

## Final Project - Quiz Application

Code:

‘FinalProject.java’

```
FinalProject.java Account.java Quiz.java File.java
1  /*
2   * Name: Vinh Tran
3   * Date: 11/26/2019
4   * Final Project
5   * This project is about the quiz application that allows users to take or create a quiz
6   */
7
8  public class FinalProject {
9      public static void main(String[] args) {
10         Quiz quiz = new Quiz();
11         quiz.setVisible(true);
12     }
13 }
14
```

‘Account.java’

```
FinalProject.java Account.java Quiz.java File.java
1  // Libraries for I/O File
2  import java.nio.file.*;
3  import java.io.*;
4  import static java.nio.file.StandardOpenOption.*;
5  import java.util.*;
6
7  // 'Account' class will help the application keep track user's accounts
8  // by storing, loading, checking them
9  public class Account {
10     // A vector of 'Account' objects
11     Vector<Account> account = new Vector<Account>();
12     // Data fields for an 'Account' object
13     String username;
14     String password;
15     String CWID;
16
17     // Default constructor of 'Account' class
18     public Account() {}
19
20     // Non-default constructor of 'Account' class
21     // This constructor will store the value for data fields
22     public Account(String u, String p, String id) {
23         username = u;
24         password = p;
25         CWID = id;
26     }
27
28     // 'createAccount' method will store the registered accounts
29     // by writing all information to the file named 'users.txt'
30     public void createAccount(String userName, String passWord, String ID) {
31         try {
32             FileOutputStream output = new FileOutputStream("D:\\users.txt", true);
33             BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(output));
34             writer.write(userName + " " + passWord + " " + ID);
35             writer.newLine();
36             writer.flush();
37             writer.close();
38         } catch (Exception e) {
39             System.out.println("Message: " + e);
40         }
41     }
42 }
```

```

42
43 // 'readAccount' method will read all information related to the account line by line from 'user.txt'
44 // then save them into a vector of 'Account' objects called 'account'
45 public void readAccount() {
46     String[] eachString = new String[3];
47     String u;
48     String p;
49     String id;
50     String readString;
51     Path file = Paths.get("D:\\users.txt");
52     InputStream input = null;
53     try {
54         input = Files.newInputStream(file);
55         BufferedReader reader = new BufferedReader(new InputStreamReader(input));
56         readString = reader.readLine();
57         while (readString != null) {
58             eachString = readString.split(" ");
59             u = eachString[0];
60             p = eachString[1];
61             id = eachString[2];
62             Account newAccount = new Account(u, p, id);
63             account.addElement(newAccount);
64             readString = reader.readLine();
65         }
66         input.close();
67     } catch (IOException e) {
68         System.out.println(e);
69     }
70 }

71
72 // 'getID' method will return an ID number as a string related to account's username and password
73 public String getID(String u, String p) {
74     String id = null;
75     for (int i = 0; i < account.size(); i++) {
76         if (account.get(i).username.equals(u) && account.get(i).password.equals(p)) {
77             id = account.get(i).CWID;
78         }
79     }
80     return id;
81 }

82
83 // 'checkAccount' method will check if the account does exist or not
84 // return 'true' if the account exists; otherwise, return 'false'
85 public boolean checkAccount(String u, String p) {
86     for (int i = 0; i < account.size(); i++) {
87         if (account.get(i).username.equals(u) && account.get(i).password.equals(p)) {
88             return true;
89         }
90     }
91     return false;
92 }
93 }
94

```

**'File.java'**

```

FinalProject.java Account.java Quiz.java File.java ✕
1 // Libraries for I/O File and GUI
2 import java.nio.file.*;
3 import java.io.*;
4 import static java.nio.file.StandardOpenOption.*;
5 import java.util.*;
6 import javax.swing.*;
7
8 // 'File' class will help the application keep track the Quiz
9 // by creating, loading, checking it
10 public class File {
11     // A vector of 'File' objects that store questions
12     Vector<File> saveQuestions = new Vector<File>();
13     // A vector of 'File' objects that load questions
14     Vector<File> loadQuestions = new Vector<File>();
15     // Data fields for a 'File' object
16     String question;
17     String choice1;
18     String choice2;
19     String choice3;
20     String choice4;
21     String answer;
22     // Paths that link to the file
23     Path createPath;
24     Path checkPath;
25
26     // Default constructor
27     public File() {}
28
29     // Non-default constructor
30     // This constructor will store the value of data fields
31     public File(String q, String c1, String c2, String c3, String c4, String a) {
32         question = q;
33         choice1 = c1;
34         choice2 = c2;
35         choice3 = c3;
36         choice4 = c4;
37         answer = a;
38     }
39
40     // 'createFile' method will create a file based on the parameter 'f'
41     public boolean createFile(String f) {
42         createPath = Paths.get("D:\\\" + f);
43         try {
44             OutputStream createFile = new BufferedOutputStream(Files.newOutputStream(createPath, CREATE_NEW));
45             return true;
46         } catch (Exception e) {
47             JOptionPane.showMessageDialog(null, "Can't create since " + f + " is already existent!!", "Error", JOptionPane.WARNING_MESSAGE);
48         }
49         return false;
50     }
51
52     // 'checkFile' method will check if the file does exist or not based on the parameter 'f'
53     // return 'true' if the file exists; otherwise, return 'false'
54     public boolean checkFile(String f) {
55         checkPath = Paths.get("D:\\\" + f);
56         if (Files.exists(checkPath)) {
57             return true;
58         }
59         return false;
60     }
61
62     // 'createQuestions' method will create each question based on the parameters
63     // Then store all questions into a vector of 'File' object called 'saveQuestions'
64     public void createQuestions(String q, String c1, String c2, String c3, String c4, String a) {
65         File newQuestion = new File(q, c1, c2, c3, c4, a);
66         saveQuestions.addElement(newQuestion);
67     }
68
69     // 'deleteQuestions' method will delete a question at the specific index
70     public void deleteQuestions(int index) {
71         saveQuestions.remove(index);
72     }

```

```

73
74 // 'createQuiz' method will create a Quiz by writing into a file based on the path
75 public void createQuiz() {
76     try {
77         OutputStream output = new BufferedOutputStream(Files.newOutputStream(createPath, APPEND));
78         BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(output));
79         for (int i = 0; i < saveQuestions.size(); i++) {
80             writer.write(saveQuestions.get(i).question);
81             writer.newLine();
82             writer.write(saveQuestions.get(i).choice1 + " " + saveQuestions.get(i).choice2 + " " +
83                 saveQuestions.get(i).choice3 + " " + saveQuestions.get(i).choice4 + " " +
84                 saveQuestions.get(i).answer);
85             writer.newLine();
86         }
87         writer.flush();
88         writer.close();
89     } catch (Exception e) {
90         System.out.println("Message: " + e);
91     }
92 }
93
94 // 'loadQuiz' method will load a Quiz by reading a file based on the path
95 public void loadQuiz(String f) {
96     createPath = Paths.get("D:\\\" + f);
97     String[] loadString = new String[5];
98     String q;
99     String c1;
100    String c2;
101    String c3;
102    String c4;
103    String a;
104    String readString;
105    InputStream input = null;
106    try {
107        input = Files.newInputStream(createPath);
108        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
109        readString = reader.readLine();
110        while (readString != null) {
111            q = readString;
112            readString = reader.readLine();
113            loadString = readString.split(" ");
114            c1 = loadString[0];
115            c2 = loadString[1];
116            c3 = loadString[2];
117            c4 = loadString[3];
118            a = loadString[4];
119            File openQuestion = new File(q, c1, c2, c3, c4, a);
120            loadQuestions.addElement(openQuestion);
121            readString = reader.readLine();
122        }
123        input.close();
124    } catch (IOException e) {
125        System.out.println(e);
126    }
127 }
128 }

```

**‘Quiz.java’**



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

```
1 // Libraries for creating the application
2 import javax.swing.*;
3 import java.awt.*;
4 import java.awt.event.*;
5
6 // 'Quiz' class will be the main control of the application
7 // This class will allow users to:
8 // 1. Create an account
9 // 2. Log in with an existent account
10 // 3. See the info of the account
11 // 4. Create a Quiz
12 // 5. Open a Quiz
13 // 6. Take/Re-take a Quiz
14 public class Quiz extends JFrame implements ActionListener, ItemListener {
15     // Data fields
16     // Initialize an object 'account' for 'Account' class
17     Account account = new Account();
18     // Initialize an object 'fileQuiz' for 'File' class
19     File fileQuiz = new File();
20     // Number of correct questions and score after taking a quiz
21     private int countCorrect = 0;
22     private int score = 0;
23     // Store the user's choice while taking a Quiz
24     boolean[] checkAnswers = new boolean[5];
25
26     // Set up the menu for the application
27     JMenuBar menuBar = new JMenuBar();
28     JMenu file = new JMenu("File");
29     JMenu user = new JMenu("User");
30     JMenuItem signUp = new JMenuItem("Sign Up");
31     JMenuItem info = new JMenuItem("Info");
32     JMenuItem newFile = new JMenuItem("New");
33     JMenuItem openFile = new JMenuItem("Open");
34     JMenuItem close = new JMenuItem("Close");
35 }
```

```

36 // Create a deck of cards for the application
37 CardLayout cards = new CardLayout();
38 // Create each card for the application
39 // 'card1' is the launching application
40 JPanel card1 = new JPanel();
41 // 'card2' is the register panel
42 JPanel card2 = new JPanel();
43 // 'card3' is the panel after logging in
44 JPanel card3 = new JPanel();
45 // 'card4' is the creation of a Quiz
46 JPanel card4 = new JPanel();
47 // 'card5' is the open of a Quiz
48 JPanel card5 = new JPanel();
49 // 'createQuestionCards' is the array of panels of new questions
50 JPanel[] createQuestionCards = new JPanel[5];
51 // 'card11' is the panel after opening a quiz
52 JPanel card11 = new JPanel();
53 // 'loadQuestionCards' is the array of panels of open questions
54 JPanel[] loadQuestionCards = new JPanel[5];
55 // 'card17' is the result of a Quiz
56 JPanel card17 = new JPanel();
57
58 // Initialize the contents for 'card1'
59 JPanel northPanel1 = new JPanel();
60 JLabel heading = new JLabel("Welcome to Quiz Application");
61
62 JPanel centerPanel1 = new JPanel();
63 JLabel signIn = new JLabel("----- Sign In -----");
64 JLabel userName = new JLabel("Username");
65 JLabel passWord = new JLabel("Password");
66 JTextField userText = new JTextField(10);
67 JTextField passText = new JTextField(10);
68
69 JPanel southPanel1 = new JPanel();
70 JButton login = new JButton("Login");
71

```

```

72 // Initialize the contents for 'card2'
73 JPanel northPanel2 = new JPanel();
74 JLabel heading2 = new JLabel("Sign Up An Account");
75
76 JPanel centerPanel2 = new JPanel();
77 JLabel accountName = new JLabel("Account Name");
78 JLabel accountPass = new JLabel("Password");
79 JLabel id = new JLabel("CWID");
80 JTextField nameText = new JTextField(10);
81 JTextField passText2 = new JTextField(10);
82 JTextField idText = new JTextField(10);
83
84 JPanel southPanel2 = new JPanel();
85 JButton create = new JButton("Create");
86 JButton cancel = new JButton("Cancel");
87
88 // Initialize the contents for 'card3'
89 JPanel northPanel3 = new JPanel();
90 JLabel heading3 = new JLabel();
91
92 JPanel centerPanel3 = new JPanel();
93 JButton newQuiz = new JButton("Create a Quiz");
94 JButton openQuiz = new JButton("Open a Quiz");
95 JButton logout = new JButton("Log Out");
96
97 // Initialize the contents for 'card4'
98 JPanel northPanel4 = new JPanel();
99 JLabel heading4 = new JLabel("Create a New Quiz");
100
101 JPanel centerPanel4 = new JPanel();
102 JLabel fileName = new JLabel("File name");
103 JLabel numOfQuestions = new JLabel("Number of Questions");
104 JTextField fileText = new JTextField(10);
105 JTextField questionsText = new JTextField(10);
106
107 JPanel southPanel4 = new JPanel();
108 JButton next = new JButton("Next");
109 JButton cancel2 = new JButton("Cancel");
110

```

```

111 // Initialize the contents for 'card5'
112 JPanel northPanel5 = new JPanel();
113 JLabel heading5 = new JLabel("Open a Quiz");
114
115 JPanel centerPanel5 = new JPanel();
116 JLabel fileName2 = new JLabel("File name");
117 JTextField fileText2 = new JTextField(10);
118
119 JPanel southPanel5 = new JPanel();
120 JButton open = new JButton("Open");
121 JButton cancel3 = new JButton("Cancel");
122
123 // Initialize the contents for the array of 'createQuestionCards'
124 JButton[] nexts = new JButton[4];
125 String[] numberCard = new String[5];
126 JLabel[] questions = new JLabel[5];
127 JTextField[] questionTexts = new JTextField[5];
128 JLabel[] choices = new JLabel[20];
129 JTextField[] choiceTexts = new JTextField[20];
130 JLabel[] keys = new JLabel[5];
131 JTextField[] keyTexts = new JTextField[5];
132 JButton[] backs = new JButton[4];
133 JButton[] cancels = new JButton[5];
134 JButton add = new JButton("Add");
135
136 // Initialize the contents for 'card11'
137 JPanel northPanel6 = new JPanel();
138 JLabel heading6 = new JLabel();
139
140 JPanel centerPanel6 = new JPanel();
141 JLabel ready = new JLabel("Are you ready?");
142 JButton start = new JButton("Start");
143 JButton quit = new JButton("Quit");
144

```



```

145 // Initialize the contents for the array of 'loadQuestionCards'
146 String[] numberCard2 = new String[5];
147 JLabel[] questionNumber = new JLabel[5];
148 JLabel[] titleQuestion = new JLabel[5];
149 JCheckBox[] multipleChoice = new JCheckBox[20];
150 JButton[] nexts2 = new JButton[4];
151 JButton[] backs2 = new JButton[4];
152 JButton submit = new JButton("Submit");
153 ButtonGroup[] groups = new ButtonGroup[5];
154
155 // Initialize the contents for 'card17'
156 JPanel northPanel7 = new JPanel();
157 JLabel heading7 = new JLabel();
158
159 JPanel centerPanel7 = new JPanel();
160 JLabel correctQuestions = new JLabel();
161 JLabel result = new JLabel();
162
163 JPanel southPanel7 = new JPanel();
164 JButton tryAgain = new JButton("Try again");
165 JButton logout2 = new JButton("Log Out");
166 JButton close2 = new JButton("Close");
167
168 public Quiz() {
169     // Create a frame for the application
170     super("Quiz Application");
171     setSize(500, 280);
172     setResizable(false);
173     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
174     setLocationRelativeTo(null);
175     setLayout(cards);
176
177     // Set up the menu for the application
178     setJMenuBar(menuBar);
179     newFile.setFont(new Font("Arial", Font.PLAIN, 12));
180     openFile.setFont(new Font("Arial", Font.PLAIN, 12));
181     close.setFont(new Font("Arial", Font.PLAIN, 12));
182     signUp.setFont(new Font("Arial", Font.PLAIN, 12));
183     info.setFont(new Font("Arial", Font.PLAIN, 12));
184     menuBar.add(file);
185     menuBar.add(user);
186     file.add(newFile);
187     file.add(openFile);
188     file.add(close);
189     user.add(signUp);
190     user.add(info);
191     newFile.setEnabled(false);
192     openFile.setEnabled(false);
193

```

```
194 // Design and build 'card1'
195 // Create the heading for 'card1'
196 card1.setLayout(new BorderLayout());
197 card1.add(northPanel1, BorderLayout.NORTH);
198 heading.setFont(new Font("Arial", Font.BOLD, 30));
199 northPanel1.setLayout(new FlowLayout());
200 northPanel1.add(heading);
201 // Create the login content for 'card1'
202 card1.add(centerPanel1, BorderLayout.CENTER);
203 signIn.setFont(new Font("Serif", Font.BOLD, 25));
204 signIn.setBounds(150, 10, 200, 30);
205 userName.setFont(new Font("Arial", Font.PLAIN, 17));
206 userName.setBounds(120, 50, 150, 30);
207 passWord.setFont(new Font("Arial", Font.PLAIN, 17));
208 passWord.setBounds(120, 90, 150, 30);
209 userText.setFont(new Font("Arial", Font.PLAIN, 14));
210 userText.setBounds(210, 50, 150, 30);
211 passText.setFont(new Font("Arial", Font.PLAIN, 14));
212 passText.setBounds(210, 90, 150, 30);
213 centerPanel1.setLayout(null);
214 centerPanel1.add(signIn);
215 centerPanel1.add(userName);
216 centerPanel1.add(userText);
217 centerPanel1.add(passWord);
218 centerPanel1.add(passText);
219 // Create the button for 'card1'
220 card1.add(southPanel1, BorderLayout.SOUTH);
221 southPanel1.setLayout(new FlowLayout());
222 login.setFont(new Font("Arial", Font.BOLD, 16));
223 southPanel1.add(login);
224 // Add 'card1' to the application
225 add(card1, "Card 1");
226
```

```
227 // Design and build 'card2'
228 // Create the heading for 'card2'
229 card2.setLayout(new BorderLayout());
230 card2.add(northPanel2, BorderLayout.NORTH);
231 heading2.setFont(new Font("Arial", Font.BOLD, 20));
232 northPanel2.setLayout(new FlowLayout());
233 northPanel2.add(heading2);
234 // Create the creation content for 'card2'
235 card2.add(centerPanel2, BorderLayout.CENTER);
236 accountName.setFont(new Font("Arial", Font.PLAIN, 15));
237 accountName.setBounds(110, 10, 150, 30);
238 accountPass.setFont(new Font("Arial", Font.PLAIN, 15));
239 accountPass.setBounds(110, 50, 150, 30);
240 id.setFont(new Font("Arial", Font.PLAIN, 15));
241 id.setBounds(110, 90, 150, 30);
242 nameText.setFont(new Font("Arial", Font.PLAIN, 14));
243 nameText.setBounds(220, 10, 150, 30);
244 passText2.setFont(new Font("Arial", Font.PLAIN, 14));
245 passText2.setBounds(220, 50, 150, 30);
246 idText.setFont(new Font("Arial", Font.PLAIN, 14));
247 idText.setBounds(220, 90, 150, 30);
248 centerPanel2.setLayout(null);
249 centerPanel2.add(accountName);
250 centerPanel2.add(nameText);
251 centerPanel2.add(accountPass);
252 centerPanel2.add(passText2);
253 centerPanel2.add(id);
254 centerPanel2.add(idText);
255 // Create the button for 'card2'
256 card2.add(southPanel2, BorderLayout.SOUTH);
257 southPanel2.setLayout(new FlowLayout());
258 create.setFont(new Font("Arial", Font.BOLD, 16));
259 cancel.setFont(new Font("Arial", Font.BOLD, 16));
260 southPanel2.add(create);
261 southPanel2.add(cancel);
262 // Add 'card2' to the application
263 add(card2, "Card 2");
264
```



```

265 // Design and build 'card3'
266 // Create the heading for 'card3'
267 card3.setLayout(new BorderLayout());
268 heading3.setFont(new Font("Arial", Font.BOLD, 25));
269 card3.add(northPanel3, BorderLayout.NORTH);
270 northPanel3.setLayout(new FlowLayout());
271 northPanel3.add(heading3);
272 // Create the contents after logging in for 'card3'
273 card3.add(centerPanel3, BorderLayout.CENTER);
274 newQuiz.setFont(new Font("Arial", Font.BOLD, 16));
275 newQuiz.setBounds(90, 20, 300, 40);
276 openQuiz.setFont(new Font("Arial", Font.BOLD, 16));
277 openQuiz.setBounds(90, 70, 300, 40);
278 logout.setFont(new Font("Arial", Font.BOLD, 16));
279 logout.setBounds(90, 120, 300, 40);
280 centerPanel3.setLayout(null);
281 centerPanel3.add(newQuiz);
282 centerPanel3.add(openQuiz);
283 centerPanel3.add(logout);
284 // Add 'card3' to the application
285 add(card3, "Card 3");
286
287 // Design and build 'card4'
288 // Create the heading for 'card4'
289 card4.setLayout(new BorderLayout());
290 heading4.setFont(new Font("Arial", Font.BOLD, 25));
291 card4.add(northPanel4, BorderLayout.NORTH);
292 northPanel4.add(heading4);
293 // Create the contents of creation for 'card4'
294 card4.add(centerPanel4, BorderLayout.CENTER);
295 fileName.setFont(new Font("Arial", Font.ITALIC, 16));
296 fileName.setBounds(130, 20, 300, 40);
297 numOfQuestions.setFont(new Font("Arial", Font.ITALIC, 16));
298 numOfQuestions.setBounds(130, 60, 300, 40);
299 fileText.setFont(new Font("Arial", Font.PLAIN, 14));
300 fileText.setBounds(220, 20, 100, 30);
301 questionsText.setText("5");
302 questionsText.setEnabled(false);
303 questionsText.setFont(new Font("Arial", Font.PLAIN, 14));
304 questionsText.setBounds(300, 60, 30, 30);
305 centerPanel4.setLayout(null);
306 centerPanel4.add(fileName);
307 centerPanel4.add(fileText);
308 centerPanel4.add(numOfQuestions);
309 centerPanel4.add(questionsText);
310 // Create the buttons for 'card4'
311 card4.add(southPanel4, BorderLayout.SOUTH);
312 southPanel4.setLayout(new FlowLayout());
313 next.setFont(new Font("Arial", Font.BOLD, 16));
314 cancel2.setFont(new Font("Arial", Font.BOLD, 16));
315 southPanel4.add(next);
316 southPanel4.add(cancel2);
317 // Add 'card2' to the application
318 add(card4, "Card 4");
319

```



```
320 // Design and build 'card5'
321 // Create the heading for 'card5'
322 card5.setLayout(new BorderLayout());
323 heading5.setFont(new Font("Arial", Font.BOLD, 25));
324 card5.add(northPanel5, BorderLayout.NORTH);
325 northPanel5.add(heading5);
326 // Create the contents of opening for 'card5'
327 card5.add(centerPanel5, BorderLayout.CENTER);
328 fileName2.setFont(new Font("Arial", Font.ITALIC, 16));
329 fileName2.setBounds(130, 20, 300, 40);
330 fileText2.setFont(new Font("Arial", Font.PLAIN, 14));
331 fileText2.setBounds(220, 20, 100, 30);
332 centerPanel5.setLayout(null);
333 centerPanel5.add(fileName2);
334 centerPanel5.add(fileText2);
335 // Create the buttons for 'card5'
336 card5.add(southPanel5, BorderLayout.SOUTH);
337 southPanel5.setLayout(new FlowLayout());
338 open.setFont(new Font("Arial", Font.BOLD, 16));
339 cancel3.setFont(new Font("Arial", Font.BOLD, 16));
340 southPanel5.add(open);
341 southPanel5.add(cancel3);
342 // Add 'card2' to the application
343 add(card5, "Card 5");
344
```

```

345 // Design and build 'createQuestionsCards'
346 // Create the contents for 'createQuestionCards'
347 for (int i = 0, j = 0, yPos = 0; i < 5; i++) {
348     // Initialize the panel for each 'createQuestionCards'
349     createQuestionCards[i] = new JPanel();
350     createQuestionCards[i].setLayout(null);
351
352     // Set up the contents for 'createQuestionCards'
353     // These contents will allow users to create a Quiz
354     // by typing the questions, the possible choices, and the answer
355     questions[i] = new JLabel();
356     questions[i].setText("Question #" + (i + 1));
357     questions[i].setFont(new Font("Arial", Font.BOLD, 16));
358     questions[i].setBounds(5, 10, 100, 20);
359
360     questionTexts[i] = new JTextField(10);
361     questionTexts[i].setFont(new Font("Arial", Font.PLAIN, 15));
362     questionTexts[i].setBounds(100, 10, 380, 20);
363
364     createQuestionCards[i].add(questions[i]);
365     createQuestionCards[i].add(questionTexts[i]);
366
367     for (; j < (i + 1) * 4; j++) {
368         choices[j] = new JLabel();
369         choices[j].setText("Option " + (j + 1 - (4 * i)));
370         choices[j].setFont(new Font("Arial", Font.BOLD, 15));
371         choices[j].setBounds(20, 50 + yPos, 100, 20);
372
373         choiceTexts[j] = new JTextField(10);
374         choiceTexts[j].setFont(new Font("Arial", Font.PLAIN, 15));
375         choiceTexts[j].setBounds(100, 50 + yPos, 150, 20);
376         createQuestionCards[i].add(choices[j]);
377         createQuestionCards[i].add(choiceTexts[j]);
378         yPos += 25;
379     }
380

```

```

381 keys[i] = new JLabel("Answer Key");
382 keys[i].setFont(new Font("Arial", Font.BOLD, 15));
383 keys[i].setBounds(20, 50 + yPos, 100, 20);
384 createQuestionCards[i].add(keys[i]);
385
386 keyTexts[i] = new JTextField(10);
387 keyTexts[i].setFont(new Font("Arial", Font.PLAIN, 15));
388 keyTexts[i].setBounds(120, 50 + yPos, 150, 20);
389 createQuestionCards[i].add(keyTexts[i]);
390 yPos = 0;
391
392 // Set up buttons for creating a Quiz to flip the cards
393 if (i < 4) {
394     nexts[i] = new JButton("Next");
395     nexts[i].setFont(new Font("Arial", Font.BOLD, 14));
396     nexts[i].setBounds(100, 180, 70, 30);
397     createQuestionCards[i].add(nexts[i]);
398 }
399 if (i > 0 && i < 5) {
400     backs[i - 1] = new JButton("Back");
401     backs[i - 1].setFont(new Font("Arial", Font.BOLD, 14));
402     backs[i - 1].setBounds(180, 180, 70, 30);
403     createQuestionCards[i].add(backs[i - 1]);
404 }
405 if (i == 4) {
406     add.setFont(new Font("Arial", Font.BOLD, 15));
407     add.setBounds(400, 180, 70, 30);
408     createQuestionCards[i].add(add);
409 }
410
411 cancels[i] = new JButton("Cancel");
412 cancels[i].setFont(new Font("Arial", Font.BOLD, 14));
413 cancels[i].setBounds(260, 180, 90, 30);
414 createQuestionCards[i].add(cancels[i]);
415 numberCard[i] = "Card " + (6 + i);
416 // Add each card to the application
417 add(createQuestionCards[i], numberCard[i]);
418 }
419

```

```

420 // Design and build 'card11'
421 // Create the heading for 'card11'
422 card11.setLayout(new BorderLayout());
423 heading6.setFont(new Font("Arial", Font.BOLD, 25));
424 card11.add(northPanel6, BorderLayout.NORTH);
425 northPanel6.add(heading6);
426 // Create the contents for 'card11'
427 card11.add(centerPanel6, BorderLayout.CENTER);
428 ready.setFont(new Font("Arial", Font.ITALIC, 20));
429 ready.setBounds(180, 20, 300, 40);
430 start.setFont(new Font("Arial", Font.BOLD, 14));
431 start.setBounds(160, 60, 70, 30);
432 quit.setFont(new Font("Arial", Font.BOLD, 14));
433 quit.setBounds(240, 60, 70, 30);
434 centerPanel6.setLayout(null);
435 centerPanel6.add(ready);
436 centerPanel6.add(start);
437 centerPanel6.add(quit);
438 // Add 'card2' to the application
439 add(card11, "Card 11");
440
441 // Create the contents for 'loadQuestionCards'
442 // Create the buttons for 'loadQuestionCards'
443 for (int i = 0; i < 4; i++) {
444     nexts2[i] = new JButton("Next");
445     backs2[i] = new JButton("Back");
446 }
447 // Create CheckBoxes for 'loadQuestionCards'
448 for (int i = 0; i < 20; i++) {
449     multipleChoice[i] = new JCheckBox();
450 }
451

```



```
452 // Design and build 'card17'
453 // Create the heading for 'card17'
454 card17.setLayout(new BorderLayout());
455 heading7.setFont(new Font("Arial", Font.BOLD, 25));
456 card17.add(northPanel7, BorderLayout.NORTH);
457 northPanel7.add(heading7);
458 // Create the contents for the result of 'card17'
459 card17.add(centerPanel7, BorderLayout.CENTER);
460 correctQuestions.setFont(new Font("Arial", Font.PLAIN, 16));
461 correctQuestions.setBounds(130, 20, 300, 40);
462 result.setFont(new Font("Arial", Font.PLAIN, 16));
463 result.setBounds(130, 60, 300, 40);
464 centerPanel7.setLayout(null);
465 centerPanel7.add(correctQuestions);
466 centerPanel7.add(result);
467 // Create the buttons for 'card17'
468 card17.add(southPanel7, BorderLayout.SOUTH);
469 southPanel7.setLayout(new FlowLayout());
470 tryAgain.setFont(new Font("Arial", Font.BOLD, 16));
471 logout2.setFont(new Font("Arial", Font.BOLD, 16));
472 close2.setFont(new Font("Arial", Font.BOLD, 16));
473 southPanel7.add(tryAgain);
474 southPanel7.add(logout2);
475 southPanel7.add(close2);
476 // Add 'card2' to the application
477 add(card17, "Card 17");
478
```

```
479 // Add the event/logic for buttons and items of the application
480 signUp.addActionListener(this);
481 login.addActionListener(this);
482 close.addActionListener(this);
483 create.addActionListener(this);
484 cancel.addActionListener(this);
485 logout.addActionListener(this);
486 info.addActionListener(this);
487 next.addActionListener(this);
488 cancel2.addActionListener(this);
489 newFile.addActionListener(this);
490 newQuiz.addActionListener(this);
491 openFile.addActionListener(this);
492 openQuiz.addActionListener(this);
493 cancel3.addActionListener(this);
494 add.addActionListener(this);
495 open.addActionListener(this);
496 quit.addActionListener(this);
497 submit.addActionListener(this);
498 start.addActionListener(this);
499 tryAgain.addActionListener(this);
500 logout2.addActionListener(this);
501 close2.addActionListener(this);
502 for (int i = 0; i < 4; i++) {
503     nexts[i].addActionListener(this);
504     backs[i].addActionListener(this);
505     nexts2[i].addActionListener(this);
506     backs2[i].addActionListener(this);
507 }
508 for (int i = 0; i < 5; i++) {
509     cancels[i].addActionListener(this);
510 }
511 for (int i = 0; i < 20; i++) {
512     multipleChoice[i].addItemListener(this);
513 }
514 }
515 }
```

```

516 // This method will keep track users's corrections and score when taking a Quiz
517 // If they select the correct answer, they gain 2 points
518 // If they select the incorrect answer, they gain 0 points
519 @Override
520 public void itemStateChanged(ItemEvent e) {
521     Object object = e.getItem();
522     int select = e.getStateChange();
523
524     // Check user's choice for Question #1
525     for (int i = 0; i < 4; i++) {
526         if (object == multipleChoice[i]) {
527             String choice = multipleChoice[i].getText();
528             if (select == ItemEvent.SELECTED) {
529                 if (choice.equals(fileQuiz.loadQuestions.get(0).answer)) {
530                     checkAnswers[0] = true;
531                 } else {
532                     checkAnswers[0] = false;
533                 }
534             }
535         }
536     }
537
538     // Check user's choice for Question #2
539     for (int i = 4; i < 8; i++) {
540         if (object == multipleChoice[i]) {
541             String choice = multipleChoice[i].getText();
542             if (select == ItemEvent.SELECTED) {
543                 if (choice.equals(fileQuiz.loadQuestions.get(1).answer)) {
544                     checkAnswers[1] = true;
545                 } else {
546                     checkAnswers[1] = false;
547                 }
548             }
549         }
550     }
551 }

```

```

552 // Check user's choice for Question #3
553 for (int i = 8; i < 12; i++) {
554     if (object == multipleChoice[i]) {
555         String choice = multipleChoice[i].getText();
556         if (select == ItemEvent.SELECTED) {
557             if (choice.equals(fileQuiz.loadQuestions.get(2).answer)) {
558                 checkAnswers[2] = true;
559             } else {
560                 checkAnswers[2] = false;
561             }
562         }
563     }
564 }
565
566 // Check user's choice for Question #4
567 for (int i = 12; i < 16; i++) {
568     if (object == multipleChoice[i]) {
569         String choice = multipleChoice[i].getText();
570         if (select == ItemEvent.SELECTED) {
571             if (choice.equals(fileQuiz.loadQuestions.get(3).answer)) {
572                 checkAnswers[3] = true;
573             } else {
574                 checkAnswers[3] = false;
575             }
576         }
577     }
578 }
579
580 // Check user's choice for Question #5
581 for (int i = 16; i < 20; i++) {
582     if (object == multipleChoice[i]) {
583         String choice = multipleChoice[i].getText();
584         if (select == ItemEvent.SELECTED) {
585             if (choice.equals(fileQuiz.loadQuestions.get(4).answer)) {
586                 checkAnswers[4] = true;
587             } else {
588                 checkAnswers[4] = false;
589             }
590         }
591     }
592 }
593 }
594
595 // This method will be a controllers for the application
596 // It will perform several tasks when users click on the menu, buttons, or checkboxes
597 @Override
598 public void actionPerformed(ActionEvent e) {
599     Object source = e.getSource();
600     // Task: Close the application
601     if (source == close || source == close2) {
602         System.exit(0);
603     }
604     // Task: Access the application by typing username and password
605     else if (source == login) {
606         String username = userText.getText();
607         String pass = passText.getText();
608         account.readAccount();
609         if (account.checkAccount(username, pass)) {
610             JOptionPane.showMessageDialog(null, "You successfully logged in!", "Login", JOptionPane.INFORMATION_MESSAGE);
611             heading3.setText("Welcome, " + username);
612             newFile.setEnabled(true);
613             openFile.setEnabled(true);
614             signUp.setEnabled(false);
615             cards.show(getContentPane(), "Card 3");
616         } else {
617             JOptionPane.showMessageDialog(null, "You typed wrong username or/and password!", "Login", JOptionPane.WARNING_MESSAGE);
618         }
619     }

```



```

620 // Task: Create a new account by typing username, password, and CWID
621 else if (source == signUp) {
622     cards.show(getContentPane(), "Card 2");
623 }
624 else if (source == create) {
625     String username = nameText.getText();
626     String pass = passText2.getText();
627     String id = idText.getText();
628     if (username.equals("") || pass.equals("") || id.equals("")) {
629         JOptionPane.showMessageDialog(null, "Please fill out all information!", "Sign Up", JOptionPane.WARNING_MESSAGE);
630     } else {
631         JOptionPane.showMessageDialog(null, "You successfully created an account", "Sign Up", JOptionPane.INFORMATION_MESSAGE);
632         account.createAccount(username, pass, id);
633         nameText.setText("");
634         passText2.setText("");
635         idText.setText("");
636         cards.show(getContentPane(), "Card 1");
637     }
638 }
639 // Task: Go back to the main
640 else if (source == cancel) {
641     nameText.setText("");
642     passText2.setText("");
643     idText.setText("");
644     String username = userText.getText();
645     String pass = passText.getText();
646     newFile.setEnabled(false);
647     openFile.setEnabled(false);
648     if (username.equals("") && pass.equals("")) {
649         cards.show(getContentPane(), "Card 1");
650     } else {
651         cards.show(getContentPane(), "Card 3");
652     }
653 }
654 // Task: Log out and go back to the main
655 else if (source == logout || source == logout2) {
656     userText.setText("");
657     passText.setText("");
658     newFile.setEnabled(false);
659     openFile.setEnabled(false);
660     signUp.setEnabled(true);
661     cards.show(getContentPane(), "Card 1");
662 }
663 // Task: View the information of the account
664 else if (source == info) {
665     String username = userText.getText();
666     String pass = passText.getText();
667     String id = account.getID(username, pass);
668     if (username.equals("") || pass.equals("") || !account.checkAccount(username, pass)) {
669         JOptionPane.showMessageDialog(null, "Please log in to see the detail", "Info", JOptionPane.WARNING_MESSAGE);
670     } else {
671         JOptionPane.showMessageDialog(null, "Username: " + username + "\nPassword: " + pass + "\nCwid: " + id,
672             "Info", JOptionPane.INFORMATION_MESSAGE);
673     }
674 }
675 // Task: Move to the content that allows to create a new Quiz
676 else if (source == newFile || source == newQuiz) {
677     cards.show(getContentPane(), "Card 4");
678 }
679 // Task: Go back to the content after logging in
680 else if (source == cancel2 || source == cancel3 || source == quit) {
681     cards.show(getContentPane(), "Card 3");
682 }
683 // Task: Move to the content that allows to open a Quiz
684 else if (source == openFile || source == openQuiz) {
685     cards.show(getContentPane(), "Card 5");
686 }

```

```

687 // Task: Create a New Quiz
688 else if (source == next) {
689     String file = fileText.getText();
690     if (file.equals("")) {
691         JOptionPane.showMessageDialog(null, "Please type a file name", "Create", JOptionPane.WARNING_MESSAGE);
692     } else {
693         if (fileQuiz.createFile(file)) {
694             cards.show(getContentPane(), numberCard[0]);
695         }
696     }
697 }
698 // Task: Move to or go back between the contents of creating questions
699 else if (source == nexts[0] || source == backs[0]) {
700     if (source == nexts[0]) {
701         String question = questionTexts[0].getText();
702         String option1 = choiceTexts[0].getText();
703         String option2 = choiceTexts[1].getText();
704         String option3 = choiceTexts[2].getText();
705         String option4 = choiceTexts[3].getText();
706         String answer = keyTexts[0].getText();
707         fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
708         cards.show(getContentPane(), numberCard[1]);
709     } else {
710         fileQuiz.deleteQuestions(0);
711         cards.show(getContentPane(), numberCard[0]);
712     }
713 } else if (source == nexts[1] || source == backs[1]) {
714     if (source == nexts[1]) {
715         String question = questionTexts[1].getText();
716         String option1 = choiceTexts[4].getText();
717         String option2 = choiceTexts[5].getText();
718         String option3 = choiceTexts[6].getText();
719         String option4 = choiceTexts[7].getText();
720         String answer = keyTexts[1].getText();
721         fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
722         cards.show(getContentPane(), numberCard[2]);
723     } else {
724         fileQuiz.deleteQuestions(1);
725         cards.show(getContentPane(), numberCard[1]);
726     }
727 } else if (source == nexts[2] || source == backs[2]) {
728     if (source == nexts[2]) {
729         String question = questionTexts[2].getText();
730         String option1 = choiceTexts[8].getText();
731         String option2 = choiceTexts[9].getText();
732         String option3 = choiceTexts[10].getText();
733         String option4 = choiceTexts[11].getText();
734         String answer = keyTexts[2].getText();
735         fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
736         cards.show(getContentPane(), numberCard[3]);
737     } else {
738         fileQuiz.deleteQuestions(2);
739         cards.show(getContentPane(), numberCard[2]);
740     }
741 } else if (source == nexts[3] || source == backs[3]) {
742     if (source == nexts[3]) {
743         String question = questionTexts[3].getText();
744         String option1 = choiceTexts[12].getText();
745         String option2 = choiceTexts[13].getText();
746         String option3 = choiceTexts[14].getText();
747         String option4 = choiceTexts[15].getText();
748         String answer = keyTexts[3].getText();
749         fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
750         cards.show(getContentPane(), numberCard[4]);
751     } else {
752         fileQuiz.deleteQuestions(3);
753         cards.show(getContentPane(), numberCard[3]);
754     }
755 }
756 // Task: Go back to the content that creates a new Quiz
757 else if (source == cancels[0] || source == cancels[1] || source == cancels[2] || source == cancels[3] || source == cancels[4]) {
758     cards.show(getContentPane(), "Card 4");
759 }

```

```

760 // Task: Complete the creation of a Quiz
761 else if (source == add) {
762     String question = questionTexts[4].getText();
763     String option1 = choiceTexts[16].getText();
764     String option2 = choiceTexts[17].getText();
765     String option3 = choiceTexts[18].getText();
766     String option4 = choiceTexts[19].getText();
767     String answer = keyTexts[4].getText();
768     fileQuiz.createQuestions(question, option1, option2, option3, option4, answer);
769     fileQuiz.createQuiz();
770     JOptionPane.showMessageDialog(null, "Created Successfully", "Create a new Quiz", JOptionPane.INFORMATION_MESSAGE);
771     cards.show(getContentPane(), "Card 3");
772 }
773 // Task: Move to the content that loads a Quiz
774 else if (source == open) {
775     String file = fileText2.getText();
776     if (file.equals("")) {
777         JOptionPane.showMessageDialog(null, "Please type a file name", "Open", JOptionPane.WARNING_MESSAGE);
778     } else if (fileQuiz.checkFile(file)) {
779         fileQuiz.loadQuiz(file);
780         String username = userText.getText();
781         heading6.setText("Welcome, " + username);
782         cards.show(getContentPane(), "Card 11");
783     } else {
784         JOptionPane.showMessageDialog(null, "Can't open since " + file + " does not exist!!!", "Error", JOptionPane.WARNING_MESSAGE);
785     }
786 }

```

```

787 // Task: Move to the content that take a Quiz
788 else if (source == start) {
789     String[] choices2 = new String[4];
790     for (int i = 0, j = 0, yPos = 0; i < 5; i++) {
791         // Create the panel for each 'loadQuestionCards'
792         loadQuestionCards[i] = new JPanel();
793         loadQuestionCards[i].setLayout(null);
794
795         // Design and build each questions after opening a Quiz
796         // by loading questions and multiple choices
797         questionNumber[i] = new JLabel();
798         questionNumber[i].setText("Question #" + (i + 1) + ":");
799         questionNumber[i].setFont(new Font("Arial", Font.BOLD, 14));
800         questionNumber[i].setBounds(5, 10, 90, 20);
801
802         titleQuestion[i] = new JLabel();
803         titleQuestion[i].setText(fileQuiz.loadQuestions.get(i).question);
804         titleQuestion[i].setFont(new Font("Arial", Font.BOLD, 14));
805         titleQuestion[i].setBounds(100, 10, 380, 20);
806
807         loadQuestionCards[i].add(questionNumber[i]);
808         loadQuestionCards[i].add(titleQuestion[i]);
809
810         groups[i] = new ButtonGroup();
811
812         choices2[0] = fileQuiz.loadQuestions.get(i).choice1;
813         choices2[1] = fileQuiz.loadQuestions.get(i).choice2;
814         choices2[2] = fileQuiz.loadQuestions.get(i).choice3;
815         choices2[3] = fileQuiz.loadQuestions.get(i).choice4;
816
817         for (int k = 0; j < (i + 1) * 4; j++, k++) {
818             multipleChoice[j].setText(choices2[k]);
819             multipleChoice[j].setFont(new Font("Arial", Font.BOLD, 15));
820             multipleChoice[j].setBounds(20, 50 + yPos, 100, 20);
821             groups[i].add(multipleChoice[j]);
822             loadQuestionCards[i].add(multipleChoice[j]);
823             yPos += 25;
824         }
825         yPos = 0;
826     }

```



```

827 // Set up the buttons for each questions
828 if (i < 4) {
829     nexts2[i].setFont(new Font("Arial", Font.BOLD, 14));
830     nexts2[i].setBounds(100, 180, 70, 30);
831     loadQuestionCards[i].add(nexts2[i]);
832 }
833 if (i > 0 && i < 5) {
834     backs2[i - 1].setFont(new Font("Arial", Font.BOLD, 14));
835     backs2[i - 1].setBounds(180, 180, 70, 30);
836     loadQuestionCards[i].add(backs2[i - 1]);
837 }
838 if (i == 4) {
839     submit.setFont(new Font("Arial", Font.BOLD, 15));
840     submit.setBounds(380, 180, 90, 30);
841     loadQuestionCards[i].add(submit);
842 }
843
844 numberCard2[i] = "Card " + (12 + i);
845 // Add each question into the application
846 add(loadQuestionCards[i], numberCard2[i]);
847 }
848 cards.show(getContentPane(), numberCard2[0]);
849 }

```



```

850 // Task: Move to and go back between each question
851 else if (source == nexts2[0] || source == backs2[0]) {
852     if (source == nexts2[0]) {
853         cards.show(getContentPane(), numberCard2[1]);
854     } else {
855         cards.show(getContentPane(), numberCard2[0]);
856     }
857 } else if (source == nexts2[1] || source == backs2[1]) {
858     if (source == nexts2[1]) {
859         cards.show(getContentPane(), numberCard2[2]);
860     } else {
861         cards.show(getContentPane(), numberCard2[1]);
862     }
863 } else if (source == nexts2[2] || source == backs2[2]) {
864     if (source == nexts2[2]) {
865         cards.show(getContentPane(), numberCard2[3]);
866     } else {
867         cards.show(getContentPane(), numberCard2[2]);
868     }
869 } else if (source == nexts2[3] || source == backs2[3]) {
870     if (source == nexts2[3]) {
871         cards.show(getContentPane(), numberCard2[4]);
872     } else {
873         cards.show(getContentPane(), numberCard2[3]);
874     }
875 }
876 // Task: Complete taking a Quiz, then move to the result
877 else if (source == submit) {
878     for (int i = 0; i < 5; i++) {
879         if (checkAnswers[i]) {
880             countCorrect += 1;
881             score += 2;
882         }
883     }
884     String username = userText.getText();
885     heading7.setText("Hi, " + username);
886     correctQuestions.setText("You got: " + countCorrect + "/5");
887     result.setText("Score: " + score + "/10");
888     cards.show(getContentPane(), "Card 17");
889 }
890 // Task: Allow users to re-take a Quiz
891 else if (source == tryAgain) {
892     // Reset the calculation
893     countCorrect = 0;
894     score = 0;
895     for (int i = 0; i < 5; i++) {
896         checkAnswers[i] = false;
897         groups[i].clearSelection();
898     }
899     cards.show(getContentPane(), numberCard2[0]);
900 }
901 }
902 }

```

**Result:**

## Launching Application



The screenshot shows a window titled "Quiz Application" with a menu bar containing "File" and "User". The main content area displays a large heading "Welcome to Quiz Application" followed by a sub-heading "----- Sign In -----". Below this, there are two input fields: "Username" and "Password". At the bottom center, there is a blue "Login" button.

## Menu



The screenshot shows the same "Quiz Application" window, but the "File" menu is open, displaying a list of options: "New", "Open", and "Close". The rest of the interface, including the "Welcome to Quiz Application" heading, "----- Sign In -----" sub-heading, "Username" and "Password" input fields, and the "Login" button, remains visible in the background.



The image shows a window titled "Quiz Application" with a menu bar containing "File" and "User". The "User" menu is open, showing "Sign Up" and "Info" options. The main content area displays "Welcome to Quiz Application" and "----- Sign In -----". Below this, there are input fields for "Username" and "Password", and a "Login" button.

Quiz Application

File User

Sign Up  
Info

Welcome to Quiz Application

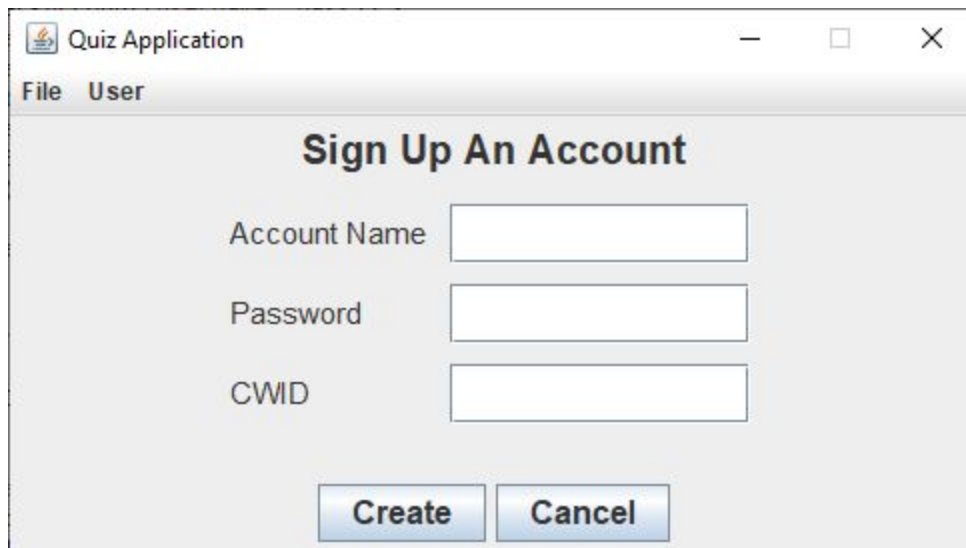
----- Sign In -----

Username

Password

Login

### Register an account



The image shows a window titled "Quiz Application" with a menu bar containing "File" and "User". The main content area displays "Sign Up An Account". Below this, there are input fields for "Account Name", "Password", and "CWID", and "Create" and "Cancel" buttons.

Quiz Application

File User

Sign Up An Account

Account Name

Password

CWID

Create Cancel

Quiz Application

File User

### Sign Up An Account

Account Name

Password


CWID

Quiz Application

File User

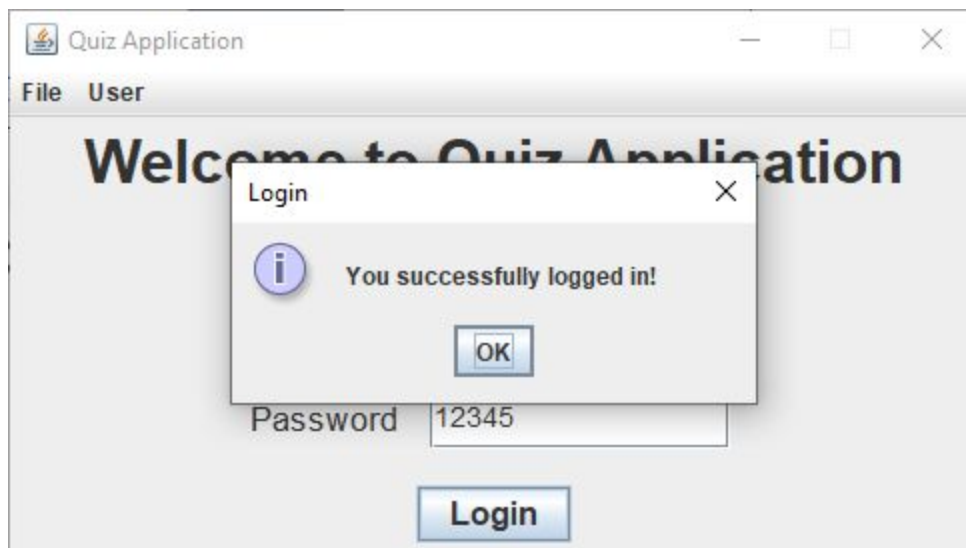
### Sign Up An Account

Sign Up

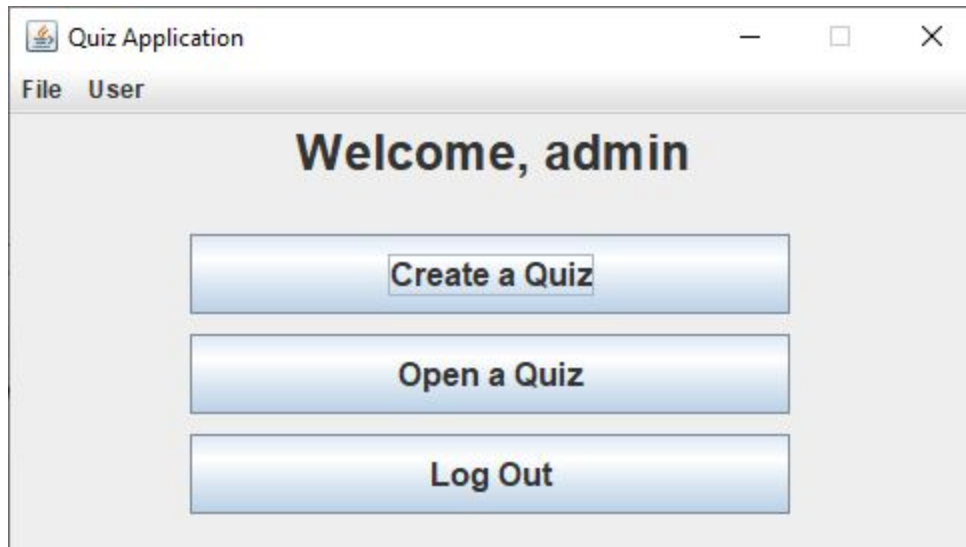
 You successfully created an account

**Logging In**

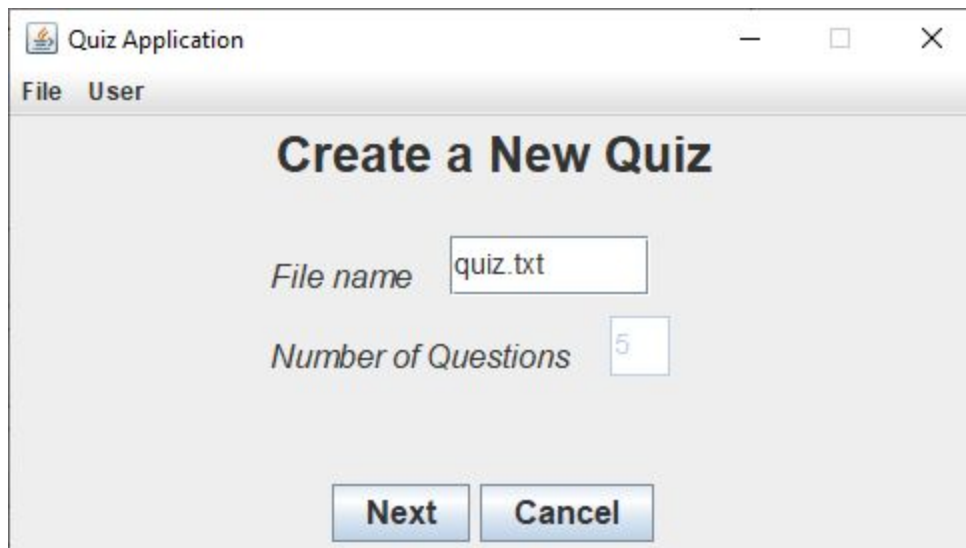




**After logging**



## Create a Quiz



Quiz Application

File User

**Question #1** What is the binary of 15?

Option 1 1111

Option 2 1011

Option 3 1110

Option 4 1111

Answer Key 1111

Next Cancel

Quiz Application

File User

**Question #2** Most class data fields are \_\_\_\_\_

Option 1 final

Option 2 static

Option 3 private

Option 4 public

Answer Key private

Next Back Cancel

Quiz Application

File User

**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

Next Back Cancel

Quiz Application

File User

**Question #4** An instance of a class is a(n) \_\_\_\_\_

Option 1

Option 2

Option 3

Option 4

Answer Key

Quiz Application

File User

**Question #5** A KeyEvent method getKeyChar() returns a(n) \_\_\_\_\_

Option 1

Option 2

Option 3

Option 4

Answer Key

Quiz Application

File User

**Question #5** A KeyEvent method getKeyChar() returns a(n) \_\_\_\_\_

Option 1


Option 2

Option 3

Option 4

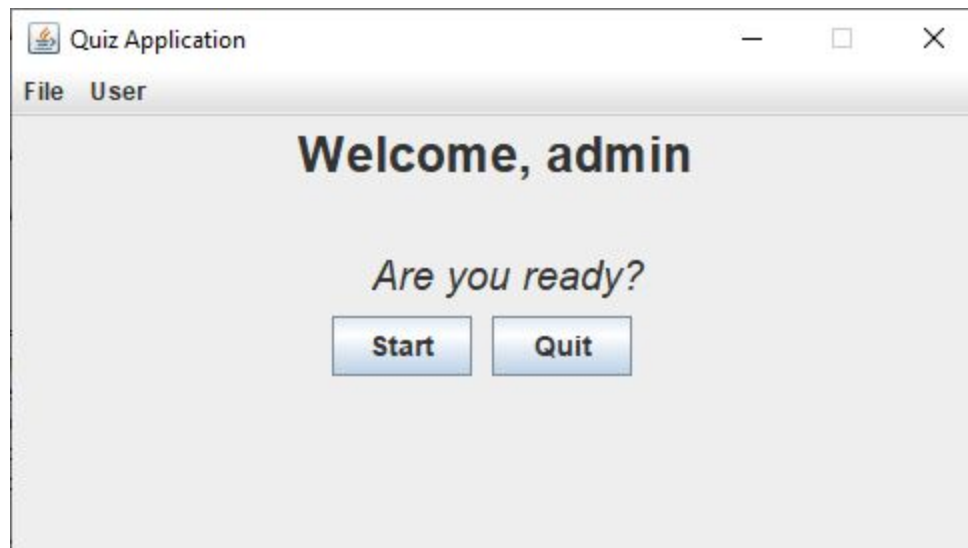
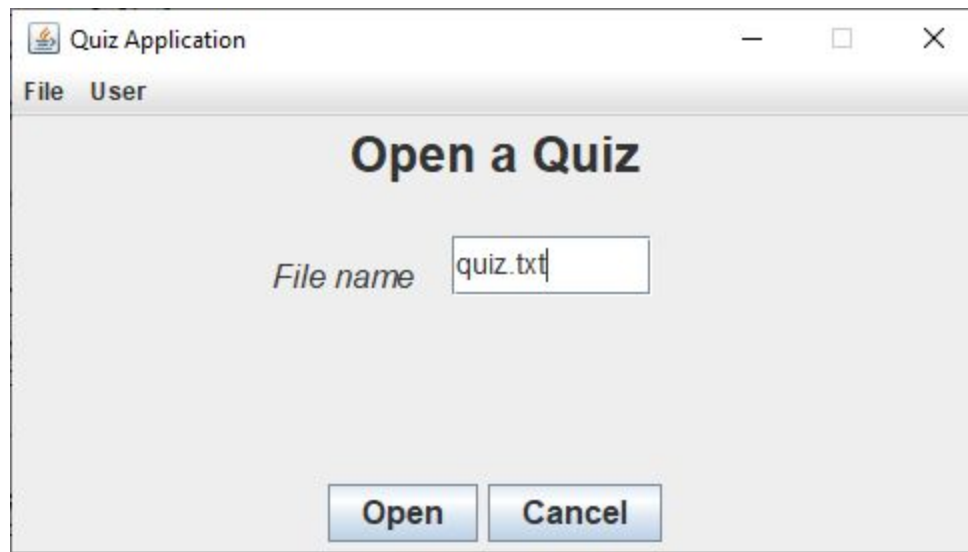
Answer Key

Create a new Quiz

 Created Successfully



## Open a Quiz



## Taking a Quiz

Quiz Application

File User

Question #1: What is the binary of 15?

☒ 1111

☐ 1011

☐ 1110

☐ 1111

Next

Quiz Application

File User

Question #2: Most class data fields are \_\_\_\_\_

☐ final

☐ static

☒ private

☐ public

Next Back

Quiz Application

File User

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

☐ bundle

☒ package

☐ packet

☐ gaggle

Next Back

Quiz Application

File User

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

☐ method

☐ procedure

☐ case

☒ object

Next Back

Quiz Application

File User

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

☒ int

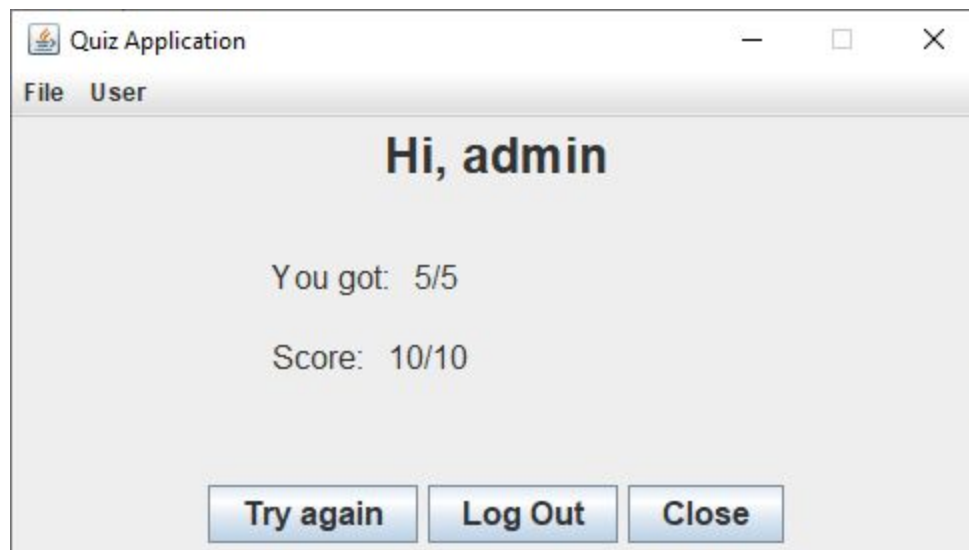
☐ char

☐ KeyEvent

☐ AWTEvent

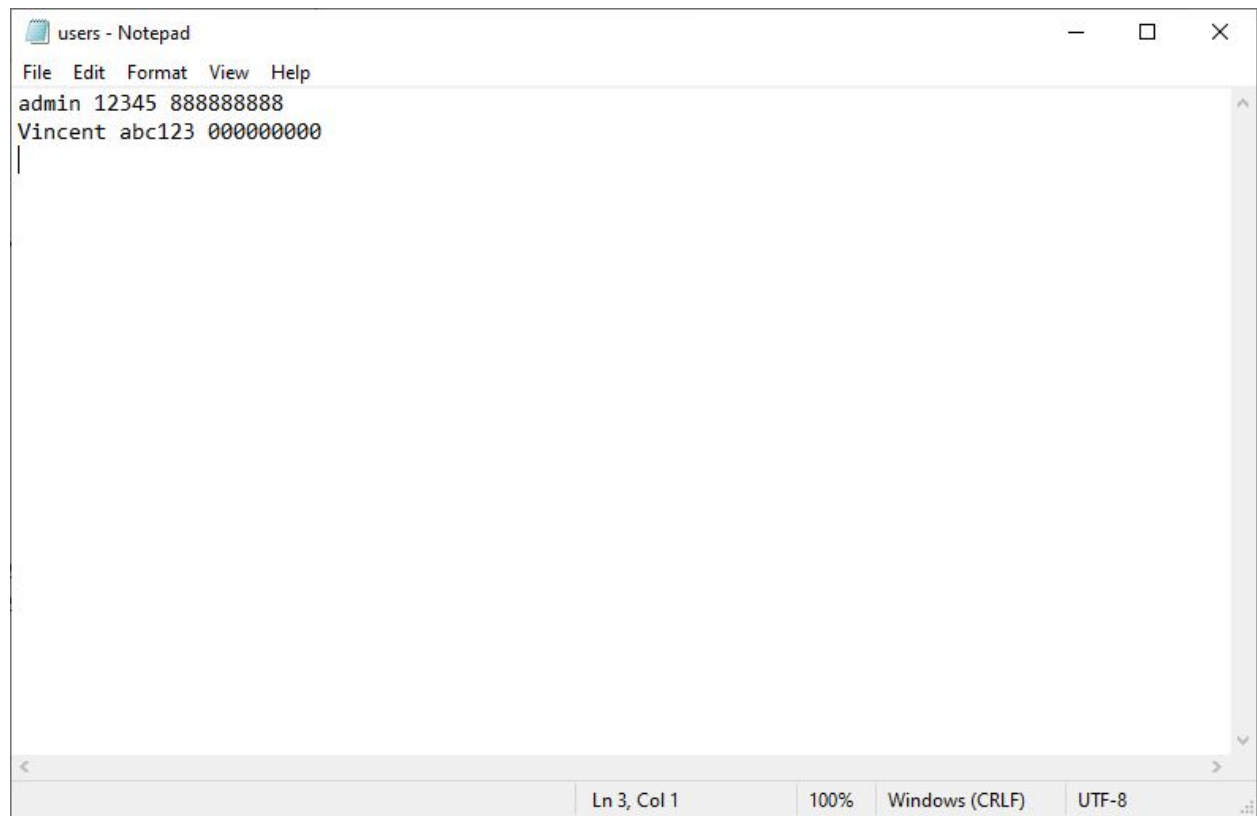
Back Submit

**After taking a Quiz**





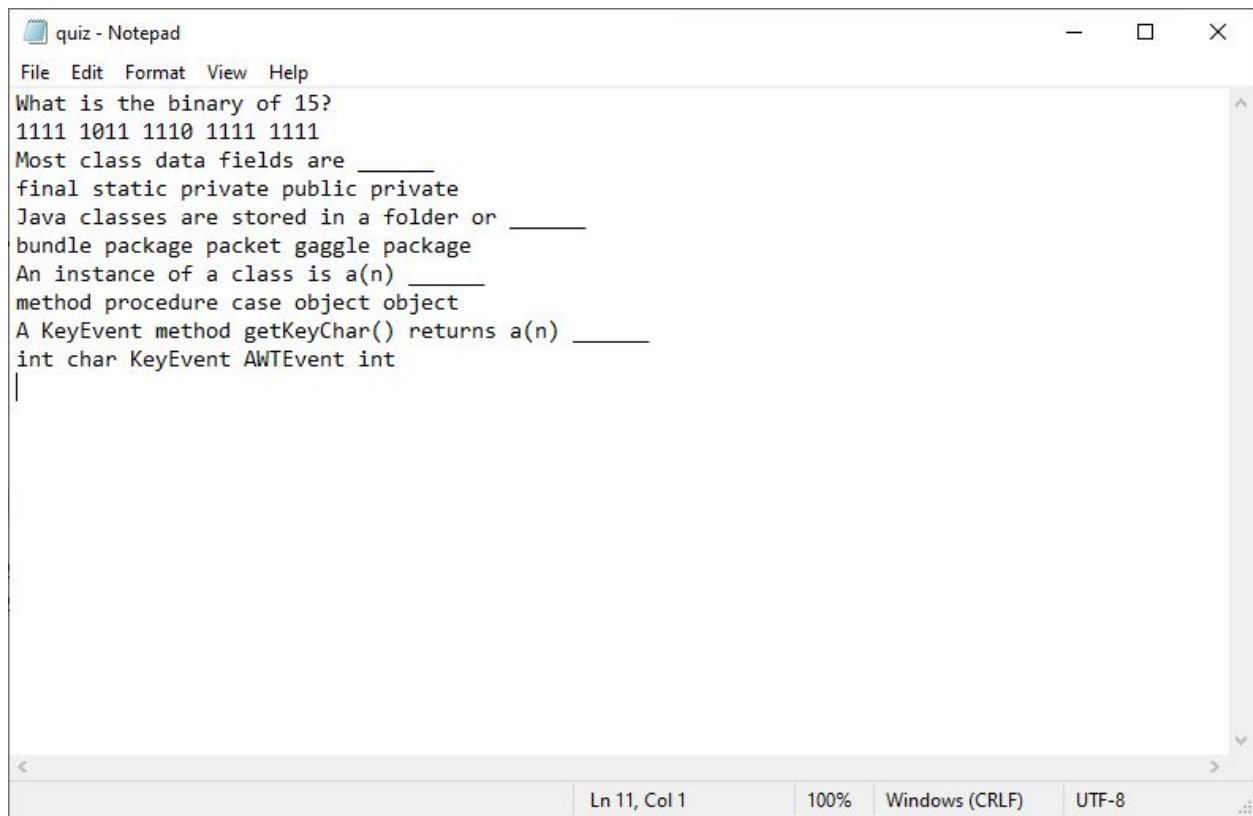
## 'users.txt'



```
users - Notepad
File Edit Format View Help
admin 12345 8888888888
Vincent abc123 0000000000
|

Ln 3, Col 1 100% Windows (CRLF) UTF-8
```

## ‘quiz.txt’



The image shows a Notepad window with the title bar 'quiz - Notepad'. The menu bar includes 'File', 'Edit', 'Format', 'View', and 'Help'. The text content of the file is as follows:

```
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 KeyEvent int
```

The status bar at the bottom indicates the cursor is at 'Ln 11, Col 1', the zoom is '100%', the encoding is 'Windows (CRLF)', and the character set is 'UTF-8'.