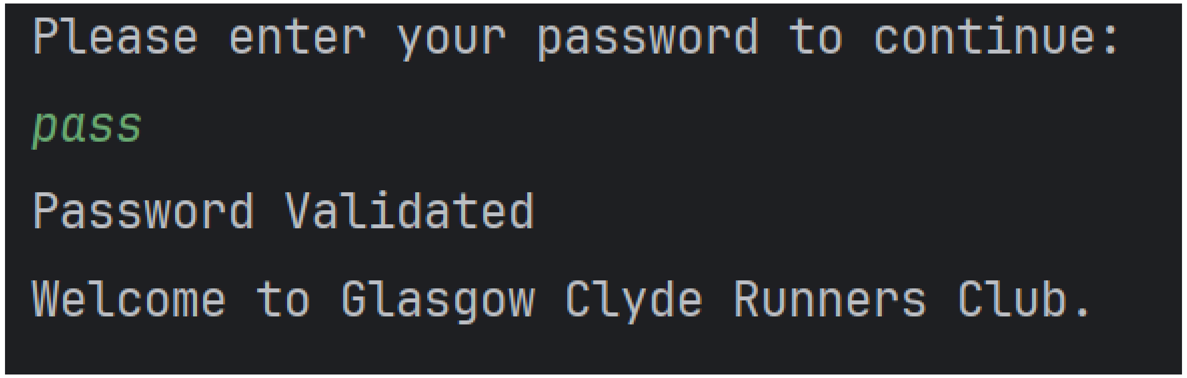
Unit testing (password code)

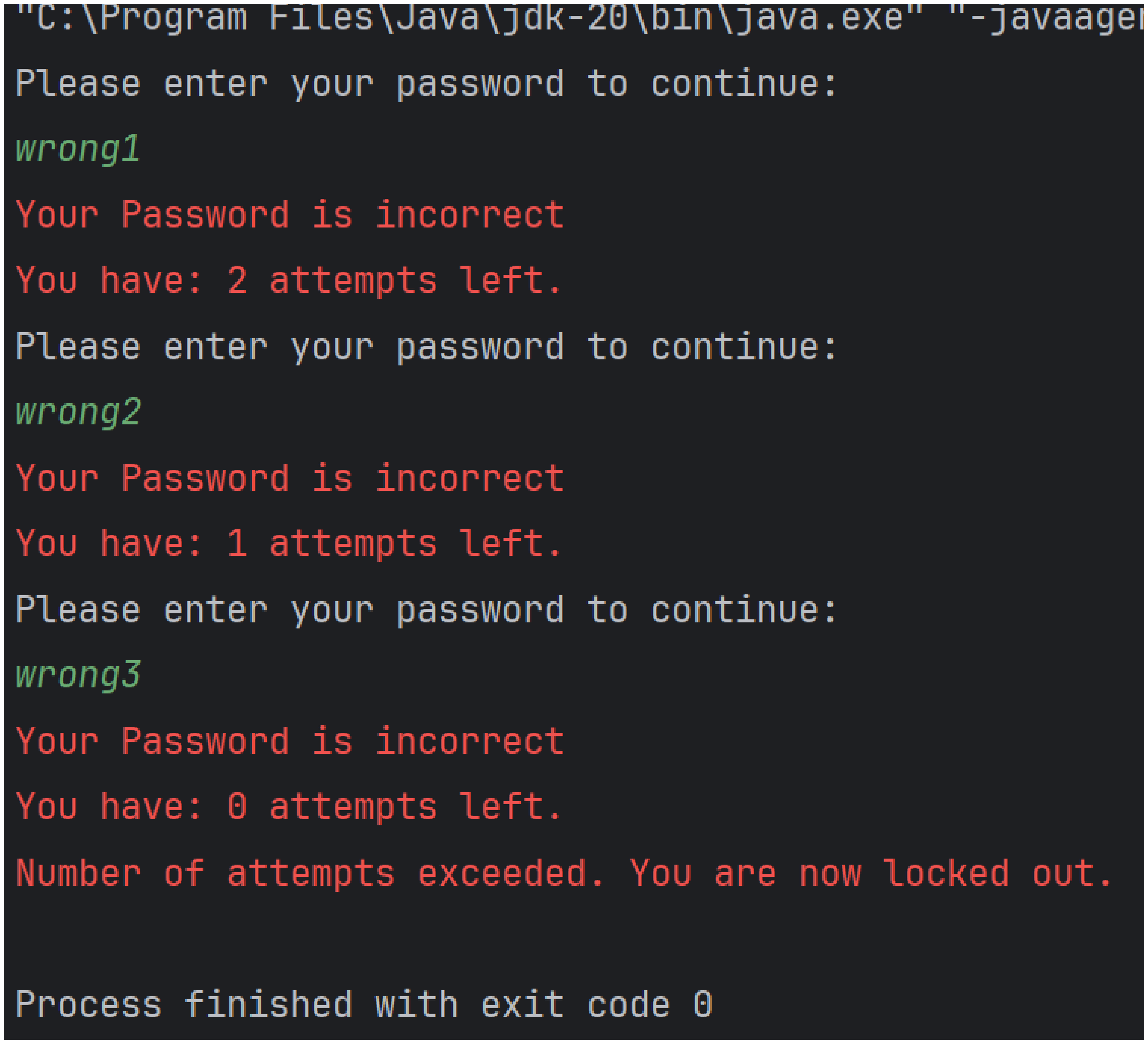
In the table is the use of a unit test to test the functionality of the authorization module using normal and abnormal data. Extreme data was not used because the password is of type String. This is a White Box.

Method Name	Test Case (Description)	Test Data	Expected Result	Actual Result	Comments
Login	Valid password is assigned and validated.	“pass”	Message displayed to user “Password Validated Welcome to Glasgow Clyde Runners Club.”	Pass	
Login	Invalid password is assigned and validated.	“wrong1”	Message displayed to user “Your Password is incorrect You have: 2 attempts left. Please enter your password to continue:”	Pass	
Login	Invalid password is assigned and validated.	“wrong2”	Message displayed to user “Your Password is incorrect You have: 1 attempts left. Please enter your password to continue:”	Pass	
Login	Invalid password is assigned and validated.	“wrong3”	Message displayed to user “Your Password is incorrect You have: 0 attempts left. Number of attempts exceeded. You are now locked out.”	Pass	

Below the table are screenshots, this is an example of the same test, but this time it’s a Black Box



```
Please enter your password to continue:
pass
Password Validated
Welcome to Glasgow Clyde Runners Club.
```



```
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaager
Please enter your password to continue:
wrong1
Your Password is incorrect
You have: 2 attempts left.
Please enter your password to continue:
wrong2
Your Password is incorrect
You have: 1 attempts left.
Please enter your password to continue:
wrong3
Your Password is incorrect
You have: 0 attempts left.
Number of attempts exceeded. You are now locked out.

Process finished with exit code 0
```

Integration testing

The integration test table is an example of a White Box and this test is used to test the performance of specific methods using specific scenarios of user actions and program behaviour during these actions

Method Names	Test Case (Description)	Test Data	Expected Result	Actual Result	Comments
App mainMenu.showMenu();	After entering the password correctly, the user is shown a menu	After Input password: pass	The menu is displayed to the user: “Menu of Glasgow Clyde Runners Club App: 1. Read and Display File 2. Sort and Print Recorded Times 3. Find and Print Fastest Time 4. Find and Print the Slowest Time 5. Search 6. Time Occurrence 7. Exit Program Please, choose your menu option (1 - 7):”	Pass	
Main Menu: processingMenu readAndDisplayFile ();	The user decided to look at the results of the final race The results should be displayed to him as they are entered into the data file	Main Menu: Option 1	The user should see: • what information did he ask for • where the data file is located on disk and its name • contents of the file as written	Pass	

Main Menu: processingMenu sortAndPrintRecordedTimes(listRunners);	The user chose to see the list of finalists sorted from slowest to fastest The sorted list should be displayed to him and the result should be written in a file	Main Menu: Option 2	The user should see what information he asked for Data must be written to a file After recording they must be read and displayed	Pass	
Main Menu: processingMenu findAndPrintFastestSlowestTime(listRunners,selectItem);	The user decided to find out who is the fastest participant in the final It should display the best result	Main Menu: Option 3	The user should see what information he asked for Data must be written to a file After recording they must be read and displayed	Pass	
Main Menu: processingMenu findAndPrintFastestSlowestTime(listRunners,selectItem);	The user decided to find out who the slowest participant was in the finals It should display the worst result	Main Menu: Option 4	The user should see what information he asked for Data must be written to a file After recording they must be read and displayed	Pass	
Main Menu: processingMenu searchOccurrenceTime(listRunners,selectItem);	The user decided to find out if there was a participant in the final who ran with a certain result	Main Menu: Option 5	The user should see what information he asked for Data must be written to a file	Pass	

	He enters the runner's result time of interest Should display a race participant with this result time		After recording they must be read and displayed		
Main Menu searchOccurrenceTime(listRunners,selectItem);	The user decided to find out whether there were participants in the final who ran with a certain result He enters the runners' result time of interest All race participants with this result should be displayed.	Main Menu: Option 6	The user should see what information he asked for Data must be written to a file After recording they must be read and displayed	Pass	

App
 	
// Processing the menu
while (selectItem != 7) {
mainMenu.showMenu();
selectItem = mainMenu.getMenuItemSelection(7);
processingMenu(selectItem);
}
	The user decided to quit the program
	
	A corresponding message is displayed to him
	
	The program exits
		Main Menu: Option 7	The user is shown a message to exit the program
			
			The program exits correctly
				Pass	

Dry run

I use dry runs, tracing, or step-by-step execution at the development stage when mastering new technologies or when errors occur when the program does not behave as planned or as designed. In this example, I documented the use of this type of testing when studying lambda streams to select records according to specified filters.

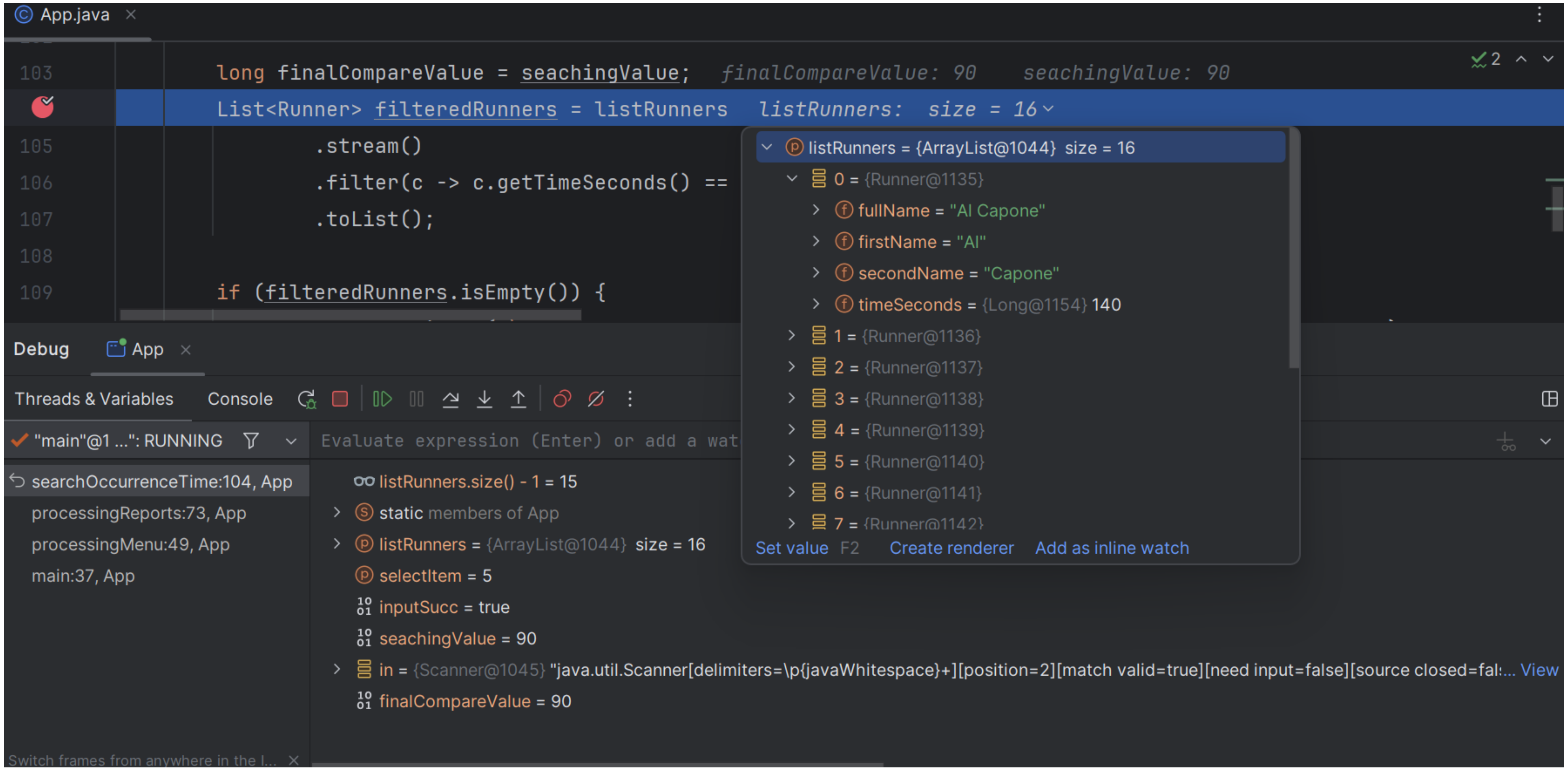
Step-by-step trace table for a search stream

```
long finalCompareValue = seachingValue;
List<Runner> filteredRunners = listRunners
        .stream()
        .filter(c -> c.getTimeSeconds() == finalCompareValue)
        .toList();

if (selectItem == 5) { //Write the search time
    filteredRunners = Collections.singletonList(filteredRunners
            .stream()
            .filter(c -> c.getTimeSeconds() == finalCompareValue)
           .findFirst()
            .orElse(null));
}
```

finalCompareValue ← 90

listRunners.Runner.timeSeconds ← [140, 120, 110, 103, 97, 95, 90, 90, 80, 80, 78, 75, 72, 70, 70, 68]

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	140	120	110	103	97	95	90	90	80	80	78	75	72	70	70	68



```
        long finalCompareValue = seachingValue;   finalCompareValue: 90    seachingValue: 90
        List<Runner> filteredRunners = listRunners   listRunners:  size = 16
                .stream()
                .filter(c -> c.getTimeSeconds() ==
                .toList();

        if (filteredRunners.isEmpty()) {
```

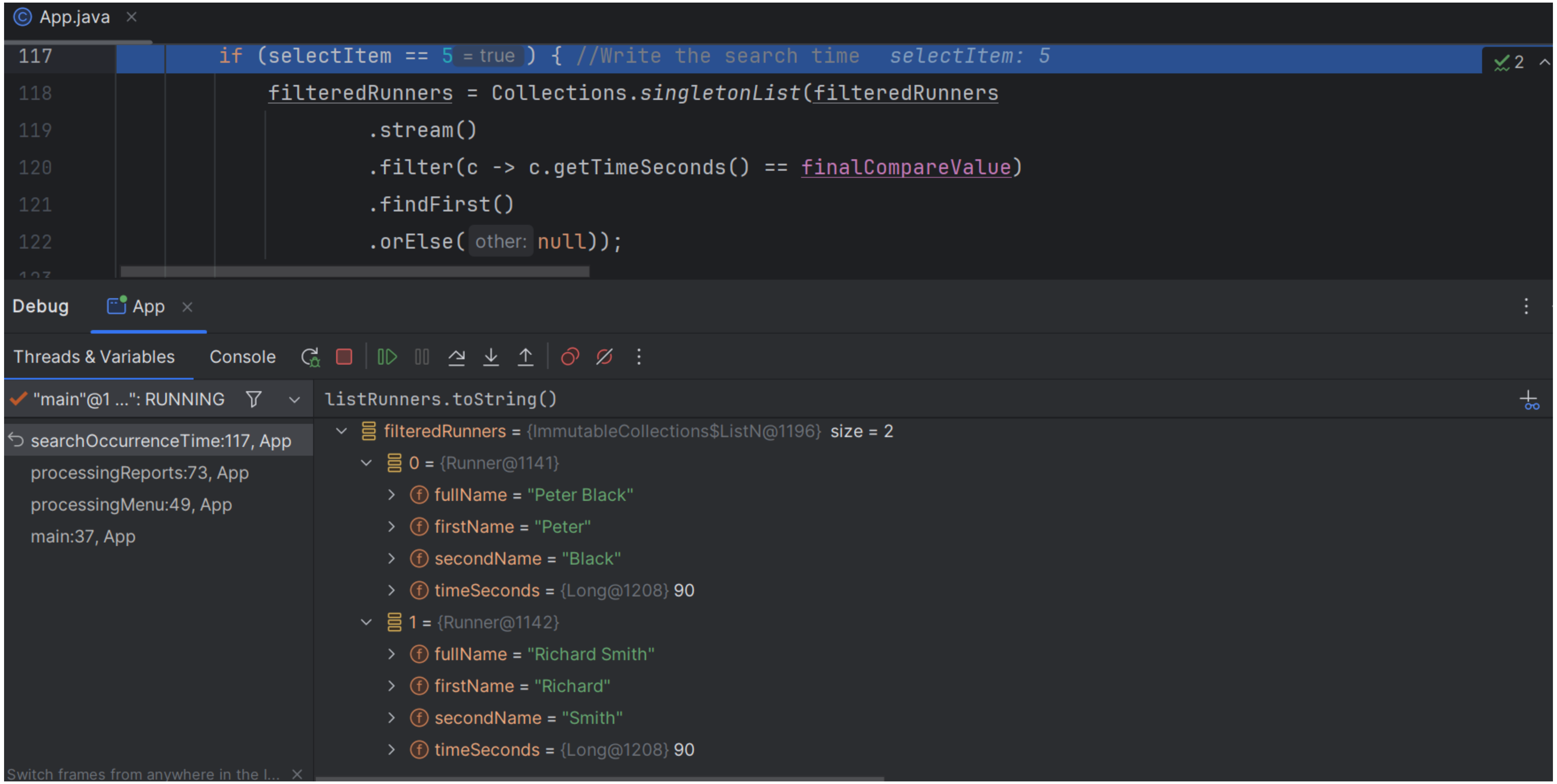
```
listRunners = {ArrayList@1044}  size = 16
  0 = {Runner@1135}
    fullName = "Al Capone"
    firstName = "Al"
    secondName = "Capone"
    timeSeconds = {Long@1154} 140
  1 = {Runner@1136}
  2 = {Runner@1137}
  3 = {Runner@1138}
  4 = {Runner@1139}
  5 = {Runner@1140}
  6 = {Runner@1141}
  7 = {Runner@1142}
```

```
"main"@1 ...": RUNNING

searchOccurrenceTime:104, App
processingReports:73, App
processingMenu:49, App
main:37, App

Switch frames from anywhere in the I...
```

```
listRunners.size() - 1 = 15
static members of App
listRunners = {ArrayList@1044}  size = 16
selectItem = 5
inputSucc = true
seachingValue = 90
in = {Scanner@1045} "java.util.Scanner[delimiters=\p{javaWhitespace}+][position=2][match valid=true][need input=false][source closed=fal:... View
finalCompareValue = 90
```

Step 1

found	Index	listRunners.Runner.timeSeconds	listRunners.Runner.fullName
True	6	90	Peter Black
True	7	90	Richard Smith



App.java

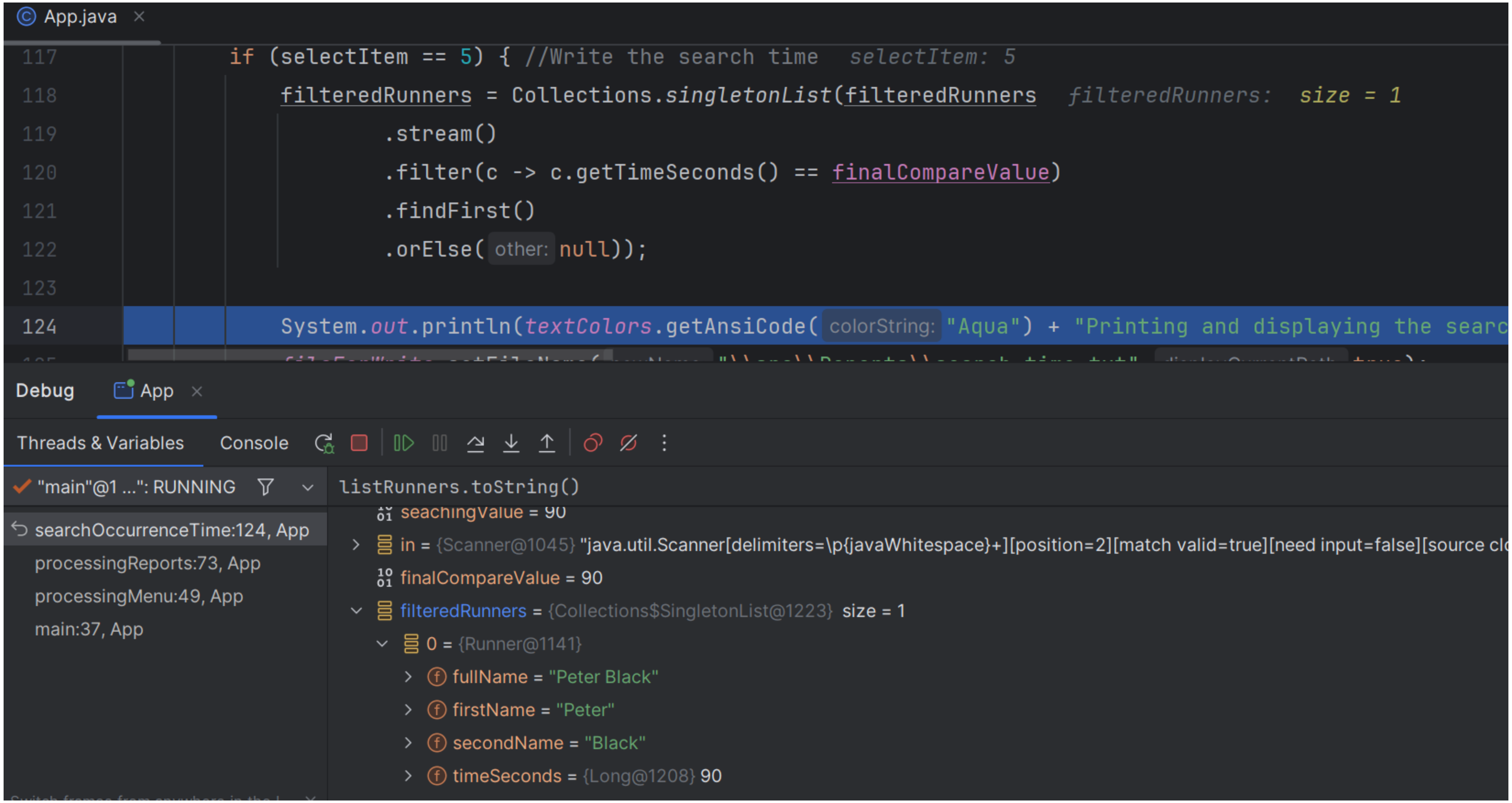
```
    if (selectItem == 5 = true ) { //Write the search time   selectItem: 5
        filteredRunners = Collections.singletonList(filteredRunners
                .stream()
                .filter(c -> c.getTimeSeconds() == finalCompareValue)
                .findFirst()
                .orElse( other: null));
```

```
searchOccurrenceTime:117, App
processingReports:73, App
processingMenu:49, App
main:37, App
```

```
listRunners.toString()
filteredRunners = {ImmutableCollections$ListN@1196}  size = 2
  0 = {Runner@1141}
    fullName = "Peter Black"
    firstName = "Peter"
    secondName = "Black"
    timeSeconds = {Long@1208} 90
  1 = {Runner@1142}
    fullName = "Richard Smith"
    firstName = "Richard"
    secondName = "Smith"
    timeSeconds = {Long@1208} 90
```

Step 2

found	Index	listRunners.Runner.timeSeconds	listRunners.Runner.fullName
True	6	90	Peter Black



App.java

```
        if (selectItem == 5) { //Write the search time   selectItem: 5
            filteredRunners = Collections.singletonList(filteredRunners   filteredRunners:  size = 1
                    .stream()
                    .filter(c -> c.getTimeSeconds() == finalCompareValue)
                    .findFirst()
                    .orElse( other: null));

            System.out.println(textColors.getAnsiCode( colorString: "Aqua") + "Printing and displaying the searc
```

Debug   App

Threads & Variables   Console

"main"@1 ...": RUNNING

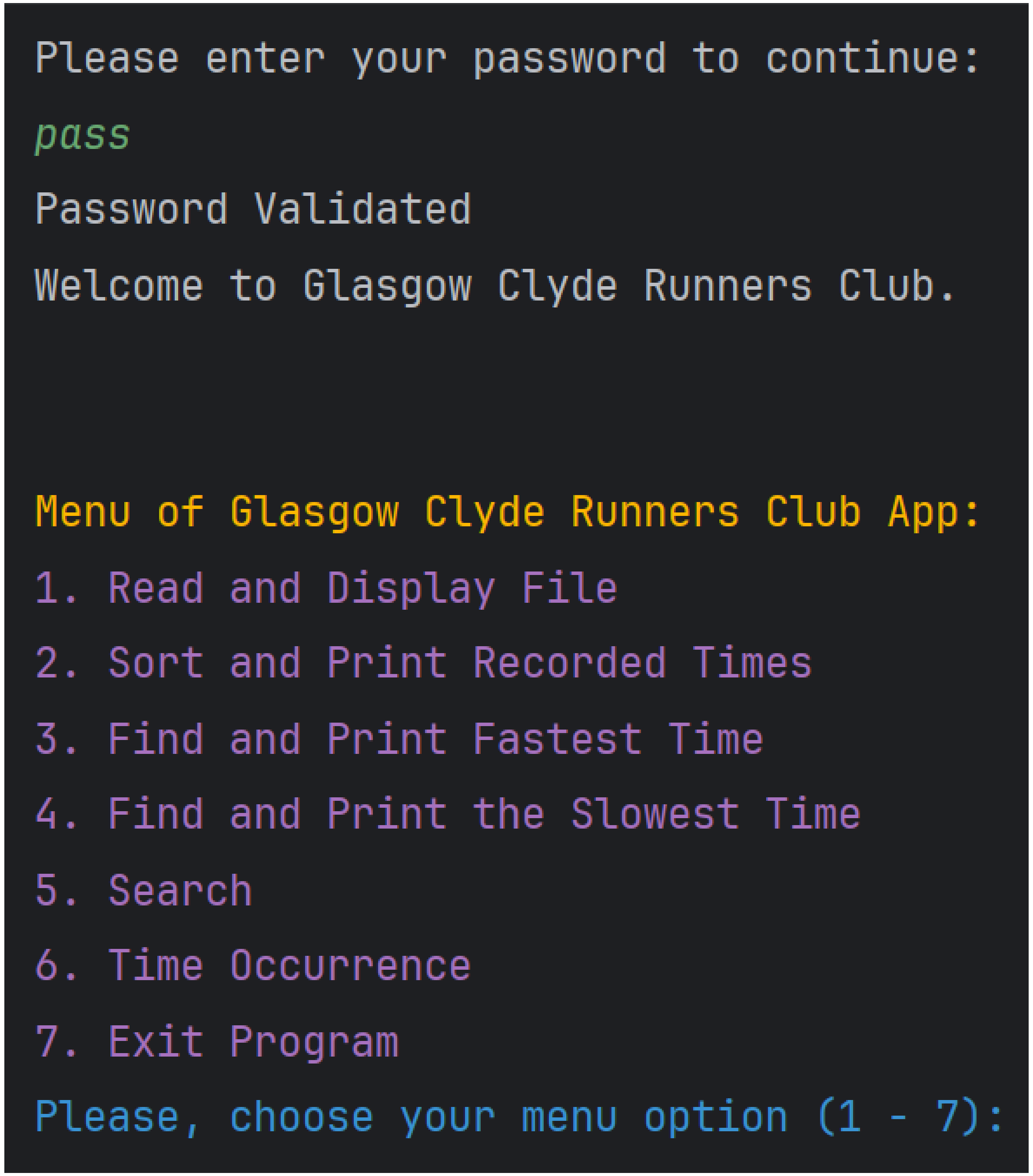
```
searchOccurrenceTime:124, App
processingReports:73, App
processingMenu:49, App
main:37, App
```

```
listRunners.toString()
    seachingValue = 90
  > in = {Scanner@1045} "java.util.Scanner[delimiters=\p{javaWhitespace}+][position=2][match valid=true][need input=false][source cl
    finalCompareValue = 90
  v filteredRunners = {Collections$SingletonList@1223}  size = 1
      v 0 = {Runner@1141}
          > fullName = "Peter Black"
          > firstName = "Peter"
          > secondName = "Black"
          > timeSeconds = {Long@1208} 90
```

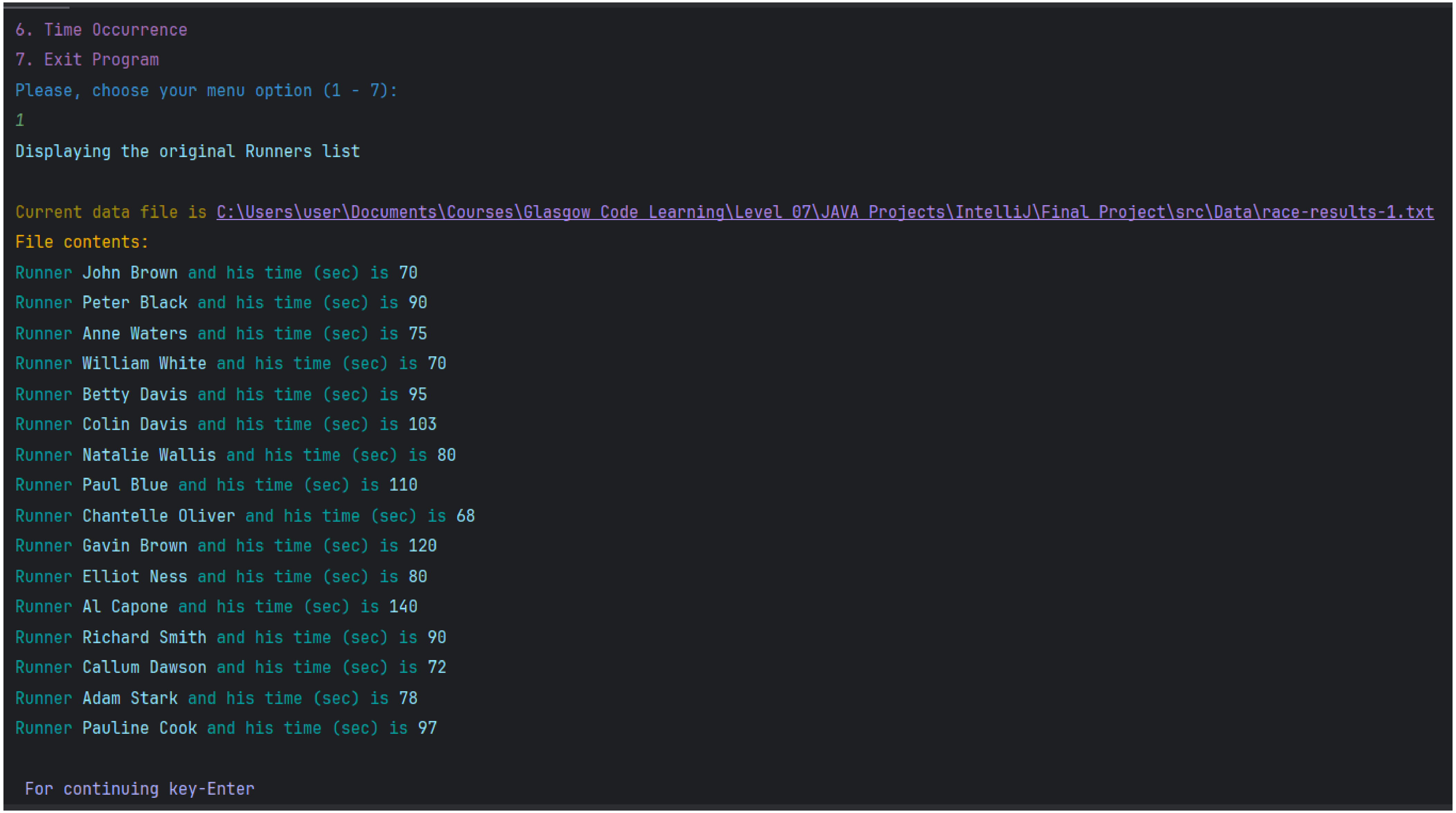
Walkthrough

This is an example of functional testing using a walkthrough, Black Box

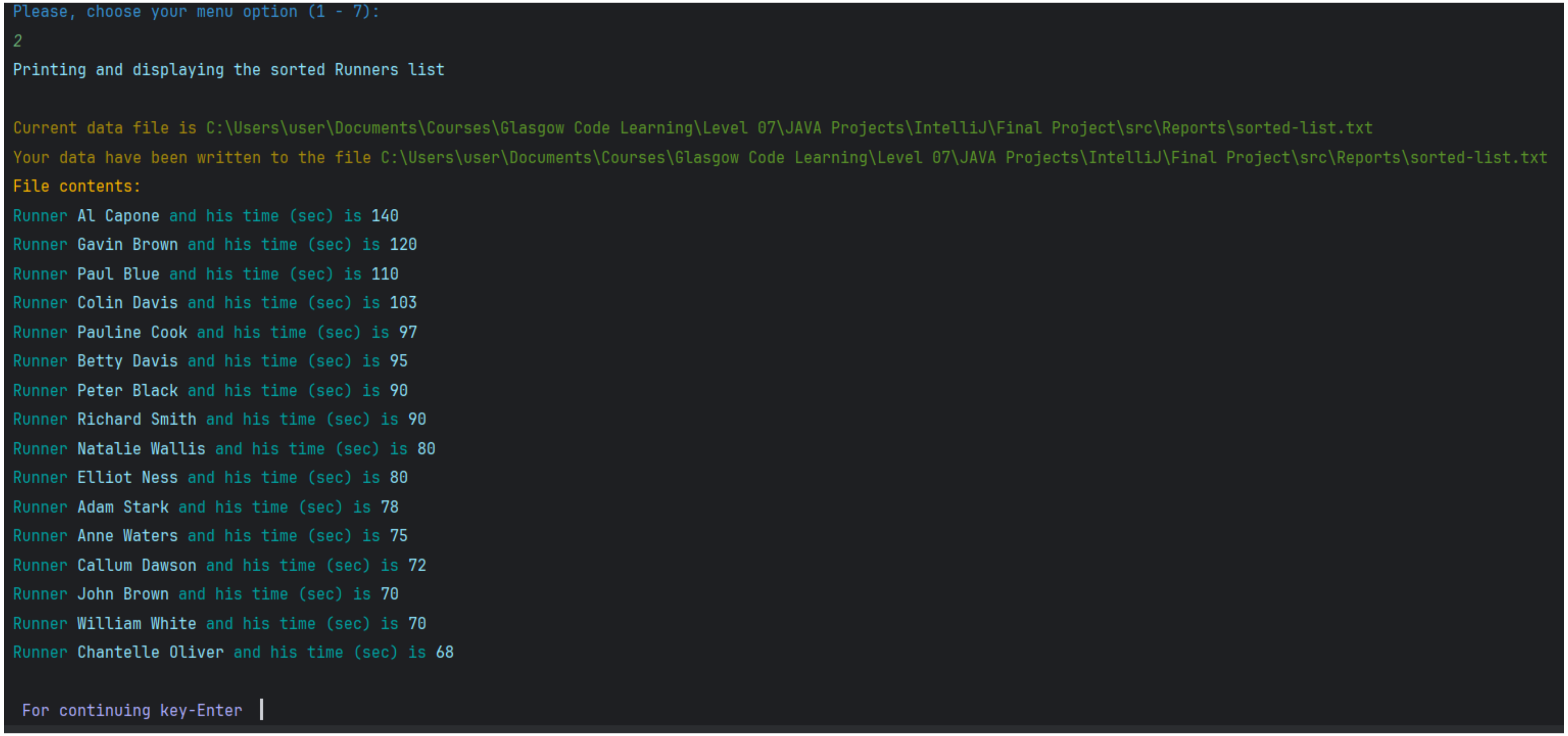
- You must enter the correct password "pass"
- Select menu item "1. Read and Display File"
- After reviewing the contents of the data file, press Enter
- Select menu item "2. Sort and Print Recorded Times"
- After reviewing the sorting result, press Enter
- Select menu item "3. Find and Print Fastest Time"
- After you have become familiar with who the fastest participant is, press Enter
- Select menu item "4. Find and Print the Slowest Time"
- After you have become familiar with who the slowest participant is, press Enter
- Select menu item "5. Search"
- When prompted for search time, enter the value "110"
- After reviewing the selection result, press Enter
- Select menu item "6. Time Occurrence"
- When prompted for search time, enter the value "90"
- After reviewing the selection result, press Enter
- Select menu item "7. Exit Program"



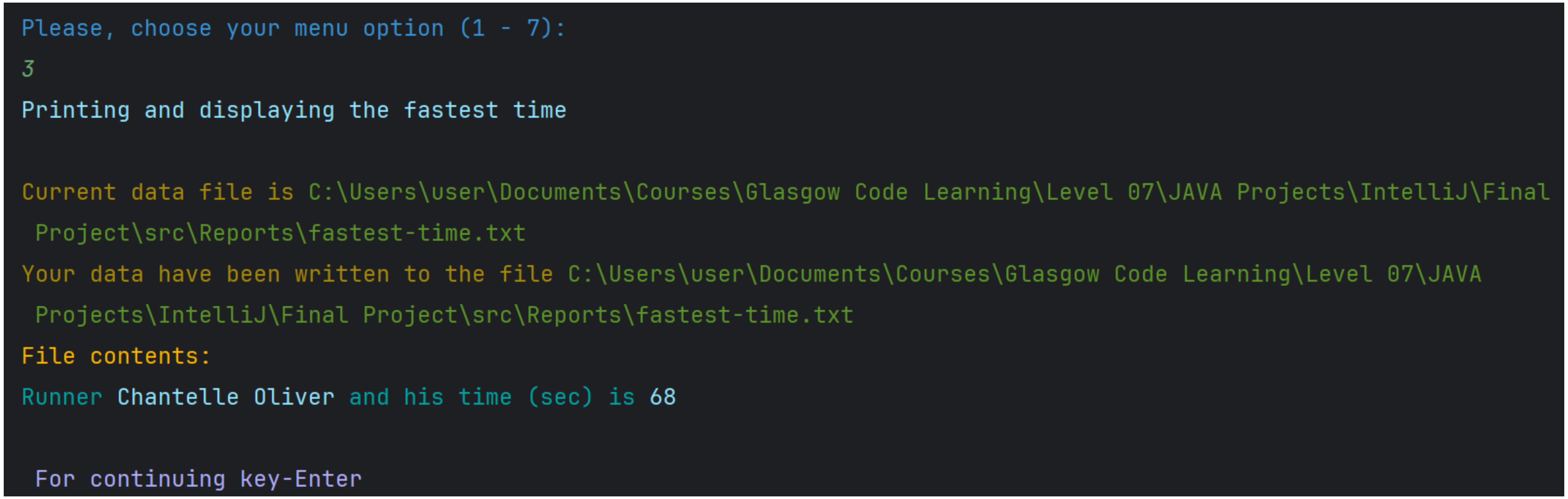
```
Please enter your password to continue:
pass
Password Validated
Welcome to Glasgow Clyde Runners Club.



Menu of Glasgow Clyde Runners Club App:
1. Read and Display File
2. Sort and Print Recorded Times
3. Find and Print Fastest Time
4. Find and Print the Slowest Time
5. Search
6. Time Occurrence
7. Exit Program
Please, choose your menu option (1 - 7):
```



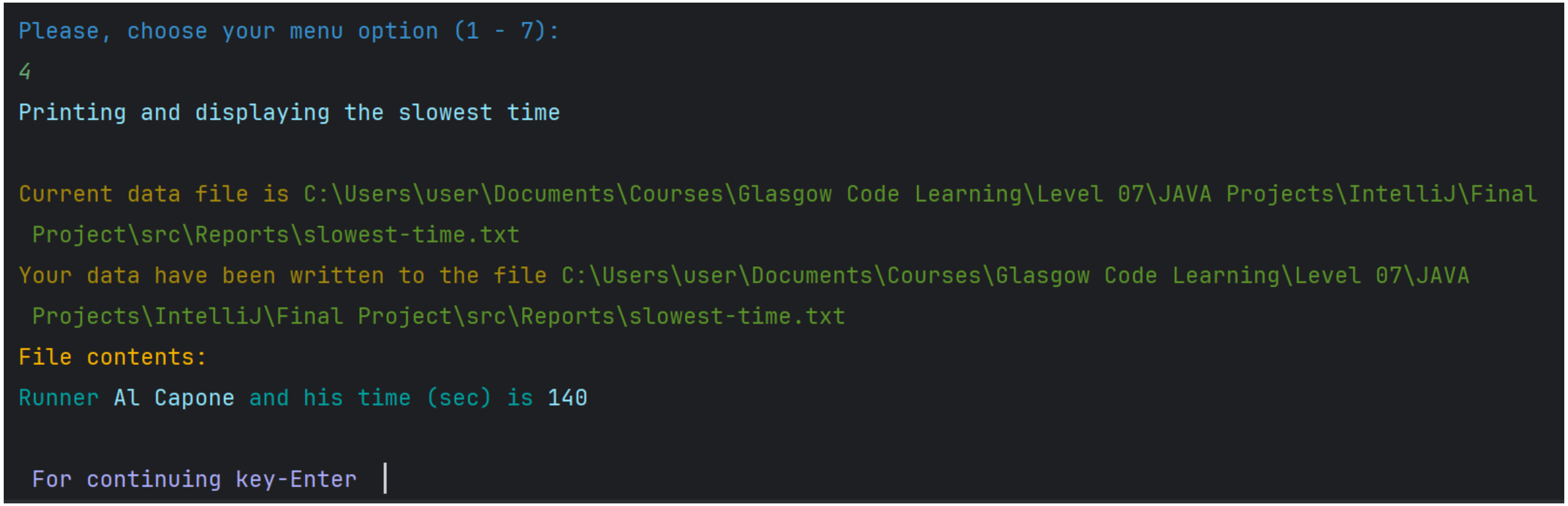
```
6. Time Occurrence
7. Exit Program
Please, choose your menu option (1 - 7):
1
Displaying the original Runners list

Current data file is C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final Project\src\Data\race-results-1.txt
File contents:
Runner John Brown and his time (sec) is 70
Runner Peter Black and his time (sec) is 90
Runner Anne Waters and his time (sec) is 75
Runner William White and his time (sec) is 70
Runner Betty Davis and his time (sec) is 95
Runner Colin Davis and his time (sec) is 103
Runner Natalie Wallis and his time (sec) is 80
Runner Paul Blue and his time (sec) is 110
Runner Chantelle Oliver and his time (sec) is 68
Runner Gavin Brown and his time (sec) is 120
Runner Elliot Ness and his time (sec) is 80
Runner Al Capone and his time (sec) is 140
Runner Richard Smith and his time (sec) is 90
Runner Callum Dawson and his time (sec) is 72
Runner Adam Stark and his time (sec) is 78
Runner Pauline Cook and his time (sec) is 97

 For continuing key-Enter
```



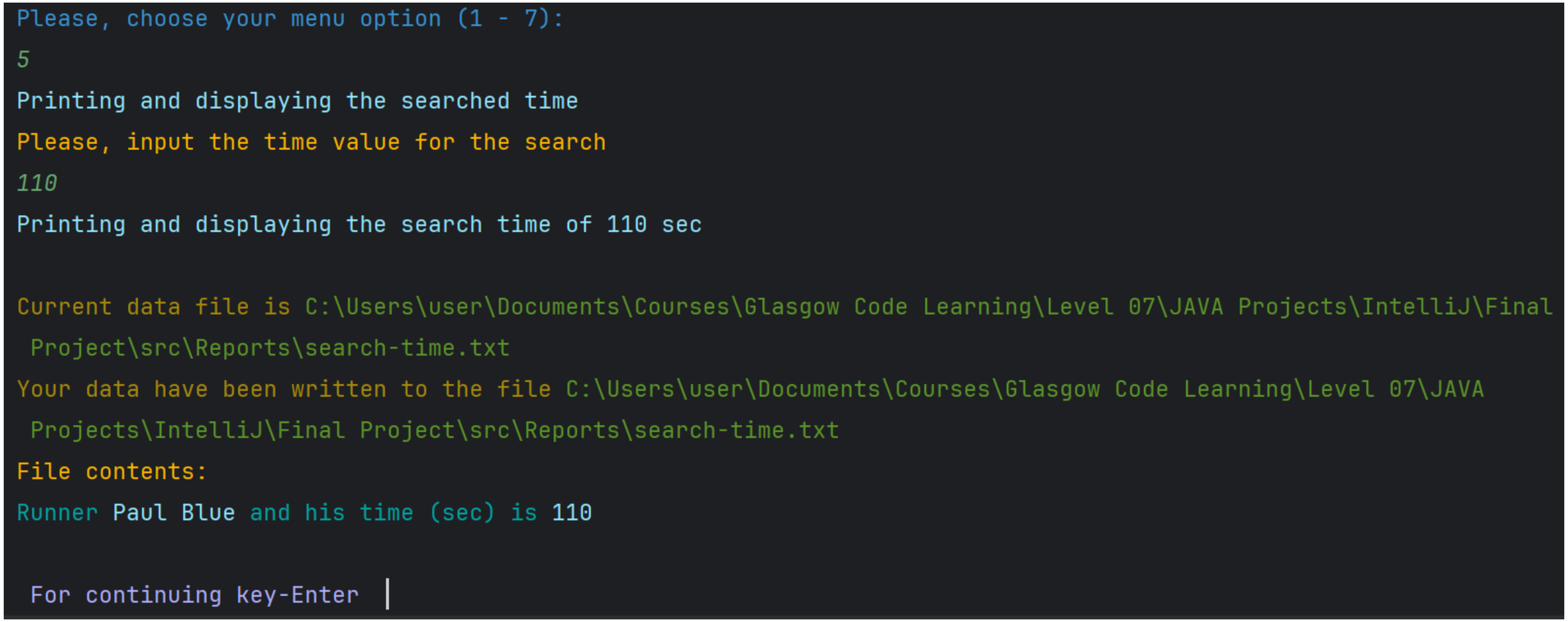
```
Please, choose your menu option (1 - 7):
2
Printing and displaying the sorted Runners list

Current data file is C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final Project\src\Reports\sorted-list.txt
Your data have been written to the file C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final Project\src\Reports\sorted-list.txt
File contents:
Runner Al Capone and his time (sec) is 140
Runner Gavin Brown and his time (sec) is 120
Runner Paul Blue and his time (sec) is 110
Runner Colin Davis and his time (sec) is 103
Runner Pauline Cook and his time (sec) is 97
Runner Betty Davis and his time (sec) is 95
Runner Peter Black and his time (sec) is 90
Runner Richard Smith and his time (sec) is 90
Runner Natalie Wallis and his time (sec) is 80
Runner Elliot Ness and his time (sec) is 80
Runner Adam Stark and his time (sec) is 78
Runner Anne Waters and his time (sec) is 75
Runner Callum Dawson and his time (sec) is 72
Runner John Brown and his time (sec) is 70
Runner William White and his time (sec) is 70
Runner Chantelle Oliver and his time (sec) is 68

 For continuing key-Enter
```



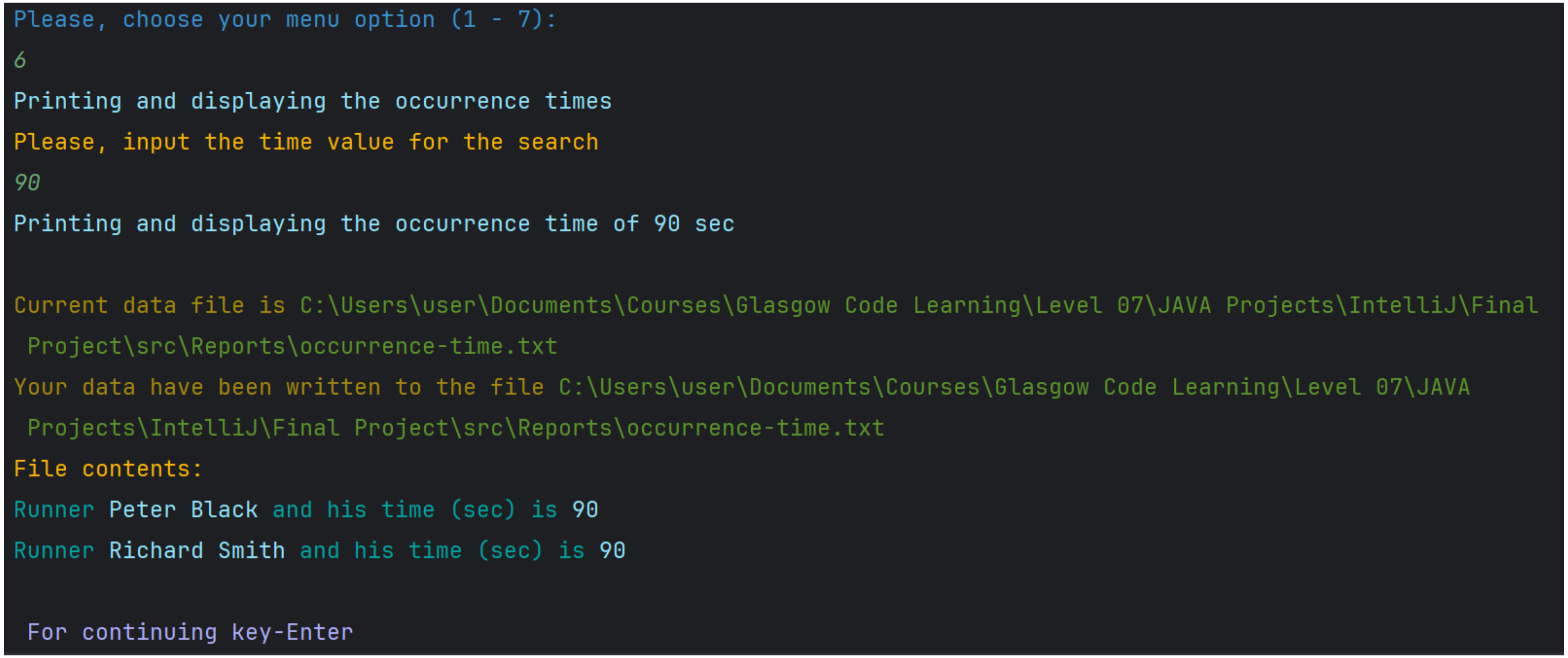
```
Please, choose your menu option (1 - 7):
3
Printing and displaying the fastest time

Current data file is C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final
 Project\src\Reports\fastest-time.txt
Your data have been written to the file C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA
 Projects\IntelliJ\Final Project\src\Reports\fastest-time.txt
File contents:
Runner Chantelle Oliver and his time (sec) is 68

 For continuing key-Enter
```



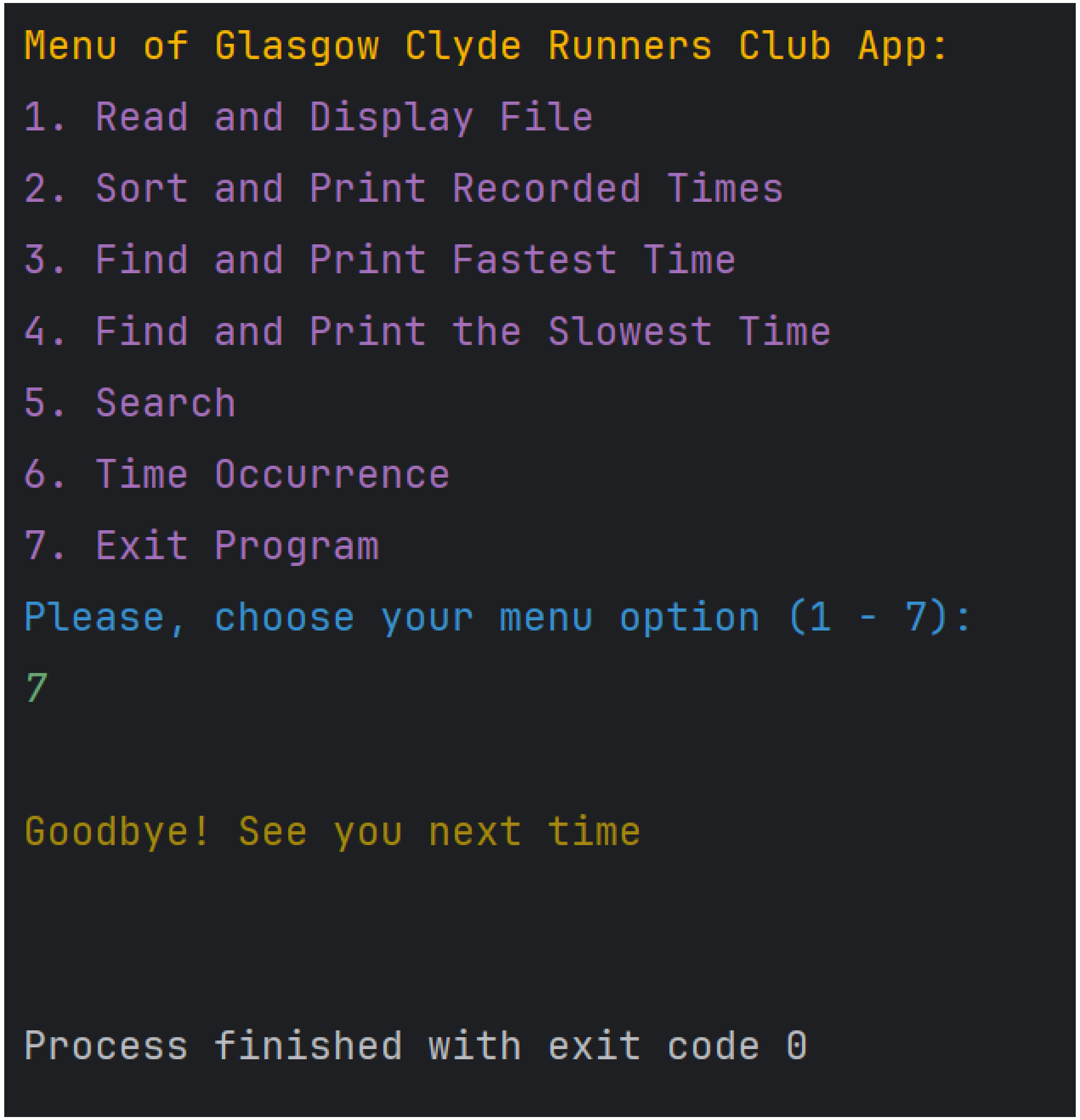
```
Please, choose your menu option (1 - 7):
4
Printing and displaying the slowest time

Current data file is C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final
 Project\src\Reports\slowest-time.txt
Your data have been written to the file C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA
 Projects\IntelliJ\Final Project\src\Reports\slowest-time.txt
File contents:
Runner Al Capone and his time (sec) is 140

 For continuing key-Enter
```



```
Please, choose your menu option (1 - 7):
5
Printing and displaying the searched time
Please, input the time value for the search
110
Printing and displaying the search time of 110 sec

Current data file is C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final
 Project\src\Reports\search-time.txt
Your data have been written to the file C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA
 Projects\IntelliJ\Final Project\src\Reports\search-time.txt
File contents:
Runner Paul Blue and his time (sec) is 110

 For continuing key-Enter
```



```
Please, choose your menu option (1 - 7):
6
Printing and displaying the occurrence times
Please, input the time value for the search
90
Printing and displaying the occurrence time of 90 sec

Current data file is C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA Projects\IntelliJ\Final
 Project\src\Reports\occurrence-time.txt
Your data have been written to the file C:\Users\user\Documents\Courses\Glasgow Code Learning\Level 07\JAVA
 Projects\IntelliJ\Final Project\src\Reports\occurrence-time.txt
File contents:
Runner Peter Black and his time (sec) is 90
Runner Richard Smith and his time (sec) is 90

 For continuing key-Enter
```



```
Menu of Glasgow Clyde Runners Club App:
1. Read and Display File
2. Sort and Print Recorded Times
3. Find and Print Fastest Time
4. Find and Print the Slowest Time
5. Search
6. Time Occurrence
7. Exit Program
Please, choose your menu option (1 - 7):
7


Goodbye! See you next time



Process finished with exit code 0
```