```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.concurrent.TimeUnit;
import javax.swing.Timer;

public class Main extends JFrame implements ActionListener, KeyListener, MouseListener{
    public static Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
    public static int FRAME_HEIGHT = (dim.height*9)/10;
    public static int FRAME_WIDTH = (dim.width*9)/10;
    public static Main frame = new Main();
    public static Menu menuInterface = new Menu();
    public static DailySchedule dailyScheduleInterface = new DailySchedule();
    public static WeeklySchedule weeklyScheduleInterface = new WeeklySchedule();
    public static CardLayout cl = new CardLayout();
    public static Timer timer;
    public static JPanel panelContainer;
    public static int JframeX, JframeY;
    private static Point mousePos;

    public Main() {
        super("Workout Planner");
        timer = new Timer(50, this);
        timer.start();
        addKeyListener(this);
        addMouseListener(this);
    }

    public static void main(String[] args) {
        ImageIcon frameIcon = new ImageIcon("grab.jpg");

        //sets up JFrame
        frame.setSize(FRAME_WIDTH,FRAME_HEIGHT);
        frame.setIconImage(frameIcon.getImage());
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.setResizable(false);

        //Add jpanels to Jframe
        panelContainer = new JPanel(new CardLayout());
        panelContainer.add(menuInterface, "Menu Panel");
        panelContainer.add(dailyScheduleInterface, "Daily Schedule Panel");
        panelContainer.add(weeklyScheduleInterface, "Weekly Schedule Panel");

        frame.add(panelContainer);

        cl = (CardLayout)panelContainer.getLayout();
        cl.show(panelContainer, "Menu Panel");

        //Sets position of Jframe
        JframeX = (dim.width-FRAME_WIDTH)/2;
        JframeY = (dim.height-FRAME_HEIGHT)/2;
        frame.setLocation(JframeX, JframeY);

        menuInterface.run();
    }

    public void keyTyped(KeyEvent e) {
    }

    public void keyPressed(KeyEvent e) {
    }

    public void keyReleased(KeyEvent e) {
```

```
65      }
66
67      public void mouseClicked(MouseEvent arg0) {
68
69      }
70
71      public void mouseExited(MouseEvent arg0) {
72
73      }
74
75      public void mouseEntered(MouseEvent arg0) {
76
77      }
78
79      public void mousePressed(MouseEvent arg0) {
80      }
81
82      public void mouseReleased(MouseEvent arg0) {
83      }
84
85      public void paint(Graphics g) {
86          super.paint(g);
87          frame.setLocation(JframeX, JframeY);
88      }
89
90      public void actionPerformed(ActionEvent ev){
91          if(menuInterface.loginSuccess){
92              cl.show(panelContainer, "Weekly Schedule Panel");
93              weeklyScheduleInterface.requestFocus();
94              weeklyScheduleInterface.setFocusable(true);
95              menuInterface.loginSuccess = false;
96              weeklyScheduleInterface.user = menuInterface.username;
97              dailyScheduleInterface.user = menuInterface.username;
98              weeklyScheduleInterface.run();
99          }
100
101         if(!dailyScheduleInterface.selected){
102             cl.show(panelContainer, "Weekly Schedule Panel");
103             weeklyScheduleInterface.requestFocus();
104             weeklyScheduleInterface.setFocusable(true);
105             dailyScheduleInterface.selected = true;
106             weeklyScheduleInterface.run();
107         }
108         if(!weeklyScheduleInterface.selected){
109             cl.show(panelContainer, "Daily Schedule Panel");
110             dailyScheduleInterface.requestFocus();
111             dailyScheduleInterface.setFocusable(true);
112             weeklyScheduleInterface.selected = true;
113             dailyScheduleInterface.selectedDay = weeklyScheduleInterface.selectedDay;
114             dailyScheduleInterface.run();
115         }
116         repaint();
117     }
118 }
```

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.Timer;
import java.io.*;

public class Menu extends JPanel implements ActionListener{
    public static Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
    public static int PANEL_HEIGHT = (dim.height*9)/10;
    public static int PANEL_WIDTH = (dim.width*9)/10;
    public static Timer timer;
    public static JButton buttons[] = new JButton[2], back;
    public static JTextField accDetails[] = new JTextField[4], user, pass;
    public static boolean login = false, signUp = false, accountFound = false,
    loginAttempt = false, signUpAttempt = false, loginSuccess = false,
    newUserRegistered = false, dataRegistered = false, enterAccDetails = false,
    homeScreen = true;
    public static Account accounts[] = new Account[100];
    public static int accountCounter = 0, accountNum = 0, min, mid, max;
    public static String username = "", currentLine = "", allUserPassData = "",
    allScheduleData = "", allAccDetailsData = "";
    public static String[] tempData = new String[1500], allUserPassDataArr,
    allScheduleDataArr, allAccDetailsDataArr;
    private static ImageIcon menuIcons[] = new ImageIcon[2];
    private static Point mousePos;
    private static FileWriter userAndPassWrite, accountDetailsWrite, scheduleWrite;
    private static Scanner userAndPassScan, accountDetailsScan, scheduleScan;

    public Menu() {
        timer = new Timer(10, this);
        mousePos = MouseInfo.getPointerInfo().getLocation();

        menuIcons[0] = new ImageIcon("Login.jpg");
        menuIcons[1] = new ImageIcon("SignUp.jpg");

        for(int i = 0; i < menuIcons.length; i++){
            Image img = menuIcons[i].getImage();
            Image newImg = img.getScaledInstance(200, 90, java.awt.Image.SCALE_SMOOTH);
            menuIcons[i] = new ImageIcon(newImg);
            buttons[i] = new JButton(menuIcons[i]);
        }

        this.setLayout(null);

        for(int i = 0; i < buttons.length; i++){
            buttons[i].setBounds(i*350+400,300,200,90);
        }

        for(int i = 0; i < buttons.length; i++){
            buttons[i].addActionListener(this);
            buttons[i].setOpaque(false);
            buttons[i].setContentAreaFilled(false);
            buttons[i].setBorderPainted(true);
            buttons[i].setFocusPainted(false);
            this.add(buttons[i]);
        }

        back = new JButton("Back");
        back.setBounds(900, 500, 200, 90);
        back.addActionListener(this);
        back.setOpaque(false);
        back.setContentAreaFilled(false);
        back.setBorderPainted(true);
```

```
60          back.setFocusPainted(false);
61          this.add(back);
62
63          user = new JTextField();
64          pass = new JTextField();
65
66          user.setBounds((PANEL_WIDTH/2)-150, (PANEL_HEIGHT/2)-100, 300, 60);
67          pass.setBounds((PANEL_WIDTH/2)-150, (PANEL_HEIGHT/2)+20, 300, 60);
68
69          user.addActionListener(this);
70          pass.addActionListener(this);
71
72          this.add(user);
73          this.add(pass);
74
75          for(int i = 0; i < accDetails.length; i++){
76              accDetails[i] = new JTextField();
77              accDetails[i].setBounds((PANEL_WIDTH/2)-150, (PANEL_HEIGHT/5)+(i*125), 300,
                60);
78              accDetails[i].addActionListener(this);
79              this.add(accDetails[i]);
80          }
81
82          reset();
83      }
84
85      public void run(){
86          timer.start();
87      }
88
89      public void paintComponent(Graphics g) {
90          super.paintComponent(g);
91
92          g.setColor(new Color(205, 255, 255));
93          g.fillRect(0, 0, PANEL_WIDTH, PANEL_HEIGHT);
94          g.setColor(new Color(0, 0, 0));
95          g.setFont(new Font("SansSerif: ", 1, 24));
96          if(loginAttempt){
97              if(!accountFound){
98                  g.drawString("Incorrect username or password", 35, 260);
99              }
100             else{
101                 g.drawString("Success!", 35, 260);
102             }
103         }
104         if(signUpAttempt){
105             if(!accountFound){
106                 g.drawString("Works!", 35, 260);
107             }
108             else{
109                 g.drawString("Sorry, username is taken.", 35, 260);
110             }
111         }
112
113         g.setFont(new Font("SansSerif: ", 1, 20));
114         if(signUp || login){
115             g.drawString("Username", 640, 280);
116             g.drawString("Password", 640, 400);
117         }
118         if(enterAccDetails){
119             g.drawString("Height (kg)", 640, 140);
120             g.drawString("Weight (m)", 640, 260);
121             g.drawString("Age", 640, 390);
122             g.drawString("Gender", 640, 520);
```

```
123             }
124             if(homeScreen) {
125                 g.setFont(new Font("SansSerif: ", 1, 60));
126                 g.drawString("Workout Planner", 440, 120);
127             }
128         }
129 
130         public void actionPerformed(ActionEvent ev){
131             mousePos = MouseInfo.getPointerInfo().getLocation();
132             buttonChecker(ev);
133             textInputChecker(ev);
134             repaint();
135         }
136 
137         public static void buttonChecker(ActionEvent ev){
138             if(ev.getSource() == buttons[0]){//Login
139                 login = true;
140                 homeScreen = false;
141                 for(int i = 0; i < buttons.length; i++){
142                     buttons[i].setVisible(false);
143                 }
144                 user.setVisible(true);
145                 pass.setVisible(true);
146                 back.setVisible(true);
147             }
148             else if(ev.getSource() == buttons[1]){//Sign up
149                 signUp = true;
150                 homeScreen = false;
151                 for(int i = 0; i < buttons.length; i++){
152                     buttons[i].setVisible(false);
153                 }
154                 user.setVisible(true);
155                 pass.setVisible(true);
156                 back.setVisible(true);
157             }
158 
159             if(ev.getSource() == back){
160                 reset();
161             }
162         }
163 
164         public static void textInputChecker(ActionEvent ev){
165             if(ev.getSource() == pass){
166                 if(login){
167                     login();
168                 }
169                 else if(signUp){
170                     signUp();
171                     if(signUpAttempt && !accountFound){
172                         accountSetup();
173                     }
174                 }
175             }
176             if(ev.getSource() == accDetails[3]){
177                 accountDetails();
178                 reset();
179             }
180         }
181 
182         public static void reset(){
183             homeScreen = true;
184             user.setVisible(false);
185             pass.setVisible(false);
186             back.setVisible(false);
```

```
187             user.setText("");
188             pass.setText("");
189             for(int i = 0; i < buttons.length; i++){
190                 buttons[i].setVisible(true);
191             }
192             login = false;
193             signUp = false;
194             loginAttempt = false;
195             signUpAttempt = false;
196 
197             for(int i = 0; i < accDetails.length; i++){
198                 accDetails[i].setVisible(false);
199             }
200         }
201 
202         public static void accountSetup(){
203             enterAccDetails = true;
204             signUp = false;
205             user.setVisible(false);
206             pass.setVisible(false);
207             back.setVisible(false);
208             for(int i = 0; i < accDetails.length; i++){
209                 accDetails[i].setVisible(true);
210             }
211         }
212 
213         public static void login(){
214             loginAttempt = true;
215             try {
216                 userAndPassScan = new Scanner(new File("UserAndPass.txt"));
217             }
218             catch (Exception error) {
219                 System.out.println("An error occurred:" + error);
220                 error.printStackTrace();
221             }
222 
223             accountFound = false;
224             currentLine = "";
225             allUserPassData = "";
226             while(userAndPassScan.hasNextLine()){
227                 currentLine = userAndPassScan.nextLine();
228                 allUserPassData += currentLine + "%";
229             }
230             allUserPassDataArr = allUserPassData.split("%");
231             userAndPassScan.close();
232             min = 0;
233             max = allUserPassDataArr.length - 1;
234             while(min <= max){
235                 mid = (min + max)/2;
236                 String userPass[] = allUserPassDataArr[mid].split(" ");
237                 if(user.getText().compareTo(userPass[0]) < 0){
238                     max = mid -1;
239                 }
240                 else if(user.getText().compareTo(userPass[0]) > 0){
241                     min = mid + 1;
242                 }
243                 else if((user.getText() + " " +
                    pass.getText()).equals(allUserPassDataArr[mid])){
244                     username = user.getText();
245                     accountFound = true;
246                     loginSuccess = true;
247                     break;
248                 }
249             }
```

```
250
251         userAndPassScan.close();
252     }
253
254     public static void signUp(){
255         signUpAttempt = true;
256         try {
257             userAndPassScan = new Scanner(new File("UserAndPass.txt"));
258             scheduleScan = new Scanner(new File("ScheduleData.txt"));
259             accountDetailsScan = new Scanner(new File("AccountDetails.txt"));
260         }
261         catch (Exception error) {
262             System.out.println("An error occurred:" + error);
263             error.printStackTrace();
264         }
265
266         accountCounter = 0;
267         accountFound = false;
268         currentLine = "";
269         allUserPassData = "";
270         allScheduleData = "";
271         allAccDetailsData = "";
272         while(userAndPassScan.hasNextLine()){
273             currentLine = userAndPassScan.nextLine();
274             allUserPassData += currentLine + "%";
275             String userPass[] = currentLine.split(" ");
276             if((user.getText()).equals(userPass[0])){
277                 accountFound = true;
278                 newUserRegistered = false;
279                 break;
280             }
281             currentLine = scheduleScan.nextLine();
282             allScheduleData += currentLine + "%";
283             currentLine = accountDetailsScan.nextLine();
284             allAccDetailsData += currentLine + "%";
285             accountCounter++;
286         }
287         allUserPassDataArr = allUserPassData.split("%");
288         userAndPassScan.close();
289         allScheduleDataArr = allScheduleData.split("%");
290         scheduleScan.close();
291         allAccDetailsDataArr = allAccDetailsData.split("%");
292         accountDetailsScan.close();
293
294         accounts[accountCounter] = new Account();
295         accounts[accountCounter].setUser(user.getText());
296         accounts[accountCounter].setPass(pass.getText());
297
298         if(!accountFound){
299             try {
300                 userAndPassWrite = new FileWriter("UserAndPass.txt", false);
301                 BufferedWriter bUAPW = new BufferedWriter(userAndPassWrite);
302                 scheduleWrite = new FileWriter("ScheduleData.txt", false);
303                 BufferedWriter bSW = new BufferedWriter(scheduleWrite);
304                 accountDetailsWrite = new FileWriter("AccountDetails.txt", false);
305                 BufferedWriter bADW = new BufferedWriter(accountDetailsWrite);
306
307                 try {
308                     for(int i = 0; i < allUserPassDataArr.length; i++){
309                         tempData = allUserPassDataArr[i].split(" ");
310                         if(!newUserRegistered){
311
                                 if(accounts[accountCounter].getUser().compareTo(tempData[0])
                                 < 0){
```

```
                                bSW.write(accounts[accountCounter].getUser());
                                bSW.newLine();
                                bADW.write(accounts[accountCounter].getUser());
                                bADW.newLine();
                                bUAPW.write(accounts[accountCounter].getUser() + " " +
                                accounts[accountCounter].getPass());
                                bUAPW.newLine();
                                accountNum = i;
                                i--;
                                newUserRegistered = true;
                            }
                            else{
                                bSW.write(allScheduleDataArr[i]);
                                bSW.newLine();
                                bADW.write(allAccDetailsDataArr[i]);
                                bADW.newLine();
                                bUAPW.write(allUserPassDataArr[i]);
                                bUAPW.newLine();
                            }
                        }
                        else{
                            bSW.write(allScheduleDataArr[i]);
                            bSW.newLine();
                            bADW.write(allAccDetailsDataArr[i]);
                            bADW.newLine();
                            bUAPW.write(allUserPassDataArr[i]);
                            bUAPW.newLine();
                        }
                    }
                    if(!newUserRegistered){
                        accountNum = allUserPassDataArr.length + 1;
                        bSW.write(accounts[accountCounter].getUser());
                        bSW.newLine();
                        bADW.write(accounts[accountCounter].getUser());
                        bADW.newLine();
                        bUAPW.write(accounts[accountCounter].getUser() + " " +
                        accounts[accountCounter].getPass());
                        bUAPW.newLine();
                    }
                bUAPW.close();
                bSW.close();
                bADW.close();
                }
                catch (IOException error) {
                    System.out.println("An error occurred:" + error);
                    error.printStackTrace();
                }
            }
            catch (IOException error) {
                System.out.println("An error occurred:" + error);
                error.printStackTrace();
            }
        }
    }

    public static void accountDetails(){
        try {
            accountDetailsScan = new Scanner(new File("AccountDetails.txt"));
        }
        catch (Exception error) {
            System.out.println("An error occurred:" + error);
            error.printStackTrace();
        }

```

```
374             accounts[accountCounter].setHeight(Integer.parseInt(accDetails[0].getText()));
375             accounts[accountCounter].setWeight(Integer.parseInt(accDetails[1].getText()));
376             accounts[accountCounter].setAge(Integer.parseInt(accDetails[2].getText()));
377             accounts[accountCounter].setGender(accDetails[3].getText());
378 
379             try {
380                 accountDetailsWrite = new FileWriter("AccountDetails.txt", false);
381                 BufferedWriter bADW = new BufferedWriter(accountDetailsWrite);
382                 dataRegistered = false;
383                 for(int i = 0; i < allAccDetailsDataArr.length; i++){
384                     if(i == accountNum && !dataRegistered){
385                         bADW.write(accounts[accountCounter].getUser() + ", " +
                            accounts[accountCounter].getHeight() + ", " +
                            accounts[accountCounter].getWeight() + ", " +
                            accounts[accountCounter].getAge() + ", " +
                            accounts[accountCounter].getGender());
386                         bADW.newLine();
387                         i--;
388                         dataRegistered = true;
389                     }
390                     else{
391                         bADW.write(allAccDetailsDataArr[i]);
392                         bADW.newLine();
393                     }
394                 }
395                 bADW.close();
396             }
397             catch (IOException e) {
398                 System.out.println("An error occurred:" + e);
399                 e.printStackTrace();
400             }
401             accountDetailsScan.close();
402             enterAccDetails = false;
403         }
404     }
```

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.Timer;
import java.io.*;

@SuppressWarnings("unchecked")
public class DailySchedule extends JPanel implements ActionListener, MouseListener, MouseWheelListener{
    public static Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
    public static int PANEL_HEIGHT = (dim.height*9)/10;
    public static int PANEL_WIDTH = (dim.width*9)/10;
    public static Timer timer;
    public static SlotsAndOptions exerciseOptions[] = new SlotsAndOptions[61], exerciseSlots[] = new SlotsAndOptions[61], dailySlots[] = new SlotsAndOptions[49], timeSlots[] = new SlotsAndOptions[49];
    public static Workouts workouts[] = new Workouts[61];
    public static String[] viewLayouts = {"Daily", "Weekly"}, workoutDetails = new String[4], weekNames = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"}, allDataArr;
    public static String currentLine = "", user = "", selectedDay = "", allData = "", tempLine = "";
    public static ArrayList<String> scheduleData = new ArrayList<String>(), tempData;
    public static JComboBox selectMenu;
    public static JLabel slotsCont, optionsCont, menuBar;
    private static Point mousePos;
    public static int JpanelX, JpanelY, mouseLatestX = 0, mouseLatestY = 0, labelHeld = 0, clipX = 0, clipY = 0, tickCounter = 0, scrollSensitivity = 40, counter = 0, accountCounter = 0, accountNum = 0;
    public static boolean mouseHold = false, clip = false, selected = true, scrollUp = false, scrollDown = false, accountFound = false, paintLoad = false;
    private Scanner scFile = null, dataScan;
    private static FileWriter dataWriter;

    public DailySchedule() {
        timer = new Timer(10, this);
        mousePos = MouseInfo.getPointerInfo().getLocation();

        this.setLayout(null);
        addMouseListener(this);
        addMouseWheelListener(this);

        JpanelX = ((dim.width-PANEL_WIDTH)/2)+5;
        JpanelY = ((dim.height-PANEL_HEIGHT)/2)+30;

        /*
        menuBar = new MenuBar(0, 0, PANEL_WIDTH-14, PANEL_HEIGHT/10);
        menuBar.setBounds(menuBar.posX, menuBar.posY, menuBar.sizeX, menuBar.sizeY);
        menuBar.setOpaque(true);
        menuBar.setBackground(new Color(255, 165, 205));
        menuBar.setBorder(BorderFactory.createLineBorder((new Color(0,0,0)), 4));
        this.add(menuBar);
        */

        selectMenu = new JComboBox(viewLayouts);
        selectMenu.setVisible(true);
        selectMenu.setEditable(false);
        selectMenu.setBounds(8*(PANEL_WIDTH/10), (PANEL_HEIGHT/20)-10, 100, 20);
        this.add(selectMenu);

        menuBar = new JLabel();
        menuBar.setBounds(0, 0, PANEL_WIDTH-14, PANEL_HEIGHT/10);
        menuBar.setOpaque(true);
```

```
56          menuBar.setBackground(new Color(255, 165, 205));
57          this.add(menuBar);
58
59          try {
60              scFile = new Scanner(new File("WorkoutList.txt"));
61          }
62          catch (Exception error) {
63              System.out.println("An error occurred:" + error);
64              error.printStackTrace();
65          }
66
67          for(int i = 0; i < exerciseOptions.length; i++){
68              currentLine = scFile.nextLine();
69              workoutDetails = currentLine.split("% ");
70              workouts[i] = new Workouts(workoutDetails[0], workoutDetails[1],
                workoutDetails[2], workoutDetails[3], Integer.parseInt(workoutDetails[4]));
71              exerciseOptions[i] = new SlotsAndOptions(1000, 200+(i*60), 300, 50);
72              exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                exerciseOptions[i].posY, exerciseOptions[i].sizeX, exerciseOptions[i].sizeY);
73              exerciseOptions[i].setOpaque(true);
74              if(workouts[i].difficulty.equals("Beginner")){
75                  exerciseOptions[i].setBackground(new Color(205, 255, 205));
76              }
77              else if(workouts[i].difficulty.equals("Intermediate")){
78                  exerciseOptions[i].setBackground(new Color(255, 255, 205));
79              }
80              else if(workouts[i].difficulty.equals("Advanced")){
81                  exerciseOptions[i].setBackground(new Color(255, 205, 205));
82              }
83              exerciseOptions[i].setBorder(BorderFactory.createLineBorder((new
                Color(0,0,0)), 2));
84              exerciseOptions[i].setFont(new Font("SansSerif", 1, 18));
85              exerciseOptions[i].setText(exerciseOptions[i].getText() + " " +
                workouts[i].name + "- " + workouts[i].muscleGroup);
86              this.add(exerciseOptions[i]);
87              exerciseSlots[i] = new SlotsAndOptions(980, 195+(i*60), 340, 60);
88              exerciseSlots[i].setBounds(exerciseSlots[i].posX, exerciseSlots[i].posY,
                exerciseSlots[i].sizeX, exerciseSlots[i].sizeY);
89              exerciseSlots[i].setOpaque(true);
90              if(workouts[i].difficulty.equals("Beginner")){
91                  exerciseSlots[i].setBackground(new Color(135, 205, 135, 85));
92              }
93              else if(workouts[i].difficulty.equals("Intermediate")){
94                  exerciseSlots[i].setBackground(new Color(205, 205, 135, 85));
95              }
96              else if(workouts[i].difficulty.equals("Advanced")){
97                  exerciseSlots[i].setBackground(new Color(205, 135, 135, 85));
98              }
99              exerciseSlots[i].setBorder(BorderFactory.createLineBorder((new
                Color(0,0,0)), 1));
100             this.add(exerciseSlots[i]);
101         }
102         scFile.close();
103
104         for(int i = 0; i < dailySlots.length; i++){
105             dailySlots[i] = new SlotsAndOptions(125, 200+(i*68), 800, 70);
106             dailySlots[i].setBounds(dailySlots[i].posX, dailySlots[i].posY,
                dailySlots[i].sizeX, dailySlots[i].sizeY);
107             dailySlots[i].setOpaque(true);
108             dailySlots[i].setBackground(new Color(255, 255, 255));
109             dailySlots[i].setBorder(BorderFactory.createLineBorder((new Color(0,0,0)),
                3));
110             this.add(dailySlots[i]);
111             timeSlots[i] = new SlotsAndOptions(50, 200+(i*68), 95, 70);
```

```
112             timeSlots[i].setBounds(timeSlots[i].posX, timeSlots[i].posY,
                timeSlots[i].sizeX, timeSlots[i].sizeY);
113             timeSlots[i].setOpaque(true);
114             timeSlots[i].setBackground(new Color(255, 255, 255));
115             timeSlots[i].setBorder(BorderFactory.createLineBorder((new Color(0,0,0)),
                3));
116             this.add(timeSlots[i]);
117         }
118
119         counter = 0;
120         for(double i = 8; counter < dailySlots.length; i+=0.25){
121             timeSlots[counter].setFont(new Font("SansSerif", 1, 20));
122             if(i%1==0){
123                 timeSlots[counter].setText(timeSlots[counter].getText()+" "+(int)i +
                    ":00");
124             }
125             else{
126                 timeSlots[counter].setText(timeSlots[counter].getText()+" "+(int)i +
                    ":" + (((int)((i%1)/0.25)*15)));
127             }
128             counter++;
129         }
130
131         slotsCont = new JLabel();
132         slotsCont.setBounds(0, PANEL_HEIGHT/10, 950, PANEL_HEIGHT);
133         slotsCont.setOpaque(true);
134         slotsCont.setBackground(new Color(205, 205, 245));
135         slotsCont.setBorder(BorderFactory.createLineBorder((new Color(0,0,0)), 1));
136         this.add(slotsCont);
137
138         optionsCont = new JLabel();
139         optionsCont.setBounds(950, PANEL_HEIGHT/10, 430, PANEL_HEIGHT);
140         optionsCont.setOpaque(true);
141         optionsCont.setBackground(new Color(205, 245, 245));
142         optionsCont.setBorder(BorderFactory.createLineBorder((new Color(0,0,0)), 1));
143         this.add(optionsCont);
144     }
145
146     public void run(){
147         timer.start();
148         selected = true;
149         selectMenu.setSelectedItem("Daily");
150         loadData(selectedDay);
151     }
152
153     public void stop(){
154         reset();
155         selected = false;
156         timer.stop();
157     }
158
159     public void paintComponent(Graphics g) {
160         super.paintComponent(g);
161
162         g.setColor(new Color(255, 255, 205));
163         g.fillRect(0, 0, PANEL_WIDTH, PANEL_HEIGHT);
164
165         if(paintLoad){
166             for(int i = 1; i < scheduleData.size(); i+=3){
167                 if(Integer.parseInt(scheduleData.get(i)) !=
                    java.util.Arrays.asList(weekNames).indexOf(selectedDay)){
168                     continue;
169                 }
170                 clip = true;
```

```
171                 labelHeld = Integer.parseInt(scheduleData.get(i+2));
172                 dailySlots[Integer.parseInt(scheduleData.get(i+1))].taken = true;
173                 dailySlots[Integer.parseInt(scheduleData.get(i+1))].holding = labelHeld;
174                 clipX =
                    dailySlots[Integer.parseInt(scheduleData.get(i+1))].posX+(dailySlots[Inte
                    ger.parseInt(scheduleData.get(i+1))].sizeX/2)-(exerciseOptions[labelHeld]
                    .sizeX/2);
175                 clipY =
                    dailySlots[Integer.parseInt(scheduleData.get(i+1))].posY+(dailySlots[Inte
                    ger.parseInt(scheduleData.get(i+1))].sizeY/2)-(exerciseOptions[labelHeld]
                    .sizeY/2);
176                 exerciseOptions[labelHeld].posX = clipX;
177                 exerciseOptions[labelHeld].posY = clipY;
178                 while(exerciseOptions[labelHeld].sizeX < dailySlots[0].sizeX-12){
179
                        exerciseOptions[labelHeld].setBounds(clipX-=(((dailySlots[0].sizeX-ex
                        erciseOptions[labelHeld].sizeX)/20)+1), clipY,
                        exerciseOptions[labelHeld].sizeX+=2*(((dailySlots[0].sizeX-exerciseOp
                        tions[labelHeld].sizeX)/20)+1), exerciseOptions[labelHeld].sizeY);
180                     exerciseOptions[labelHeld].posX = clipX;
181                 }
182             }
183             dataScan.close();
184             clip = false;
185             paintLoad = false;
186         }
187
188         if(mouseHold){
189             if(exerciseOptions[labelHeld].held){
190                 exerciseOptions[labelHeld].posX =
                    mousePos.x-JpanelX-(exerciseOptions[labelHeld].sizeX)/2;
191                 exerciseOptions[labelHeld].posY =
                    mousePos.y-JpanelY-(exerciseOptions[labelHeld].sizeY)/2;
192                 exerciseOptions[labelHeld].sizeX = exerciseOptions[labelHeld].origSizeX;
193
194                 if(!clip){
195
                        exerciseOptions[labelHeld].setBounds(exerciseOptions[labelHeld].posX,
                         exerciseOptions[labelHeld].posY,
                        exerciseOptions[labelHeld].origSizeX,
                        exerciseOptions[labelHeld].origSizeY);
196                 }
197                 else if(clip){
198                     exerciseOptions[labelHeld].setBounds(clipX, clipY,
                        exerciseOptions[labelHeld].sizeX, exerciseOptions[labelHeld].sizeY);
199                 }
200             }
201
202             clip = false;
203             for(int i = 0; i < dailySlots.length; i++){
204                 if((dailySlots[i].getBounds().contains(new Point(mousePos.x-JpanelX,
                    mousePos.y-JpanelY))) && !dailySlots[i].taken){
205                     clip = true;
206                     clipX =
                        dailySlots[i].posX+(dailySlots[i].sizeX/2)-(exerciseOptions[labelHeld
                        ].sizeX/2);
207                     clipY =
                        dailySlots[i].posY+(dailySlots[i].sizeY/2)-(exerciseOptions[labelHeld
                        ].sizeY/2);
208                     exerciseOptions[labelHeld].posX = clipX;
209                     exerciseOptions[labelHeld].posY = clipY;
210                 }
211             }
212         }
```

```
213         if(exerciseOptions[labelHeld].sizeX < dailySlots[0].sizeX-12 && clip){
214 
                exerciseOptions[labelHeld].setBounds(clipX-=(((dailySlots[0].sizeX-exerciseOp
                tions[labelHeld].sizeX)/20)+1), clipY,
                exerciseOptions[labelHeld].sizeX+=2*(((dailySlots[0].sizeX-exerciseOptions[la
                belHeld].sizeX)/20)+1), exerciseOptions[labelHeld].sizeY);
215             exerciseOptions[labelHeld].posX = clipX;
216         }
217 
218         if(scrollUp){
219             if(slotsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
                mousePos.y-JpanelY)) && dailySlots[0].posY < dailySlots[0].origY){
220                 for(int j = 0; j < dailySlots.length; j++){
221                     dailySlots[j].posY+=scrollSensitivity;
222                     dailySlots[j].setBounds(dailySlots[j].posX, dailySlots[j].posY,
                    dailySlots[j].sizeX, dailySlots[j].sizeY);
223                     timeSlots[j].posY+=scrollSensitivity;
224                     timeSlots[j].setBounds(timeSlots[j].posX, timeSlots[j].posY,
                    timeSlots[j].sizeX, timeSlots[j].sizeY);
225                 }
226                 for(int i = 0; i < exerciseOptions.length; i++){
227                     if(exerciseOptions[i].posX != exerciseOptions[i].origX &&
                    exerciseOptions[i].posY != exerciseOptions[i].origY && !mouseHold){
228                         exerciseOptions[i].posY+=scrollSensitivity;
229                         exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
230                     }
231                 }
232             }
233             else if(optionsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
                mousePos.y-JpanelY)) && exerciseSlots[0].posY < exerciseSlots[0].origY){
234                 for(int i = 0; i < exerciseOptions.length; i++){
235                     if(exerciseOptions[i].posX == exerciseOptions[i].origX){
236                         exerciseOptions[i].posY+=scrollSensitivity;
237                         exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
238                     }
239                     exerciseSlots[i].posY+=scrollSensitivity;
240                     exerciseSlots[i].setBounds(exerciseSlots[i].posX,
                    exerciseSlots[i].posY, exerciseSlots[i].sizeX,
                    exerciseSlots[i].sizeY);
241                 }
242             }
243             scrollUp = false;
244         }
245         else if(scrollDown){
246             if(slotsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
                mousePos.y-JpanelY)) && dailySlots[48].posY > 650){
247                 for(int j = 0; j < dailySlots.length; j++){
248                     dailySlots[j].posY-=scrollSensitivity;
249                     timeSlots[j].posY-=scrollSensitivity;
250                     dailySlots[j].setBounds(dailySlots[j].posX, dailySlots[j].posY,
                    dailySlots[j].sizeX, dailySlots[j].sizeY);
251                     timeSlots[j].setBounds(timeSlots[j].posX, timeSlots[j].posY,
                    timeSlots[j].sizeX, timeSlots[j].sizeY);
252                 }
253                 for(int i = 0; i < exerciseOptions.length; i++){
254                     if(exerciseOptions[i].posX != exerciseOptions[i].origX &&
                    exerciseOptions[i].posY != exerciseOptions[i].origY && !mouseHold){
255                         exerciseOptions[i].posY-=scrollSensitivity;
256                         exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
```

```
                        exerciseOptions[i].sizeY);
257                 }
258             }
259         }
260         else if(optionsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
            mousePos.y-JpanelY)) && exerciseSlots[60].posY > 670){
261             for(int i = 0; i < exerciseOptions.length; i++){
262                 if(exerciseOptions[i].posX == exerciseOptions[i].origX){
263                     exerciseOptions[i].posY-=scrollSensitivity;
264                     exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
265                 }
266                 exerciseSlots[i].posY-=scrollSensitivity;
267                 exerciseSlots[i].setBounds(exerciseSlots[i].posX,
                    exerciseSlots[i].posY, exerciseSlots[i].sizeX,
                    exerciseSlots[i].sizeY);
268             }
269         }
270         scrollDown = false;
271     }
272     tickCounter++;
273 }
274
275 public void paint(Graphics g) {
276     super.paint(g);
277     g.setColor(new Color(255, 165, 205));
278     g.fillRect(0, 0, PANEL_WIDTH-400, PANEL_HEIGHT/10);
279     g.setColor(new Color(0, 0, 0));
280     g.setFont(new Font("SansSerif: ", 1, 28));
281     g.drawString("Daily Schedule", ((PANEL_WIDTH-14)/2)-80,
        ((PANEL_HEIGHT/10)/2)+10);
282
283     g.setColor(new Color(195, 195, 195));
284     g.fillRect(0, PANEL_HEIGHT/10, 950, 100);
285     g.fillRect(950, PANEL_HEIGHT/10, 430, 100);
286     g.setColor(new Color(0, 0, 0));
287     g.drawRect(950, PANEL_HEIGHT/10, 430, 100);
288     g.drawRect(0, PANEL_HEIGHT/10, 950, 100);
289     g.setColor(new Color(0, 0, 0));
290     g.setFont(new Font("SansSerif: ", 1, 40));
291     g.drawString("Schedule Timeslots", 275, (PANEL_HEIGHT/10)+65);
292     g.drawString("Workout Options", 1000, (PANEL_HEIGHT/10)+65);
293 }
294
295 public void actionPerformed(ActionEvent ev){
296     mousePos = MouseInfo.getPointerInfo().getLocation();
297     buttonChecker(ev);
298
299     if (((String)selectMenu.getSelectedItem()).equals("Weekly")){
300         stop();
301     }
302
303     repaint();
304 }
305
306 public static void buttonChecker(ActionEvent ev){
307     //(jButton1.getModel().isPressed())
308 }
309
310
311 public void mouseClicked(MouseEvent e) {
312
313 }
```

```
314
315     public void mouseExited(MouseEvent e) {
316
317     }
318
319     public void mouseEntered(MouseEvent e) {
320
321     }
322
323     public void mousePressed(MouseEvent e) {
324         mouseLatestX = e.getX();
325         mouseLatestY = e.getY();
326         clip = false;
327
328         for(int i = 0; i < exerciseOptions.length; i++){
329             if(exerciseOptions[i].getBounds().contains(new Point(mouseLatestX,
                mouseLatestY))){
330                 mouseHold = true;
331                 exerciseOptions[i].held = true;
332                 labelHeld = i;
333             }
334         }
335         for(int i = 0; i < dailySlots.length; i++){
336             if(dailySlots[i].getBounds().contains(new Point(mousePos.x-JpanelX,
                mousePos.y-JpanelY))){
337                 dailySlots[i].taken = false;
338                 dailySlots[i].holding = -1;
339             }
340         }
341     }
342
343     public void mouseReleased(MouseEvent e) {
344         if(!clip && mouseHold){
345             exerciseOptions[labelHeld].posX = exerciseSlots[labelHeld].posX+20;
346             exerciseOptions[labelHeld].posY = exerciseSlots[labelHeld].posY+5;
347             exerciseOptions[labelHeld].setBounds(exerciseOptions[labelHeld].posX,
                exerciseOptions[labelHeld].posY, exerciseOptions[labelHeld].origSizeX,
                exerciseOptions[labelHeld].origSizeY);
348         }
349         for(int i = 0; i < exerciseOptions.length; i++){
350             exerciseOptions[i].held = false;
351         }
352         for(int i = 0; i < dailySlots.length; i++){
353             if(dailySlots[i].getBounds().contains(new Point(mousePos.x-JpanelX,
                mousePos.y-JpanelY)) && mouseHold){
354                 dailySlots[i].taken = true;
355                 dailySlots[i].holding = labelHeld;
356             }
357         }
358         mouseHold = false;
359     }
360
361     public void mouseWheelMoved(MouseWheelEvent e) {
362         if(e.getWheelRotation() < 0){
363             scrollUp = true;
364         }
365         else if(e.getWheelRotation() > 0){
366             scrollDown = true;
367         }
368     }
369
370     public void reset(){
371         for(int i = 0; i < exerciseOptions.length; i++){
372             exerciseOptions[i].held = false;
```

```
373                 exerciseOptions[i].posX = exerciseOptions[i].origX;
374                 exerciseOptions[i].posY = exerciseOptions[i].origY;
375                 exerciseOptions[i].sizeX = exerciseOptions[i].origSizeX;
376                 exerciseOptions[i].sizeY = exerciseOptions[i].origSizeY;
377                 exerciseSlots[i].posX = exerciseSlots[i].origX;
378                 exerciseSlots[i].posY = exerciseSlots[i].origY;
379                 exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                    exerciseOptions[i].posY, exerciseOptions[i].sizeX, exerciseOptions[i].sizeY);
380                 exerciseSlots[i].setBounds(exerciseSlots[i].posX, exerciseSlots[i].posY,
                    exerciseSlots[i].sizeX, exerciseSlots[i].sizeY);
381             }
382             for(int i = 0; i < dailySlots.length; i++){
383                 dailySlots[i].taken = false;
384             }
385         }
386 
387         public void loadData(String day){
388             try {
389                 dataScan = new Scanner(new File("ScheduleData.txt"));
390             }
391             catch (Exception error) {
392                 System.out.println("An error occurred:" + error);
393                 error.printStackTrace();
394             }
395             accountCounter = 0;
396             allData = "";
397             currentLine = "";
398             accountFound = false;
399             while(dataScan.hasNextLine()){
400                 currentLine = dataScan.nextLine();
401                 allData += currentLine + "%";
402                 tempData = new ArrayList<>(Arrays.asList(currentLine.split(", ")));
403                 if(user.equals(tempData.get(0))){
404                     accountFound = true;
405                     accountNum = accountCounter;
406                     scheduleData = tempData;
407                 }
408                 accountCounter++;
409             }
410             allDataArr = allData.split("%");
411             paintLoad = true;
412             dataScan.close();
413         }
414     }
```

```
1   import java.awt.*;
2   import java.awt.event.*;
3   import javax.swing.*;
4   import java.util.*;
5   import javax.swing.Timer;
6   import java.io.*;
7   import java.time.*;
8
9   @SuppressWarnings("unchecked")
10  public class WeeklySchedule extends JPanel implements ActionListener, MouseListener,
    MouseWheelListener{
11      public static Calendar calendar = Calendar.getInstance();
12      public static Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
13      public static int PANEL_HEIGHT = (dim.height*9)/10;
14      public static int PANEL_WIDTH = (dim.width*9)/10;
15      public static Timer timer;
16      public static SlotsAndOptions exerciseOptions[] = new SlotsAndOptions[61],
        exerciseSlots[] = new SlotsAndOptions[61], dailySlots[][] = new
        SlotsAndOptions[7][49], timeSlots[] = new SlotsAndOptions[49];
17      public static Workouts workouts[] = new Workouts[61], tempWorkout = new Workouts();
18      public static String[] viewLayouts = {"Daily", "Weekly"}, workoutDetails = new
        String[4], sortNames = {"Alphanetical", "Muscle Group", "Difficulty"}, weekNames =
        {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"},
        allDataArr;
19      public static String currentLine = "", allData = "", tempLine = "", selectedDay =
        weekNames[calendar.get(Calendar.DAY_OF_WEEK)-1];
20      public static ArrayList<String> scheduleData = new ArrayList<String>(), tempData;
21      public String user = "";
22      public static JComboBox selectMenu;
23      public static JLabel slotsCont, optionsCont, menuBar;
24      public static JButton daysOfWeek[] = new JButton[7], sortOptions[] = new JButton[3];
25      private static Point mousePos;
26      public static int JpanelX, JpanelY, mouseLatestX = 0, mouseLatestY = 0, labelHeld =
        0, clipX = 0, clipY = 0, tickCounter = 0, scrollSensitivity = 40, counter = 0,
        accountCounter = 0, accountNum = 0;
27      public static boolean mouseHold = false, clip = false, selected = true, scrollUp =
        false, scrollDown = false, accountFound = false, paintLoad = false, enterDayView =
        false, update = false, scheduleEmpty = true;
28      private static Scanner scFile = null, dataScan;
29      private static FileWriter dataWriter;
30
31      public WeeklySchedule() {
32          timer = new Timer(10, this);
33          mousePos = MouseInfo.getPointerInfo().getLocation();
34
35          this.setLayout(null);
36          addMouseListener(this);
37          addMouseWheelListener(this);
38
39          JpanelX = ((dim.width-PANEL_WIDTH)/2)+5;
40          JpanelY = ((dim.height-PANEL_HEIGHT)/2)+30;
41
42          /*
43          menuBar = new MenuBar(0, 0, PANEL_WIDTH-14, PANEL_HEIGHT/10);
44          menuBar.setBounds(menuBar.posX, menuBar.posY, menuBar.sizeX, menuBar.sizeY);
45          menuBar.setOpaque(true);
46          menuBar.setBackground(new Color(255, 165, 205));
47          menuBar.setBorder(BorderFactory.createLineBorder((new Color(0,0,0)), 4));
48          this.add(menuBar);
49          */
50          selectMenu = new JComboBox(viewLayouts);
51          selectMenu.setVisible(true);
52          selectMenu.setEditable(false);
53          selectMenu.setBounds(8*(PANEL_WIDTH/10), (PANEL_HEIGHT/20)-10, 100, 20);
```

```
54          this.add(selectMenu);
55
56          menuBar = new JLabel();
57          menuBar.setBounds(0, 0, PANEL_WIDTH-14, PANEL_HEIGHT/10);
58          menuBar.setOpaque(true);
59          menuBar.setBackground(new Color(255, 165, 205));
60          this.add(menuBar);
61
62          for(int i = 0; i < daysOfWeek.length; i++){
63              daysOfWeek[i] = new JButton();
64              daysOfWeek[i].setFont(new Font("SansSerif", 1, 14));
65              daysOfWeek[i].setText(weekNames[i]);
66              daysOfWeek[i].setBounds(100+(i*120), 178, 120, 70);
67              daysOfWeek[i].addActionListener(this);
68              daysOfWeek[i].setOpaque(true);
69              daysOfWeek[i].setContentAreaFilled(true);
70              daysOfWeek[i].setBackground(new Color(255, 255, 255));
71              daysOfWeek[i].setBorder(BorderFactory.createLineBorder((new Color(0,0,0)),
                2));
72              daysOfWeek[i].setBorderPainted(true);
73              daysOfWeek[i].setFocusPainted(false);
74              this.add(daysOfWeek[i]);
75          }
76
77          for(int i = 0; i < sortOptions.length; i++){
78              sortOptions[i] = new JButton();
79              sortOptions[i].setFont(new Font("SansSerif", 1, 14));
80              sortOptions[i].setText(sortNames[i]);
81              sortOptions[i].setBounds(951+(i*139), 145, 139, 34);
82              sortOptions[i].addActionListener(this);
83              sortOptions[i].setOpaque(true);
84              sortOptions[i].setContentAreaFilled(true);
85              sortOptions[i].setBackground(new Color(255, 255, 255));
86              sortOptions[i].setBorder(BorderFactory.createLineBorder((new Color(0,0,0)),
                2));
87              sortOptions[i].setBorderPainted(true);
88              sortOptions[i].setFocusPainted(false);
89              this.add(sortOptions[i]);
90          }
91
92          try {
93              scFile = new Scanner(new File("WorkoutList.txt"));
94          }
95          catch (Exception error) {
96              System.out.println("An error occurred:" + error);
97              error.printStackTrace();
98          }
99
100         for(int i = 0; i < exerciseOptions.length; i++){
101             currentLine = scFile.nextLine();
102             workoutDetails = currentLine.split("% ");
103             workouts[i] = new Workouts(workoutDetails[0], workoutDetails[1],
                workoutDetails[2], workoutDetails[3], Integer.parseInt(workoutDetails[4]));
104             exerciseOptions[i] = new SlotsAndOptions(1000, 200+(i*60), 300, 50);
105             exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                exerciseOptions[i].posY, exerciseOptions[i].sizeX, exerciseOptions[i].sizeY);
106             exerciseOptions[i].setOpaque(true);
107             if(workouts[i].difficulty.equals("Beginner")){
108                 exerciseOptions[i].setBackground(new Color(205, 255, 205));
109             }
110             else if(workouts[i].difficulty.equals("Experienced")){
111                 exerciseOptions[i].setBackground(new Color(255, 255, 205));
112             }
113             else if(workouts[i].difficulty.equals("Expert")){
```

```
114                 exerciseOptions[i].setBackground(new Color(255, 205, 205));
115             }
116             exerciseOptions[i].setBorder(BorderFactory.createLineBorder((new
                Color(0,0,0)), 2));
117             exerciseOptions[i].setFont(new Font("SansSerif", 1, 18));
118             exerciseOptions[i].setText(exerciseOptions[i].getText() + " " +
                workouts[i].name + "- " + workouts[i].muscleGroup);
119             this.add(exerciseOptions[i]);
120 
121             exerciseSlots[i] = new SlotsAndOptions(980, 195+(i*60), 340, 60);
122             exerciseSlots[i].setBounds(exerciseSlots[i].posX, exerciseSlots[i].posY,
                exerciseSlots[i].sizeX, exerciseSlots[i].sizeY);
123             exerciseSlots[i].setOpaque(true);
124             if(workouts[i].difficulty.equals("Beginner")){
125                 exerciseSlots[i].setBackground(new Color(135, 205, 135, 85));
126             }
127             else if(workouts[i].difficulty.equals("Experienced")){
128                 exerciseSlots[i].setBackground(new Color(205, 205, 135, 85));
129             }
130             else if(workouts[i].difficulty.equals("Expert")){
131                 exerciseSlots[i].setBackground(new Color(205, 135, 135, 85));
132             }
133             exerciseSlots[i].setBorder(BorderFactory.createLineBorder((new
                Color(0,0,0)), 1));
134             this.add(exerciseSlots[i]);
135         }
136         scFile.close();
137 
138         for(int i = 0; i < dailySlots.length; i++){
139             for(int j = 0; j <dailySlots[i].length; j++){
140                 dailySlots[i][j] = new SlotsAndOptions(100+(i*120), 246+(j*68), 120, 70);
141                 dailySlots[i][j].setBounds(dailySlots[i][j].posX,
                    dailySlots[i][j].posY, dailySlots[i][j].sizeX, dailySlots[i][j].sizeY);
142                 dailySlots[i][j].setOpaque(true);
143                 dailySlots[i][j].setBackground(new Color(255, 255, 255));
144                 dailySlots[i][j].setBorder(BorderFactory.createLineBorder((new
                    Color(0,0,0)), 2));
145                 this.add(dailySlots[i][j]);
146             }
147         }
148 
149         counter = 0;
150         for(double i = 8; counter < timeSlots.length; i+=0.25){
151             timeSlots[counter] = new SlotsAndOptions(10, 246+(counter*68), 95, 70);
152             timeSlots[counter].setBounds(timeSlots[counter].posX,
                timeSlots[counter].posY, timeSlots[counter].sizeX, timeSlots[counter].sizeY);
153             timeSlots[counter].setOpaque(true);
154             timeSlots[counter].setBackground(new Color(255, 255, 255));
155             timeSlots[counter].setBorder(BorderFactory.createLineBorder((new
                Color(0,0,0)), 3));
156             this.add(timeSlots[counter]);
157 
158             timeSlots[counter].setFont(new Font("SansSerif", 1, 20));
159             if(i%1==0){
160                 timeSlots[counter].setText(timeSlots[counter].getText()+"   "+(int)i +
                    ":00");
161             }
162             else{
163                 timeSlots[counter].setText(timeSlots[counter].getText()+"   "+(int)i +
                    ":" + (((int)((i%1)/0.25)*15)));
164             }
165             counter++;
166         }
167 
```

```
168             slotsCont = new JLabel();
169             slotsCont.setBounds(0, PANEL_HEIGHT/10, 950, PANEL_HEIGHT);
170             slotsCont.setOpaque(true);
171             slotsCont.setBackground(new Color(205, 205, 245));
172             slotsCont.setBorder(BorderFactory.createLineBorder((new Color(0,0,0)), 1));
173             this.add(slotsCont);
174 
175             optionsCont = new JLabel();
176             optionsCont.setBounds(950, PANEL_HEIGHT/10, 430, PANEL_HEIGHT);
177             optionsCont.setOpaque(true);
178             optionsCont.setBackground(new Color(205, 245, 245));
179             optionsCont.setBorder(BorderFactory.createLineBorder((new Color(0,0,0)), 1));
180             this.add(optionsCont);
181         }
182 
183         public void run(){
184             timer.start();
185             selected = true;
186             selectMenu.setSelectedItem("Weekly");
187             selectedDay = weekNames[calendar.get(Calendar.DAY_OF_WEEK)-1];
188             loadData();
189         }
190 
191         public void stop(){
192             selected = false;
193             timer.stop();
194         }
195 
196         public void paintComponent(Graphics g) {
197             super.paintComponent(g);
198 
199             g.setColor(new Color(255, 255, 205));
200             g.fillRect(0, 0, PANEL_WIDTH, PANEL_HEIGHT);
201 
202             if(paintLoad){
203                 for(int i = 1; i < scheduleData.size(); i+=3){
204                     scheduleEmpty = false;
205                     clip = true;
206                     labelHeld = Integer.parseInt(scheduleData.get(i+2));
207 
                        dailySlots[Integer.parseInt(scheduleData.get(i))][Integer.parseInt(schedu
                        leData.get(i+1))].taken = true;
208 
                        dailySlots[Integer.parseInt(scheduleData.get(i))][Integer.parseInt(schedu
                        leData.get(i+1))].holding = workouts[labelHeld].num;
209                     clipX =
                        dailySlots[Integer.parseInt(scheduleData.get(i))][Integer.parseInt(schedu
                        leData.get(i+1))].posX+(dailySlots[Integer.parseInt(scheduleData.get(i))]
                        [Integer.parseInt(scheduleData.get(i+1))].sizeX/2)-(exerciseOptions[label
                        Held].sizeX/2);
210                     clipY =
                        dailySlots[Integer.parseInt(scheduleData.get(i))][Integer.parseInt(schedu
                        leData.get(i+1))].posY+(dailySlots[Integer.parseInt(scheduleData.get(i))]
                        [Integer.parseInt(scheduleData.get(i+1))].sizeY/2)-(exerciseOptions[label
                        Held].sizeY/2);
211                     exerciseOptions[labelHeld].posX = clipX;
212                     exerciseOptions[labelHeld].posY = clipY;
213                     exerciseOptions[labelHeld].inUse = true;
214                     while(exerciseOptions[labelHeld].sizeX > dailySlots[0][0].sizeX-12){
215 
                            exerciseOptions[labelHeld].setBounds(clipX+=(((exerciseOptions[labelH
                            eld].sizeX-dailySlots[0][0].sizeX)/20)+1), clipY,
                            exerciseOptions[labelHeld].sizeX-=2*(((exerciseOptions[labelHeld].siz
                            eX-dailySlots[0][0].sizeX)/20)+1), exerciseOptions[labelHeld].sizeY);
```

```
                    exerciseOptions[labelHeld].posX = clipX;
                }
            }
            dataScan.close();
            paintLoad = false;
        }

        if(mouseHold){
            if(exerciseOptions[labelHeld].held){
                exerciseOptions[labelHeld].posX =
                mousePos.x-JpanelX-(exerciseOptions[labelHeld].sizeX)/2;
                exerciseOptions[labelHeld].posY =
                mousePos.y-JpanelY-(exerciseOptions[labelHeld].sizeY)/2;
                exerciseOptions[labelHeld].sizeX = exerciseOptions[labelHeld].origSizeX;

                if(!clip){

                    exerciseOptions[labelHeld].setBounds(exerciseOptions[labelHeld].posX,
                     exerciseOptions[labelHeld].posY,
                    exerciseOptions[labelHeld].origSizeX,
                    exerciseOptions[labelHeld].origSizeY);
                }
                else if(clip){
                    exerciseOptions[labelHeld].setBounds(clipX, clipY,
                    exerciseOptions[labelHeld].sizeX, exerciseOptions[labelHeld].sizeY);
                }
            }

            clip = false;
            for(int i = 0; i < dailySlots.length; i++){
                for(int j = 0; j <dailySlots[i].length; j++){
                    if((dailySlots[i][j].getBounds().contains(new
                    Point(mousePos.x-JpanelX, mousePos.y-JpanelY))) &&
                    !dailySlots[i][j].taken){
                        clip = true;
                        clipX =
                        dailySlots[i][j].posX+(dailySlots[i][j].sizeX/2)-(exerciseOptions
                        [labelHeld].sizeX/2);
                        clipY =
                        dailySlots[i][j].posY+(dailySlots[i][j].sizeY/2)-(exerciseOptions
                        [labelHeld].sizeY/2);
                        exerciseOptions[labelHeld].posX = clipX;
                        exerciseOptions[labelHeld].posY = clipY;
                    }
                }
            }
        }
        if(exerciseOptions[labelHeld].sizeX > dailySlots[0][0].sizeX-12 && clip){

            exerciseOptions[labelHeld].setBounds(clipX+=(((exerciseOptions[labelHeld].siz
            eX-dailySlots[0][0].sizeX)/20)+1), clipY,
            exerciseOptions[labelHeld].sizeX-=2*(((exerciseOptions[labelHeld].sizeX-daily
            Slots[0][0].sizeX)/20)+1), exerciseOptions[labelHeld].sizeY);
            exerciseOptions[labelHeld].posX = clipX;
        }

        if(scrollUp){
            if(slotsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
            mousePos.y-JpanelY)) && dailySlots[0][0].posY < dailySlots[0][0].origY){
                for(int i = 0; i < dailySlots.length; i++){
                    for(int j = 0; j <dailySlots[i].length; j++){
                        dailySlots[i][j].posY+=scrollSensitivity;
                        dailySlots[i][j].setBounds(dailySlots[i][j].posX,
                        dailySlots[i][j].posY, dailySlots[i][j].sizeX,
```

```
                        dailySlots[i][j].sizeY);
261                 }
262             }
263             for(int i = 0; i < timeSlots.length; i++){
264                 timeSlots[i].posY+=scrollSensitivity;
265                 timeSlots[i].setBounds(timeSlots[i].posX, timeSlots[i].posY,
                    timeSlots[i].sizeX, timeSlots[i].sizeY);
266             }
267             for(int i = 0; i < exerciseOptions.length; i++){
268                 if(exerciseOptions[i].posX != exerciseOptions[i].origX &&
                    exerciseOptions[i].posY != exerciseOptions[i].origY && !mouseHold){
269                     exerciseOptions[i].posY+=scrollSensitivity;
270                     exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
271                 }
272             }
273         }
274         else if(optionsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
            mousePos.y-JpanelY)) && exerciseSlots[0].posY < exerciseSlots[0].origY){
275             for(int i = 0; i < exerciseOptions.length; i++){
276                 if(exerciseOptions[i].posX == exerciseOptions[i].origX){
277                     exerciseOptions[i].posY+=scrollSensitivity;
278                     exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
279                 }
280                 exerciseSlots[i].posY+=scrollSensitivity;
281                 exerciseSlots[i].setBounds(exerciseSlots[i].posX,
                    exerciseSlots[i].posY, exerciseSlots[i].sizeX,
                    exerciseSlots[i].sizeY);
282             }
283         }
284         scrollUp = false;
285     }
286     else if(scrollDown){
287         if(slotsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
            mousePos.y-JpanelY)) && dailySlots[0][48].posY > 650){
288             for(int i = 0; i < dailySlots.length; i++){
289                 for(int j = 0; j <dailySlots[i].length; j++){
290                     dailySlots[i][j].posY-=scrollSensitivity;
291                     dailySlots[i][j].setBounds(dailySlots[i][j].posX,
                        dailySlots[i][j].posY, dailySlots[i][j].sizeX,
                        dailySlots[i][j].sizeY);
292                 }
293             }
294             for(int i = 0; i < timeSlots.length; i++){
295                 timeSlots[i].posY-=scrollSensitivity;
296                 timeSlots[i].setBounds(timeSlots[i].posX, timeSlots[i].posY,
                    timeSlots[i].sizeX, timeSlots[i].sizeY);
297             }
298             for(int i = 0; i < exerciseOptions.length; i++){
299                 if(exerciseOptions[i].posX != exerciseOptions[i].origX &&
                    exerciseOptions[i].posY != exerciseOptions[i].origY && !mouseHold){
300                     exerciseOptions[i].posY-=scrollSensitivity;
301                     exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
302                 }
303             }
304         }
305         else if(optionsCont.getBounds().contains(new Point(mousePos.x-JpanelX,
            mousePos.y-JpanelY)) && exerciseSlots[60].posY > 670){
306             for(int i = 0; i < exerciseOptions.length; i++){
```

```
                    if(exerciseOptions[i].posX == exerciseOptions[i].origX){
                        exerciseOptions[i].posY-=scrollSensitivity;
                        exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                        exerciseOptions[i].posY, exerciseOptions[i].sizeX,
                        exerciseOptions[i].sizeY);
                    }
                    exerciseSlots[i].posY-=scrollSensitivity;
                    exerciseSlots[i].setBounds(exerciseSlots[i].posX,
                    exerciseSlots[i].posY, exerciseSlots[i].sizeX,
                    exerciseSlots[i].sizeY);
                }
            }
            scrollDown = false;
        }

        if(tickCounter%100 == 0){
            saveData();
        }

        tickCounter++;
    }

    public void paint(Graphics g) {
        super.paint(g);
        g.setColor(new Color(255, 165, 205));
        g.fillRect(0, 0, PANEL_WIDTH-400, PANEL_HEIGHT/10);
        g.setColor(new Color(0, 0, 0));
        g.setFont(new Font("SansSerif: ", 1, 28));
        g.drawString("Weekly Schedule", ((PANEL_WIDTH-14)/2)-80,
        ((PANEL_HEIGHT/10)/2)+10);

        g.setColor(new Color(195, 195, 195));
        g.fillRect(0, PANEL_HEIGHT/10, 950, 100);
        g.fillRect(950, PANEL_HEIGHT/10, 430, 70);
        g.setColor(new Color(0, 0, 0));
        g.drawRect(950, PANEL_HEIGHT/10, 430, 70);
        g.drawRect(0, PANEL_HEIGHT/10, 950, 100);
        g.setColor(new Color(0, 0, 0));
        g.setFont(new Font("SansSerif: ", 1, 40));
        g.drawString("Schedule Timeslots", 275, (PANEL_HEIGHT/10)+65);
        g.drawString("Workout Options", 1000, (PANEL_HEIGHT/10)+45);

        if(mouseHold){
            g.setColor(new Color(252, 195, 3));
            g.fillRect(exerciseOptions[labelHeld].posX,
            exerciseOptions[labelHeld].posY, exerciseOptions[labelHeld].sizeX,
            exerciseOptions[labelHeld].sizeY);
            g.setColor(new Color(0, 0, 0));
            g.drawRect(exerciseOptions[labelHeld].posX,
            exerciseOptions[labelHeld].posY, exerciseOptions[labelHeld].sizeX,
            exerciseOptions[labelHeld].sizeY);
            g.setColor(new Color(0, 0, 0));
            g.setFont(new Font("SansSerif: ", 1, 20));
            g.drawString(workouts[labelHeld].name + "- " +
            workouts[labelHeld].muscleGroup,
            exerciseOptions[labelHeld].posX+(exerciseOptions[labelHeld].sizeX/20),
            exerciseOptions[labelHeld].posY+(2*exerciseOptions[labelHeld].sizeY/3));
        }

        if(update){
            refresh();
            update = false;
        }
    }
```

```
359
360     public void actionPerformed(ActionEvent ev){
361         mousePos = MouseInfo.getPointerInfo().getLocation();
362         buttonChecker(ev);
363 
364         if (((String)selectMenu.getSelectedItem()).equals("Daily") || enterDayView){
365             selectMenu.setSelectedItem("Weekly");
366             enterDayView = false;
367             stop();
368         }
369 
370         repaint();
371     }
372 
373     public static void buttonChecker(ActionEvent ev){
374         for(int i = 0; i < daysOfWeek.length; i++){
375             if (ev.getSource() == daysOfWeek[i]){
376                 selectedDay = weekNames[i];
377                 enterDayView = true;
378                 break;
379             }
380         }
381 
382         if(ev.getSource() == sortOptions[0] && scheduleEmpty){
383             for(int i = 0; i < workouts.length-1; i++){
384                 for(int j =0; j<i+1; j++){
385                     if(workouts[i+1-j].name.compareTo(workouts[i-j].name) < 0){
386                         tempWorkout = workouts[i+1-j];
387                         workouts[i+1-j] = workouts[i-j];
388                         workouts[i-j] = tempWorkout;
389                     }
390                 }
391             }
392             update = true;
393         }
394 
395         if(ev.getSource() == sortOptions[1] && scheduleEmpty){
396             for(int i = 0; i < workouts.length-1; i++){
397                 for(int j =0; j<i+1; j++){
398                     if(workouts[i+1-j].muscleGroup.compareTo(workouts[i-j].muscleGroup)
                    < 0){
399                         tempWorkout = workouts[i+1-j];
400                         workouts[i+1-j] = workouts[i-j];
401                         workouts[i-j] = tempWorkout;
402                     }
403                 }
404             }
405             update = true;
406         }
407 
408         if(ev.getSource() == sortOptions[2] && scheduleEmpty){
409             for(int i = 0; i < workouts.length-1; i++){
410                 for(int j =0; j<i+1; j++){
411                     if(workouts[i+1-j].difficulty.compareTo(workouts[i-j].difficulty) <
                    0){
412                         tempWorkout = workouts[i+1-j];
413                         workouts[i+1-j] = workouts[i-j];
414                         workouts[i-j] = tempWorkout;
415                     }
416                 }
417             }
418             update = true;
419         }
420     }
```

```
421
422
423         public void mouseClicked(MouseEvent e) {
424
425         }
426
427         public void mouseExited(MouseEvent e) {
428
429         }
430
431         public void mouseEntered(MouseEvent e) {
432
433         }
434
435         public void mousePressed(MouseEvent e) {
436             mouseLatestX = e.getX();
437             mouseLatestY = e.getY();
438             clip = false;
439
440             for(int i = 0; i < exerciseOptions.length; i++){
441                 if(exerciseOptions[i].getBounds().contains(new Point(mouseLatestX,
                    mouseLatestY))){
442                     mouseHold = true;
443                     exerciseOptions[i].held = true;
444                     labelHeld = i;
445                 }
446             }
447             for(int i = 0; i < dailySlots.length; i++){
448                 for(int j = 0; j <dailySlots[i].length; j++){
449                     if(dailySlots[i][j].getBounds().contains(new Point(mousePos.x-JpanelX,
                        mousePos.y-JpanelY))){
450                         dailySlots[i][j].taken = false;
451                         exerciseOptions[labelHeld].inUse = false;
452                         dailySlots[i][j].holding = -1;
453                     }
454                 }
455             }
456         }
457
458         public void mouseReleased(MouseEvent e) {
459             if(!clip && mouseHold){
460                 exerciseOptions[labelHeld].posX = exerciseSlots[labelHeld].posX+20;
461                 exerciseOptions[labelHeld].posY = exerciseSlots[labelHeld].posY+5;
462                 exerciseOptions[labelHeld].setBounds(exerciseOptions[labelHeld].posX,
                    exerciseOptions[labelHeld].posY, exerciseOptions[labelHeld].origSizeX,
                    exerciseOptions[labelHeld].origSizeY);
463             }
464
465             scheduleEmpty = true;
466             for(int i = 0; i < exerciseOptions.length; i++){
467                 exerciseOptions[i].held = false;
468                 if(exerciseOptions[i].inUse){
469                     scheduleEmpty = false;
470                 }
471             }
472             for(int i = 0; i < dailySlots.length; i++){
473                 for(int j = 0; j <dailySlots[i].length; j++){
474                     if(dailySlots[i][j].getBounds().contains(new Point(mousePos.x-JpanelX,
                        mousePos.y-JpanelY)) && mouseHold){
475                         dailySlots[i][j].taken = true;
476                         exerciseOptions[labelHeld].inUse = true;
477                         dailySlots[i][j].holding = workouts[labelHeld].num;
478                     }
479                 }
```

```
480             }
481             mouseHold = false;
482         }
483 
484         public void mouseWheelMoved(MouseWheelEvent e) {
485             if(e.getWheelRotation() < 0){
486                 scrollUp = true;
487             }
488             else if(e.getWheelRotation() > 0){
489                 scrollDown = true;
490             }
491         }
492 
493         public void loadData(){
494             try {
495                 dataScan = new Scanner(new File("ScheduleData.txt"));
496             }
497             catch (Exception error) {
498                 System.out.println("An error occurred:" + error);
499                 error.printStackTrace();
500             }
501             accountCounter = 0;
502             allData = "";
503             currentLine = "";
504             accountFound = false;
505             while(dataScan.hasNextLine()){
506                 currentLine = dataScan.nextLine();
507                 allData += currentLine + "%";
508                 tempData = new ArrayList<>(Arrays.asList(currentLine.split(", ")));
509                 if(user.equals(tempData.get(0))){
510                     accountFound = true;
511                     accountNum = accountCounter;
512                     scheduleData = tempData;
513                 }
514                 accountCounter++;
515             }
516             allDataArr = allData.split("%");
517             paintLoad = true;
518             dataScan.close();
519         }
520 
521         public void saveData(){
522             tempLine = "";
523             try {
524                 tempLine += user;
525 
526                 for(int i = 0; i < dailySlots.length; i++){
527                     for(int j = 0; j <dailySlots[i].length; j++){
528                         if(dailySlots[i][j].taken){
529                             tempLine += (", " + i + ", " + j + ", " +
                                dailySlots[i][j].holding);
530                         }
531                     }
532                 }
533                 allDataArr[accountNum] = tempLine;
534 
535                 dataWriter = new FileWriter("ScheduleData.txt", false);
536                 BufferedWriter bDW = new BufferedWriter(dataWriter);
537                 for(int i = 0; i < allDataArr.length; i++){
538                     bDW.write(allDataArr[i]);
539                     bDW.newLine();
540                 }
541 
542                 bDW.close();
```

```
543             }
544             catch (IOException error) {
545                 System.out.println("An error occurred:" + error);
546                 error.printStackTrace();
547             }
548         }
549 
550         public void refresh(){
551             this.remove(selectMenu);
552             this.remove(menuBar);
553 
554             for(int i = 0; i < daysOfWeek.length; i++){
555                 this.remove(daysOfWeek[i]);
556             }
557 
558             for(int i = 0; i < sortOptions.length; i++){
559                 this.remove(sortOptions[i]);
560             }
561 
562             for(int i = 0; i < exerciseOptions.length; i++){
563                 this.remove(exerciseOptions[i]);
564                 this.remove(exerciseSlots[i]);
565             }
566 
567             for(int i = 0; i < dailySlots.length; i++){
568                 for(int j = 0; j <dailySlots[i].length; j++){
569                     this.remove(dailySlots[i][j]);
570                 }
571             }
572 
573             counter = 0;
574             for(double i = 8; counter < timeSlots.length; i+=0.25){
575                 this.remove(timeSlots[counter]);
576                 counter++;
577             }
578             this.remove(slotsCont);
579             this.remove(optionsCont);
580 
581             this.add(selectMenu);
582             this.add(menuBar);
583 
584             for(int i = 0; i < daysOfWeek.length; i++){
585                 this.add(daysOfWeek[i]);
586             }
587 
588             for(int i = 0; i < sortOptions.length; i++){
589                 this.add(sortOptions[i]);
590             }
591 
592             for(int i = 0; i < exerciseOptions.length; i++){
593                 exerciseOptions[i].setBounds(exerciseOptions[i].posX,
                    exerciseOptions[i].posY, exerciseOptions[i].sizeX, exerciseOptions[i].sizeY);
594                 exerciseOptions[i].setOpaque(true);
595                 if(workouts[i].difficulty.equals("Beginner")){
596                     exerciseOptions[i].setBackground(new Color(205, 255, 205));
597                 }
598                 else if(workouts[i].difficulty.equals("Experienced")){
599                     exerciseOptions[i].setBackground(new Color(255, 255, 205));
600                 }
601                 else if(workouts[i].difficulty.equals("Expert")){
602                     exerciseOptions[i].setBackground(new Color(255, 205, 205));
603                 }
604                 exerciseOptions[i].setBorder(BorderFactory.createLineBorder((new
                    Color(0,0,0)), 2));
```

```
605                 exerciseOptions[i].setFont(new Font("SansSerif", 1, 18));
606                 exerciseOptions[i].setText(" " + workouts[i].name + "- " +
                    workouts[i].muscleGroup);
607                 this.add(exerciseOptions[i]);
608                 exerciseSlots[i].setBounds(exerciseSlots[i].posX, exerciseSlots[i].posY,
                    exerciseSlots[i].sizeX, exerciseSlots[i].sizeY);
609                 exerciseSlots[i].setOpaque(true);
610                 if(workouts[i].difficulty.equals("Beginner")){
611                     exerciseSlots[i].setBackground(new Color(135, 205, 135, 85));
612                 }
613                 else if(workouts[i].difficulty.equals("Experienced")){
614                     exerciseSlots[i].setBackground(new Color(205, 205, 135, 85));
615                 }
616                 else if(workouts[i].difficulty.equals("Expert")){
617                     exerciseSlots[i].setBackground(new Color(205, 135, 135, 85));
618                 }
619                 exerciseSlots[i].setBorder(BorderFactory.createLineBorder((new
                    Color(0,0,0)), 1));
620                 this.add(exerciseSlots[i]);
621             }
622 
623             for(int i = 0; i < dailySlots.length; i++){
624                 for(int j = 0; j <dailySlots[i].length; j++){
625                     this.add(dailySlots[i][j]);
626                 }
627             }
628 
629             counter = 0;
630             for(double i = 8; counter < timeSlots.length; i+=0.25){
631                 this.add(timeSlots[counter]);
632                 counter++;
633             }
634             this.add(slotsCont);
635             this.add(optionsCont);
636         }
637     }
```

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Account{
    private String user, pass, gender;
    private int height, weight, age;

    public Account(){
        this("null", "null", 0, 0, 0, "");
    }

    public Account(String user, String pass, int height, int weight, int age, String
    gender){
        this.user = user;
        this.pass = pass;
        this.height = height;
        this.weight = weight;
        this.age = age;
        this.gender = gender;
    }

    public void setUser(String user){
        this.user = user;
    }

    public void setPass(String pass){
        this.pass = pass;
    }

    public void setGender(String gender){
        this.gender = gender;
    }

    public void setHeight(int height){
        this.height = height;
    }

    public void setWeight(int weight){
        this.weight = weight;
    }

    public void setAge(int age){
        this.age = age;
    }

    public String getUser(){
        return user;
    }

    public String getPass(){
        return pass;
    }

    public String getGender(){
        return gender;
    }

    public int getHeight(){
        return height;
    }

    public int getWeight(){
        return weight;
```

```
64        }
65
66        public int getAge(){
67            return age;
68        }
69    }
```

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class SlotsAndOptions extends JLabel{
    public int posX, posY, sizeX, sizeY, origX, origY, origSizeX, origSizeY, holding;
    public boolean held = false, taken = false, inUse = false;

    public SlotsAndOptions(){
        this(0, 0, false, false, false, 0, 0, 0, 0, 0, 0, -1);
    }

    public SlotsAndOptions(int posX, int posY, int sizeX, int sizeY){
        this(posX, posY, false, false, false, sizeX, sizeY, posX, posY, sizeX, sizeY,
        -1);
    }

    public SlotsAndOptions(int posX, int posY, boolean held, boolean taken, boolean
    inUse, int sizeX, int sizeY, int origX, int origY, int origSizeX, int origSizeY,
    int holding){
        this.posX = posX;
        this.posY = posY;
        this.held = held;
        this.taken = taken;
        this.inUse = inUse;
        this.sizeX = sizeX;
        this.sizeY = sizeY;
        this.origX = posX;
        this.origY = posY;
        this.origSizeX = sizeX;
        this.origSizeY = sizeY;
        this.holding = holding;
    }

}
```

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Workouts{
    public String name, muscleGroup, difficulty, description;
    public int num;

    public Workouts(){
        this("", "", "", "", 0);
    }

    public Workouts(String name, String muscleGroup, String difficulty, String
    description, int num){
        this.name = name;
        this.muscleGroup = muscleGroup;
        this.difficulty = difficulty;
        this.description = description;
        this.num = num;
    }
}
```