Appendix A: Initial Consultation with the Client

I asked Mr. Hammond multiple questions about the software to best design it.

Question: What is the problem you are experiencing that led to the idea of using a computerized system rather than a manual system?

Mr. Hammond: The current system requires a lot of workforces that are hard to find because of the life-threatening situation due to the dinosaurs at the park. It is also very inefficient and unsafe for a manual locking system at a dinosaur park. Therefore, a computerized system would allow my employees and me to work safely and efficiently. Do you think you can create software for my park?

Me: Yes! I will be willing to help you with the situation. So, what does the park look like? Can you please give me a detailed description of the doors and the building with their relevant location?

Mr. Hammond: The park has 5 buildings, and each building has 10 doors. I want the system to mimic the buildings and when you enter a building, you should see the 10 doors.

I want there to be a security system as well, where it only lets the employee access the main program if and only if they are registered inside a User Database. I should also be able to register a new employee if I want into the database.

The way the doors can be unlocked or locked should be through a textbox entry and not just a click from the mouse, because I do not want there to be an accidental locking or unlocking of the doors.

While Mr. Hammond was talking, I made a rough 5-minute prototype of the product Mr. Hammond was looking for (Prototypes can be seen in Appendix B)

Me: Here is the rough prototype you requested! I made it while you described the park. Tell me what you think of this prototype.

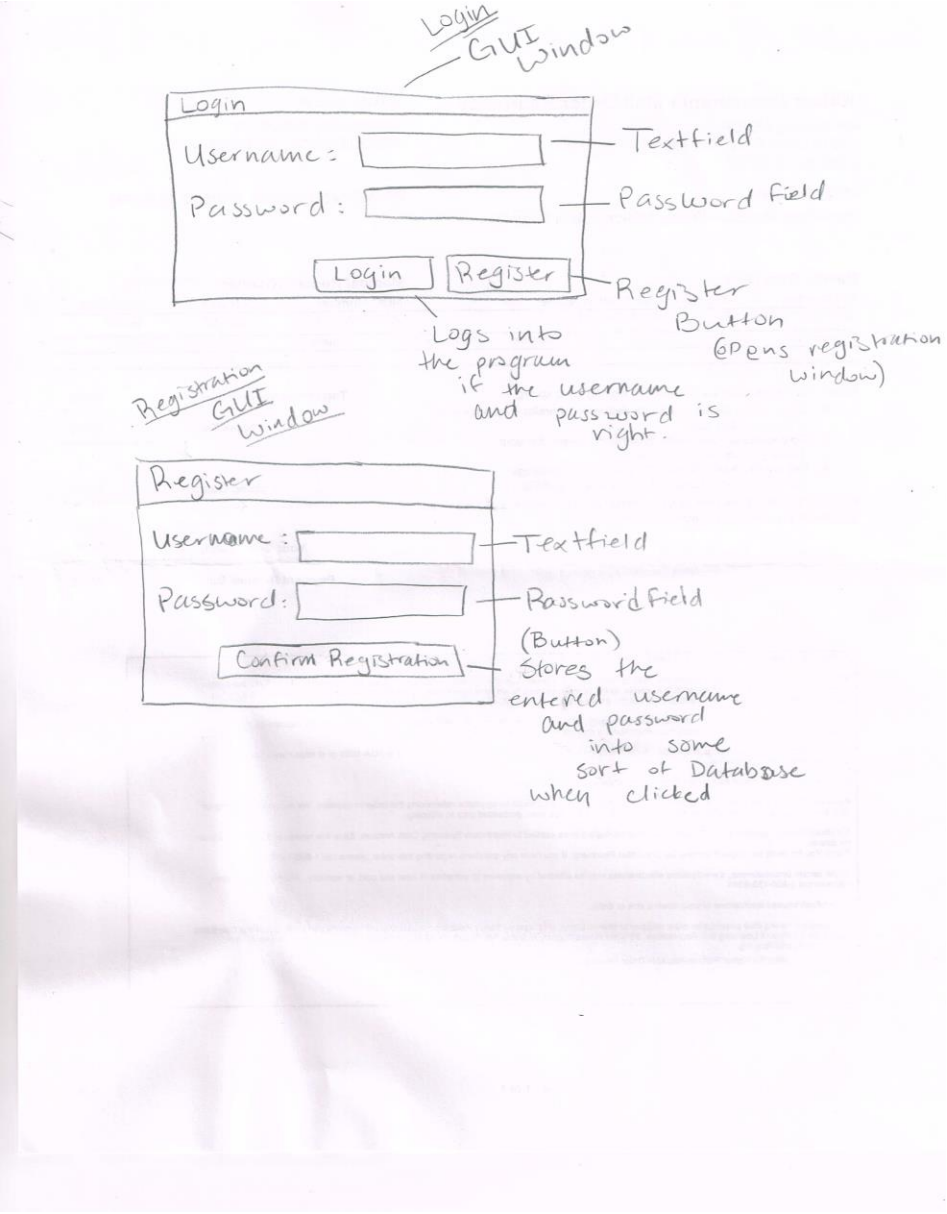
Mr. Hammond: The prototype works great and is exactly what I am looking for. However, it is a bit bland, and needs to have the colors of my company which is red and black. The software should have a touch of the company branding as well for marketing purposes. I want you to create an Icon for the software, that is the same as the park logo.

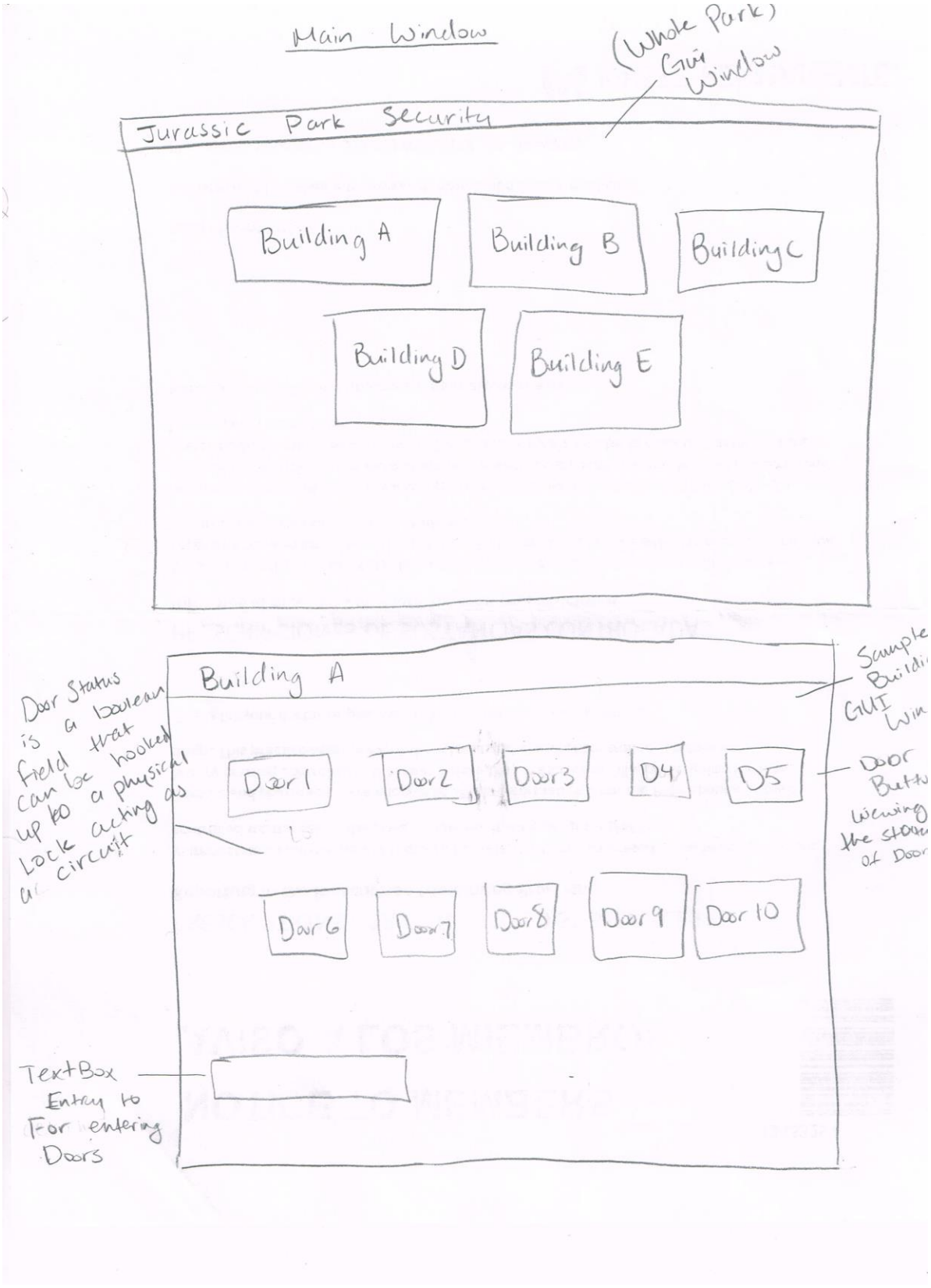
I realized that I would have to do some corrective maintenance to the product as well

Me: Noted! Anything else you would like me to develop?

Mr. Hammond: No, that would be all, thank you!

Appendix B: Prototypes




Main Window
(Whole Park) Gui Window
Jurassic Park Security
Building A
Building B
Building C
Building D
Building E
Building A
Door Status is a boolean field that can be hooked up to a physical Lock acting as a circuit
Door 1
Door 2
Door 3
D4
D5
Door 6
Door 7
Door 8
Door 9
Door 10
TextBox Entry to For entering Doors

Appendix C: Demonstration Video Script

This is a video demonstrating the product I developed for Mr. John Hammond.

As you can see it is a runnable application that is installed on the desktop.

So, let's run it and see the result.

There we go, there is the login window.

So, let us go ahead and register a new employee here.

Click register.

And enter the username and the password

Click confirm registration

And it says, the user is registered, and we need to restart the program to sign in.

Now assuming it registered (Which we will test later) this **checks off the 4th success criterion.**

So, let us restart it.

Now before we login with the user we created, lets enter a random username and password to see if it will let us login to the system or not

And “user or password did not match.”

So, as you can see, it is not letting everyone in and checking if the username and password is registered in the database or not.

Now let’s login to the system with the username and password we created.

Notice how the password is hidden unlike the text, for protection.

Click login.

And there pops up the main window with the title Jurassic Park Security

That checks the first 2 success criteria. Because it only lets us into the system with a valid username and password. It also checks off the live database registration part because as you can see it stores to the live database and updates

But lets say the user forgot their password. Well they can click on the forgot password link right here to create a new one. Lets click it.

So it is asking us for a badge number which is initialized when a new user registers and cannot be changed because it is a unique employee id.

So lets enter our employee id and reset out password

So my new password is going to be “newPassword”

Now if they don’t match then it will tell me they don’t match.

Lets click reset.

And we login again

And here we see that it works

This checks off the 4th success criteria

Now you can see there are 5 buildings titled A. B, C, D, and finally E.

Let's click different buildings to see happens.

Building A

And it shows us the 10 different doors in building A

There is the 6th success criterion checked off because there is a graphical user interface

Lets go inside a building

Now there are 10 buttons. Each button displays the door status when clicked.

So, let us click them to see if they work

And all of them are locked by default.

Now notice in the bottom left corner, there is a text box where we enter the door number we want to lock and unlock

So, let us enter a1 because we are inside building a and the door number, we want to unlock is 1.

As you can see, door 1 has been changed.

Now this is not just a printed statement saying locked or unlocked.

Each door is a Boolean variable of the doors class where it is representing a circuit of each door that is true or false meaning unlocked and locked.

So, each variable can be attached to a physical door lock, where it locks and unlocks the door based on its status.

It is a similar system in all of the buildings.

Let’s doors throughout the park to see if they are functional.

There we go, all of the doors are working fine.

Now let me enter a wrong input.

And as you can see, there is an error message that says the door you entered does not exist.

And when you do enter the right door number with the right building of course, the error message goes away, and it locks/unlocks the entered door number.

That was the 6th success criterion.

For the **Last criterion**, we have all of these error messages telling us that what is wrong so that success criterion is also checked off.

So, let's exist. And log back in. The username and password still work because they are stored inside a database.

And there we have the final success criterion met, wrapping up the program demonstration.

Appendix D: Source Code

Doors Class:

```
public class Doors {


private static boolean[] building1;

private static boolean[] building2;

private static boolean[] building3;

private static boolean[] building4;

private static boolean[] building5;

private static boolean doorStatus;


public Doors(int buildingNumber) { // default constructor - creates a building array for all of the buildings

// with all doors set to false (Locked) by default

if (buildingNumber == 1) { // building 1 initializer

building1 = new boolean[10];

for (int d = 0; d < building1.length; d++) {

building1[d] = false;

}

} else if (buildingNumber == 2) { // building 2 initializer

building2 = new boolean[10];

for (int d = 0; d < building2.length; d++) {
```

```
building2[d] = false;
}
} else if (buildingNumber == 3) { // building 3 initializer
building3 = new boolean[10];
for (int d = 0; d < building3.length; d++) {
building3[d] = false;
}
} else if (buildingNumber == 4) { // building 4 initializer
building4 = new boolean[10];
for (int d = 0; d < building4.length; d++) {
building4[d] = false;
}
} else if (buildingNumber == 5) { // building 4 initializer
building5 = new boolean[10];
for (int d = 0; d < building5.length; d++) {
building5[d] = false;
}
}

}// Buildings Setup ends

public static boolean getDoorStatus(int buildingNumber, int doorNumber) { // accesses the status of the door
doorNumber--;
if (buildingNumber == 1) {
doorStatus = building1[doorNumber];
} else if (buildingNumber == 2) {
doorStatus = building2[doorNumber];
} else if (buildingNumber == 3) {
```

```
doorStatus = building3[doorNumber];
} else if (buildingNumber == 4) {
doorStatus = building4[doorNumber];
} else if (buildingNumber == 5) {
doorStatus = building5[doorNumber];
}
return doorStatus;
}

public static boolean setDoorStatus(int buildingNumber, int doorNumber) { // Locks or unlocks the entered door
doorNumber--; // only 10 elements in the building array, therefore subtracted the doorNumber
// to avoid
// out of bounds error and match with the array indices

if (buildingNumber == 1) {
if (building1[doorNumber] == false) {
building1[doorNumber] = true;
doorStatus = building1[doorNumber];
} else if (building1[doorNumber] == true) {
building1[doorNumber] = false;
doorStatus = building1[doorNumber];
}
} else if (buildingNumber == 2) {
if (building2[doorNumber] == false) {
building2[doorNumber] = true;
doorStatus = building2[doorNumber];
} else if (building2[doorNumber] == true) {
building2[doorNumber] = false;
```

```
doorStatus = building2[doorNumber];
}
} else if (buildingNumber == 3) {
if (building3[doorNumber] == false) {
building3[doorNumber] = true;
doorStatus = building3[doorNumber];
} else if (building3[doorNumber] == true) {
building3[doorNumber] = false;
doorStatus = building3[doorNumber];
}
} else if (buildingNumber == 4) {
if (building4[doorNumber] == false) {
building4[doorNumber] = true;
doorStatus = building4[doorNumber];
} else if (building4[doorNumber] == true) {
building4[doorNumber] = false;
doorStatus = building4[doorNumber];
}
} else if (buildingNumber == 5) {
if (building5[doorNumber] == false) {
building5[doorNumber] = true;
doorStatus = building5[doorNumber];
} else if (building5[doorNumber] == true) {
building5[doorNumber] = false;
doorStatus = building5[doorNumber];
}
}
return doorStatus;
}
```

```
}
```

User Class:

```
public class User {

  private final String username;
  private String password;
  private Badge employeeBadge;

  // default constructor creates a user object with the parameters of 2 strings
```

```
  public User(String u, String p) {
    username = u;
    password = p;

    Badge possibleBadge = new Badge();

    for(int i = 0; i < Login.allCreatedBadges.size(); i++) {
      if(Login.allCreatedBadges.equals(possibleBadge)) {
        possibleBadge = new Badge();
      }
    }

    employeeBadge = possibleBadge;

  }

  public boolean equals(User u) {
    if (username.equals(u.getUsername()) && password.equals(u.getPassword())) {
      return true;
    }
    return false;
  }

  public String getUsername() { // Can't change the user set username therefore no setUser
method added
    return username;
  }

  public String getPassword() {
```

```
    return password;
  }

  public Badge getEmployeeBadge() {
    return employeeBadge;
  }

  public void setPassword(String p) {
    password = p;
  }

  public void setEmployeeBadge(Badge eB) {
    employeeBadge = eB;
  }

  public String toString() {
    return username + " " + password;
  }

}
```

Badge Class:

```
public class Badge {

    private final String badgeNum;

    public Badge() {
```

```
        badgeNum = assignEmployeeBadge();
    }

    public Badge(String s) {
        badgeNum = s;
    }

    private String assignEmployeeBadge() {
double random = (int)(999999 * Math.random() + 100000);
String id = random + "";
return id;
}

    public String getBadgeNum() {
        return badgeNum;
    }

    public boolean equals(Badge bN) {
        if(bN.getBadgeNum().equals(badgeNum)) {
            return true;
        }
        return false;
    }

    public String toString() {
        return badgeNum + "";
    }
}
```

Login Class:

```
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.PasswordAuthentication;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.Map;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import org.json.JSONArray;
import org.json.JSONObject;

public class Login extends JFrame {
```

```
private static HttpURLConnection connection;

private static ArrayList<User> Database = new ArrayList<>(); // creates an arrayList of users
static ArrayList<Badge> allCreatedBadges = new ArrayList<>();
private static JFrame frame;
private static JPanel panel;
private static JLabel userLabel;
private static JLabel idLabel;
private static JLabel passResetLabel;
private static JTextField userText;
private static JTextField idText;
private static JLabel passwordLabel;
private static JPasswordField passwordText;
private static JPasswordField rPass1;
private static JPasswordField rPass2;
private static JButton loginButton;
private static JButton registerButton;
private static JButton confirmRegistration;
private static JButton forgotPassword;
private static JButton checkBadge;
private static JButton reset;
private static JLabel dialogueBox;
private static JLabel dialogueBox1;

public static void main(String[] args) {

// Database reader, where it reads the database from the api endpoint
BufferedReader reader;
String line;
```

```
StringBuffer responseContent = new StringBuffer();

try {
URL url = new URL("https://sheet.best/api/sheets/2e0a3a65-f899-49f3-9477-780c8f6521fe");
connection = (HttpURLConnection) url.openConnection();

connection.setRequestMethod("GET");
connection.setConnectTimeout(5000);
connection.setReadTimeout(5000);

int status = connection.getResponseCode();

if (status > 299) {
reader = new BufferedReader(new InputStreamReader(connection.getErrorStream()));
while ((line = reader.readLine()) != null) {
responseContent.append(line);
}
reader.close();
} else {
reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
while ((line = reader.readLine()) != null) {
responseContent.append(line);
}
reader.close();
}
} catch (MalformedURLException e) {
dialogueBox.setText("Database not connected!");
} catch (IOException e) {
dialogueBox.setText("Database does not contain any users!");
```

```
} finally {
connection.disconnect();
}

parse(responseContent.toString());

panel = new JPanel();
panel.setLayout(null);
panel.setBackground(Color.black);

frame = new JFrame();// creates a frame panel
frame.setTitle("Login"); // sets the title of the gui window
frame.setResizable(true);
frame.setSize(420, 200); // sets the size of the gui window
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // closes the application when "x" is clicked
frame.add(panel); // add panel to frame

// appearance settings
ImageIcon parkLogo = new ImageIcon("parkLogo.png"); // imports icon image from project file and stores it in
// parkLogo
frame.setIconImage(parkLogo.getImage()); // gets the image from parkLogo and sets it to the window icon
frame.getContentPane().setBackground(Color.black); // sets the background color to black

// creates a username label
userLabel = new JLabel("Username: "); // creates the username label for the login
userLabel.setForeground(Color.white);
userLabel.setBounds(10, 20, 80, 25);
```

```
panel.add(userLabel); // adds the label to the panel so it displays

// creates a text box to enter the username
userText = new JTextField(20);
userText.setBounds(100, 20, 165, 25);
panel.add(userText);

// creates a username label
passwordLabel = new JLabel("Password: "); // creates the password label for the login
passwordLabel.setForeground(Color.white);
passwordLabel.setBounds(10, 50, 80, 25);// sets dimensions
panel.add(passwordLabel);

// creates a text box to enter the password (length of 20)
passwordText = new JPasswordField(20);
passwordText.setBounds(100, 50, 165, 25);
panel.add(passwordText);

// creates a login button
loginButton = new JButton("Login");
loginButton.setBounds(10, 100, 80, 25);
loginButton.setFocusable(false);
loginButton.setBackground(Color.red);
loginButton.addActionListener(new LoginHandler());
panel.add(loginButton);

// creates a register button
registerButton = new JButton("Register");
registerButton.setBounds(100, 100, 100, 25);
```

```
registerButton.setFocusable(false);
registerButton.setBackground(Color.red);
registerButton.addActionListener(new LoginHandler());
panel.add(registerButton);

// creates a dialogue label which returns the status of the application to the
// user to direct them on what to do
dialogueBox = new JLabel("");
dialogueBox.setBounds(10, 110, 300, 35);
dialogueBox.setForeground(Color.white);
panel.add(dialogueBox);

// create a new button that calls the forgotPassword method
forgotPassword = new JButton("I Forgot My Password...");
forgotPassword.setForeground(Color.blue);
forgotPassword.setBounds(10, 80, 180, 15);
forgotPassword.setBackground(Color.black);
forgotPassword.setBorder(null);
forgotPassword.addActionListener(new LoginHandler());
panel.add(forgotPassword);

frame.setVisible(true); // makes the frame visible

}

public static void createVerificationFrame() {
panel = new JPanel();
panel.setLayout(null);
panel.setBackground(Color.black);
```

```
frame = new JFrame();// creates a frame panel

frame.setTitle("Badge Verification"); // sets the title of the gui window

frame.setResizable(true);

frame.setSize(420, 200); // sets the size of the gui window

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // closes the application when "x" is clicked

frame.add(panel); // add panel to frame

// appearance settings

ImageIcon parkLogo = new ImageIcon("parkLogo.png"); // imports icon image from project file and stores it in pa

frame.setIconImage(parkLogo.getImage()); // gets the image from parkLogo and sets it to the window icon

frame.getContentPane().setBackground(Color.black); // sets the background color to black


idLabel = new JLabel("Enter your Badge #"); // creates the username label for the login

idLabel.setForeground(Color.white);

idLabel.setBounds(10, 20, 120, 25);

panel.add(idLabel); // adds the label to the panel so it displays


// creates a text box to enter the username

idText = new JTextField(20);

idText.setBounds(150, 20, 165, 25);

panel.add(idText);


checkBadge = new JButton();

checkBadge = new JButton("Verify");

checkBadge.setBounds(100, 80, 200, 25);

checkBadge.setFocusable(false);

checkBadge.setBackground(Color.red);
```

```
checkBadge.addActionListener(new LoginHandler());
panel.add(checkBadge);

dialogueBox = new JLabel("");
dialogueBox.setBounds(10, 110, 350, 25);
dialogueBox.setForeground(Color.white);
panel.add(dialogueBox);

frame.setVisible(true);
}

public static void createRegisterFrame() {
panel = new JPanel();
panel.setLayout(null);
panel.setBackground(Color.black);

frame = new JFrame();// creates a frame panel
frame.setTitle("Registration"); // sets the title of the gui window
frame.setResizable(true);
frame.setSize(420, 200); // sets the size of the gui window
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // closes the application when "x" is clicked
frame.add(panel); // add panel to frame

// appearance settings
ImageIcon parkLogo = new ImageIcon("parkLogo.png"); // imports icon image from project file and stores it in
// parkLogo
frame.setIconImage(parkLogo.getImage()); // gets the image from parkLogo and sets it to the window icon
```

```
frame.getContentPane().setBackground(Color.black); // sets the background color to black

// creates a username label
userLabel = new JLabel("Username: "); // creates the username label for the login
userLabel.setForeground(Color.white);
userLabel.setBounds(10, 20, 80, 25);
panel.add(userLabel); // adds the label to the panel so it displays

// creates a text box to enter the username
userText = new JTextField(20);
userText.setBounds(100, 20, 165, 25);
panel.add(userText);

// creates a username label
passwordLabel = new JLabel("Password: "); // creates the password label for the login
passwordLabel.setForeground(Color.white);
passwordLabel.setBounds(10, 50, 80, 25);// sets dimensions
panel.add(passwordLabel);

// creates a text box to enter the password (length of 20)
passwordText = new JPasswordField(20);
passwordText.setBounds(100, 50, 165, 25);
panel.add(passwordText);

confirmRegistration = new JButton("Confirm Registration");
confirmRegistration.setBounds(100, 80, 200, 25);
confirmRegistration.setFocusable(false);
confirmRegistration.setBackground(Color.red);
confirmRegistration.addActionListener(new LoginHandler());
```

```
panel.add(confirmRegistration);

dialogueBox = new JLabel("");
dialogueBox.setBounds(10, 110, 350, 25);
dialogueBox.setForeground(Color.white);
panel.add(dialogueBox);

frame.setVisible(true); // makes the frame visible
}

public static void createPasswordResetFrame() {

panel = new JPanel();
panel.setLayout(null);
panel.setBackground(Color.black);

frame = new JFrame();// creates a frame panel
frame.setTitle("Create a New Password"); // sets the title of the gui window
frame.setResizable(true);
frame.setSize(420, 220); // sets the size of the gui window
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // closes the application when "x" is clicked
frame.add(panel); // add panel to frame

// appearance settings
ImageIcon parkLogo = new ImageIcon("parkLogo.png"); // imports icon image from project file and stores it in

frame.setIconImage(parkLogo.getImage()); // gets the image from parkLogo and sets it to the window icon
```

```
frame.getContentPane().setBackground(Color.black); // sets the background color to black


passResetLabel = new JLabel("Enter a New Password"); // creates the username label for the login
passResetLabel.setForeground(Color.white);
passResetLabel.setBounds(70, 20, 200, 25);
panel.add(passResetLabel); // adds the label to the panel so it displays

// creates a text box to enter the password (length of 20)
rPass1 = new JPasswordField(20);
rPass1.setBounds(70, 45, 165, 25);
panel.add(rPass1);

passResetLabel = new JLabel("Enter it Again"); // creates the username label for the login
passResetLabel.setForeground(Color.white);
passResetLabel.setBounds(70, 80, 200, 25);
panel.add(passResetLabel); // adds the label to the panel so it displays

// creates a text box to enter the password (length of 20)
rPass2 = new JPasswordField(20);
rPass2.setBounds(70, 110, 165, 25);
panel.add(rPass2);

// creates a register button
reset = new JButton("Reset");
reset.setBounds(250, 110, 100, 25);
reset.setFocusable(false);
reset.setBackground(Color.red);
reset.addActionListener(new LoginHandler());
```

```
panel.add(reset);

dialogueBox1 = new JLabel("");
dialogueBox1.setBounds(30, 140, 350, 25);
dialogueBox1.setForeground(Color.white);
panel.add(dialogueBox1);



frame.setVisible(true);
}

private static class LoginHandler implements ActionListener {

public void actionPerformed(ActionEvent e) {

// different methods of what the different buttons do when they are clicked

if (e.getSource() == loginButton) { // e.getSource determines which button was clicked
String user = userText.getText();
String password = passwordText.getText(); // getting the password field content

User existingEmployee = new User(user, password);
allCreatedBadges.add(existingEmployee.getEmployeeBadge());

boolean result = LoginSuccess(existingEmployee);
if (result) {
frame.dispose();
JurassicParkSecurity program = new JurassicParkSecurity();
} else
```

```
dialogueBox.setText("Login or Password did not Match");
}

else if (e.getSource() == registerButton) {
frame.dispose();
createRegisterFrame();

}

else if (e.getSource() == confirmRegistration) {

String user = userText.getText();
String password = passwordText.getText();

User newEmployee = new User(user, password);
allCreatedBadges.add(newEmployee.getEmployeeBadge());

boolean registered = Register(newEmployee);

if (registered) {
dialogueBox.setText("User registered! Close this Window and restart to login!");
// frame.dispose();
// printDatabase();
} else {
dialogueBox.setText("User did not register, please try again!");
}
}

else if (e.getSource() == forgotPassword) {
```

```
frame.dispose();
createVerificationFrame();
}

else if (e.getSource() == checkBadge) {

Badge badgeNum = new Badge(idText.getText());

for(Badge b : allCreatedBadges) {

if (b.equals(badgeNum)) {
frame.dispose();
createPasswordResetFrame();
} else
dialogueBox.setText("The Badge Number Didn't Match!");

}
}

else if (e.getSource() == reset) {

Badge badgeNum = new Badge(idText.getText());
String newPass1 = rPass1.getText();
String newPass2 = rPass2.getText();

if(newPass1.equals(newPass2)) {
try {
if(resetPassword(badgeNum, newPass2)) {
dialogueBox1.setText("Password Updated, Please close this window and log back in.");
```

```
} else { dialogueBox1.setText("The Password could not be updated!"); }
} catch (IOException e1) {
e1.printStackTrace();
}
} else { dialogueBox1.setText("The entered passwords do not match!"); }

}
}// actionPerformed Method ends

}// Class LoginHandler ends

public static boolean LoginSuccess(User employee) { // login method to confirm if the login was successful or not

boolean result = false;

for (int u = 0; u < Database.size(); u++) {
if (Database.get(u).equals(employee)) {
result = true;
}
}
return result;
}

public static boolean Register(User newEmployee) {

boolean result = false;

Database.add(newEmployee);
```

```
post(newEmployee);

for (int u = 0; u < Database.size(); u++) {
if (Database.get(u).equals(newEmployee)) {
result = true;
}
}
return result;
}

public static boolean resetPassword(Badge badge, String pass) throws IOException {

boolean result = false;

for (int i = 0; i < Database.size(); i++) {
Badge existingBadge = Database.get(i).getEmployeeBadge();
if (badge.equals(existingBadge)) {
Database.get(i).setPassword(pass);
result = true;
User user = Database.get(i);
patch(user, pass, i);
}
}

return result;
}

public static void printDatabase() {
```

```
for (int i = 0; i < Database.size(); i++) {
System.out.println(Database.get(i));
}
}

public static void parse(String responseBody) {

JSONArray users = new JSONArray(responseBody);

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

JSONObject user = users.getJSONObject(i);

if (user != null) {

String username = user.getString("Username");
String password = user.getString("Password");
Badge badgeNum = new Badge(user.getString("Badge"));

User person = new User(username, password);
person.setEmployeeBadge(badgeNum);
allCreatedBadges.add(badgeNum);

Database.add(person);
}
}

// printDatabase();
}
```

```
public static void post(User person) {
try {
String user = person.getUsername();
String pass = person.getPassword();
Badge id = person.getEmployeeBadge();

URL url = new URL("https://sheet.best/api/sheets/2e0a3a65-f899-49f3-9477-780c8f6521fe");

Map<String, Object> params = new LinkedHashMap<>();
params.put("Username", user);
params.put("Password", pass);
params.put("Badge", id);

StringBuilder dataPost = new StringBuilder();
for (Map.Entry<String, Object> param : params.entrySet()) {
if (dataPost.length() != 0)
dataPost.append('&');
dataPost.append(URLEncoder.encode(param.getKey(), "UTF-8"));
dataPost.append('=');
dataPost.append(URLEncoder.encode(String.valueOf(param.getValue()), "UTF-8"));
}
byte[] dataPostBytes = dataPost.toString().getBytes("UTF-8");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.setRequestMethod("POST");
connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
connection.setRequestProperty("Content-Length", String.valueOf(dataPostBytes.length));
connection.setDoOutput(true);
connection.getOutputStream().write(dataPostBytes);
```

```
Reader in = new BufferedReader(new InputStreamReader(connection.getInputStream(), "UTF-8"));

} catch (MalformedURLException e) {
dialogueBox.setText("Cannot connect to the database");
} catch (IOException e) {
dialogueBox.setText("The User was not resgistered!");
} finally {
connection.disconnect();
}
}

public static void patch(User user, String pass, int i) throws IOException {
try {
Badge badge = user.getEmployeeBadge();
String username = user.getUsername();
String id = badge.toString();

URL url = new URL("https://sheet.best/api/sheets/2e0a3a65-f899-49f3-9477-780c8f6521fe");

Map<String, Object> params = new LinkedHashMap<>();
params.put("Username", username);
params.put("Password", pass);
params.put("Badge", id);

Badge dataBadge = new Badge(params.get("Badge").toString());
if(dataBadge.equals(badge)) {
params.put("Password", pass);
```

```
}

StringBuilder dataPost = new StringBuilder();
for (Map.Entry<String, Object> param : params.entrySet()) {
if (dataPost.length() != 0)
dataPost.append('&');
dataPost.append(URLEncoder.encode(param.getKey(), "UTF-8"));
dataPost.append('=');
dataPost.append(URLEncoder.encode(String.valueOf(param.getValue()), "UTF-8"));
}
byte[] dataPostBytes = dataPost.toString().getBytes("UTF-8");
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.setRequestMethod("POST");
connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
connection.setRequestProperty("Content-Length", String.valueOf(dataPostBytes.length));
connection.setDoOutput(true);
connection.getOutputStream().write(dataPostBytes);

Reader in = new BufferedReader(new InputStreamReader(connection.getInputStream(), "UTF-
8"));

} catch (MalformedURLException e) {
dialogueBox.setText("Cannot connect to the database");
} catch (IOException e) {
dialogueBox.setText("The User was not resgistered!");
} finally {
connection.disconnect();
}
```

```
URL url2 = new URL("https://sheet.best/api/sheets/2e0a3a65-f899-49f3-9477-780c8f6521fe/" +
i);

HttpURLConnection http = (HttpURLConnection) url2.openConnection();

http.setRequestMethod("DELETE");

http.setRequestProperty("Accept", "*/*");

http.setRequestProperty("Authorization", "Bearer
mt0dgHmLJMVQhvjpNXDyA83vA_PxH23Y");


System.out.println(http.getResponseCode() + " " + http.getResponseMessage());

http.disconnect();


}

}
```

Jurassic Park Security Class:

```
import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextField;


public class JurassicParkSecurity extends JFrame {


  private static ImageIcon parkLogo = new ImageIcon("parkLogo.png"); //imports icon image
from project file and stores it in parkLogo
```

```
  //JPanels
  private static JPanel panel = new JPanel();
  private static JPanel bP1 = new JPanel();
  private static JPanel bP2 = new JPanel();
  private static JPanel bP3 = new JPanel();  //creates 5 JPanels for each building bP = building Panel
  private static JPanel bP4 = new JPanel();
  private static JPanel bP5 = new JPanel();

  //JFrames
  private static JFrame frame = new JFrame();// creates a main frame for the park
  private static JFrame bF1 = new JFrame();
  private static JFrame bF2 = new JFrame();   // create 5 frames for each building bF = building Frame
  private static JFrame bF3= new JFrame();
  private static JFrame bF4 = new JFrame();
  private static JFrame bF5 = new JFrame();

  //Building buttons
  private static JButton building1;
  private static JButton building2;
  private static JButton building3;
  private static JButton building4;
  private static JButton building5;

  //door buttons
  private static JButton door1;
  private static JButton door2;
  private static JButton door3;
```

```
private static JButton door4;
private static JButton door5;
private static JButton door6;
private static JButton door7;
private static JButton door8;
private static JButton door9;
private static JButton door10;
private static JButton door11;
private static JButton door12;
private static JButton door13;
private static JButton door14;
private static JButton door15;
private static JButton door16;
private static JButton door17;
private static JButton door18;
private static JButton door19;
private static JButton door20;
private static JButton door21;
private static JButton door22;
private static JButton door23;
private static JButton door24;
private static JButton door25;
private static JButton door26;
private static JButton door27;
private static JButton door28;
private static JButton door29;
private static JButton door30;
private static JButton door31;
private static JButton door32;
```

```
private static JButton door33;
private static JButton door34;
private static JButton door35;
private static JButton door36;
private static JButton door37;
private static JButton door38;
private static JButton door39;
private static JButton door40;
private static JButton door41;
private static JButton door42;
private static JButton door43;
private static JButton door44;
private static JButton door45;
private static JButton door46;
private static JButton door47;
private static JButton door48;
private static JButton door49;
private static JButton door50;

//"Go Back" Buttons
private static JButton back1;
private static JButton back2;
private static JButton back3;
private static JButton back4;
private static JButton back5;

//Door status labels
private static JLabel door1Label;
private static JLabel door2Label;
```

```
private static JLabel door3Label;
private static JLabel door4Label;
private static JLabel door5Label;
private static JLabel door6Label;
private static JLabel door7Label;
private static JLabel door8Label;
private static JLabel door9Label;
private static JLabel door10Label;
private static JLabel door11Label;
private static JLabel door12Label;
private static JLabel door13Label;
private static JLabel door14Label;
private static JLabel door15Label;
private static JLabel door16Label;
private static JLabel door17Label;
private static JLabel door18Label;
private static JLabel door19Label;
private static JLabel door20Label;
private static JLabel door21Label;
private static JLabel door22Label;
private static JLabel door23Label;
private static JLabel door24Label;
private static JLabel door25Label;
private static JLabel door26Label;
private static JLabel door27Label;
private static JLabel door28Label;
private static JLabel door29Label;
private static JLabel door30Label;
private static JLabel door31Label;
```

```
private static JLabel door32Label;
private static JLabel door33Label;
private static JLabel door34Label;
private static JLabel door35Label;
private static JLabel door36Label;
private static JLabel door37Label;
private static JLabel door38Label;
private static JLabel door39Label;
private static JLabel door40Label;
private static JLabel door41Label;
private static JLabel door42Label;
private static JLabel door43Label;
private static JLabel door44Label;
private static JLabel door45Label;
private static JLabel door46Label;
private static JLabel door47Label;
private static JLabel door48Label;
private static JLabel door49Label;
private static JLabel door50Label;

//Instruction Text Field
private static JTextField instruction1;
private static JTextField instruction2;
private static JTextField instruction3;
private static JTextField instruction4;
private static JTextField instruction5;

//Instruction Labels
private static JLabel instruction1Label;
```

```
  private static JLabel instruction2Label;
  private static JLabel instruction3Label;
  private static JLabel instruction4Label;
  private static JLabel instruction5Label;

  //Run buttons
  private static JButton run1;
  private static JButton run2;
  private static JButton run3;
  private static JButton run4;
  private static JButton run5;

  //Error Box Labels
  private static JLabel errorBox1;
  private static JLabel errorBox2;
  private static JLabel errorBox3;
  private static JLabel errorBox4;
  private static JLabel errorBox5;

public JurassicParkSecurity() {

Doors buildingA = new Doors(1);
Doors buildingB = new Doors(2);
Doors buildingC = new Doors(3);
Doors buildingD = new Doors(4);
Doors buildingE = new Doors(5);

// whole park setup all 5 buildings
panel.setLayout(null);
```

```
    panel.setBackground(Color.black);


frame.setTitle("Jurassic Park Security"); //sets the title of the gui window

frame.setResizable(true);

frame.setSize(1280, 800); // sets the size of the gui window

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //closes the application when "x" is clicked

frame.add(panel); // add panel to frame


//appearance settings

frame.setIconImage(parkLogo.getImage()); //gets the image from parkLogo and sets it to the window icon

frame.getContentPane().setBackground(Color.black);  //sets the background color to black


building1 = new JButton("Building A");

building1.setBounds(300, 150, 200, 150);

building1.setFocusable(false);

building1.setBackground(Color.red);

building1.setFont(new Font("Comic Sans", Font.BOLD,18));

building1.addActionListener(new MainHandler());

    panel.add(building1);


building2 = new JButton("Building B");

building2.setBounds(540, 150, 200, 150);

building2.setFocusable(false);

building2.setBackground(Color.red);

building2.setFont(new Font("Comic Sans", Font.BOLD,18));

building2.addActionListener(new MainHandler());

    panel.add(building2);
```

```
building3 = new JButton("Building C");
building3.setBounds(780, 150, 200, 150);
building3.setFocusable(false);
building3.setBackground(Color.red);
building3.setFont(new Font("Comic Sans", Font.BOLD,18));
building3.addActionListener(new MainHandler());
   panel.add(building3);

building4 = new JButton("Building D");
building4.setBounds(420, 350, 200, 150);
building4.setFocusable(false);
building4.setBackground(Color.red);
building4.setFont(new Font("Comic Sans", Font.BOLD,18));
building4.addActionListener(new MainHandler());
   panel.add(building4);

building5 = new JButton("Building E");
building5.setBounds(660, 350, 200, 150);
building5.setFocusable(false);
building5.setBackground(Color.red);
building5.setFont(new Font("Comic Sans", Font.BOLD,18));
building5.addActionListener(new MainHandler());
   panel.add(building5);

frame.setVisible(true); //makes the frame visible
//Park Setup ends

bP1.setLayout(null);
bP1.setBackground(Color.black);
```

```
    bF1.setTitle("Building A");
bF1.setResizable(true);
bF1.setSize(1280, 800);
bF1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
bF1.setIconImage(parkLogo.getImage());
bF1.getContentPane().setBackground(Color.black);
bF1.add(bP1);

// adding doors
door1 = new JButton("Door 1");
door1.setBounds(200, 100, 100, 100);
door1.setFocusable(false);
door1.setBackground(Color.red);
door1.setFont(new Font("Comic Sans", Font.BOLD,18));
door1.addActionListener(new MainHandler());
bP1.add(door1);

door1Label = new JLabel("");
door1Label.setBounds(200, 250, 350, 25);
door1Label.setForeground(Color.white);
bP1.add(door1Label);

door2 = new JButton("Door 2");
door2.setBounds(400, 100, 100, 100);
door2.setFocusable(false);
door2.setBackground(Color.red);
door2.setFont(new Font("Comic Sans", Font.BOLD,18));
door2.addActionListener(new MainHandler());
```

```
bP1.add(door2);

door2Label = new JLabel("");
door2Label.setBounds(400, 250, 350, 25);
door2Label.setForeground(Color.white);
bP1.add(door2Label);

door3 = new JButton("Door 3");
door3.setBounds(600, 100, 100, 100);
door3.setFocusable(false);
door3.setBackground(Color.red);
door3.setFont(new Font("Comic Sans", Font.BOLD,18));
door3.addActionListener(new MainHandler());
bP1.add(door3);

door3Label = new JLabel("");
door3Label.setBounds(600, 250, 350, 25);
door3Label.setForeground(Color.white);
bP1.add(door3Label);

door4 = new JButton("Door 4");
door4.setBounds(800, 100, 100, 100);
door4.setFocusable(false);
door4.setBackground(Color.red);
door4.setFont(new Font("Comic Sans", Font.BOLD,18));
door4.addActionListener(new MainHandler());
bP1.add(door4);

door4Label = new JLabel("");
```

```
door4Label.setBounds(800, 250, 350, 25);
door4Label.setForeground(Color.white);
bP1.add(door4Label);

door5 = new JButton("Door 5");
door5.setBounds(1000, 100, 100, 100);
door5.setFocusable(false);
door5.setBackground(Color.red);
door5.setFont(new Font("Comic Sans", Font.BOLD,18));
door5.addActionListener(new MainHandler());
bP1.add(door5);

door5Label = new JLabel("");
door5Label.setBounds(1000, 250, 350, 25);
door5Label.setForeground(Color.white);
bP1.add(door5Label);

door6 = new JButton("Door 6");
door6.setBounds(200, 400, 100, 100);
door6.setFocusable(false);
door6.setBackground(Color.red);
door6.setFont(new Font("Comic Sans", Font.BOLD,18));
door6.addActionListener(new MainHandler());
bP1.add(door6);

door6Label = new JLabel("");
door6Label.setBounds(200, 550, 350, 25);
door6Label.setForeground(Color.white);
bP1.add(door6Label);
```

```
door7 = new JButton("Door 7");
door7.setBounds(400, 400, 100, 100);
door7.setFocusable(false);
door7.setBackground(Color.red);
door7.setFont(new Font("Comic Sans", Font.BOLD,18));
door7.addActionListener(new MainHandler());
bP1.add(door7);

door7Label = new JLabel("");
door7Label.setBounds(400, 550, 350, 25);
door7Label.setForeground(Color.white);
bP1.add(door7Label);

door8 = new JButton("Door 8");
door8.setBounds(600, 400, 100, 100);
door8.setFocusable(false);
door8.setBackground(Color.red);
door8.setFont(new Font("Comic Sans", Font.BOLD,18));
door8.addActionListener(new MainHandler());
bP1.add(door8);

door8Label = new JLabel("");
door8Label.setBounds(600, 550, 350, 25);
door8Label.setForeground(Color.white);
bP1.add(door8Label);

door9 = new JButton("Door 9");
door9.setBounds(800, 400, 100, 100);
```

```
door9.setFocusable(false);
door9.setBackground(Color.red);
door9.setFont(new Font("Comic Sans", Font.BOLD,18));
door9.addActionListener(new MainHandler());
   bP1.add(door9);

door9Label = new JLabel("");
door9Label.setBounds(800, 550, 350, 25);
door9Label.setForeground(Color.white);
bP1.add(door9Label);

door10 = new JButton("Door 10");
door10.setBounds(1000, 400, 110, 100);
door10.setFocusable(false);
door10.setBackground(Color.red);
door10.setFont(new Font("Comic Sans", Font.BOLD,18));
door10.addActionListener(new MainHandler());
bP1.add(door10);

door10Label = new JLabel("");
door10Label.setBounds(1000, 550, 350, 25);
door10Label.setForeground(Color.white);
bP1.add(door10Label);
//Building 1 setup ends

bP2.setLayout(null);
bP2.setBackground(Color.black);

   bF2.setTitle("Building B");
```

```
bF2.setResizable(true);
bF2.setSize(1280, 800);
bF2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
bF2.setIconImage(parkLogo.getImage());
bF2.getContentPane().setBackground(Color.black);
bF2.add(bP2);

door11 = new JButton("Door 1");
door11.setBounds(200, 100, 100, 100);
door11.setFocusable(false);
door11.setBackground(Color.red);
door11.setFont(new Font("Comic Sans", Font.BOLD,18));
door11.addActionListener(new MainHandler());
bP2.add(door11);

door11Label = new JLabel("");
door11Label.setBounds(200, 250, 350, 25);
door11Label.setForeground(Color.white);
bP2.add(door11Label);

door12 = new JButton("Door 2");
door12.setBounds(400, 100, 100, 100);
door12.setFocusable(false);
door12.setBackground(Color.red);
door12.setFont(new Font("Comic Sans", Font.BOLD,18));
door12.addActionListener(new MainHandler());
bP2.add(door12);

door12Label = new JLabel("");
```

```
door12Label.setBounds(400, 250, 350, 25);
door12Label.setForeground(Color.white);
bP2.add(door12Label);

door13 = new JButton("Door 3");
door13.setBounds(600, 100, 100, 100);
door13.setFocusable(false);
door13.setBackground(Color.red);
door13.setFont(new Font("Comic Sans", Font.BOLD,18));
door13.addActionListener(new MainHandler());
bP2.add(door13);

door13Label = new JLabel("");
door13Label.setBounds(600, 250, 350, 25);
door13Label.setForeground(Color.white);
bP2.add(door13Label);

door14 = new JButton("Door 4");
door14.setBounds(800, 100, 100, 100);
door14.setFocusable(false);
door14.setBackground(Color.red);
door14.setFont(new Font("Comic Sans", Font.BOLD,18));
door14.addActionListener(new MainHandler());
bP2.add(door14);

door14Label = new JLabel("");
door14Label.setBounds(800, 250, 350, 25);
door14Label.setForeground(Color.white);
bP2.add(door14Label);
```

```
door15 = new JButton("Door 5");
door15.setBounds(1000, 100, 100, 100);
door15.setFocusable(false);
door15.setBackground(Color.red);
door15.setFont(new Font("Comic Sans", Font.BOLD,18));
door15.addActionListener(new MainHandler());
bP2.add(door15);

door15Label = new JLabel("");
door15Label.setBounds(1000, 250, 350, 25);
door15Label.setForeground(Color.white);
bP2.add(door15Label);

door16 = new JButton("Door 6");
door16.setBounds(200, 400, 100, 100);
door16.setFocusable(false);
door16.setBackground(Color.red);
door16.setFont(new Font("Comic Sans", Font.BOLD,18));
door16.addActionListener(new MainHandler());
bP2.add(door16);

door16Label = new JLabel("");
door16Label.setBounds(200, 550, 350, 25);
door16Label.setForeground(Color.white);
bP2.add(door16Label);

door17 = new JButton("Door 7");
door17.setBounds(400, 400, 100, 100);
```

```
door17.setFocusable(false);
door17.setBackground(Color.red);
door17.setFont(new Font("Comic Sans", Font.BOLD,18));
door17.addActionListener(new MainHandler());
bP2.add(door17);

door17Label = new JLabel("");
door17Label.setBounds(400, 550, 350, 25);
door17Label.setForeground(Color.white);
bP2.add(door17Label);

door18 = new JButton("Door 8");
door18.setBounds(600, 400, 100, 100);
door18.setFocusable(false);
door18.setBackground(Color.red);
door18.setFont(new Font("Comic Sans", Font.BOLD,18));
door18.addActionListener(new MainHandler());
bP2.add(door18);

door18Label = new JLabel("");
door18Label.setBounds(600, 550, 350, 25);
door18Label.setForeground(Color.white);
bP2.add(door18Label);

door19 = new JButton("Door 9");
door19.setBounds(800, 400, 100, 100);
door19.setFocusable(false);
door19.setBackground(Color.red);
door19.setFont(new Font("Comic Sans", Font.BOLD,18));
```

```
door19.addActionListener(new MainHandler());
    bP2.add(door19);

door19Label = new JLabel("");
door19Label.setBounds(800, 550, 350, 25);
door19Label.setForeground(Color.white);
bP2.add(door19Label);

door20 = new JButton("Door 10");
door20.setBounds(1000, 400, 110, 100);
door20.setFocusable(false);
door20.setBackground(Color.red);
door20.setFont(new Font("Comic Sans", Font.BOLD,18));
door20.addActionListener(new MainHandler());
bP2.add(door20);

door20Label = new JLabel("");
door20Label.setBounds(1000, 550, 350, 25);
door20Label.setForeground(Color.white);
bP2.add(door20Label);
    //building 2 setup ends

bP3.setLayout(null);
bP3.setBackground(Color.black);

    bF3.setTitle("Building C");
bF3.setResizable(true);
bF3.setSize(1280, 800);
bF3.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
bF3.setIconImage(parkLogo.getImage());
bF3.getContentPane().setBackground(Color.black);
bF3.add(bP3);

door21 = new JButton("Door 1");
door21.setBounds(200, 100, 100, 100);
door21.setFocusable(false);
door21.setBackground(Color.red);
door21.setFont(new Font("Comic Sans", Font.BOLD,18));
door21.addActionListener(new MainHandler());
bP3.add(door21);

door21Label = new JLabel("");
door21Label.setBounds(200, 250, 350, 25);
door21Label.setForeground(Color.white);
bP3.add(door21Label);

door22 = new JButton("Door 2");
door22.setBounds(400, 100, 100, 100);
door22.setFocusable(false);
door22.setBackground(Color.red);
door22.setFont(new Font("Comic Sans", Font.BOLD,18));
door22.addActionListener(new MainHandler());
bP3.add(door22);

door22Label = new JLabel("");
door22Label.setBounds(400, 250, 350, 25);
door22Label.setForeground(Color.white);
bP3.add(door22Label);
```

```
door23 = new JButton("Door 3");
door23.setBounds(600, 100, 100, 100);
door23.setFocusable(false);
door23.setBackground(Color.red);
door23.setFont(new Font("Comic Sans", Font.BOLD,18));
door23.addActionListener(new MainHandler());
bP3.add(door23);

door23Label = new JLabel("");
door23Label.setBounds(600, 250, 350, 25);
door23Label.setForeground(Color.white);
bP3.add(door23Label);

door24 = new JButton("Door 4");
door24.setBounds(800, 100, 100, 100);
door24.setFocusable(false);
door24.setBackground(Color.red);
door24.setFont(new Font("Comic Sans", Font.BOLD,18));
door24.addActionListener(new MainHandler());
bP3.add(door24);

door24Label = new JLabel("");
door24Label.setBounds(800, 250, 350, 25);
door24Label.setForeground(Color.white);
bP3.add(door24Label);

door25 = new JButton("Door 5");
door25.setBounds(1000, 100, 100, 100);
```

```
door25.setFocusable(false);
door25.setBackground(Color.red);
door25.setFont(new Font("Comic Sans", Font.BOLD,18));
door25.addActionListener(new MainHandler());
bP3.add(door25);

door25Label = new JLabel("");
door25Label.setBounds(1000, 250, 350, 25);
door25Label.setForeground(Color.white);
bP3.add(door25Label);

door26 = new JButton("Door 6");
door26.setBounds(200, 400, 100, 100);
door26.setFocusable(false);
door26.setBackground(Color.red);
door26.setFont(new Font("Comic Sans", Font.BOLD,18));
door26.addActionListener(new MainHandler());
bP3.add(door26);

door26Label = new JLabel("");
door26Label.setBounds(200, 550, 350, 25);
door26Label.setForeground(Color.white);
bP3.add(door26Label);

door27 = new JButton("Door 7");
door27.setBounds(400, 400, 100, 100);
door27.setFocusable(false);
door27.setBackground(Color.red);
door27.setFont(new Font("Comic Sans", Font.BOLD,18));
```

```
door27.addActionListener(new MainHandler());
bP3.add(door27);

door27Label = new JLabel("");
door27Label.setBounds(400, 550, 350, 25);
door27Label.setForeground(Color.white);
bP3.add(door27Label);

door28 = new JButton("Door 8");
door28.setBounds(600, 400, 100, 100);
door28.setFocusable(false);
door28.setBackground(Color.red);
door28.setFont(new Font("Comic Sans", Font.BOLD,18));
door28.addActionListener(new MainHandler());
bP3.add(door28);

door28Label = new JLabel("");
door28Label.setBounds(600, 550, 350, 25);
door28Label.setForeground(Color.white);
bP3.add(door28Label);

door29 = new JButton("Door 9");
door29.setBounds(800, 400, 100, 100);
door29.setFocusable(false);
door29.setBackground(Color.red);
door29.setFont(new Font("Comic Sans", Font.BOLD,18));
door29.addActionListener(new MainHandler());
   bP3.add(door29);
```

```
door29Label = new JLabel("");
door29Label.setBounds(800, 550, 350, 25);
door29Label.setForeground(Color.white);
bP3.add(door29Label);

door30 = new JButton("Door 10");
door30.setBounds(1000, 400, 110, 100);
door30.setFocusable(false);
door30.setBackground(Color.red);
door30.setFont(new Font("Comic Sans", Font.BOLD,18));
door30.addActionListener(new MainHandler());
bP3.add(door30);

door30Label = new JLabel("");
door30Label.setBounds(1000, 550, 350, 25);
door30Label.setForeground(Color.white);
bP3.add(door30Label);
   // building 3 setup ends

bP4.setLayout(null);
bP4.setBackground(Color.black);

   bF4.setTitle("Building D");
bF4.setResizable(true);
bF4.setSize(1280, 800);
bF4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
bF4.setIconImage(parkLogo.getImage());
bF4.getContentPane().setBackground(Color.black);
bF4.add(bP4);
```

```
door31 = new JButton("Door 1");
door31.setBounds(200, 100, 100, 100);
door31.setFocusable(false);
door31.setBackground(Color.red);
door31.setFont(new Font("Comic Sans", Font.BOLD,18));
door31.addActionListener(new MainHandler());
bP4.add(door31);

door31Label = new JLabel("");
door31Label.setBounds(200, 250, 350, 25);
door31Label.setForeground(Color.white);
bP4.add(door31Label);

door32 = new JButton("Door 2");
door32.setBounds(400, 100, 100, 100);
door32.setFocusable(false);
door32.setBackground(Color.red);
door32.setFont(new Font("Comic Sans", Font.BOLD,18));
door32.addActionListener(new MainHandler());
bP4.add(door32);

door32Label = new JLabel("");
door32Label.setBounds(400, 250, 350, 25);
door32Label.setForeground(Color.white);
bP4.add(door32Label);

door33 = new JButton("Door 3");
door33.setBounds(600, 100, 100, 100);
```

```
door33.setFocusable(false);
door33.setBackground(Color.red);
door33.setFont(new Font("Comic Sans", Font.BOLD,18));
door33.addActionListener(new MainHandler());
bP4.add(door33);

door33Label = new JLabel("");
door33Label.setBounds(600, 250, 350, 25);
door33Label.setForeground(Color.white);
bP4.add(door33Label);

door34 = new JButton("Door 4");
door34.setBounds(800, 100, 100, 100);
door34.setFocusable(false);
door34.setBackground(Color.red);
door34.setFont(new Font("Comic Sans", Font.BOLD,18));
door34.addActionListener(new MainHandler());
bP4.add(door34);

door34Label = new JLabel("");
door34Label.setBounds(800, 250, 350, 25);
door34Label.setForeground(Color.white);
bP4.add(door34Label);

door35 = new JButton("Door 5");
door35.setBounds(1000, 100, 100, 100);
door35.setFocusable(false);
door35.setBackground(Color.red);
door35.setFont(new Font("Comic Sans", Font.BOLD,18));
```

```
door35.addActionListener(new MainHandler());
bP4.add(door35);

door35Label = new JLabel("");
door35Label.setBounds(1000, 250, 350, 25);
door35Label.setForeground(Color.white);
bP4.add(door35Label);

door36 = new JButton("Door 6");
door36.setBounds(200, 400, 100, 100);
door36.setFocusable(false);
door36.setBackground(Color.red);
door36.setFont(new Font("Comic Sans", Font.BOLD,18));
door36.addActionListener(new MainHandler());
bP4.add(door36);

door36Label = new JLabel("");
door36Label.setBounds(200, 550, 350, 25);
door36Label.setForeground(Color.white);
bP4.add(door36Label);

door37 = new JButton("Door 7");
door37.setBounds(400, 400, 100, 100);
door37.setFocusable(false);
door37.setBackground(Color.red);
door37.setFont(new Font("Comic Sans", Font.BOLD,18));
door37.addActionListener(new MainHandler());
bP4.add(door37);
```

```
door37Label = new JLabel("");
door37Label.setBounds(400, 550, 350, 25);
door37Label.setForeground(Color.white);
bP4.add(door37Label);

door38 = new JButton("Door 8");
door38.setBounds(600, 400, 100, 100);
door38.setFocusable(false);
door38.setBackground(Color.red);
door38.setFont(new Font("Comic Sans", Font.BOLD,18));
door38.addActionListener(new MainHandler());
bP4.add(door38);

door38Label = new JLabel("");
door38Label.setBounds(600, 550, 350, 25);
door38Label.setForeground(Color.white);
bP4.add(door38Label);

door39 = new JButton("Door 9");
door39.setBounds(800, 400, 100, 100);
door39.setFocusable(false);
door39.setBackground(Color.red);
door39.setFont(new Font("Comic Sans", Font.BOLD,18));
door39.addActionListener(new MainHandler());
   bP4.add(door39);

door39Label = new JLabel("");
door39Label.setBounds(800, 550, 350, 25);
door39Label.setForeground(Color.white);
```

```
bP4.add(door39Label);

door40 = new JButton("Door 10");
door40.setBounds(1000, 400, 110, 100);
door40.setFocusable(false);
door40.setBackground(Color.red);
door40.setFont(new Font("Comic Sans", Font.BOLD,18));
door40.addActionListener(new MainHandler());
bP4.add(door40);

door40Label = new JLabel("");
door40Label.setBounds(1000, 550, 350, 25);
door40Label.setForeground(Color.white);
bP4.add(door40Label);
// building 4 setup ends

bP5.setLayout(null);
bP5.setBackground(Color.black);

   bF5.setTitle("Building E");
bF5.setResizable(true);
bF5.setSize(1280, 800);
bF5.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
bF5.setIconImage(parkLogo.getImage());
bF5.getContentPane().setBackground(Color.black);
bF5.add(bP5);

door41 = new JButton("Door 1");
door41.setBounds(200, 100, 100, 100);
```

```
door41.setFocusable(false);
door41.setBackground(Color.red);
door41.setFont(new Font("Comic Sans", Font.BOLD,18));
door41.addActionListener(new MainHandler());
bP5.add(door41);

door41Label = new JLabel("");
door41Label.setBounds(200, 250, 350, 25);
door41Label.setForeground(Color.white);
bP5.add(door41Label);

door42 = new JButton("Door 2");
door42.setBounds(400, 100, 100, 100);
door42.setFocusable(false);
door42.setBackground(Color.red);
door42.setFont(new Font("Comic Sans", Font.BOLD,18));
door42.addActionListener(new MainHandler());
bP5.add(door42);

door42Label = new JLabel("");
door42Label.setBounds(400, 250, 350, 25);
door42Label.setForeground(Color.white);
bP5.add(door42Label);

door43 = new JButton("Door 3");
door43.setBounds(600, 100, 100, 100);
door43.setFocusable(false);
door43.setBackground(Color.red);
door43.setFont(new Font("Comic Sans", Font.BOLD,18));
```

```
door43.addActionListener(new MainHandler());
bP5.add(door43);

door43Label = new JLabel("");
door43Label.setBounds(600, 250, 350, 25);
door43Label.setForeground(Color.white);
bP5.add(door43Label);

door44 = new JButton("Door 4");
door44.setBounds(800, 100, 100, 100);
door44.setFocusable(false);
door44.setBackground(Color.red);
door44.setFont(new Font("Comic Sans", Font.BOLD,18));
door44.addActionListener(new MainHandler());
bP5.add(door44);

door44Label = new JLabel("");
door44Label.setBounds(800, 250, 350, 25);
door44Label.setForeground(Color.white);
bP5.add(door44Label);

door45 = new JButton("Door 5");
door45.setBounds(1000, 100, 100, 100);
door45.setFocusable(false);
door45.setBackground(Color.red);
door45.setFont(new Font("Comic Sans", Font.BOLD,18));
door45.addActionListener(new MainHandler());
bP5.add(door45);
```

```
door45Label = new JLabel("");
door45Label.setBounds(1000, 250, 350, 25);
door45Label.setForeground(Color.white);
bP5.add(door45Label);

door46 = new JButton("Door 6");
door46.setBounds(200, 400, 100, 100);
door46.setFocusable(false);
door46.setBackground(Color.red);
door46.setFont(new Font("Comic Sans", Font.BOLD,18));
door46.addActionListener(new MainHandler());
bP5.add(door46);

door46Label = new JLabel("");
door46Label.setBounds(200, 550, 350, 25);
door46Label.setForeground(Color.white);
bP5.add(door46Label);

door47 = new JButton("Door 7");
door47.setBounds(400, 400, 100, 100);
door47.setFocusable(false);
door47.setBackground(Color.red);
door47.setFont(new Font("Comic Sans", Font.BOLD,18));
door47.addActionListener(new MainHandler());
bP5.add(door47);

door47Label = new JLabel("");
door47Label.setBounds(400, 550, 350, 25);
door47Label.setForeground(Color.white);
```

```
bP5.add(door47Label);

door48 = new JButton("Door 8");
door48.setBounds(600, 400, 100, 100);
door48.setFocusable(false);
door48.setBackground(Color.red);
door48.setFont(new Font("Comic Sans", Font.BOLD,18));
door48.addActionListener(new MainHandler());
bP5.add(door48);

door48Label = new JLabel("");
door48Label.setBounds(600, 550, 350, 25);
door48Label.setForeground(Color.white);
bP5.add(door48Label);

door49 = new JButton("Door 9");
door49.setBounds(800, 400, 100, 100);
door49.setFocusable(false);
door49.setBackground(Color.red);
door49.setFont(new Font("Comic Sans", Font.BOLD,18));
door49.addActionListener(new MainHandler());
   bP5.add(door49);

door49Label = new JLabel("");
door49Label.setBounds(800, 550, 350, 25);
door49Label.setForeground(Color.white);
bP5.add(door49Label);

door50 = new JButton("Door 10");
```

```
door50.setBounds(1000, 400, 110, 100);
door50.setFocusable(false);
door50.setBackground(Color.red);
door50.setFont(new Font("Comic Sans", Font.BOLD,18));
door50.addActionListener(new MainHandler());
bP5.add(door50);

door50Label = new JLabel("");
door50Label.setBounds(1000, 550, 350, 25);
door50Label.setForeground(Color.white);
bP5.add(door50Label);
//building 5 setup ends

    back1 = new JButton("Go Back");
    back1.setBounds(1100, 600, 120, 100);
    back1.setFocusable(false);
    back1.setBackground(Color.red);
    back1.setFont(new Font("Comic Sans", Font.BOLD,18));
    back1.addActionListener(new MainHandler());
    bP1.add(back1);

    back2 = new JButton("Go Back");
    back2.setBounds(1100, 600, 120, 100);
    back2.setFocusable(false);
    back2.setBackground(Color.red);
    back2.setFont(new Font("Comic Sans", Font.BOLD,18));
    back2.addActionListener(new MainHandler());
    bP2.add(back2);
```

```
    back3 = new JButton("Go Back");
    back3.setBounds(1100, 600, 120, 100);
    back3.setFocusable(false);
    back3.setBackground(Color.red);
    back3.setFont(new Font("Comic Sans", Font.BOLD,18));
    back3.addActionListener(new MainHandler());
    bP3.add(back3);

    back4 = new JButton("Go Back");
    back4.setBounds(1100, 600, 120, 100);
    back4.setFocusable(false);
    back4.setBackground(Color.red);
    back4.setFont(new Font("Comic Sans", Font.BOLD,18));
    back4.addActionListener(new MainHandler());
    bP4.add(back4);

    back5 = new JButton("Go Back");
    back5.setBounds(1100, 600, 120, 100);
    back5.setFocusable(false);
    back5.setBackground(Color.red);
    back5.setFont(new Font("Comic Sans", Font.BOLD,18));
    back5.addActionListener(new MainHandler());
    bP5.add(back5);
    //back buttons end

    //Instruction textfields and labels
instruction1 = new JTextField(20);
instruction1.setBounds(100, 650, 180, 30);
bP1.add(instruction1);
```

```
    instruction1Label = new JLabel("Enter the Door you want to Lock or Unlock");
    instruction1Label.setBounds(100, 620, 300, 35);
    instruction1Label.setForeground(Color.white);
    bP1.add(instruction1Label);

instruction2 = new JTextField(20);
instruction2.setBounds(100, 650, 180, 30);
bP2.add(instruction2);

    instruction2Label = new JLabel("Enter the Door you want to Lock or Unlock");
    instruction2Label.setBounds(100, 620, 300, 35);
    instruction2Label.setForeground(Color.white);
    bP2.add(instruction2Label);

instruction3 = new JTextField(20);
instruction3.setBounds(100, 650, 180, 30);
bP3.add(instruction3);

    instruction3Label = new JLabel("Enter the Door you want to Lock or Unlock");
    instruction3Label.setBounds(100, 620, 300, 35);
    instruction3Label.setForeground(Color.white);
    bP3.add(instruction3Label);

instruction4 = new JTextField(20);
instruction4.setBounds(100, 650, 180, 30);
bP4.add(instruction4);

    instruction4Label = new JLabel("Enter the Door you want to Lock or Unlock");
```

```
    instruction4Label.setBounds(100, 620, 300, 35);
    instruction4Label.setForeground(Color.white);
    bP4.add(instruction4Label);

instruction5 = new JTextField(20);
instruction5.setBounds(100, 650, 180, 30);
bP5.add(instruction5);

    instruction5Label = new JLabel("Enter the Door you want to Lock or Unlock");
    instruction5Label.setBounds(100, 620, 300, 35);
    instruction5Label.setForeground(Color.white);
    bP5.add(instruction5Label);

    //Run Buttons
run1 = new JButton("Lock/Unlock");
run1.setBounds(100, 685, 120, 25);
run1.setFocusable(false);
run1.setBackground(Color.red);
run1.addActionListener(new MainHandler());
    bP1.add(run1);

run2 = new JButton("Lock/Unlock");
run2.setBounds(100, 685, 120, 25);
run2.setFocusable(false);
run2.setBackground(Color.red);
run2.addActionListener(new MainHandler());
    bP2.add(run2);

run3 = new JButton("Lock/Unlock");
```

```
run3.setBounds(100, 685, 120, 25);
run3.setFocusable(false);
run3.setBackground(Color.red);
run3.addActionListener(new MainHandler());
   bP3.add(run3);

run4 = new JButton("Lock/Unlock");
run4.setBounds(100, 685, 120, 25);
run4.setFocusable(false);
run4.setBackground(Color.red);
run4.addActionListener(new MainHandler());
   bP4.add(run4);

run5 = new JButton("Lock/Unlock");
run5.setBounds(100, 685, 120, 25);
run5.setFocusable(false);
run5.setBackground(Color.red);
run5.addActionListener(new MainHandler());
   bP5.add(run5);

   //Error Boxes
   errorBox1 = new JLabel("");
   errorBox1.setBounds(100, 710, 300, 35);
   errorBox1.setForeground(Color.white);
   bP1.add(errorBox1);

   errorBox2 = new JLabel("");
   errorBox2.setBounds(100, 710, 300, 35);
   errorBox2.setForeground(Color.white);
```

```
    bP2.add(errorBox2);

    errorBox3 = new JLabel("");
    errorBox3.setBounds(100, 710, 300, 35);
    errorBox3.setForeground(Color.white);
    bP3.add(errorBox3);

    errorBox4 = new JLabel("");
    errorBox4.setBounds(100, 710, 300, 35);
    errorBox4.setForeground(Color.white);
    bP4.add(errorBox4);

    errorBox5 = new JLabel("");
    errorBox5.setBounds(100, 710, 300, 35);
    errorBox5.setForeground(Color.white);
    bP5.add(errorBox5);
  }

  class MainHandler implements ActionListener{
    public void actionPerformed(ActionEvent e) {

  // building buttons
  if(e.getSource() == building1) {
frame.setVisible(false);
bF1.setVisible(true);
  }
  else if(e.getSource() == building2) {
frame.setVisible(false);
bF2.setVisible(true);
```

```
  }
  else if(e.getSource() == building3) {
frame.setVisible(false);
bF3.setVisible(true);
  }
  else if(e.getSource() == building4) {
frame.setVisible(false);
bF4.setVisible(true);
  }
  else if(e.getSource() == building5) {
frame.setVisible(false);
bF5.setVisible(true);
  }
  //  back buttons
  else if(e.getSource() == back1) {
    bF1.setVisible(false);
    frame.setVisible(true);
  }
  else if(e.getSource() == back2) {
bF2.setVisible(false);
frame.setVisible(true);
  }                              // Different Back buttons across all frames
  else if(e.getSource() == back3) {
    bF3.setVisible(false);
    frame.setVisible(true);
  }
  else if(e.getSource() == back4) {
    bF4.setVisible(false);
    frame.setVisible(true);
```

```
  }
  else if(e.getSource() == back5) {
    bF5.setVisible(false);
    frame.setVisible(true);
  }
  // door buttons
  else if(e.getSource() == door1) {
boolean doorStatus = Doors.getDoorStatus(1,1);
    if(doorStatus == false) {
  door1Label.setText("Locked");
}
else if(doorStatus == true) {
  door1Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door2) {
    boolean doorStatus = Doors.getDoorStatus(1,2);
    if(doorStatus == false) {
  door2Label.setText("Locked");
}
else if(doorStatus == true) {
  door2Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door3) {
boolean doorStatus = Doors.getDoorStatus(1,3);
    if(doorStatus == false) {
  door3Label.setText("Locked");
}
```

```
else if(doorStatus == true) {
  door3Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door4) {
    boolean doorStatus = Doors.getDoorStatus(1,4);
    if(doorStatus == false) {
  door4Label.setText("Locked");
}
else if(doorStatus == true) {
  door4Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door5) {
boolean doorStatus = Doors.getDoorStatus(1,5);
    if(doorStatus == false) {
  door5Label.setText("Locked");
}
else if(doorStatus == true) {
  door5Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door6) {
boolean doorStatus = Doors.getDoorStatus(1,6);
    if(doorStatus == false) {
  door6Label.setText("Locked");
}
else if(doorStatus == true) {
  door6Label.setText("Unlocked");
```

```
}
  }
  else if(e.getSource() == door7) {
boolean doorStatus = Doors.getDoorStatus(1,7);
    if(doorStatus == false) {
  door7Label.setText("Locked");
}
else if(doorStatus == true) {
  door7Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door8) {
boolean doorStatus = Doors.getDoorStatus(1,8);
    if(doorStatus == false) {
  door8Label.setText("Locked");
}
else if(doorStatus == true) {
  door8Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door9) {
boolean doorStatus = Doors.getDoorStatus(1,9);
    if(doorStatus == false) {
  door9Label.setText("Locked");
}
else if(doorStatus == true) {
  door9Label.setText("Unlocked");
}
  }
```

```
  else if(e.getSource() == door10) {
boolean doorStatus = Doors.getDoorStatus(1,10);
    if(doorStatus == false) {
  door10Label.setText("Locked");
}
else if(doorStatus == true) {
  door10Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door11) {
boolean doorStatus = Doors.getDoorStatus(2,1);
    if(doorStatus == false) {
  door11Label.setText("Locked");
}
else if(doorStatus == true) {
  door11Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door12) {
    boolean doorStatus = Doors.getDoorStatus(2,2);
    if(doorStatus == false) {
  door12Label.setText("Locked");
}
else if(doorStatus == true) {
  door12Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door13) {
boolean doorStatus = Doors.getDoorStatus(2,3);
```

```
    if(doorStatus == false) {
  door13Label.setText("Locked");
}
else if(doorStatus == true) {
  door13Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door14) {
    boolean doorStatus = Doors.getDoorStatus(2,4);
    if(doorStatus == false) {
  door14Label.setText("Locked");
}
else if(doorStatus == true) {
  door14Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door15) {
boolean doorStatus = Doors.getDoorStatus(2,5);
    if(doorStatus == false) {
  door15Label.setText("Locked");
}
else if(doorStatus == true) {
  door15Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door16) {
boolean doorStatus = Doors.getDoorStatus(2,6);
    if(doorStatus == false) {
  door16Label.setText("Locked");
```

```
}
else if(doorStatus == true) {
  door16Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door17) {
boolean doorStatus = Doors.getDoorStatus(2,7);
    if(doorStatus == false) {
  door17Label.setText("Locked");
}
else if(doorStatus == true) {
  door17Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door18) {
boolean doorStatus = Doors.getDoorStatus(2,8);
    if(doorStatus == false) {
  door18Label.setText("Locked");
}
else if(doorStatus == true) {
  door18Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door19) {
boolean doorStatus = Doors.getDoorStatus(2,9);
    if(doorStatus == false) {
  door19Label.setText("Locked");
}
else if(doorStatus == true) {
```

```
  door19Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door20) {
boolean doorStatus = Doors.getDoorStatus(2,10);
    if(doorStatus == false) {
  door20Label.setText("Locked");
}
else if(doorStatus == true) {
  door20Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door21) {
boolean doorStatus = Doors.getDoorStatus(3,1);
    if(doorStatus == false) {
  door21Label.setText("Locked");
}
else if(doorStatus == true) {
  door21Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door22) {
    boolean doorStatus = Doors.getDoorStatus(3,2);
    if(doorStatus == false) {
  door22Label.setText("Locked");
}
else if(doorStatus == true) {
  door22Label.setText("Unlocked");
}
```

```
  }
  else if(e.getSource() == door23) {
boolean doorStatus = Doors.getDoorStatus(3,3);
    if(doorStatus == false) {
  door23Label.setText("Locked");
}
else if(doorStatus == true) {
  door23Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door24) {
    boolean doorStatus = Doors.getDoorStatus(3,4);
    if(doorStatus == false) {
  door24Label.setText("Locked");
}
else if(doorStatus == true) {
  door24Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door25) {
boolean doorStatus = Doors.getDoorStatus(3,5);
    if(doorStatus == false) {
  door25Label.setText("Locked");
}
else if(doorStatus == true) {
  door25Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door26) {
```

```
boolean doorStatus = Doors.getDoorStatus(3,6);
    if(doorStatus == false) {
  door26Label.setText("Locked");
}
else if(doorStatus == true) {
  door26Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door27) {
boolean doorStatus = Doors.getDoorStatus(3,7);
    if(doorStatus == false) {
  door27Label.setText("Locked");
}
else if(doorStatus == true) {
  door27Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door28) {
boolean doorStatus = Doors.getDoorStatus(3,8);
    if(doorStatus == false) {
  door28Label.setText("Locked");
}
else if(doorStatus == true) {
  door28Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door29) {
boolean doorStatus = Doors.getDoorStatus(3,9);
    if(doorStatus == false) {
```

```
  door29Label.setText("Locked");
}
else if(doorStatus == true) {
  door29Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door30) {
boolean doorStatus = Doors.getDoorStatus(3,10);
    if(doorStatus == false) {
  door30Label.setText("Locked");
}
else if(doorStatus == true) {
  door30Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door31) {
boolean doorStatus = Doors.getDoorStatus(4,1);
    if(doorStatus == false) {
  door31Label.setText("Locked");
}
else if(doorStatus == true) {
  door31Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door32) {
    boolean doorStatus = Doors.getDoorStatus(4,2);
    if(doorStatus == false) {
  door32Label.setText("Locked");
}
```

```
else if(doorStatus == true) {
  door32Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door33) {
boolean doorStatus = Doors.getDoorStatus(4,3);
    if(doorStatus == false) {
  door33Label.setText("Locked");
}
else if(doorStatus == true) {
  door33Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door34) {
    boolean doorStatus = Doors.getDoorStatus(4,4);
    if(doorStatus == false) {
  door34Label.setText("Locked");
}
else if(doorStatus == true) {
  door34Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door35) {
boolean doorStatus = Doors.getDoorStatus(4,5);
    if(doorStatus == false) {
  door35Label.setText("Locked");
}
else if(doorStatus == true) {
  door35Label.setText("Unlocked");
```

```
}
  }
  else if(e.getSource() == door36) {
boolean doorStatus = Doors.getDoorStatus(4,6);
    if(doorStatus == false) {
  door36Label.setText("Locked");
}
else if(doorStatus == true) {
  door36Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door37) {
boolean doorStatus = Doors.getDoorStatus(4,7);
    if(doorStatus == false) {
  door37Label.setText("Locked");
}
else if(doorStatus == true) {
  door37Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door38) {
boolean doorStatus = Doors.getDoorStatus(4,8);
    if(doorStatus == false) {
  door38Label.setText("Locked");
}
else if(doorStatus == true) {
  door38Label.setText("Unlocked");
}
  }
```

```
  else if(e.getSource() == door39) {
boolean doorStatus = Doors.getDoorStatus(4,9);
    if(doorStatus == false) {
  door39Label.setText("Locked");
}
else if(doorStatus == true) {
  door39Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door40) {
boolean doorStatus = Doors.getDoorStatus(4,10);
    if(doorStatus == false) {
  door40Label.setText("Locked");
}
else if(doorStatus == true) {
  door40Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door41) {
boolean doorStatus = Doors.getDoorStatus(5,1);
    if(doorStatus == false) {
  door41Label.setText("Locked");
}
else if(doorStatus == true) {
  door41Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door42) {
    boolean doorStatus = Doors.getDoorStatus(5,2);
```

```
    if(doorStatus == false) {
  door42Label.setText("Locked");
}
else if(doorStatus == true) {
  door42Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door43) {
boolean doorStatus = Doors.getDoorStatus(5,3);
    if(doorStatus == false) {
  door43Label.setText("Locked");
}
else if(doorStatus == true) {
  door43Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door44) {
    boolean doorStatus = Doors.getDoorStatus(5,4);
    if(doorStatus == false) {
  door44Label.setText("Locked");
}
else if(doorStatus == true) {
  door44Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door45) {
boolean doorStatus = Doors.getDoorStatus(5,5);
    if(doorStatus == false) {
  door45Label.setText("Locked");
```

```
}
else if(doorStatus == true) {
  door45Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door46) {
boolean doorStatus = Doors.getDoorStatus(5,6);
    if(doorStatus == false) {
  door46Label.setText("Locked");
}
else if(doorStatus == true) {
  door46Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door47) {
boolean doorStatus = Doors.getDoorStatus(5,7);
    if(doorStatus == false) {
  door47Label.setText("Locked");
}
else if(doorStatus == true) {
  door47Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door48) {
boolean doorStatus = Doors.getDoorStatus(5,8);
    if(doorStatus == false) {
  door48Label.setText("Locked");
}
else if(doorStatus == true) {
```

```
  door48Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door49) {
boolean doorStatus = Doors.getDoorStatus(5,9);
    if(doorStatus == false) {
  door49Label.setText("Locked");
}
else if(doorStatus == true) {
  door49Label.setText("Unlocked");
}
  }
  else if(e.getSource() == door50) {
boolean doorStatus = Doors.getDoorStatus(5,10);
    if(doorStatus == false) {
  door50Label.setText("Locked");
}
else if(doorStatus == true) {
  door50Label.setText("Unlocked");
}
  }
  else if(e.getSource() == run1) {
    String cmd1 = instruction1.getText();
    cmd1.toLowerCase();

    if(cmd1.equals("a1")) {
      errorBox1.setText("");
      boolean doorStatus = Doors.setDoorStatus(1,1);
```

```
    if(doorStatus == false) {
        door1Label.setText("Locked");
    }
    else if(doorStatus == true) {
      door1Label.setText("Unlocked");
    }
  }
  else if(cmd1.equals("a2")) {
    errorBox1.setText("");
    boolean doorStatus = Doors.setDoorStatus(1,2);

    if(doorStatus == false) {
        door2Label.setText("Locked");
    }
    else if(doorStatus == true) {
        door2Label.setText("Unlocked");
    }
  }
  else if(cmd1.equals("a3")) {
 errorBox1.setText("");
 boolean doorStatus = Doors.setDoorStatus(1,3);

 if(doorStatus == false) {
   door3Label.setText("Locked");
 }
 else if(doorStatus == true) {
   door3Label.setText("Unlocked");
 }
}
```

```
    else if(cmd1.equals("a4")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,4);

  if(doorStatus == false) {
door4Label.setText("Locked");
  }
  else if(doorStatus == true) {
door4Label.setText("Unlocked");
  }
}
    else if(cmd1.equals("a5")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,5);

  if(doorStatus == false) {
door5Label.setText("Locked");
  }
  else if(doorStatus == true) {
door5Label.setText("Unlocked");
  }
}
    else if(cmd1.equals("a6")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,6);

  if(doorStatus == false) {
door6Label.setText("Locked");
  }
```

```
  else if(doorStatus == true) {
door6Label.setText("Unlocked");
  }
}
    else if(cmd1.equals("a7")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,7);

  if(doorStatus == false) {
door7Label.setText("Locked");
  }
  else if(doorStatus == true) {
door7Label.setText("Unlocked");
  }
}
    else if(cmd1.equals("a8")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,8);

  if(doorStatus == false) {
door8Label.setText("Locked");
  }
  else if(doorStatus == true) {
door8Label.setText("Unlocked");
  }
}
    else if(cmd1.equals("a9")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,9);
```

```
  if(doorStatus == false) {
door9Label.setText("Locked");
  }
  else if(doorStatus == true) {
door9Label.setText("Unlocked");
  }
    }
    else if(cmd1.equals("a10")) {
  errorBox1.setText("");
  boolean doorStatus = Doors.setDoorStatus(1,10);

  if(doorStatus == false) {
door10Label.setText("Locked");
  }
  else if(doorStatus == true) {
door10Label.setText("Unlocked");
  }
}
    else{
      errorBox1.setText("Invalid Door: Entered door does not exist");
    }
  }
      else if(e.getSource() == run2) {
String cmd2 = instruction2.getText();
cmd2.toLowerCase();

    if(cmd2.equals("b1")) {
  boolean doorStatus = Doors.setDoorStatus(2,1);
```

```
  errorBox2.setText("");

  if(doorStatus == false) {
    door11Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door11Label.setText("Unlocked");
  }
}
    else if(cmd2.equals("b2")) {
  boolean doorStatus = Doors.setDoorStatus(2,2);
  errorBox2.setText("");

  if(doorStatus == false) {
    door12Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door12Label.setText("Unlocked");
  }
}
else if(cmd2.equals("b3")) {
      boolean doorStatus = Doors.setDoorStatus(2,3);
      errorBox2.setText("");

  if(doorStatus == false) {
door13Label.setText("Locked");
      }
  else if(doorStatus == true) {
door13Label.setText("Unlocked");
```

```
  }
   }
else if(cmd2.equals("b4")) {
     boolean doorStatus = Doors.setDoorStatus(2,4);
     errorBox2.setText("");

  if(doorStatus == false) {
       door14Label.setText("Locked");
  }
     else if(doorStatus == true) {
    door14Label.setText("Unlocked");
  }
   }
else if(cmd2.equals("b5")) {
  boolean doorStatus = Doors.setDoorStatus(2,5);
  errorBox2.setText("");

  if(doorStatus == false) {
    door15Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door15Label.setText("Unlocked");
  }
}
else if(cmd2.equals("b6")) {
     boolean doorStatus = Doors.setDoorStatus(2,6);
     errorBox2.setText("");

  if(doorStatus == false) {
```

```
    door16Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door16Label.setText("Unlocked");
  }
}
else if(cmd2.equals("b7")) {
      boolean doorStatus = Doors.setDoorStatus(2,7);
      errorBox2.setText("");

  if(doorStatus == false) {
door17Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door17Label.setText("Unlocked");
  }
    }
else if(cmd2.equals("b8")) {
  boolean doorStatus = Doors.setDoorStatus(2,8);
  errorBox2.setText("");

  if(doorStatus == false) {
    door18Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door18Label.setText("Unlocked");
  }
}
else if(cmd2.equals("b9")) {
```

```
 boolean doorStatus = Doors.setDoorStatus(2,9);
 errorBox2.setText("");

if(doorStatus == false) {
  door19Label.setText("Locked");
}
    else if(doorStatus == true) {
  door19Label.setText("Unlocked");
    }
  }
  else if(cmd2.equals("b10")) {
    boolean doorStatus = Doors.setDoorStatus(2,10);
    errorBox2.setText("");

if(doorStatus == false) {
  door20Label.setText("Locked");
}
else if(doorStatus == true) {
  door20Label.setText("Unlocked");
}
  }
  else {
    errorBox2.setText("Invalid Door: Entered door does not exist");
  }
}
    else if(e.getSource() == run3) {
String cmd3 = instruction3.getText();
cmd3.toLowerCase();
```

```
   if(cmd3.equals("c1")) {
  boolean doorStatus = Doors.setDoorStatus(3,1);
  errorBox3.setText("");

  if(doorStatus == false) {
    door21Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door21Label.setText("Unlocked");
  }
}
   else if(cmd3.equals("c2")) {
  boolean doorStatus = Doors.setDoorStatus(3,2);
  errorBox3.setText("");

  if(doorStatus == false) {
    door22Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door22Label.setText("Unlocked");
  }
}
else if(cmd3.equals("c3")) {
      boolean doorStatus = Doors.setDoorStatus(3,3);
      errorBox3.setText("");

  if(doorStatus == false) {
door23Label.setText("Locked");
      }
```

```
  else if(doorStatus == true) {
door23Label.setText("Unlocked");
  }
    }
else if(cmd3.equals("c4")) {
      boolean doorStatus = Doors.setDoorStatus(3,4);
      errorBox3.setText("");

  if(doorStatus == false) {
        door24Label.setText("Locked");
  }
      else if(doorStatus == true) {
    door24Label.setText("Unlocked");
  }
    }
else if(cmd3.equals("c5")) {
  boolean doorStatus = Doors.setDoorStatus(3,5);
  errorBox3.setText("");

  if(doorStatus == false) {
    door25Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door25Label.setText("Unlocked");
  }
}
else if(cmd3.equals("c6")) {
      boolean doorStatus = Doors.setDoorStatus(3,6);
      errorBox3.setText("");
```

```
  if(doorStatus == false) {
    door26Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door26Label.setText("Unlocked");
  }
}
else if(cmd3.equals("c7")) {
      boolean doorStatus = Doors.setDoorStatus(3,7);
      errorBox3.setText("");

  if(doorStatus == false) {
door27Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door27Label.setText("Unlocked");
  }
    }
else if(cmd3.equals("c8")) {
  boolean doorStatus = Doors.setDoorStatus(3,8);
  errorBox3.setText("");

  if(doorStatus == false) {
    door28Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door28Label.setText("Unlocked");
  }
```

```
}
else if(cmd3.equals("c9")) {
  boolean doorStatus = Doors.setDoorStatus(3,9);
  errorBox3.setText("");

 if(doorStatus == false) {
   door29Label.setText("Locked");
 }
     else if(doorStatus == true) {
   door29Label.setText("Unlocked");
     }
   }
   else if(cmd3.equals("c10")) {
     boolean doorStatus = Doors.setDoorStatus(3,10);
     errorBox3.setText("");

 if(doorStatus == false) {
   door30Label.setText("Locked");
 }
 else if(doorStatus == true) {
   door30Label.setText("Unlocked");
 }
   }
   else {
     errorBox3.setText("Invalid Door: Entered door does not exist");
   }
     }
     else if(e.getSource() == run4) {
    String cmd4 = instruction4.getText();
```

```
  cmd4.toLowerCase();

  if(cmd4.equals("d1")) {
    errorBox4.setText("");
    boolean doorStatus = Doors.setDoorStatus(4,1);

    if(doorStatus == false) {
        door31Label.setText("Locked");
    }
    else if(doorStatus == true) {
      door31Label.setText("Unlocked");
    }
  }
  else if(cmd4.equals("d2")) {
    errorBox4.setText("");
    boolean doorStatus = Doors.setDoorStatus(4,2);

    if(doorStatus == false) {
        door32Label.setText("Locked");
    }
    else if(doorStatus == true) {
        door32Label.setText("Unlocked");
    }
  }
  else if(cmd4.equals("d3")) {
errorBox4.setText("");
boolean doorStatus = Doors.setDoorStatus(4,3);

if(doorStatus == false) {
```

```
    door33Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door33Label.setText("Unlocked");
  }
}
    else if(cmd4.equals("d4")) {
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,4);

  if(doorStatus == false) {
door34Label.setText("Locked");
  }
  else if(doorStatus == true) {
door34Label.setText("Unlocked");
  }
}
    else if(cmd4.equals("d5")) {
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,5);

  if(doorStatus == false) {
door35Label.setText("Locked");
  }
  else if(doorStatus == true) {
door35Label.setText("Unlocked");
  }
}
    else if(cmd4.equals("d6")) {
```

```
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,6);

  if(doorStatus == false) {
door36Label.setText("Locked");
  }
  else if(doorStatus == true) {
door36Label.setText("Unlocked");
  }
}
    else if(cmd4.equals("d7")) {
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,7);

  if(doorStatus == false) {
door37Label.setText("Locked");
  }
  else if(doorStatus == true) {
door37Label.setText("Unlocked");
  }
}
    else if(cmd4.equals("d8")) {
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,8);

  if(doorStatus == false) {
door38Label.setText("Locked");
  }
  else if(doorStatus == true) {
```

```
door38Label.setText("Unlocked");
  }
}
    else if(cmd4.equals("d9")) {
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,9);

  if(doorStatus == false) {
door39Label.setText("Locked");
  }
  else if(doorStatus == true) {
door39Label.setText("Unlocked");
  }
    }
    else if(cmd4.equals("d10")) {
  errorBox4.setText("");
  boolean doorStatus = Doors.setDoorStatus(4,10);

  if(doorStatus == false) {
door40Label.setText("Locked");
  }
  else if(doorStatus == true) {
door40Label.setText("Unlocked");
  }
}
    else{
      errorBox4.setText("Invalid Door: Entered door does not exist");
    }
     }
```

```
 else if(e.getSource() == run5) {
  String cmd5 = instruction5.getText();
  cmd5.toLowerCase();

  if(cmd5.equals("e1")) {
    errorBox5.setText("");
    boolean doorStatus = Doors.setDoorStatus(5,1);

    if(doorStatus == false) {
         door41Label.setText("Locked");
    }
    else if(doorStatus == true) {
      door41Label.setText("Unlocked");
    }
  }
  else if(cmd5.equals("e2")) {
    errorBox5.setText("");
    boolean doorStatus = Doors.setDoorStatus(5,2);

    if(doorStatus == false) {
         door42Label.setText("Locked");
    }
    else if(doorStatus == true) {
         door42Label.setText("Unlocked");
    }
  }
  else if(cmd5.equals("e3")) {
errorBox5.setText("");
boolean doorStatus = Doors.setDoorStatus(5,3);
```

```
  if(doorStatus == false) {
    door43Label.setText("Locked");
  }
  else if(doorStatus == true) {
    door43Label.setText("Unlocked");
  }
}
    else if(cmd5.equals("e4")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,4);

  if(doorStatus == false) {
door44Label.setText("Locked");
  }
  else if(doorStatus == true) {
door44Label.setText("Unlocked");
  }
}
    else if(cmd5.equals("e5")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,5);

  if(doorStatus == false) {
door45Label.setText("Locked");
  }
  else if(doorStatus == true) {
door45Label.setText("Unlocked");
  }
```

```
}
    else if(cmd5.equals("e6")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,6);

  if(doorStatus == false) {
door46Label.setText("Locked");
  }
  else if(doorStatus == true) {
door46Label.setText("Unlocked");
  }
}
    else if(cmd5.equals("e7")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,7);

  if(doorStatus == false) {
door47Label.setText("Locked");
  }
  else if(doorStatus == true) {
door47Label.setText("Unlocked");
  }
}
    else if(cmd5.equals("e8")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,8);

  if(doorStatus == false) {
door48Label.setText("Locked");
```

```
  }
  else if(doorStatus == true) {
door48Label.setText("Unlocked");
  }
}
    else if(cmd5.equals("e9")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,9);

  if(doorStatus == false) {
door49Label.setText("Locked");
  }
  else if(doorStatus == true) {
door49Label.setText("Unlocked");
  }
    }
    else if(cmd5.equals("e10")) {
  errorBox5.setText("");
  boolean doorStatus = Doors.setDoorStatus(5,10);

  if(doorStatus == false) {
door50Label.setText("Locked");
  }
  else if(doorStatus == true) {
door50Label.setText("Unlocked");
  }
}
    else{
      errorBox5.setText("Invalid Door: Entered door does not exist");
```

```
    }
     }

    }// actionPerformed method ends

  }// MainHanlder Class ends

}// Jurassic park security class ends
```