**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**

**ELECTRICAL & ELECTRONICS ENGINEERING**

CSE2062.1

OBJECT-ORIENTED PROGRAMMING

**PROJECT-II**

NAME STUDENT NUMBER

Orhan AĞIRMAN 150716054

Tayyip Osman AYDINER 150716014

Alpaslan TETİK 150716020

Mehmet Akif KARACA 150716029

Submitted to: **Şakir Bingöl**

Due Date

23.05.2019

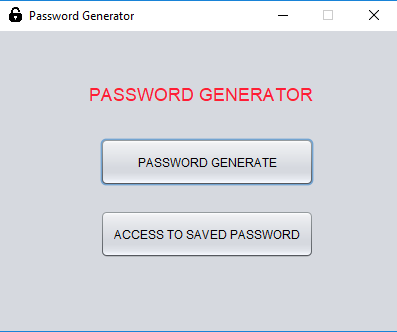
**Problem Definition**

In this Java Project, we will focus on one of the problems technology brings with our lives. In Daily life, we use various devices such as computers, tablets, mobile phones and using hundreds of programs, applications and web sites working with membership systems. Mobile bankings, Social Media, Forums are just a few of them. Finding passwords for each of them at a sufficient level of security and making our accounts safe is getting harder every day. People use the same password in many places or forget their passwords often. The main purpose of this Project is to create completely random passwords for our personal accounts and keep them under record so that our accounts are protected, and when we enter our username and password, we can see all of our passwords with their headers. In addition, we add GUI to our code and make more comfortable to use this Project.

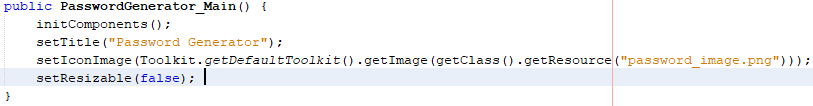
**Java Code Process**

Our code’s working process is;

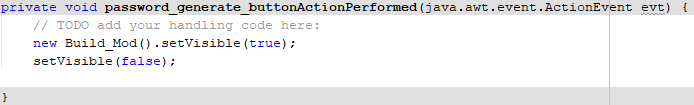
First of all, our code starting with PasswordGenerator\_Main class. When we run the code this screen opening.

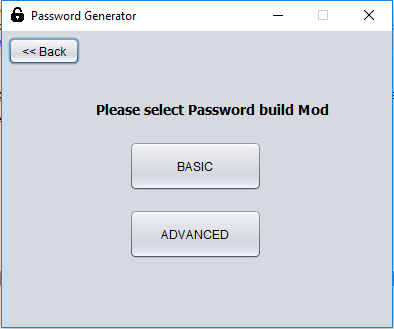


In this class we start with set title and icon image after that we make our frame’s size cant resizable.



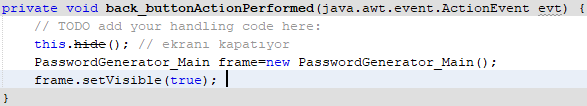
We have two buttons in the screen when we click ‘'PASSWORD GENERATE'' code direct us to Build\_Mod class, open Build\_Mod class screen, and close initial screen.



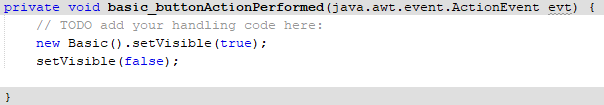


Also In this class, we start with set title and icon image after that we make our frame’s size cant resizable.

We have three buttons in this screen ‘'Back'', ‘'BASIC'', ‘'ADVANCED''. When the user clicks Back button the code close this screen and return the previous screen.



The BASIC button directs us to Basic class open, Basic class screen and close this screen.





Also In this class, we start with set title and icon image after that we make our frame’s size cant resizable.

In this screen, we have three buttons and one text field to enter the number, ‘'GENERATE'', ‘'SAVE'', ‘'BACK'' and a text field for password length.

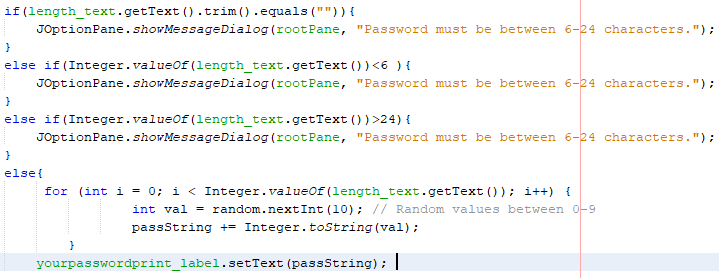
When the user clicks Back button the code close this screen and return the previous screen.

For basic password, the algorithm is simple first we create random value and null string.

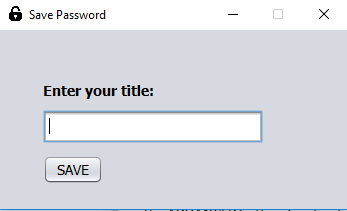


If we click ‘’GENERATE’’ button;

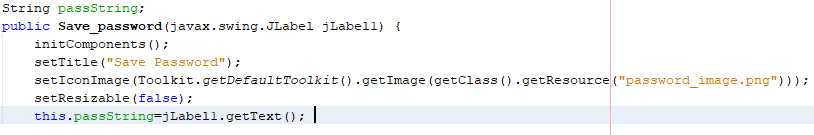
We take password length from the user. If the number isn't between 6 and 24, code show error in the dialog screen. If the user enters a valid value, Code creates a password of the desired length, consisting entirely of random numbers and shows in the label.



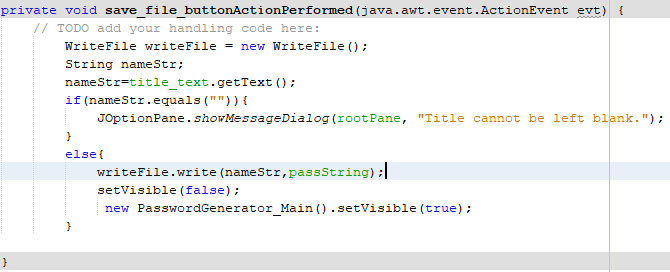
If a user wants to save the created password, click the save button and code direct us to save the class and open save screen. When we click save button before creating a password (Label is null) error shown in the dialog screen.



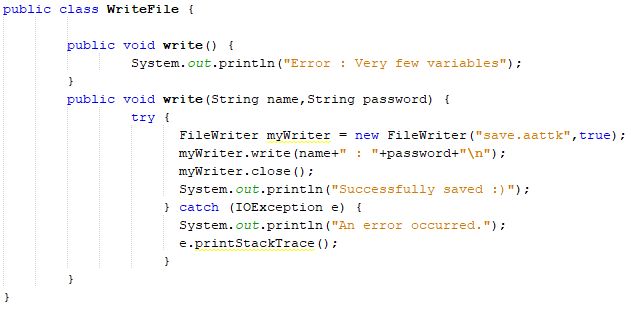
Save class will create a new string and assign the arrived password to the null string.



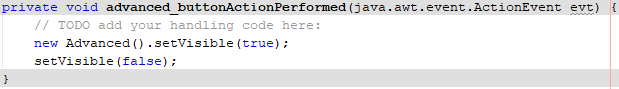
In this screen, we have one button and one text field. Save class ask us for a title if we don't enter any title shows error in the dialog screen. After the user enters the title of the password the code send title and password to writefile class. Password will be saved and code direct us to initial screen.



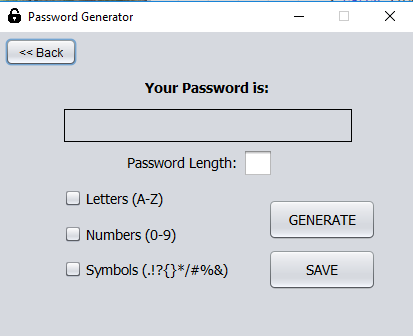
There is the writing password to file algorithm.



The ADVANCED button directs us to Advanced class, open Advanced class screen and close this screen.



The screen of the Advanced class is



Also In this class, we start with set title and icon image after that we make our frame’s size cant resizable.

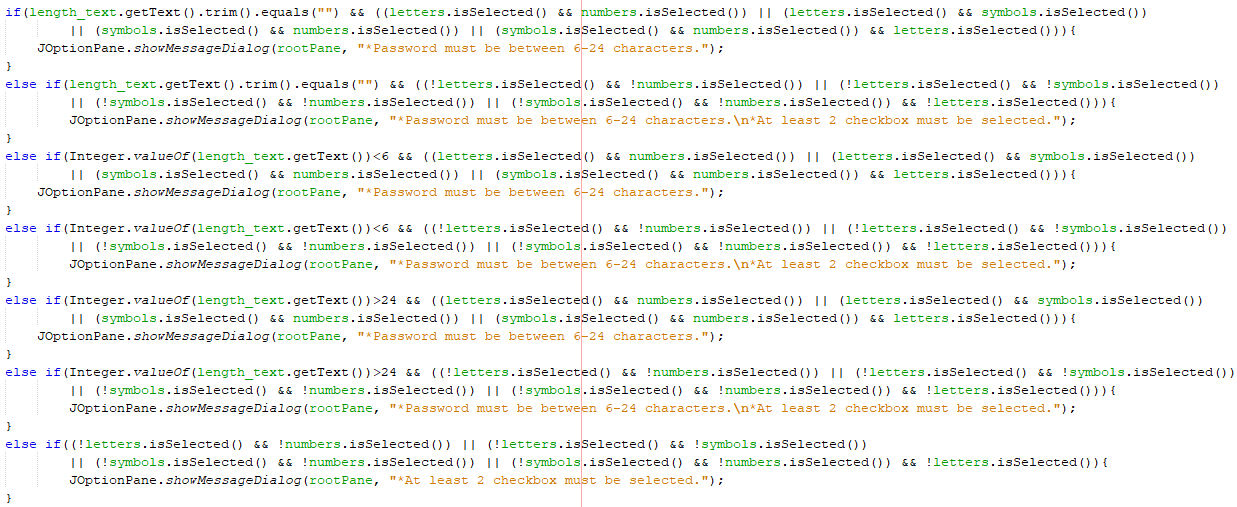
In this screen, we have three buttons, one text field, three checkboxes.

When the user clicks Back button the code close this screen and return the previous screen.

Advanced password is more complicated than a basic one because this time user can use three types of character for there password. Letters, numbers, and symbols. In the screen user sign checkbox to decide that. After sign and enter the length of the password when the user clicks the generate button. The advanced class creates random value and a null string.



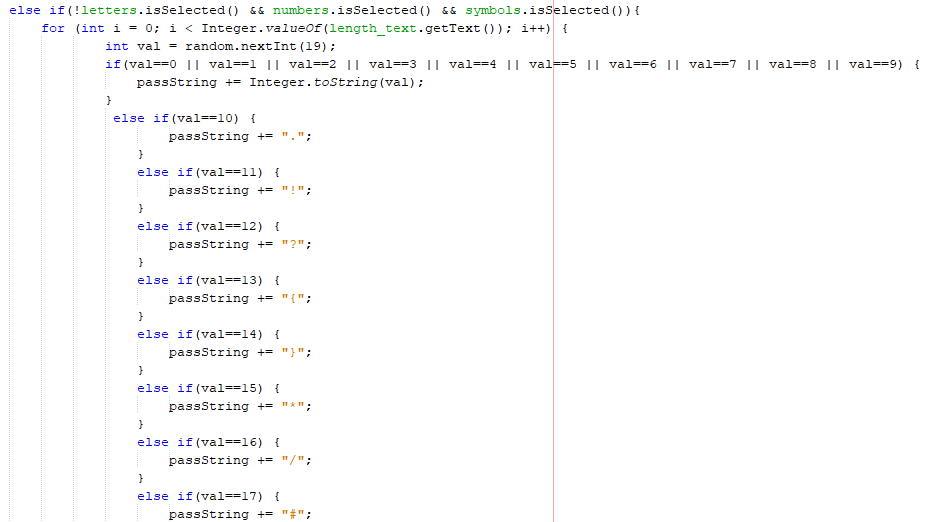
After that code look for every possible way to create advanced code. If at least two character types don't sign or length of the password isn't between 6-24 code shows error in dialog screen.

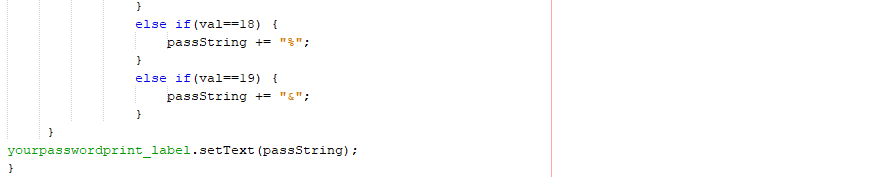


If at least two checkboxes signed and length of the password valid, Code create a password with given selections.

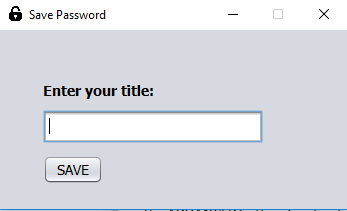
The algorithm of advanced password creation is creating a random variable from 0 to 19(not constant) and give all variables different characters and sort them to complete the string of password. At the end shows in the label.

Here is the example of Advanced password creation when password includes symbols and numbers.

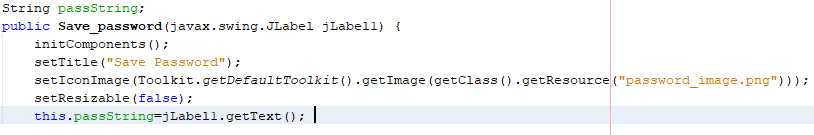




If a user wants to save the created password, click the save button and code direct us to save the class and open save screen. When we click save button before creating a password (Label is null) error shown in the dialog screen.

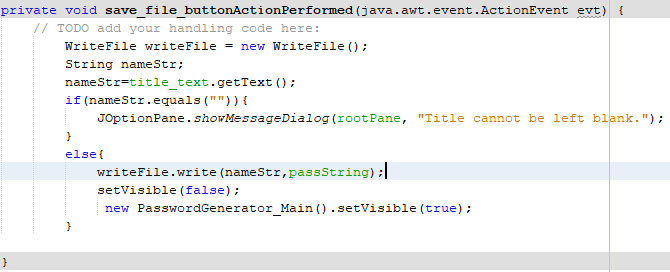


Also In this class, we start with set title and icon image after that we make our frame’s size cant resizable.

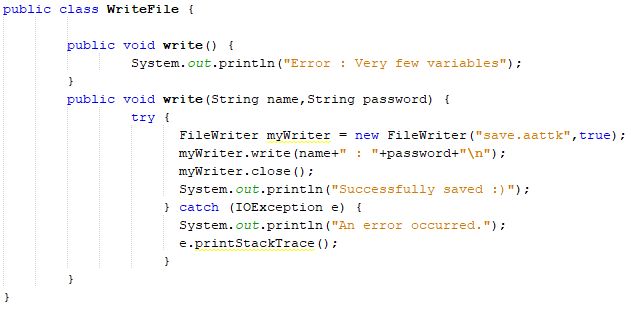


Save class will create a new string and assign the arrived password to the null string.

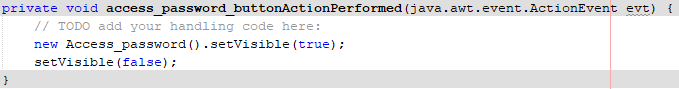
In this screen, we have one button and one text field. Save class ask us for a title if we don't enter any title shows error in the dialog screen. After the user enters the title of the password the code send title and password to writefile class. Password will be saved and code direct us to initial screen.

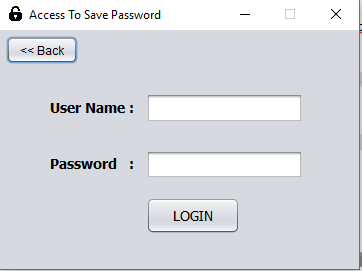


There is the writing password to file algorithm.



when we click ‘’ACCESS TO SAVED PASSWORD’’ code direct us to Access\_password class, open Access screen and close initial screen.





In this screen we have two text field, ‘'LOGIN'' and ‘'Back'' button. The user enters the User Name and Password to access saved passwords with titles.

Also In this class, we start with set title and icon image after that we make our frame’s size cant resizable.

When the user clicks Back button the code close this screen and return the previous screen. After the user enters password and user, name code compares it with a real password and user name in the file.

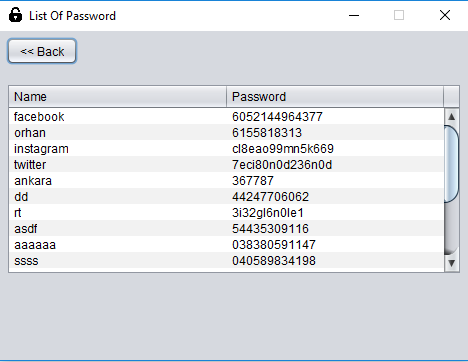


If Id and password are true code direct us to read past class, in this class algorithm read all variables and save as a strArray after that send this array to list\_password class.

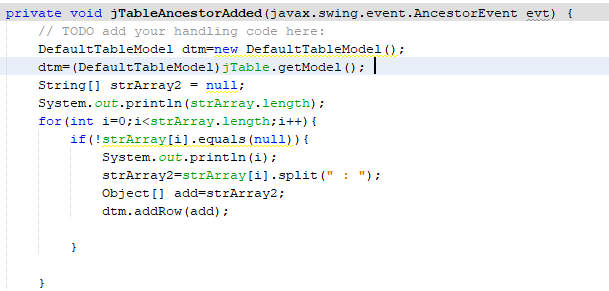


List\_password class shows us all passwords and titles.

Also In this class, we start with set title and icon image after that we make our frame’s size cant resizable.



The algorithm of listing passwords and titles is



The code takes strArray that sent from Readpass class and write every variable in the table. If the user click the back button, the code shows initial screen.

**JAVA CODE**

**Access\_Password Class**

1 package proje;

2

3 import java.awt.Toolkit;

4 import javax.swing.JOptionPane;

5

6

7 public class **Access\_password** extends javax.swing.JFrame {

8

9

10 public **Access\_password**() {

11 initComponents();

12 setTitle("Access To Save Password");

13 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

14 setResizable(false);

15 }

96 private void **back\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

97 // TODO add your handling code here:

98 this.hide();

99 PasswordGenerator\_Main frame=new PasswordGenerator\_Main();

100 frame.setVisible(true);

101 }

102

103 private void **login\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

104 // TODO add your handling code here:

105 ReadPass readPass =new ReadPass();

106 get\_password password = new get\_password() {};

107 password.read();

108 if(password.getName().equals(username\_text.getText())){

109 if(password.getPass().equals(password\_text.getText())){

110 this.hide();

111 readPass.read(); //

112 }

113 else{

114 JOptionPane.*showMessageDialog*(rootPane, "User Name and/or Password is in correct.");

115 }

116

117 }

118 else{

119 JOptionPane.*showMessageDialog*(rootPane, "User Name and/or Password is in correct.");

120 }

121

122 }

123

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

125 java.awt.EventQueue.*invokeLater*(new Runnable() {

126 public void **run**() {

127 new Access\_password().setVisible(true);

128 }

129 });

130 }

131

132 // Variables declaration - do not modify

133 private javax.swing.JButton back\_button;

134 private javax.swing.JButton login\_button;

135 private javax.swing.JLabel password\_label;

136 private javax.swing.JPasswordField password\_text;

137 private javax.swing.JLabel username\_label;

138 private javax.swing.JTextField username\_text;

139 // End of variables declaration

140 }

**Advanced Class**

1 package proje;

2 import com.sun.glass.events.KeyEvent;

3 import java.awt.Toolkit;

4 import java.util.Random;

5 import javax.swing.JOptionPane;

6

7 public class **Advanced** extends javax.swing.JFrame {

8

9 public **Advanced**() {

10 initComponents();

11 setTitle("Password Generator");

12 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

13 setResizable(false);

14 }

161 private void **length\_textKeyTyped**(java.awt.event.KeyEvent evt) {

162 // TODO add your handling code here:

163 char vchar=evt.getKeyChar();

164 if(!(Character.*isDigit*(vchar)) || (vchar==KeyEvent.*VK\_BACKSPACE*) || (vchar==KeyEvent.*VK\_DELETE*)){

165 evt.consume();

166 }

167 }

168

169 private void **length\_textActionPerformed**(java.awt.event.ActionEvent evt) {

170 // TODO add your handling code here:

171 }

172

173 private void **back\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

174 // TODO add your handling code here:

175 this.hide();

176 Build\_Mod frame=new Build\_Mod();

177 frame.setVisible(true);

178 }

179

180 private void **generate\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

181 // TODO add your handling code here:

182 Random random = new Random();

183 String passString = "";

184 if(length\_text.getText().trim().equals("") && ((letters.isSelected() && numbers.isSelected()) || (letters.isSelected() && symbols.isSelected())

185 || (symbols.isSelected() && numbers.isSelected()) || (symbols.isSelected() && numbers.isSelected()) && letters.isSelected())){

186 JOptionPane.*showMessageDialog*(rootPane, "\*Password must be between 6-24 characters.");

187 }

188 else if(length\_text.getText().trim().equals("") && ((!letters.isSelected() && !numbers.isSelected()) || (!letters.isSelected() && !symbols.isSelected())

189 || (!symbols.isSelected() && !numbers.isSelected()) || (!symbols.isSelected() && !numbers.isSelected()) && !letters.isSelected())){

190 JOptionPane.*showMessageDialog*(rootPane, "\*Password must be between 6-24 characters.**\n**\*At least 2 checkbox must be selected.");

191 }

192 else if(Integer.*valueOf*(length\_text.getText())<6 && ((letters.isSelected() && numbers.isSelected()) || (letters.isSelected() && symbols.isSelected())

193 || (symbols.isSelected() && numbers.isSelected()) || (symbols.isSelected() && numbers.isSelected()) && letters.isSelected())){

194 JOptionPane.*showMessageDialog*(rootPane, "\*Password must be between 6-24 characters.");

195 }

196 else if(Integer.*valueOf*(length\_text.getText())<6 && ((!letters.isSelected() && !numbers.isSelected()) || (!letters.isSelected() && !symbols.isSelected())

197 || (!symbols.isSelected() && !numbers.isSelected()) || (!symbols.isSelected() && !numbers.isSelected()) && !letters.isSelected())){

198 JOptionPane.*showMessageDialog*(rootPane, "\*Password must be between 6-24 characters.**\n**\*At least 2 checkbox must be selected.");

199 }

200 else if(Integer.*valueOf*(length\_text.getText())>24 && ((letters.isSelected() && numbers.isSelected()) || (letters.isSelected() && symbols.isSelected())

201 || (symbols.isSelected() && numbers.isSelected()) || (symbols.isSelected() && numbers.isSelected()) && letters.isSelected())){

202 JOptionPane.*showMessageDialog*(rootPane, "\*Password must be between 6-24 characters.");

203 }

204 else if(Integer.*valueOf*(length\_text.getText())>24 && ((!letters.isSelected() && !numbers.isSelected()) || (!letters.isSelected() && !symbols.isSelected())

205 || (!symbols.isSelected() && !numbers.isSelected()) || (!symbols.isSelected() && !numbers.isSelected()) && !letters.isSelected())){

206 JOptionPane.*showMessageDialog*(rootPane, "\*Password must be between 6-24 characters.**\n**\*At least 2 checkbox must be selected.");

207 }

208 else if((!letters.isSelected() && !numbers.isSelected()) || (!letters.isSelected() && !symbols.isSelected())

209 || (!symbols.isSelected() && !numbers.isSelected()) || (!symbols.isSelected() && !numbers.isSelected()) && !letters.isSelected()){

210 JOptionPane.*showMessageDialog*(rootPane, "\*At least 2 checkbox must be selected.");

211 }

212 else{

213

214 if(letters.isSelected() && numbers.isSelected() && symbols.isSelected()){

215 for (int i = 0; i < Integer.*valueOf*(length\_text.getText()); i++) {

216 int val = random.nextInt(35);

217 if(val==0 || val==1 || val==2 || val==3 || val==4 || val==5 || val==6 || val==7 || val==8 || val==9) {

218 passString += Integer.*toString*(val);

219 }

220 if(val==10) {

221 passString += "a";

222 }

223 else if(val==11) {

224 passString += "b";

225 }

226 else if(val==12) {

227 passString += "c";

228 }

229 else if(val==13) {

230 passString += "d";

231 }

232 else if(val==14) {

233 passString += "e";

234 }

235 else if(val==15) {

236 passString += "f";

237 }

238 else if(val==16) {

239 passString += "g";

240 }

241 else if(val==17) {

242 passString += "h";

243 }

244 else if(val==18) {

245 passString += "i";

246 }

247 else if(val==19) {

248 passString += "j";

249 }

250 else if(val==20) {

251 passString += "k";

252 }

253 else if(val==21) {

254 passString += "l";

255 }

256 else if(val==22) {

257 passString += "m";

258 }

259 else if(val==23) {

260 passString += "n";

261 }

262 else if(val==24) {

263 passString += "o";

264 }

265 else if(val==25) {

266 passString += "p";

267 }

268 else if(val==26) {

269 passString += ".";

270 }

271 else if(val==27) {

272 passString += "!";

273 }

274 else if(val==28) {

275 passString += "?";

276 }

277 else if(val==29) {

278 passString += "{";

279 }

280 else if(val==30) {

281 passString += "}";

282 }

283 else if(val==31) {

284 passString += "\*";

285 }

286 else if(val==32) {

287 passString += "/";

288 }

289 else if(val==33) {

290 passString += "#";

291 }

292 else if(val==34) {

293 passString += "%";

294 }

295 else if(val==35) {

296 passString += "&";

297 }

298 }

299 yourpasswordprint\_label.setText(passString);

300 }

301 else if(letters.isSelected() && numbers.isSelected() && !symbols.isSelected()){

302 for (int i = 0; i < Integer.*valueOf*(length\_text.getText()); i++) {

303 int val = random.nextInt(25);

304 if(val==0 || val==1 || val==2 || val==3 || val==4 || val==5 || val==6 || val==7 || val==8 || val==9) {

305 passString += Integer.*toString*(val);

306 }

307

308 if(val==10) {

309 passString += "a";

310 }

311 else if(val==11) {

312 passString += "b";

313 }

314 else if(val==12) {

315 passString += "c";

316 }

317 else if(val==13) {

318 passString += "d";

319 }

320 else if(val==14) {

321 passString += "e";

322 }

323 else if(val==15) {

324 passString += "f";

325 }

326 else if(val==16) {

327 passString += "g";

328 }

329 else if(val==17) {

330 passString += "h";

331 }

332 else if(val==18) {

333 passString += "i";

334 }

335 else if(val==19) {

336 passString += "j";

337 }

338 else if(val==20) {

339 passString += "k";

340 }

341 else if(val==21) {

342 passString += "l";

343 }

344 else if(val==22) {

345 passString += "m";

346 }

347 else if(val==23) {

348 passString += "n";

349 }

350 else if(val==24) {

351 passString += "o";

352 }

353 else if(val==25) {

354 passString += "p";

355 }

356 }

357 yourpasswordprint\_label.setText(passString);

358 }

359 else if(!letters.isSelected() && numbers.isSelected() && symbols.isSelected()){

360 for (int i = 0; i < Integer.*valueOf*(length\_text.getText()); i++) {

361 int val = random.nextInt(19);

362 if(val==0 || val==1 || val==2 || val==3 || val==4 || val==5 || val==6 || val==7 || val==8 || val==9) {

363 passString += Integer.*toString*(val);

364 }

365 else if(val==10) {

366 passString += ".";

367 }

368 else if(val==11) {

369 passString += "!";

370 }

371 else if(val==12) {

372 passString += "?";

373 }

374 else if(val==13) {

375 passString += "{";

376 }

377 else if(val==14) {

378 passString += "}";

379 }

380 else if(val==15) {

381 passString += "\*";

382 }

383 else if(val==16) {

384 passString += "/";

385 }

386 else if(val==17) {

387 passString += "#";

388 }

389 else if(val==18) {

390 passString += "%";

391 }

392 else if(val==19) {

393 passString += "&";

394 }

395 }

396 yourpasswordprint\_label.setText(passString);

397 }

398 else if(letters.isSelected() && !numbers.isSelected() && symbols.isSelected()){

399 for (int i = 0; i < Integer.*valueOf*(length\_text.getText()); i++) {

400 int val = random.nextInt(26);

401

402 if(val==1) {

403 passString += "a";

404 }

405 else if(val==2) {

406 passString += "b";

407 }

408 else if(val==3) {

409 passString += "c";

410 }

411 else if(val==4) {

412 passString += "d";

413 }

414 else if(val==5) {

415 passString += "e";

416 }

417 else if(val==6) {

418 passString += "f";

419 }

420 else if(val==7) {

421 passString += "g";

422 }

423 else if(val==8) {

424 passString += "h";

425 }

426 else if(val==9) {

427 passString += "i";

428 }

429 else if(val==10) {

430 passString += "j";

431 }

432 else if(val==11) {

433 passString += "k";

434 }

435 else if(val==12) {

436 passString += "l";

437 }

438 else if(val==13) {

439 passString += "m";

440 }

441 else if(val==14) {

442 passString += "n";

443 }

444 else if(val==15) {

445 passString += "o";

446 }

447 else if(val==16) {

448 passString += "p";

449 }

450 else if(val==17) {

451 passString += ".";

452 }

453 else if(val==18) {

454 passString += "!";

455 }

456 else if(val==19) {

457 passString += "?";

458 }

459 else if(val==20) {

460 passString += "{";

461 }

462 else if(val==21) {

463 passString += "}";

464 }

465 else if(val==22) {

466 passString += "\*";

467 }

468 else if(val==23) {

469 passString += "/";

470 }

471 else if(val==24) {

472 passString += "#";

473 }

474 else if(val==25) {

475 passString += "%";

476 }

477 else if(val==26) {

478 passString += "&";

479 }

480 }

481 yourpasswordprint\_label.setText(passString);

482 }

483 }

484

485

486 }

487

488 private void **save\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

489 // TODO add your handling code here:

490 if(yourpasswordprint\_label.getText().equals("")){

491 JOptionPane.*showMessageDialog*(rootPane, "No password to save.");

492 }

493 else{

494 setVisible(false);

495 new Save\_password(yourpasswordprint\_label).setVisible(true);

496 }

497 }

498

499 /\*\*

500 \* **@param** args the command line arguments

501 \*/

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

528 java.awt.EventQueue.*invokeLater*(new Runnable() {

529 public void **run**() {

530 new Advanced().setVisible(true);

531 }

532 });

533 }

534

535 // Variables declaration - do not modify

536 private javax.swing.JButton back\_button;

537 private javax.swing.JButton generate\_button;

538 private javax.swing.JTextField length\_text;

539 private javax.swing.JCheckBox letters;

540 private javax.swing.JCheckBox numbers;

541 private javax.swing.JLabel passwordlength\_label;

542 private javax.swing.JButton save\_button;

543 private javax.swing.JCheckBox symbols;

544 private javax.swing.JLabel yourpassword\_label;

545 private javax.swing.JLabel yourpasswordprint\_label;

546 // End of variables declaration

547 }

**Basic Class**

1 package proje;

2 import com.sun.glass.events.KeyEvent;

3 import java.awt.Toolkit;

4 import java.util.Random;

5 import javax.swing.JOptionPane;

6

7 public class **Basic** extends javax.swing.JFrame {

8

9 public **Basic**() {

10 initComponents();

11 setTitle("Password Generator");

12 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

13 setResizable(false);

14 }

128 private void **generate\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

129 // TODO add your handling code here:

130 Random random = new Random();

131 String passString = "";

132 if(length\_text.getText().trim().equals("")){

133 JOptionPane.*showMessageDialog*(rootPane, "Password must be between 6-24 characters.");

134 }

135 else if(Integer.*valueOf*(length\_text.getText())<6 ){

136 JOptionPane.*showMessageDialog*(rootPane, "Password must be between 6-24 characters.");

137 }

138 else if(Integer.*valueOf*(length\_text.getText())>24){

139 JOptionPane.*showMessageDialog*(rootPane, "Password must be between 6-24 characters.");

140 }

141 else{

142 for (int i = 0; i < Integer.*valueOf*(length\_text.getText()); i++) {

143 int val = random.nextInt(10); // Random values between 0-9

144 passString += Integer.*toString*(val);

145 }

146 yourpasswordprint\_label.setText(passString);

147

148 }

149

150 }

151

152 private void **save\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

153

154 // TODO add your handling code here:

155 if(yourpasswordprint\_label.getText().equals("")){

156 JOptionPane.*showMessageDialog*(rootPane, "No password to save.");

157 }

158 else{

159 setVisible(false);

160 new Save\_password(yourpasswordprint\_label).setVisible(true);

161 }

162

163

164 }

165

166 private void **length\_textKeyTyped**(java.awt.event.KeyEvent evt) {

167 // TODO add your handling code here:

168 char vchar=evt.getKeyChar();

169 if(!(Character.*isDigit*(vchar)) || (vchar==KeyEvent.*VK\_BACKSPACE*) || (vchar==KeyEvent.*VK\_DELETE*)){

170 evt.consume();

171 }

172 }

173

174 private void **back\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

175 // TODO add your handling code here:

176 this.hide();

177 Build\_Mod frame=new Build\_Mod();

178 frame.setVisible(true);

179 }

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

java.awt.EventQueue.*invokeLater*(new Runnable() {

211 public void **run**() {

212 new Basic().setVisible(true);

213 }

214 });

215 }

216

217 // Variables declaration - do not modify

218 private javax.swing.JButton back\_button;

219 private javax.swing.JButton generate\_button;

220 private javax.swing.JTextField length\_text;

221 private javax.swing.JLabel password\_lengthtext;

222 private javax.swing.JButton save\_button;

223 private javax.swing.JLabel yourpassword\_label;

224 private javax.swing.JLabel yourpasswordprint\_label;

225 // End of variables declaration

**Build\_Mod Class**

1 package proje;

2

3 import java.awt.Toolkit;

4

5 public class **Build\_Mod** extends javax.swing.JFrame {

6

7 public **Build\_Mod**() {

8 initComponents();

9 setTitle("Password Generator");

10 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

11 setResizable(false);

12 }

90 private void **basic\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

91 // TODO add your handling code here:

92 new Basic().setVisible(true);

93 setVisible(false);

94

95 }

96

97 private void **advanced\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

98 // TODO add your handling code here:

99 new Advanced().setVisible(true);

100 setVisible(false);

101 }

102

103 private void **back\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

104 // TODO add your handling code here:

105 this.hide();

106 PasswordGenerator\_Main frame=new PasswordGenerator\_Main();

107 frame.setVisible(true);

108 }

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

139 java.awt.EventQueue.*invokeLater*(new Runnable() {

140 public void **run**() {

141 new Build\_Mod().setVisible(true);

142

143 }

144 });

145 }

146

147 // Variables declaration - do not modify

148 private javax.swing.JButton advanced\_button;

149 private javax.swing.JButton back\_button;

150 private javax.swing.JButton basic\_button;

151 private javax.swing.JLabel buildmod\_label;

152 // End of variables declaration

**PasswordGenerator\_Main Class**

1 package proje;

2

3 import java.awt.Toolkit;

4

5 public class **PasswordGenerator\_Main** extends javax.swing.JFrame {

6

7 public **PasswordGenerator\_Main**() {

8 initComponents();

9 setTitle("Password Generator");

10 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

11 setResizable(false);

12 }

76 private void **password\_generate\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

77 // TODO add your handling code here:

78 new Build\_Mod().setVisible(true);

79 setVisible(false);

80

81 }

82

83 private void **access\_password\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

84 // TODO add your handling code here:

85 new Access\_password().setVisible(true);

86 setVisible(false);

87 }

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

118 java.awt.EventQueue.*invokeLater*(new Runnable() {

119 public void **run**() {

120 new PasswordGenerator\_Main().setVisible(true);

121 }

122 });

123 }

124

125 // Variables declaration - do not modify

126 private javax.swing.JButton access\_password\_button;

127 private javax.swing.JButton password\_generate\_button;

128 private javax.swing.JLabel passwordgenerator\_label;

129 // End of variables declaration

**ReadPass Class**

1 package proje;

2 import java.io.File;

3 import java.io.FileNotFoundException;

4 import java.util.Scanner;

5

6 public class **ReadPass** extends javax.swing.JFrame{

7

8 String data;

9 String[] strArray= new String[100];

10

11 int i=0;

12 get\_password password = new get\_password() {};

13 public void **read**() {

14 try {

15 File myObj = new File("save.aattk");

16 Scanner myReader = new Scanner(myObj);

17 while (myReader.hasNextLine()) {

18

19 data=myReader.nextLine();

20 strArray[i]=data;

21 i++;

22

23 }

24

25

26 new list\_password(strArray).setVisible(true);

27

28 myReader.close();

29 } catch (FileNotFoundException e) {

30 System.*out*.println("An error occurred.");

31 e.printStackTrace();

32 }

33 }

34

35

36

37 }

**Save\_password Class**

1 package proje;

2

3 import java.awt.Toolkit;

4 import javax.swing.JOptionPane;

5

6 public class **Save\_password** extends javax.swing.JFrame{

7

8 String passString;

9 public **Save\_password**(javax.swing.JLabel jLabel1) {

10 initComponents();

11 setTitle("Save Password");

12 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

13 setResizable(false);

14 this.passString=jLabel1.getText();

79 private void **title\_textActionPerformed**(java.awt.event.ActionEvent evt) {

80 // TODO add your handling code here:

81 }

82

83 private void **save\_file\_buttonActionPerformed**(java.awt.event.ActionEvent evt) {

84 // TODO add your handling code here:

85 WriteFile writeFile = new WriteFile();

86 String nameStr;

87 nameStr=title\_text.getText();

88 if(nameStr.equals("")){

89 JOptionPane.*showMessageDialog*(rootPane, "Title cannot be left blank.");

90 }

91 else{

92 writeFile.write(nameStr,passString);

93 setVisible(false);

94 new PasswordGenerator\_Main().setVisible(true);

95 }

96

97 }

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

128 java.awt.EventQueue.*invokeLater*(new Runnable() {

129 public void **run**() {

130

131 }

132 });

133 }

134

135 // Variables declaration - do not modify

136 private javax.swing.JButton save\_file\_button;

137 private javax.swing.JLabel title\_label;

138 private javax.swing.JTextField title\_text;

139 // End of variables declaration

140 }

**WriteFile Class**

1 package proje;

2 import java.io.FileWriter;

3 import java.io.IOException;

4

5 public class **WriteFile** {

6

7 public void **write**() {

8 System.*out*.println("Error : Very few variables");

9 }

10 public void **write**(String name,String password) {

11 try {

12 FileWriter myWriter = new FileWriter("save.aattk",true);

13 myWriter.write(name+" : "+password+"**\n**");

14 myWriter.close();

15 System.*out*.println("Successfully saved :)");

16 } catch (IOException e) {

17 System.*out*.println("An error occurred.");

18 e.printStackTrace();

19 }

20 }

21 }

**get\_password Class**

1 package proje;

2 import java.io.File;

3 import java.io.FileNotFoundException;

4 import java.util.Scanner;

5

6 public abstract class **get\_password** {

7

8

9 private String pass;

10 private String name;

11 private String data;

12 public String **getName**() {

13 return name;

14 }

15 public void **setName**(String name) {

16 this.name = name;

17 }

18 public String **getPass**() {

19 return pass;

20 }

21 public void **setPass**(String pass) {

22 this.pass = pass;

23 }

24 public void **read**() {

25 try {

26 File myObj = new File("pass.aattk");

27 Scanner myReader = new Scanner(myObj);

28 data = myReader.nextLine();

29 setName(data);

30 data = myReader.nextLine();

31 setPass(data);

32 myReader.close();

33 } catch (FileNotFoundException e) {

34 System.*out*.println("password senc error.");

35 e.printStackTrace();

36 }

37 }

**list\_password Class**

1 package proje;

2 import java.awt.Toolkit;

3 import javax.swing.table.DefaultTableModel;

4 import java.util.Arrays;

5 import java.util.Scanner;

6

7 public class **list\_password** extends javax.swing.JFrame {

8

9 public String[] strArray;

10 public **list\_password**(String[] strArray ) {

11 initComponents();

12 setTitle("List Of Password");

13 setIconImage(Toolkit.*getDefaultToolkit*().getImage(getClass().getResource("password\_image.png")));

14 setResizable(false);

15 this.strArray=strArray;

16 }

94 private void **jTableAncestorAdded**(javax.swing.event.AncestorEvent evt) {

95 // TODO add your handling code here:

96 DefaultTableModel dtm=new DefaultTableModel();

97 dtm=(DefaultTableModel)jTable.getModel();

98 String[] strArray2 = null;

99 System.*out*.println(strArray.length);

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

101 if(!strArray[i].equals(null)){

102 System.*out*.println(i);

103 strArray2=strArray[i].split(" : ");

104 Object[] add=strArray2;

105 dtm.addRow(add);

106

107 }

108

109 }

116 private void **jButton1ActionPerformed**(java.awt.event.ActionEvent evt) {

117 // TODO add your handling code here:

118 this.hide();

119 PasswordGenerator\_Main frame=new PasswordGenerator\_Main();

120 frame.setVisible(true);

121 }

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

151 java.awt.EventQueue.*invokeLater*(new Runnable() {

152 public void **run**() {

153 //new list\_password().setVisible(true);

154 }

155 });

156 }

157

158 // Variables declaration - do not modify

159 private javax.swing.JButton jButton1;

160 private javax.swing.JScrollBar jScrollBar1;

161 private javax.swing.JScrollPane jScrollPane2;

162 private javax.swing.JTable jTable;

163 // End of variables declaration

164 }