

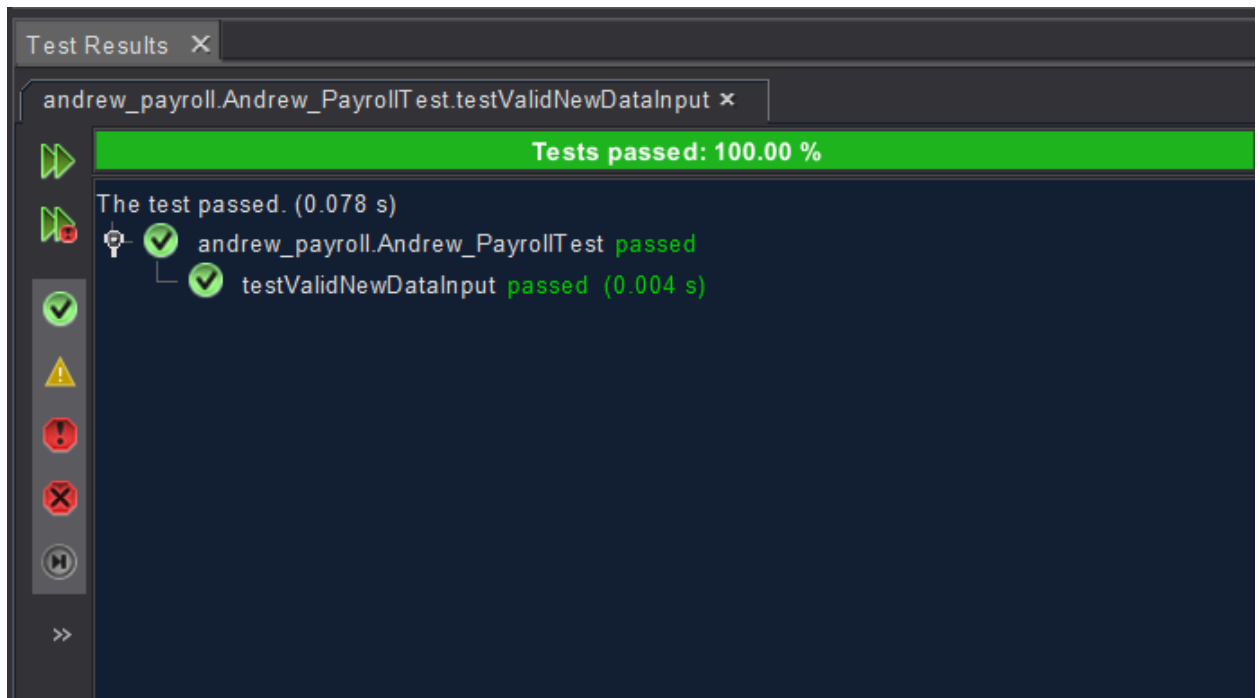
a)

Feedback

- Feedback that was given by the end-user was to adjust documentation so that it includes what type of software is used within the program and the costs of the program and how much will be used exactly when creating the program.
- The maximum amount of users was also meant to be specified to let Urban Furns know how many users can use the program.
- All feedback received was implemented within the program and the documentation and all adjustments that needed to be made were all implemented after receiving feedback from the end-user.

Test Cases

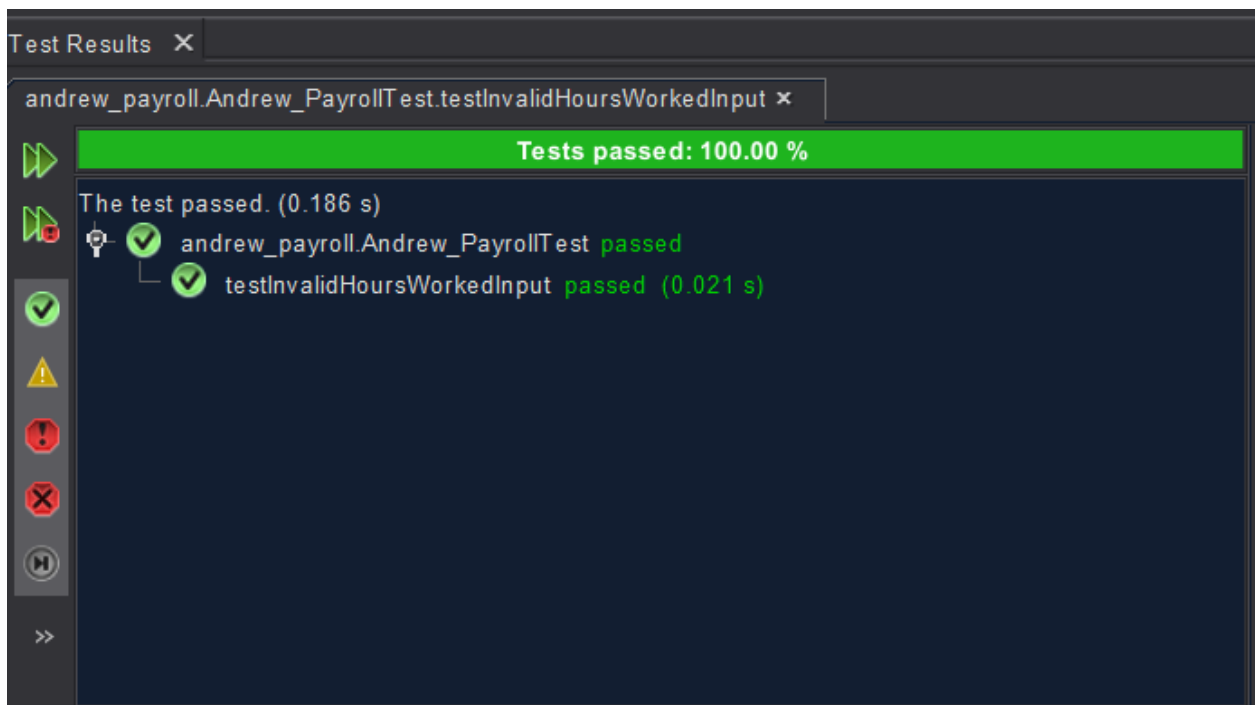
Test Case 1: Valid New Data Input



- **Expected Result:**

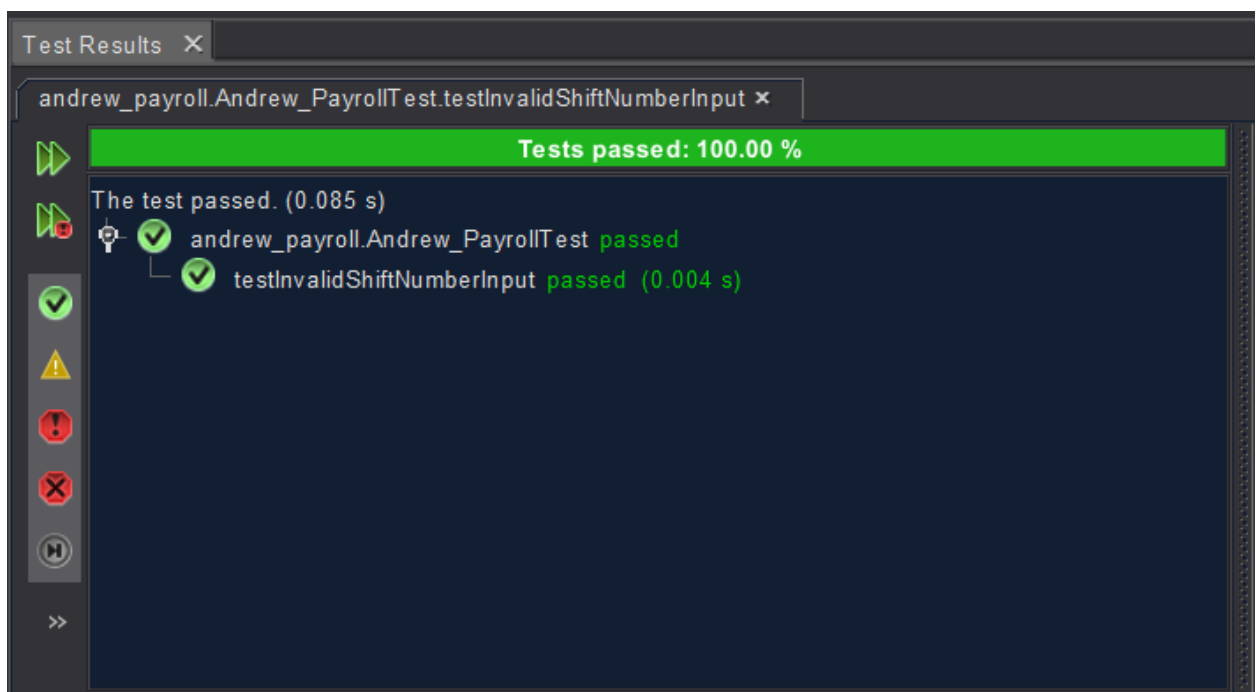
- **Regular Pay:** $40 \times 50 = R2000$
 - **Overtime Pay:** R0 (No overtime)
 - **Total Pay:** R2000
 - **Retirement Deduction:** R0 (No retirement plan)
 - **Net Pay:** R2000
- **Actual Result:**
 - **Regular Pay:** (Check if it matches R2000)
 - **Overtime Pay:** (Check if it matches R0)
 - **Total Pay:** (Check if it matches R2000)
 - **Retirement Deduction:** (Check if it matches R0)
 - **Net Pay:** (Check if it matches R2000)

Test Case 2: Invalid Hours Worked Input



- **Expected Result:**
 - Exception: `IllegalArgumentException` for negative hours.
- **Actual Result:**
 - Exception: (Check if it is `IllegalArgumentException`)

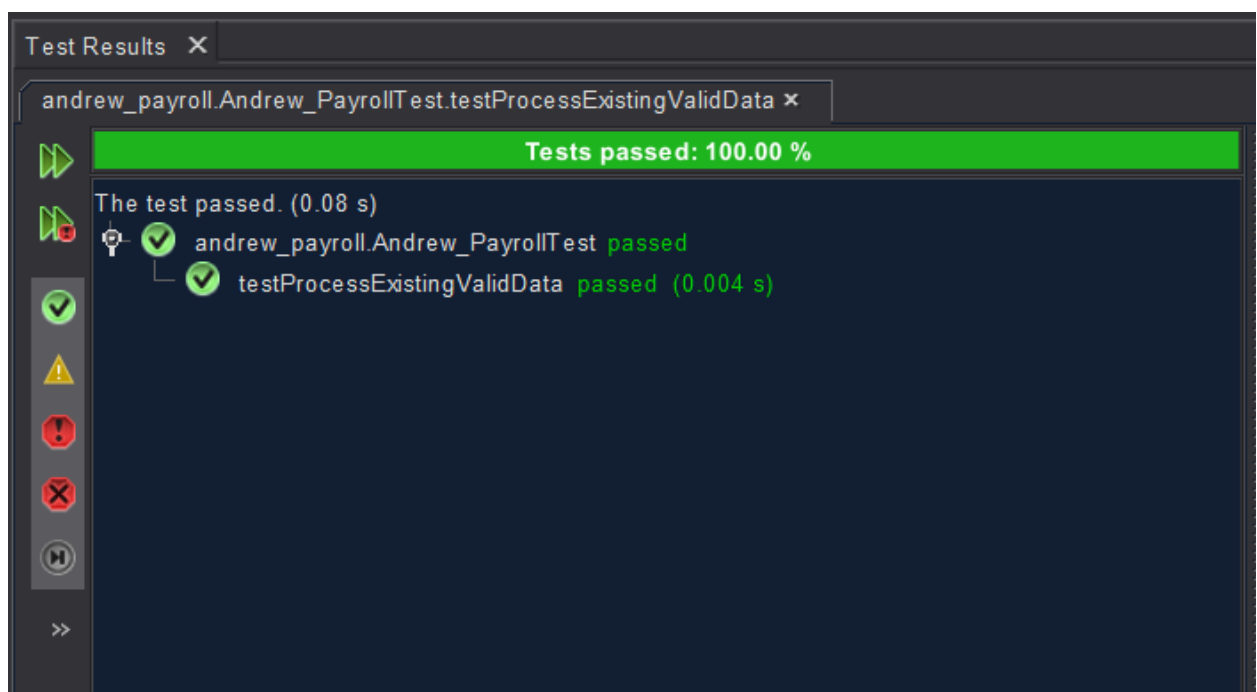
Test Case 3: Invalid Shift Number Input



- **Expected Result:**
 - **Regular Pay:** 0
 - **Overtime Pay:** 0
 - **Total Pay:** 0
 - **Retirement Deduction:** 0
 - **Net Pay:** 0

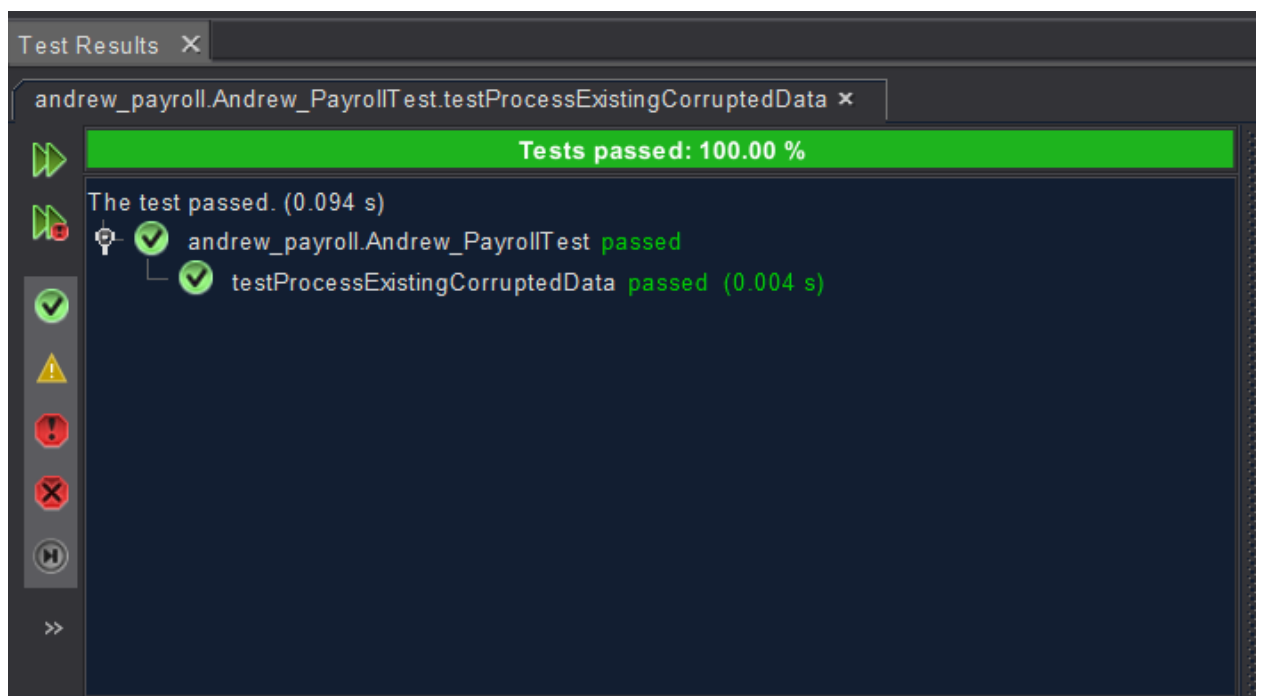
- **Actual Result:**
 - **Regular Pay:** (Check if it matches 0)
 - **Overtime Pay:** (Check if it matches 0)
 - **Total Pay:** (Check if it matches 0)
 - **Retirement Deduction:** (Check if it matches 0)
 - **Net Pay:** (Check if it matches 0)

Test Case 4: Process Existing Valid Data



- **Expected Result:**
 - The program processes the file data without errors.
- **Actual Result:**
 - (Check if the file is processed without errors)

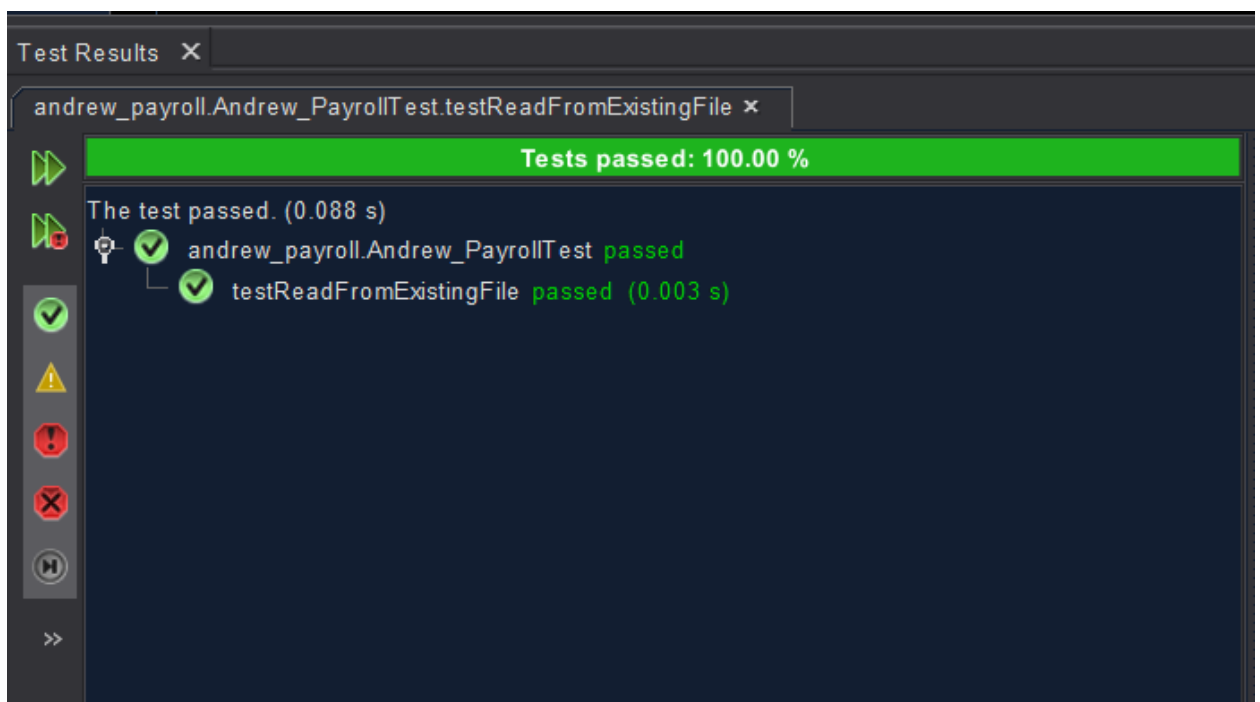
Test Case 5: Process Existing Corrupted Data



- **Expected Result:**
 - **Exception:** IOException for corrupted data.
- **Actual Result:**

- **Exception:** (Check if it is IOException)

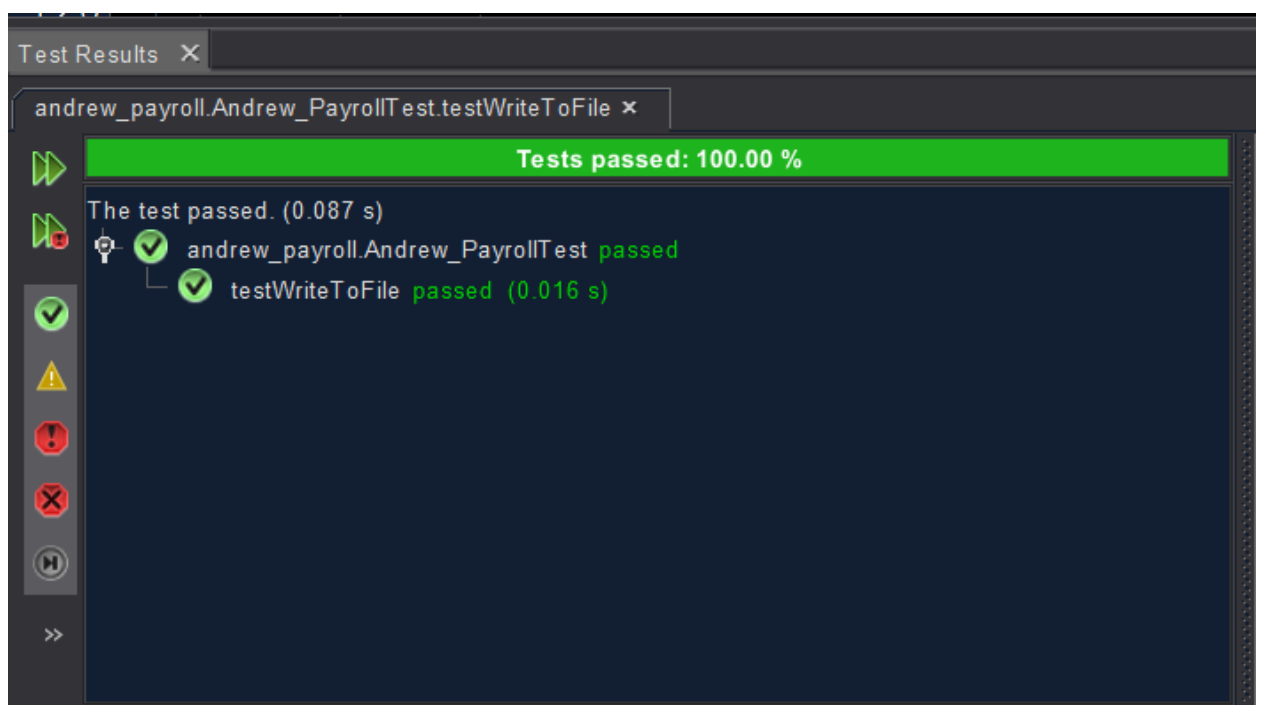
Test Case 6: Read from Existing File



- **Expected Result:**
 - The program reads and displays the data correctly.
- **Actual Result:**

- (Check if the file data is read and displayed correctly)

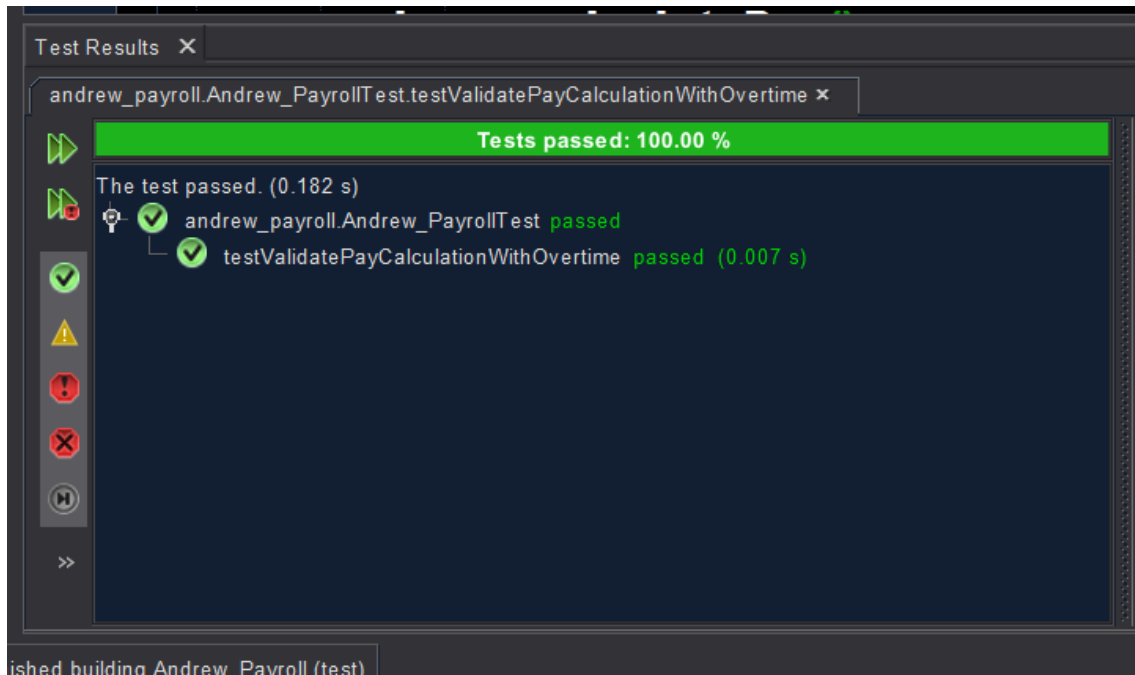
Test Case 7: Write to File



- **Expected Result:**
 - The file contains the new entry "30".
- **Actual Result:**

- (Check if the file contains the entry "30")

Test Case 7: Calculations with Overtime



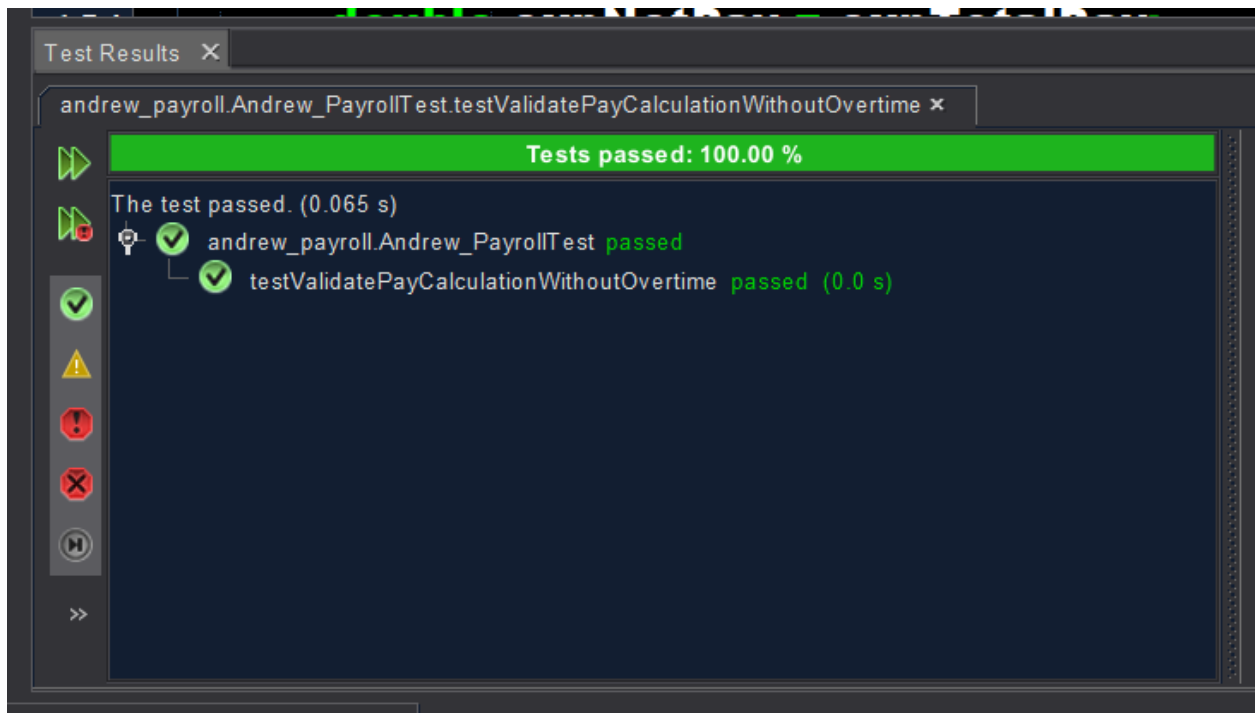
- **Expected Result:**

- **Regular Pay:** $40 \text{ hours} \times R70 = R2800$
- **Overtime Pay:** $(45 - 40) \text{ hours} \times R70 \times 1.5 = R525$
- **Total Pay:** $R2800 + R525 = R3325$
- **Retirement Deduction:** $5\% \text{ of } R3325 = R166.25$
- **Net Pay:** $R3325 - R166.25 = R3158.75$

- **Actual Result:**

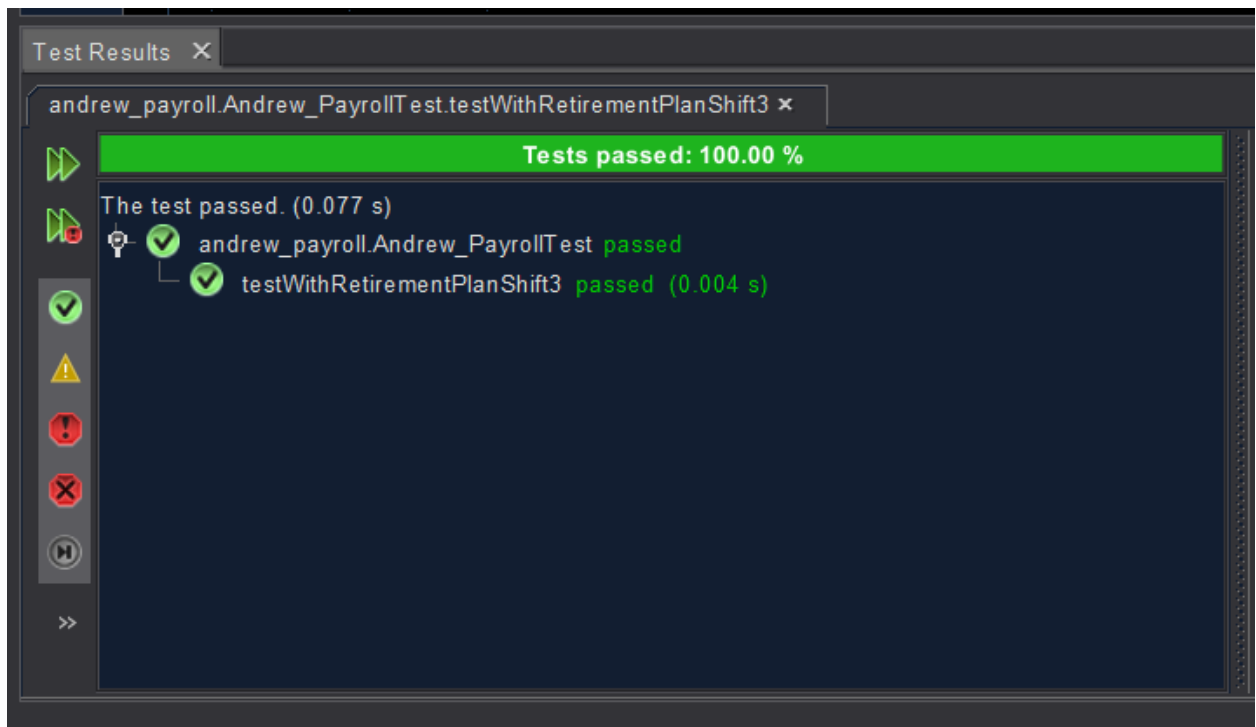
- Check if the calculations match the expected values:
 - Regular Pay should be R2800.
 - Overtime Pay should be R525.
 - Total Pay should be R3325.
 - Retirement Deduction should be R166.25.
 - Net Pay should be R3158.75.
- Ensure the file `hours_worked.txt` contains the entry "45".

Test Case 9: Calculations without Overtime



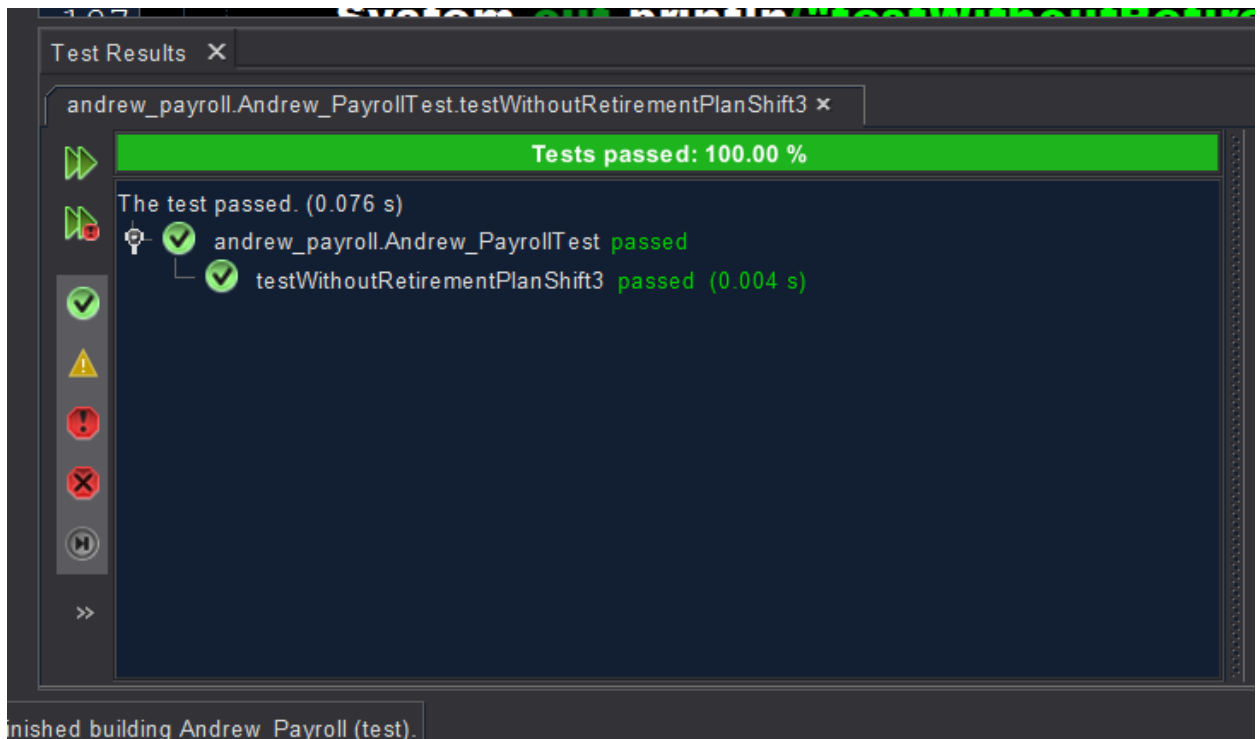
- **Expected Result:**
 - **Regular Pay:** $30 \text{ hours} \times R70 = R2100$
 - **Overtime Pay:** R0
 - **Total Pay:** $R2100 + R0 = R2100$
 - **Retirement Deduction:** $5\% \text{ of } R2100 = R105$
 - **Net Pay:** $R2100 - R105 = R1995$
- **Actual Result:**
 - Check if the calculations match the expected values:
 - Regular Pay should be R2100.
 - Overtime Pay should be R0.
 - Total Pay should be R2100.
 - Retirement Deduction should be R105.
 - Net Pay should be R1995.
 - Ensure the file `hours_worked.txt` contains the entry "30".

Test Case 10: Calculations with Retirement plan



- **Expected Result:**
 - **Regular Pay:** 40 hours \times R70 = R2800
 - **Overtime Pay:** R0
 - **Total Pay:** R2800 + R0 = R2800
 - **Retirement Deduction:** 5% of R2800 = R140
 - **Net Pay:** R2800 - R140 = R2660
- **Actual Result:**
 - Check if the calculations match the expected values:
 - Regular Pay should be R2800.
 - Overtime Pay should be R0.
 - Total Pay should be R2800.
 - Retirement Deduction should be R140.
 - Net Pay should be R2660.
 - Ensure the file `hours_worked.txt` contains the entry "40".

Test Case 11: Calculations without Retirement plan



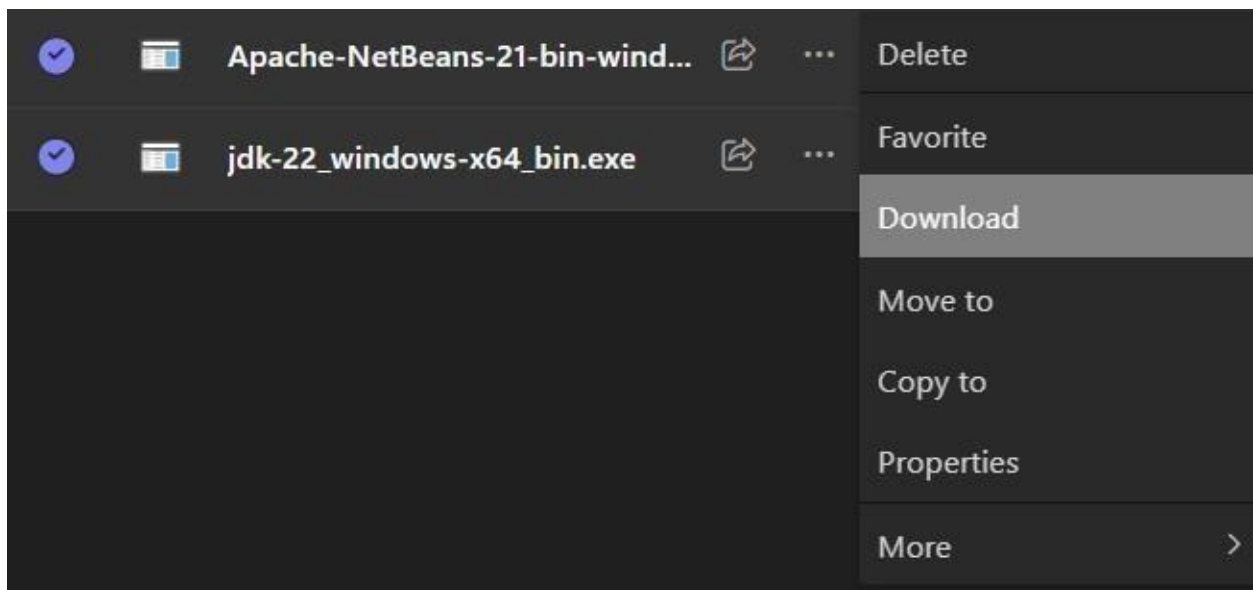
- **Expected Result:**
 - **Regular Pay:** 40 hours × R70 = R2800
 - **Overtime Pay:** R0
 - **Total Pay:** R2800 + R0 = R2800
 - **Retirement Deduction:** R0
 - **Net Pay:** R2800 - R0 = R2800
- **Actual Result:**
 - Check if the calculations match the expected values:
 - Regular Pay should be R2800.
 - Overtime Pay should be R0.
 - Total Pay should be R2800.
 - Retirement Deduction should be R0.
 - Net Pay should be R2800.
- Ensure the file hours_worked.txt contains the entry "40".

c)

Installation Plan



Step 1:

Open your **Microsoft Teams** program and download the two files that will have been sent to you by the factory's software deployment team.



Step 2:

Once the programs have been downloaded, start by double-clicking the first file to install **jdk22**. **JDK** stands for **Java Development Kit** is used to access Java libraries and is used for applications.

Name	Date modified	Type	Size
Today			
 jdk-22_windows-x64_bin	2024/07/05 12:14	Application	168 256 KB
 Apache-NetBeans-21-bin-windows-x64	2024/07/05 12:14	Application	487 515 KB

Step 3:

You will be shown this window that will welcome you to **JDK** and you will have to move your mouse and click next.





Step 4:

After clicking next, you will be told that installing **JDK** requires **420MB** on your hard drive. Ensure that that space is available then click next.



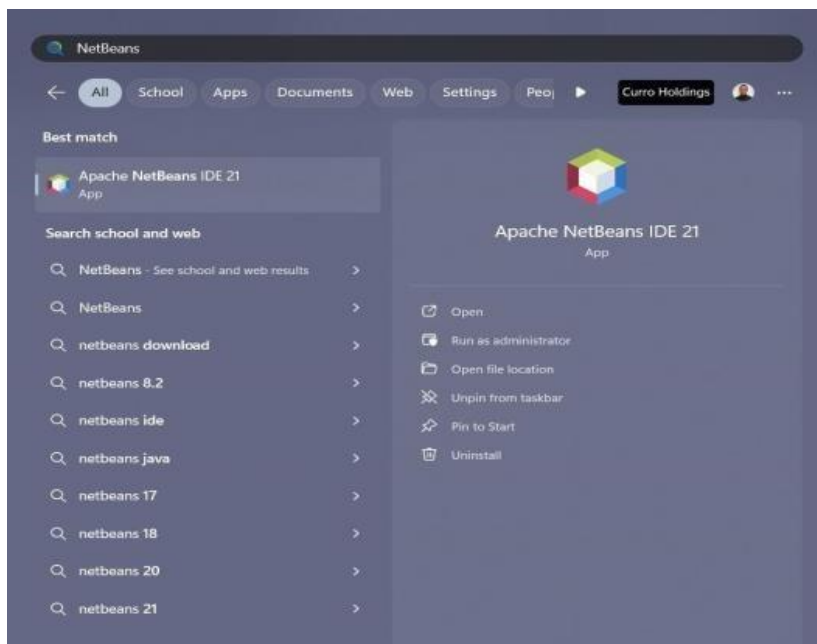
Step 5:

Once **JDK** has finished installing, move on to installing **Apache-NetBeans** which is our **Integrated Development Environment (IDE)** and where we will be running the program. Double-click on Apache NetBeans to install it.

Name	Date modified	Type	Size
Today			
 jdk-22_windows-x64_bin	2024/07/05 12:14	Application	168 256 KB
 Apache-NetBeans-21-bin-windows-x64	2024/07/05 12:14	Application	487 515 KB

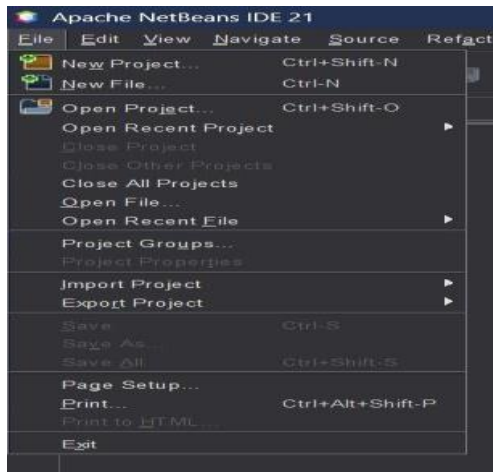
Step 6:

Once **NetBeans** has finished installing, go to your search bar and search for **NetBeans** and this will show up. After it has shown up, click on open.



Step 7:

Once you're inside **NetBeans**, click on the file in the top left corner and then click **Import Project**. You will then navigate to the program that the software deployment would have sent to you via **Microsoft Teams**.



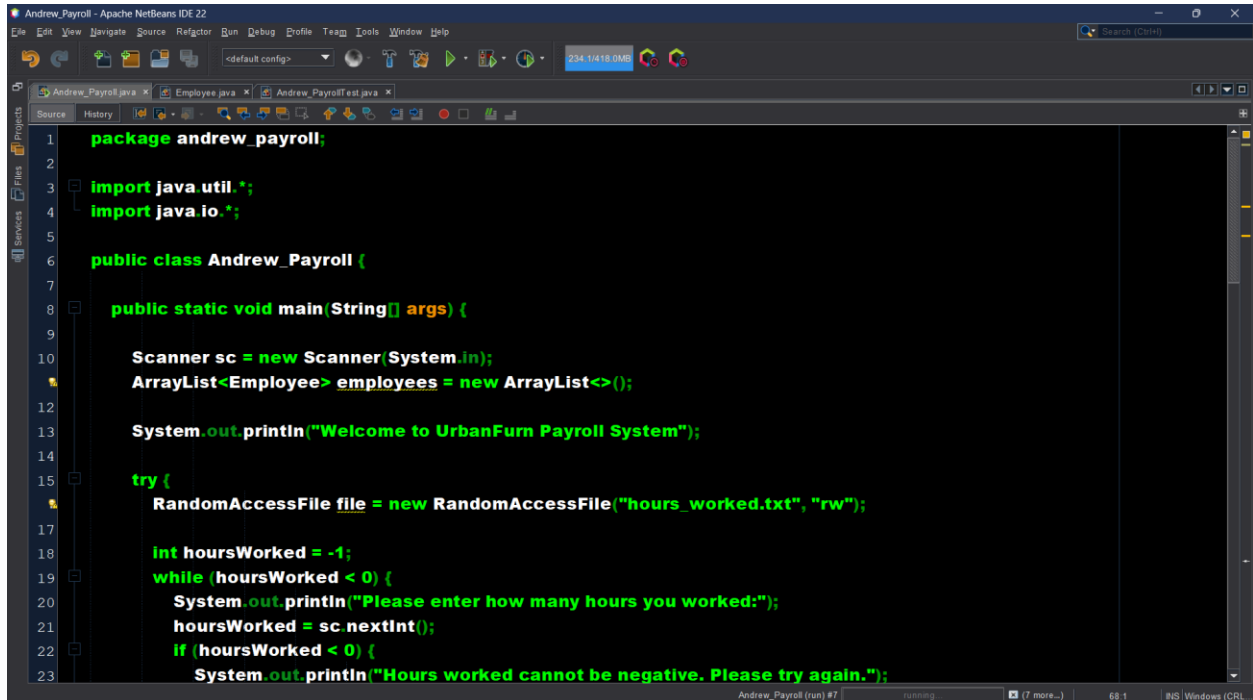
Step 8:

Once you have imported the program, you will click the **green triangle** that is facing right, which will run the program.



Step 9:

Once you have followed all the steps, this is how the program should look.



The screenshot shows the Apache NetBeans IDE interface. The main editor window displays the source code for `Andrew_Payroll.java`. The code is as follows:

```
1 package andrew_payroll;
2
3 import java.util.*;
4 import java.io.*;
5
6 public class Andrew_Payroll {
7
8     public static void main (String[] args) {
9
10         Scanner sc = new Scanner(System.in);
11         ArrayList<Employee> employees = new ArrayList<>();
12
13         System.out.println("Welcome to UrbanFurn Payroll System");
14
15         try {
16             RandomAccessFile file = new RandomAccessFile("hours_worked.txt", "rw");
17
18             int hoursWorked = -1;
19             while (hoursWorked < 0) {
20                 System.out.println("Please enter how many hours you worked:");
21                 hoursWorked = sc.nextInt();
22                 if (hoursWorked < 0) {
23                     System.out.println("Hours worked cannot be negative. Please try again.");
```

The IDE interface includes a menu bar at the top with options like File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. Below the menu bar is a toolbar with various icons for file operations, running, and debugging. The left sidebar shows the Project, Files, and Services views. The bottom status bar indicates the current file is `Andrew_Payroll (run) #7`, the status is `running`, and the cursor is at line 68, column 1.

- The timeline of the project goes over 4 days during which the program will be developed and installed for the workers.
- These four days are Monday, Tuesday, Wednesday and Thursday..
- Monday will be spent on programming then the other 3 days will be spent on documentation.
- Resources that will be required is between R36 000 and R54 000 which will be used to pay the developer R300 an hour over the week.
- Contingency plans include creating backups of the program every hour to prevent data loss in the case of an emergency where there is a power outage or the computer unexpectedly shuts down without warning.
- By having contingency plans, this will help to combat data loss and always be ahead so that should there be an emergency, the progress of the program will not be halted since there will be backups available.
- Once the program is completed and has been documented, a meeting will be set up with UrbanFurn to discuss how the program will be installed.
- A user manual will be provided to UrbanFurn on how to install the program for other employees and how the program can be used.
- The program will then be installed by me for every employee and the user manual will be left for future employees who will be hired to the company.