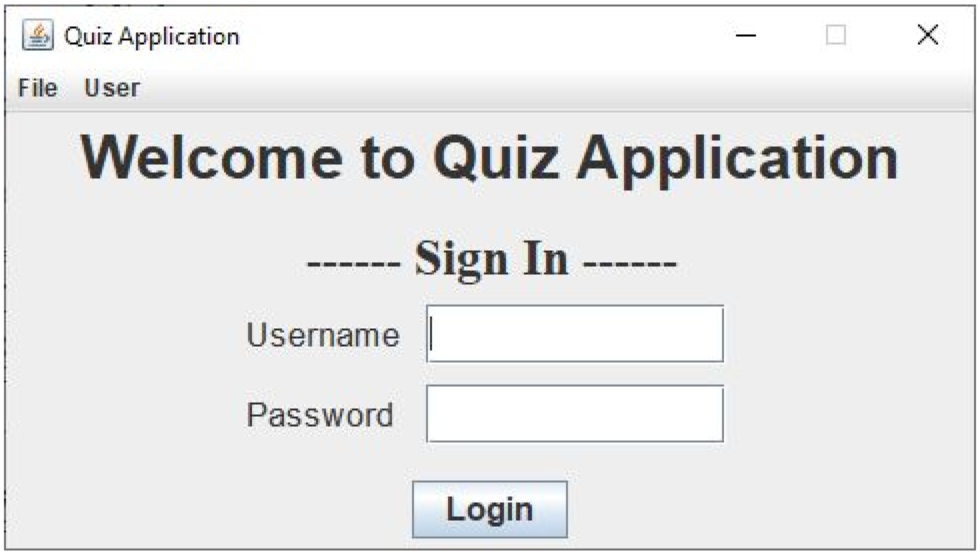
```java
        // Task: Move to and go back between each question
        else if (source == nexts2[0] || source == backs2[0]) {
            if (source == nexts2[0]) {
                cards.show(getContentPane(), numberCard2[1]);
            } else {
                cards.show(getContentPane(), numberCard2[0]);
            }
        } else if (source == nexts2[1] || source == backs2[1]) {
            if (source == nexts2[1]) {
                cards.show(getContentPane(), numberCard2[2]);
            } else {
                cards.show(getContentPane(), numberCard2[1]);
            }
        } else if (source == nexts2[2] || source == backs2[2]) {
            if (source == nexts2[2]) {
                cards.show(getContentPane(), numberCard2[3]);
            } else {
                cards.show(getContentPane(), numberCard2[2]);
            }
        } else if (source == nexts2[3] || source == backs2[3]) {
            if (source == nexts2[3]) {
                cards.show(getContentPane(), numberCard2[4]);
            } else {
                cards.show(getContentPane(), numberCard2[3]);
            }
        }
        // Task: Complete taking a Quiz, then move to the result
        else if (source == submit) {
            for (int i = 0; i < 5; i++) {
                if (checkAnswers[i]) {
                    countCorrect += 1;
                    score += 2;
                }
            }
            String username = userText.getText();
            heading7.setText("Hi, " + username);
            correctQuestions.setText("You got:   " + countCorrect + "/5");
            result.setText("Score:   " + score + "/10");
            cards.show(getContentPane(), "Card 17");
        }
```



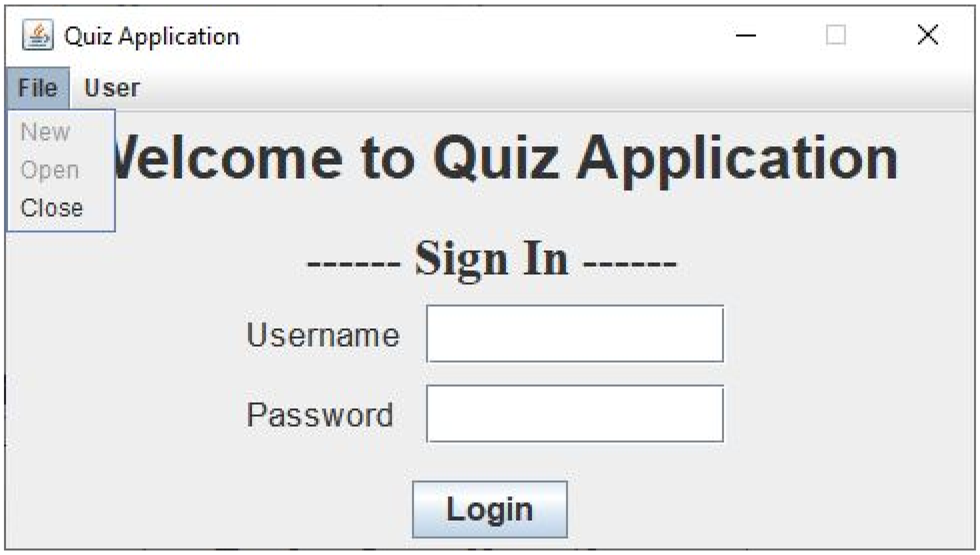
```java
        // Task: Allow users to re-take a Quiz
        else if (source == tryAgain) {
            // Reset the calculation
            countCorrect = 0;
            score = 0;
            for (int i = 0; i < 5; i++) {
                checkAnswers[i] = false;
                groups[i].clearSelection();
            }
            cards.show(getContentPane(), numberCard2[0]);
        }
    }
}
```

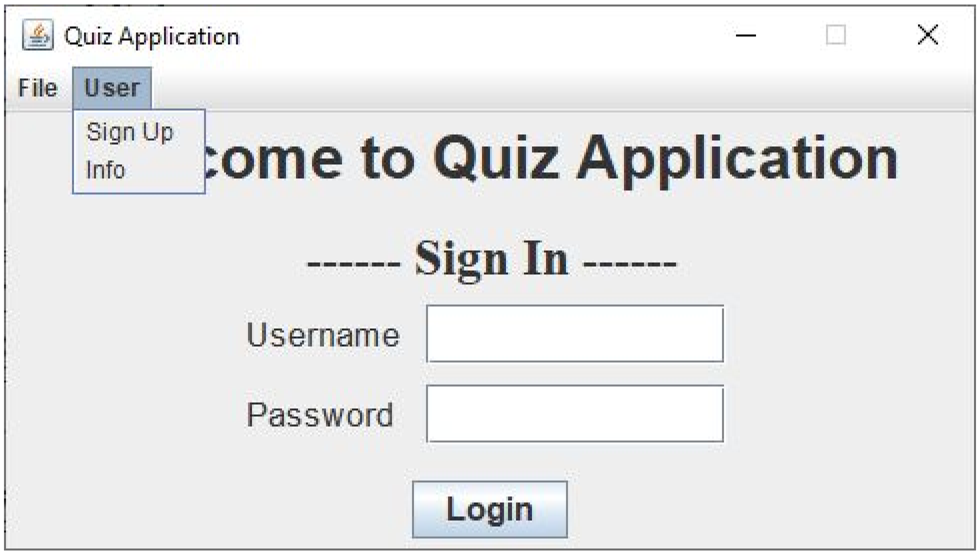
**Result:**

**Launching Application**



**Menu**





come to Quiz Application

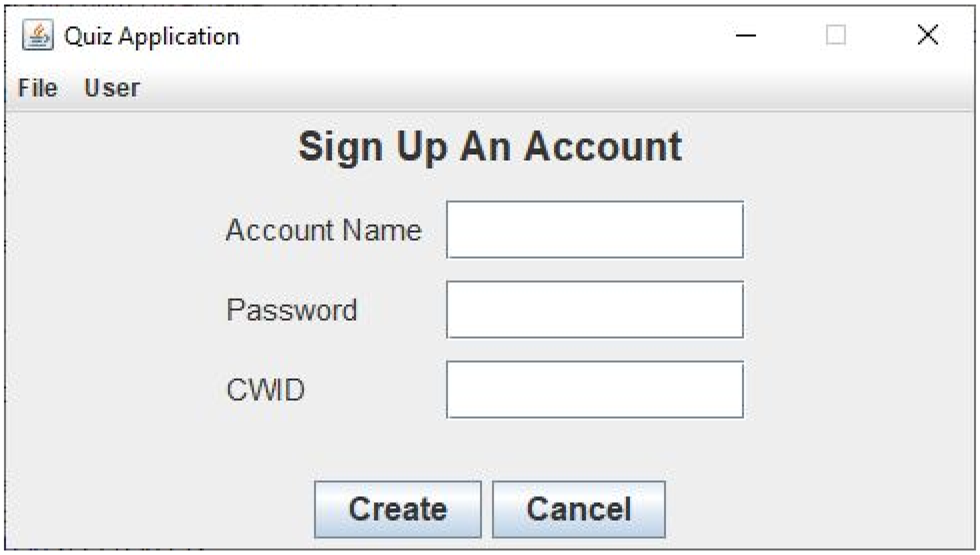
------ Sign In ------

Username

Password

Login

**Register an account**



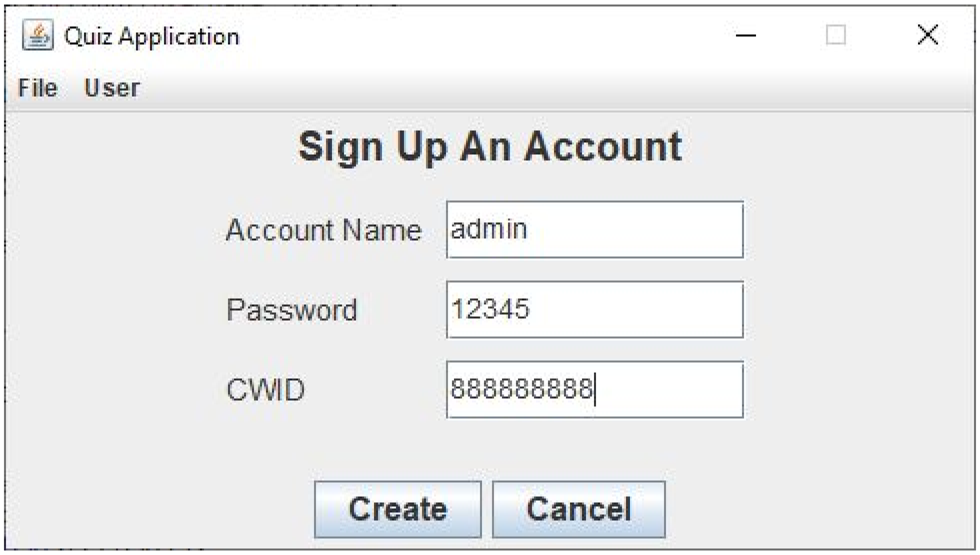
Sign Up An Account

Account Name

Password

CWID

Create Cancel

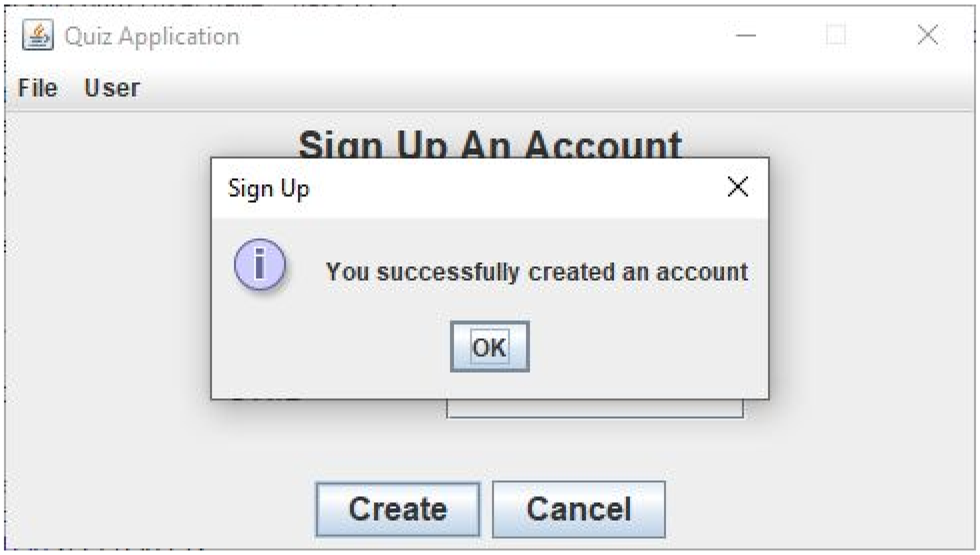


Sign Up An Account

Account Name: admin

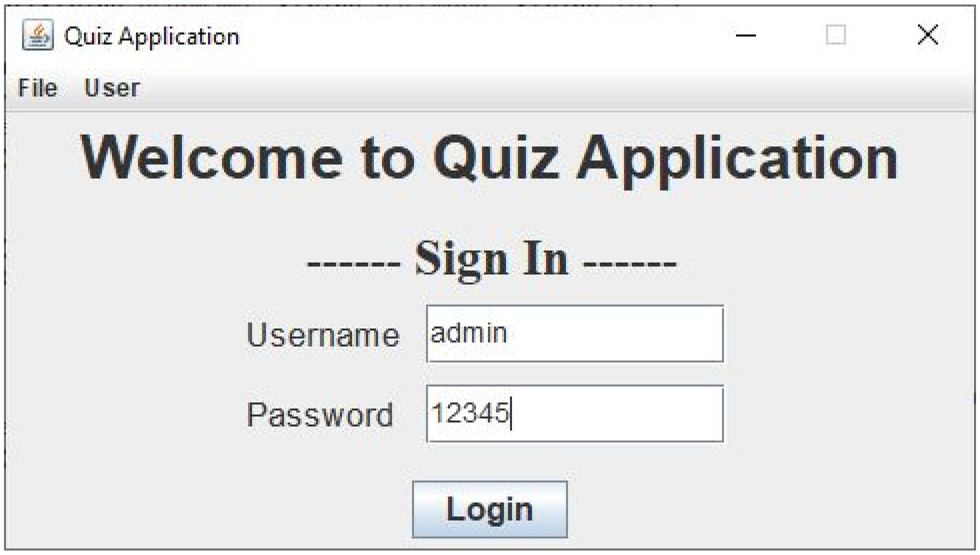
Password: 12345

CWID: 888888888



You successfully created an account

## Logging In

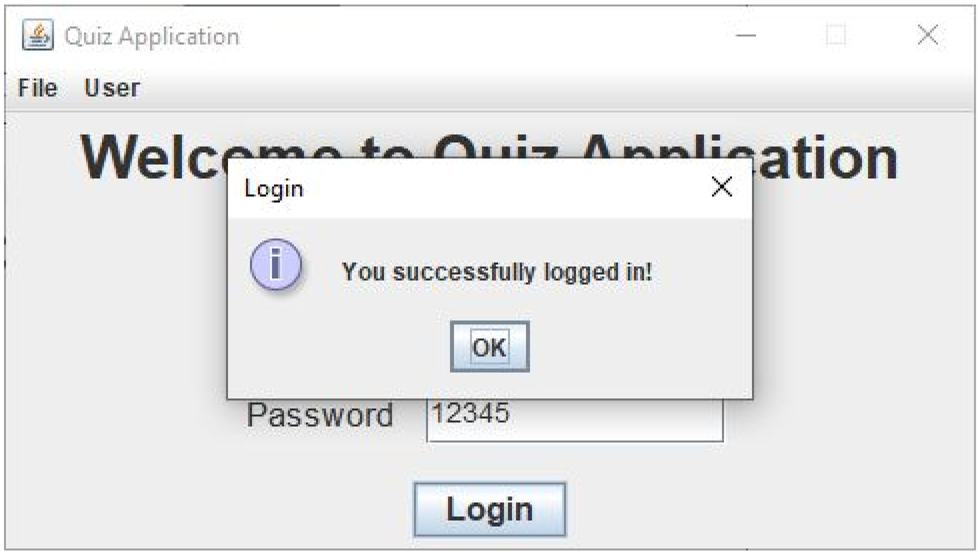


Welcome to Quiz Application

------ Sign In ------

Username admin

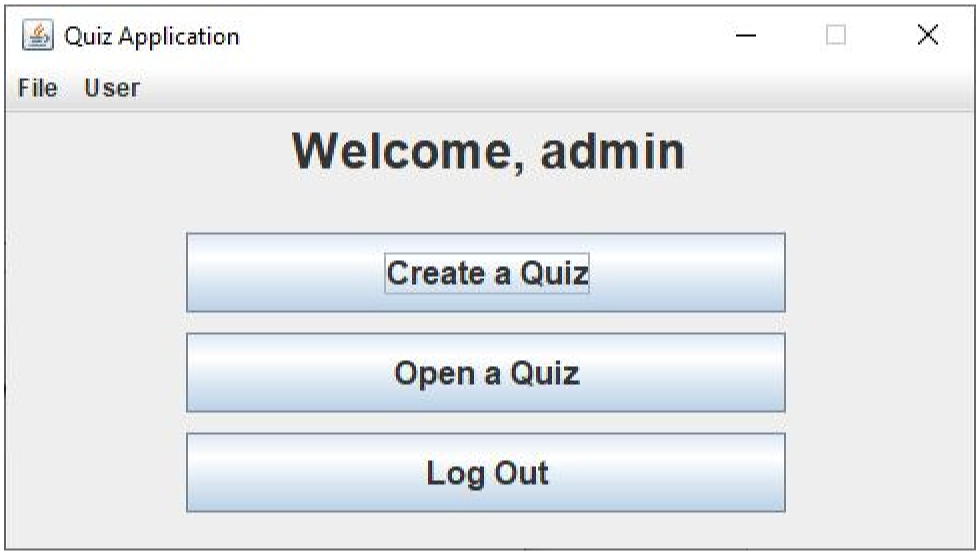
Password 12345



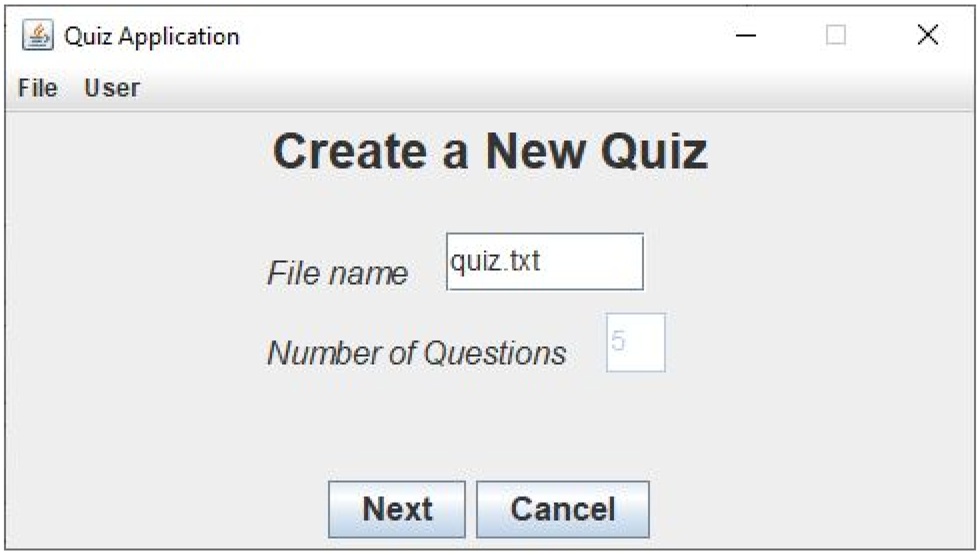
You successfully logged in!

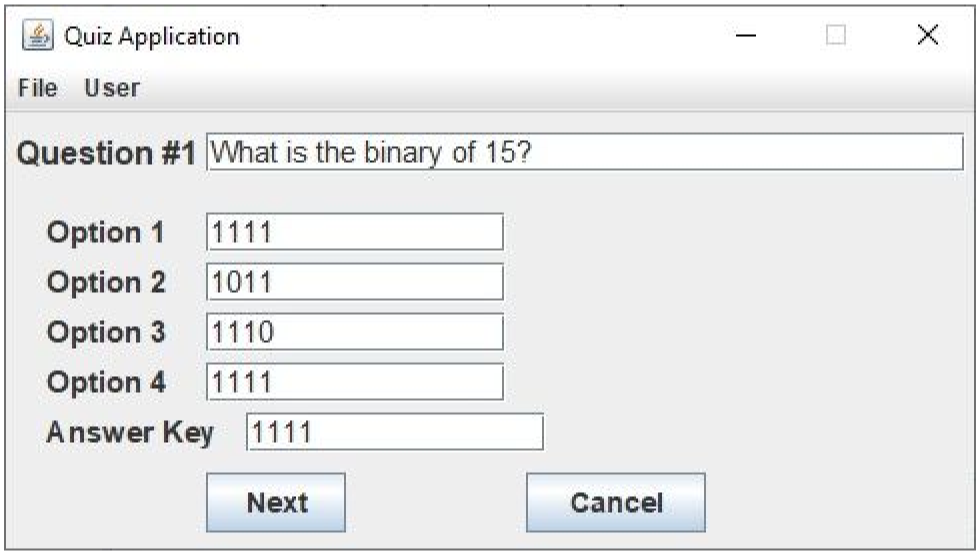
Password 12345

**After logging**



**Create a Quiz**





Question #1 What is the binary of 15?

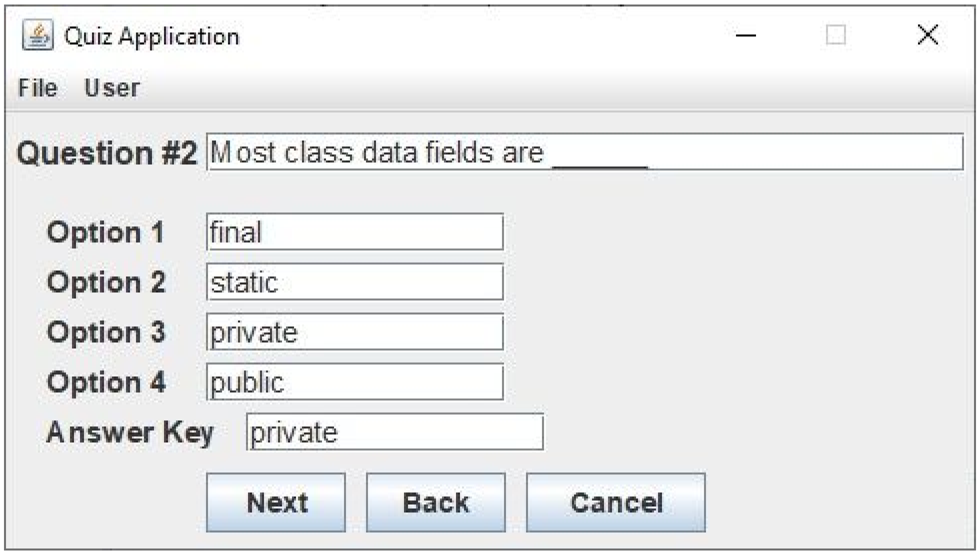
Option 1 1111

Option 2 1011

Option 3 1110

Option 4 1111

Answer Key 1111



Question #2 Most class data fields are ______

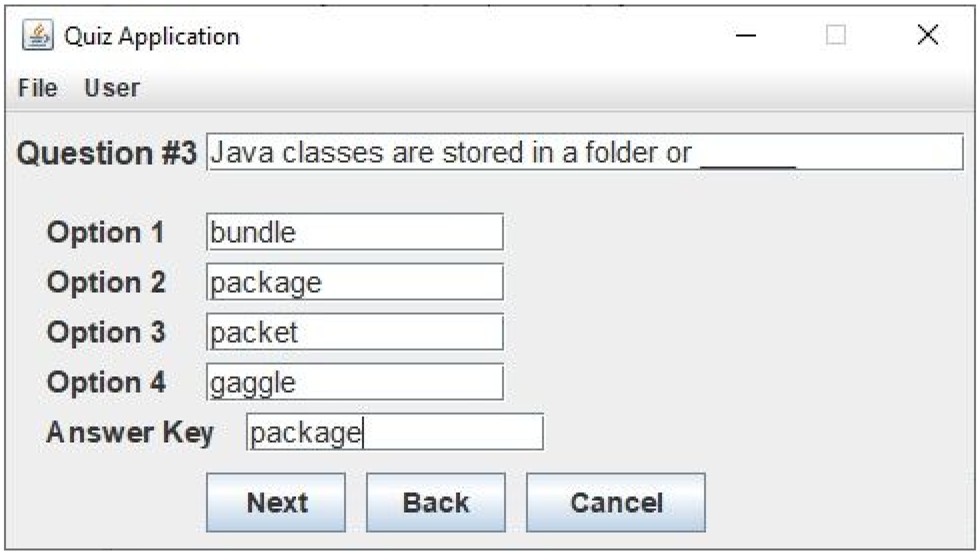
Option 1 final

Option 2 static

Option 3 private

Option 4 public

Answer Key private



Question #3 Java classes are stored in a folder or ______

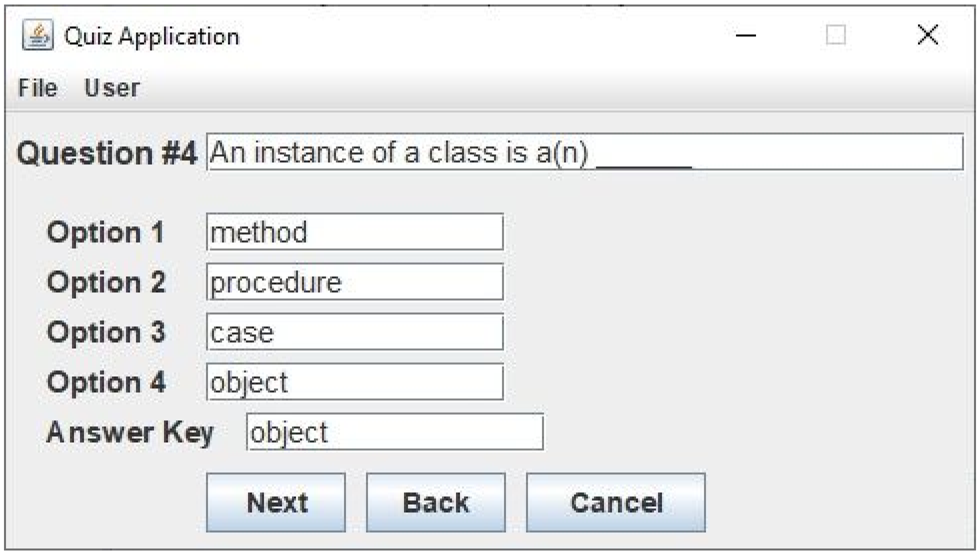
Option 1 bundle

Option 2 package

Option 3 packet

Option 4 gaggle

Answer Key package


Quiz Application

File User

Question #4 An instance of a class is a(n) ______

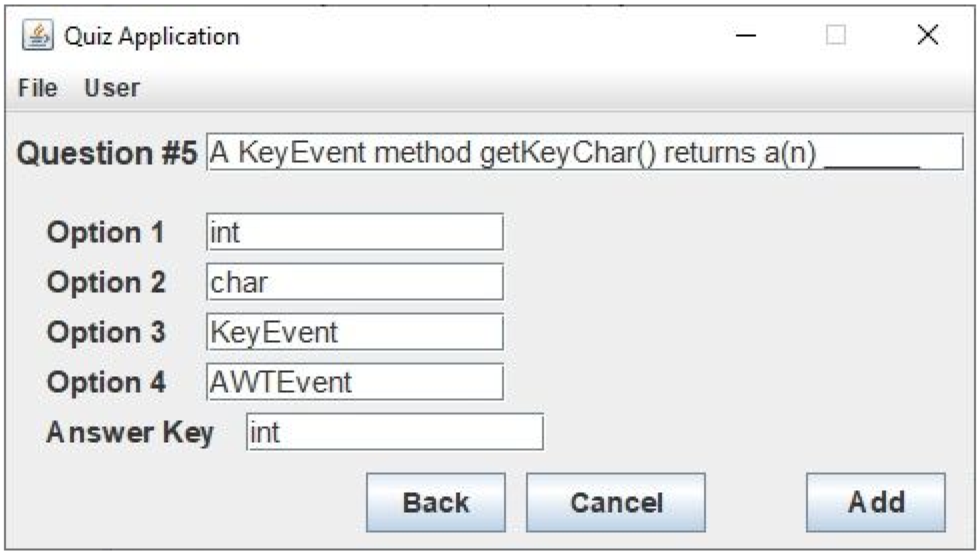
Option 1 method

Option 2 procedure

Option 3 case

Option 4 object

Answer Key object


Quiz Application

File User

Question #5 A KeyEvent method getKeyChar() returns a(n) ______

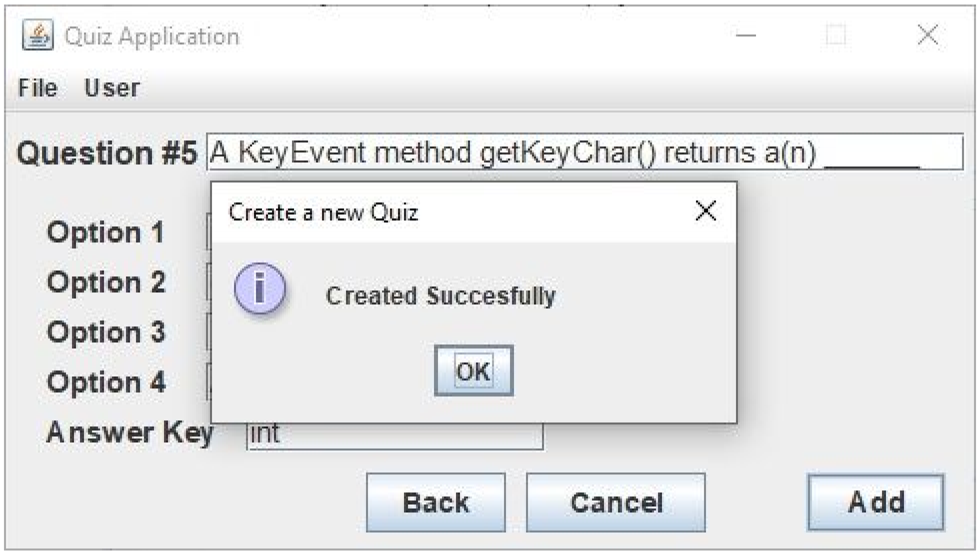
Option 1 int

Option 2 char

Option 3 KeyEvent

Option 4 AWTEvent

Answer Key int


Quiz Application

File User

Question #5 A KeyEvent method getKeyChar() returns a(n) ______

Option 1

Option 2

Option 3

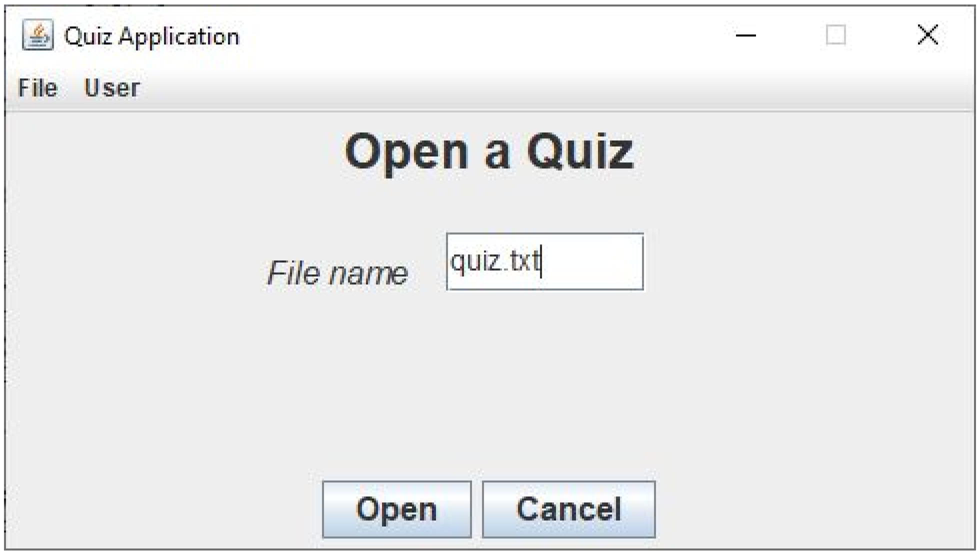
Option 4

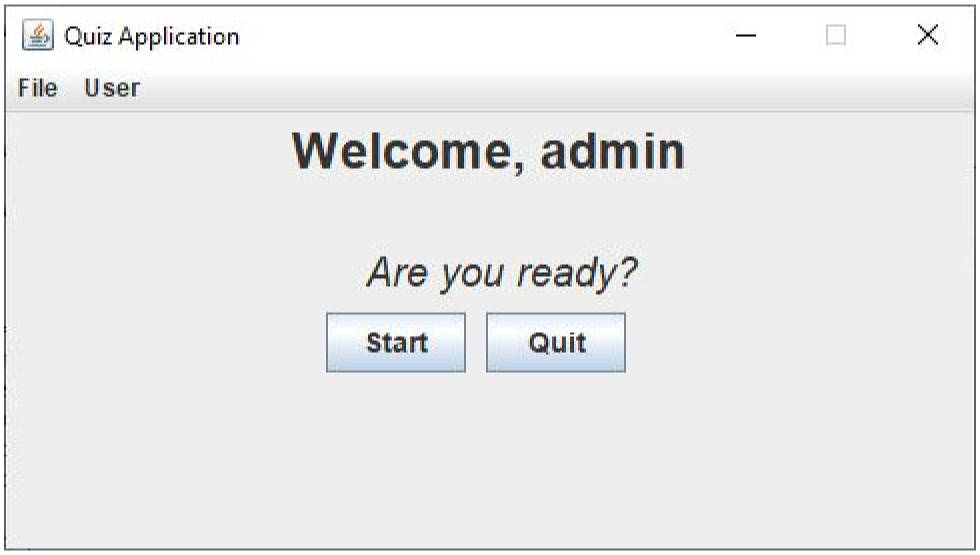
Answer Key int

Create a new Quiz

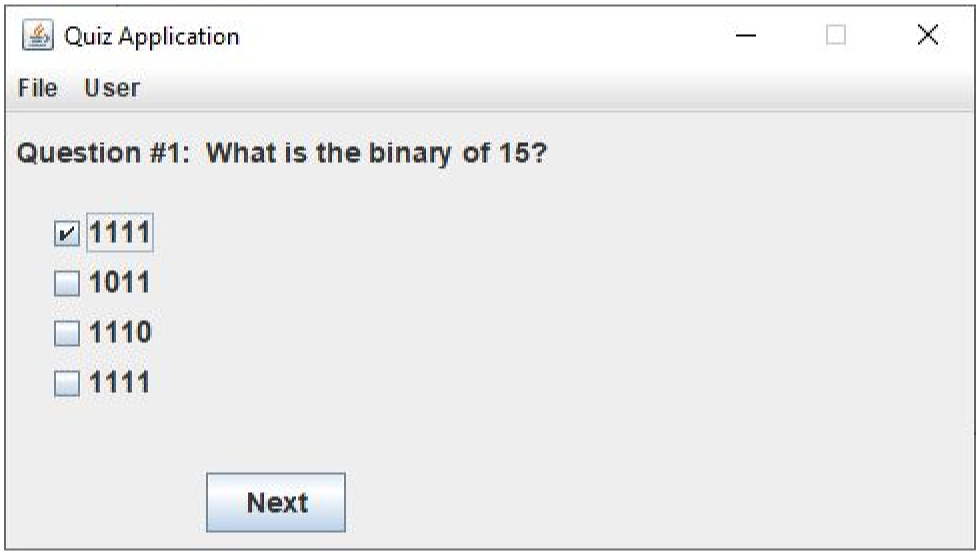
Created Succesfully

## Open a Quiz



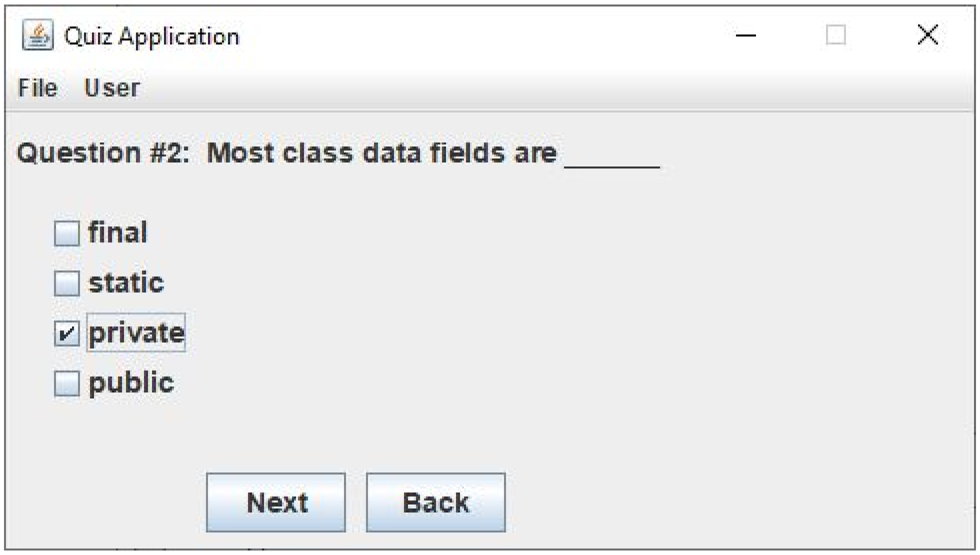


## Taking a Quiz



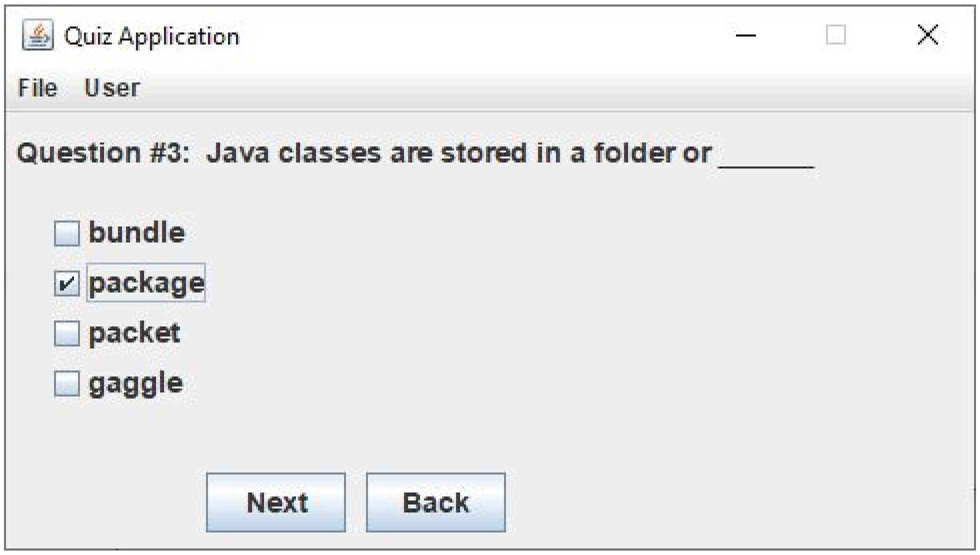
Question #1: What is the binary of 15?

[✓ 1111]
1011
1110
1111



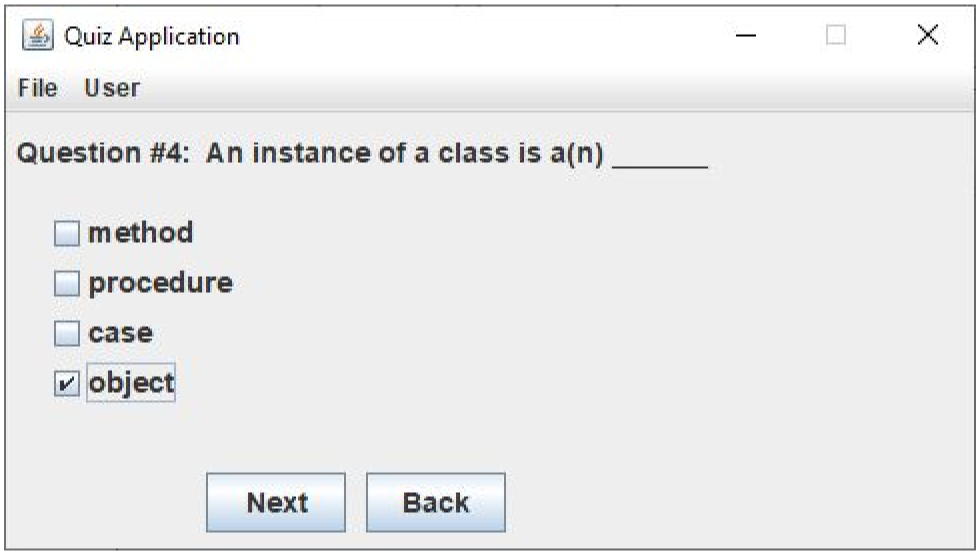
Question #2: Most class data fields are ______

final
static
[✓ private]
public



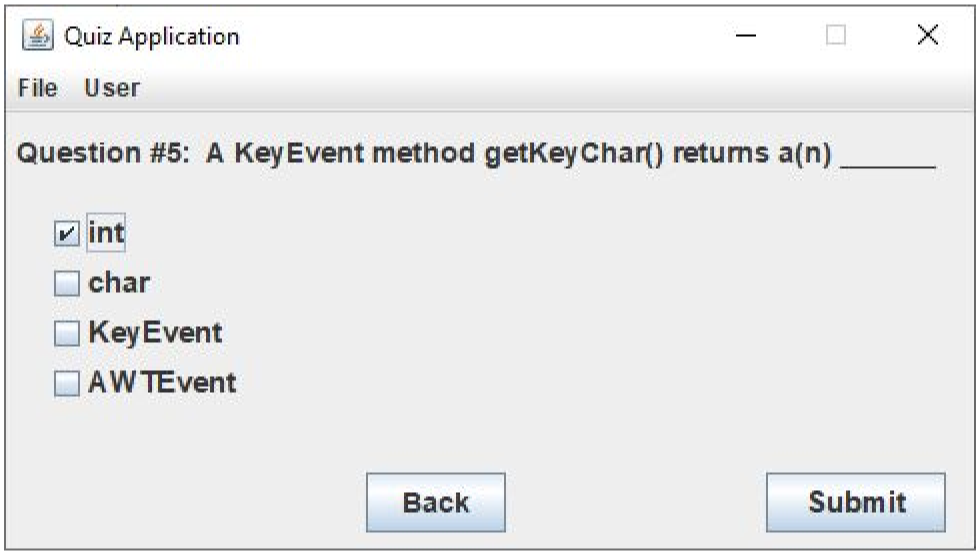
Question #3: Java classes are stored in a folder or ______

bundle
[✓ package]
packet
gaggle



Question #4: An instance of a class is a(n) ______

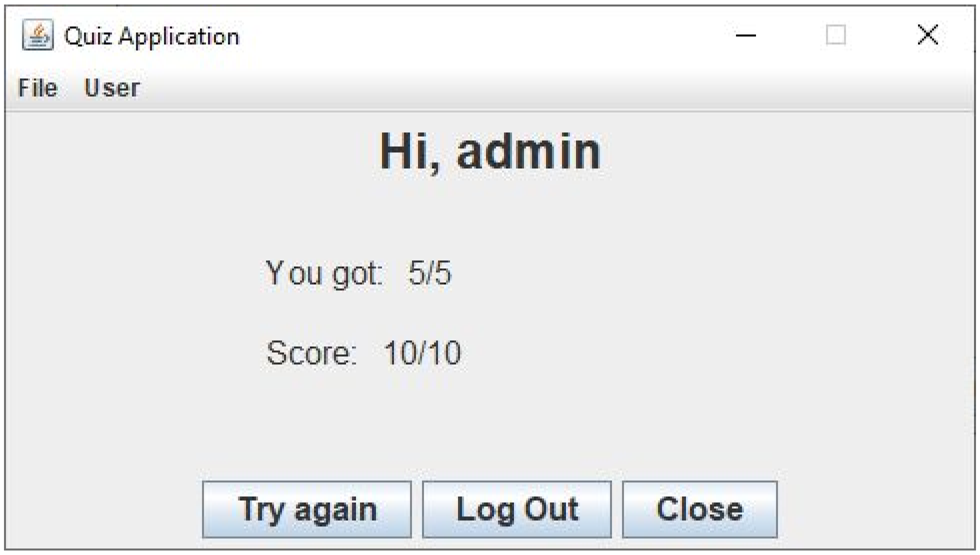
- [ ] method
- [ ] procedure
- [ ] case
- [✓] object



Question #5: A KeyEvent method getKeyChar() returns a(n) ______

- [✓] int
- [ ] char
- [ ] KeyEvent
- [ ] AWTEvent

**After taking a Quiz**

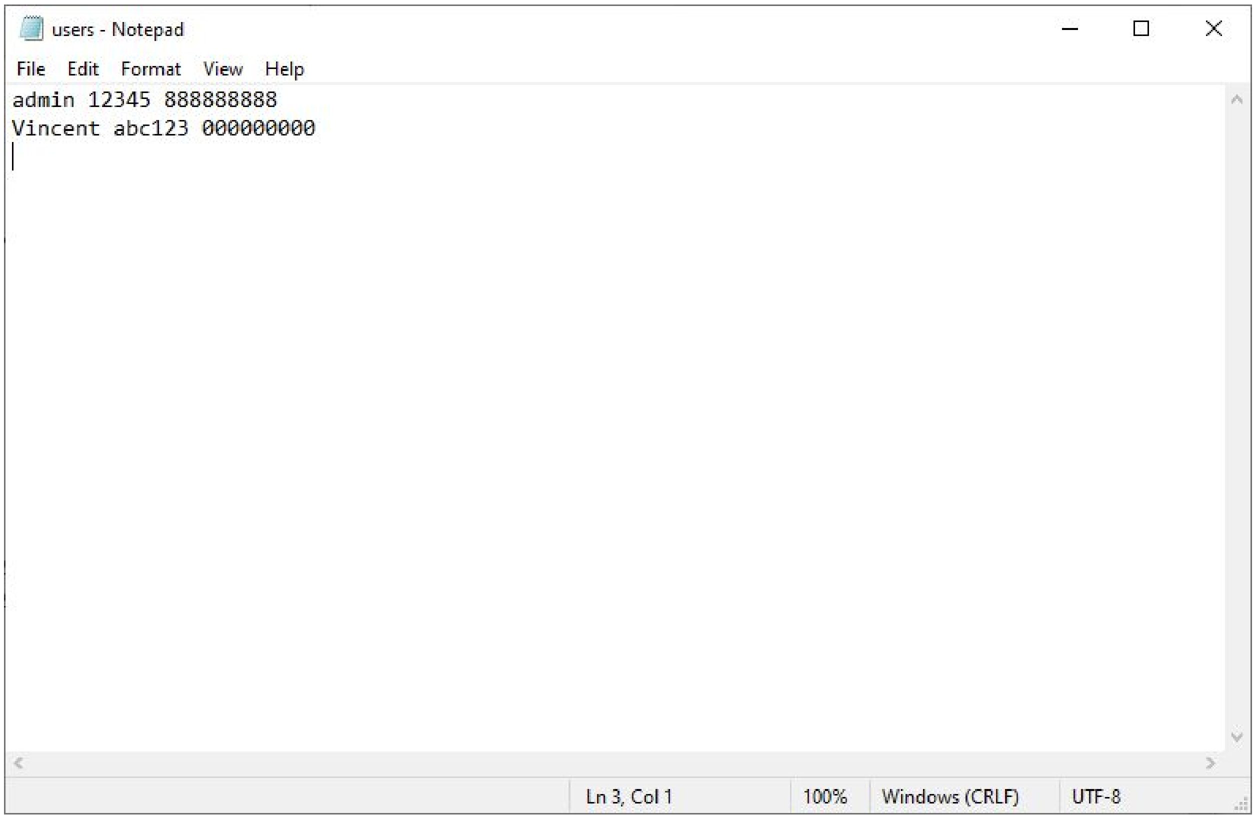


Hi, admin

You got: 5/5

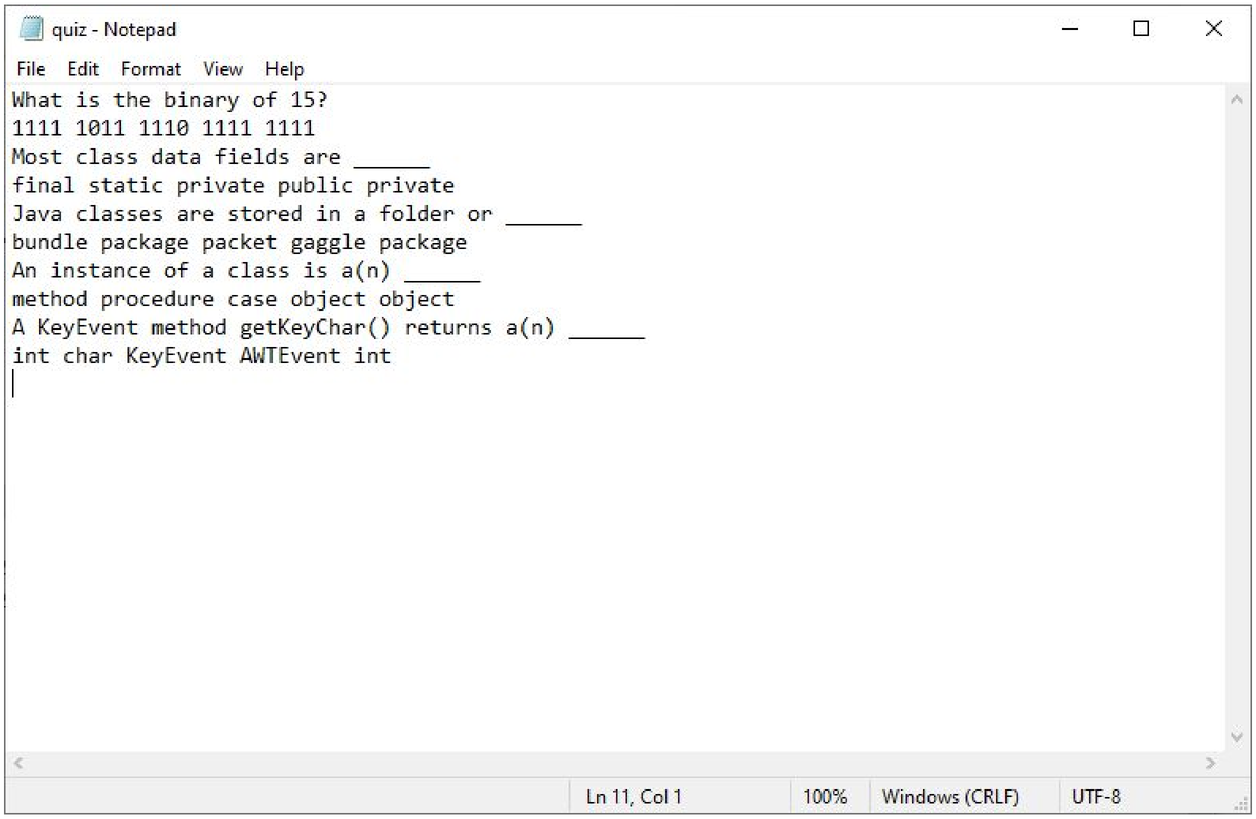
Score: 10/10

**‘users.txt’**



```
admin 12345 888888888
Vincent abc123 000000000

```

**‘quiz.txt’**



```
What is the binary of 15?
1111 1011 1110 1111 1111
Most class data fields are ______
final static private public private
Java classes are stored in a folder or ______
bundle package packet gaggle package
An instance of a class is a(n) ______
method procedure case object object
A KeyEvent method getKeyChar() returns a(n) ______
int char KeyEvent AWTEvent int
```