# **Project Design**

**Project Overview**

The purpose of this project is to design, implement and test a simple java program using the waterfall methodology.

Glasgow Clyde Runners Club manually sorts data captured from various races. They want a program that can read a text file, sort data, print the sorted data to a text file and to the console. They want to develop a trial program using test data as a proof of concept.

They would like a menu interface for their non-technical staff and results to be printed to a text file as well as a suitable message output to the user.The application to be password protected with code from a previous developer and the name of their club to be displayed in the program.

They would also like the Software to run without having to upgrade and for it to run from 03/05/2023 and be completed by 26/05/2023 for £10,000. They would like us to produce a trial program based on their last 2 recorded 400m final races.

The goal of This document is to produce:

* User Interface Design using a suitable method, eg Storyboard or Wireframe. For a console application your Login / Menu interface can be used.…**Page 2**
* Data Dictionary for the new system listing all the data items, their format and a brief description of their content…**Page 3**
* An overall design using Structure Chart. This should show the breakdown of the system into its lowest levels.…**Page 5**
* Detailed design of the system using pseudocode which explains the steps in an English-Like format…**Page 6**
* List of Algorithms and logic**…Page 7**
* Test Plan (How to plan to test your program i.e. black box / white box, types of test e.g. unit, integration, system, acceptance etc.)…**Page 12**

# **Interface Design**

Welcome to Glasgow Clyde Runners Club.

Please enter your password to continue:

Enter Password:

<User Input>

Password Validation

Enter Choice:

1: Get Names

2: Times

3: Fastest and slowest time

4: Search

5: Display all data and create sorted txt file

6: Exit

1->

Displays all names

2->

Displays all times

3->

Displays fastest and slowest time

4 - >

1: Search Name

2: Search Time

3: Back

<User Input>

1 - >

Enter letters from first OR last name (Not case sensitive):

User Input

Display matches

Call search()

2 - >

Enter Search term:

<User Input>

Display matches

Cal search()

3 - >

Enter Choice:

1: Get Names

2: Times

3: minMax

4: Search

5: Create sorted txt file

6: Exit

5->

Displays all sorted data and date created

Outputs same data to .txt file

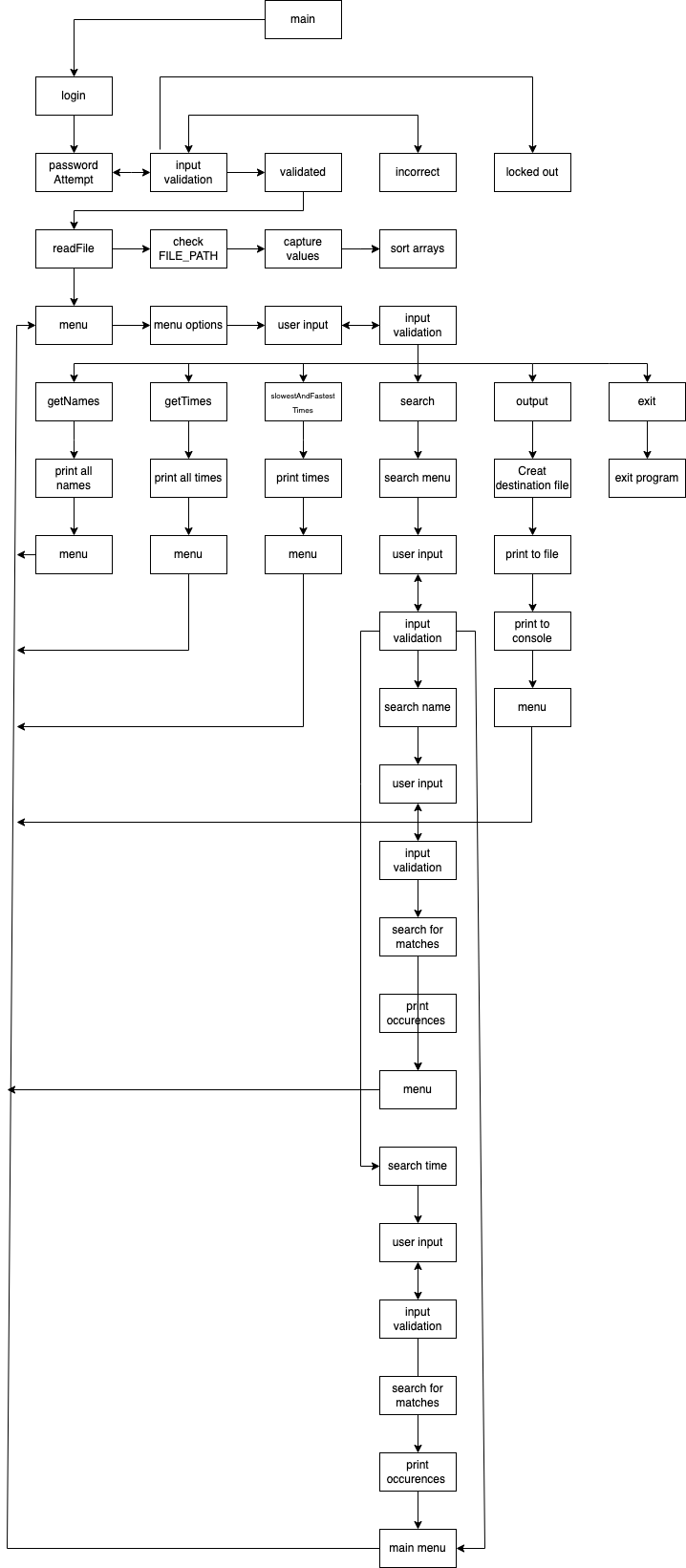
6->

Exit program

|  |  |  |
| --- | --- | --- |
| **Data Dictionary** | | |
| **Variable Name** | **Data Type** | **Description** |
| runners | 2D String Array | Stores data from text file. Each row represents a runner. Each column represents runners' data. |
| line | 1D String Array | Converts each element of a line of text file divided by a space and pushes them to runners array |

|  |  |  |
| --- | --- | --- |
| **Method Dictionary** | | |
| **Name** | **Type** | **Description** |
| Main | Main Method | Calls login method |
| Login | Method | Validates password calls readFile |
| readFile | Method | Reads and sorts text file then calls menu() |
| menu | Method | Displays options , Takes and validates user choice and calls other methods |
| getName | Method | Returns runners[n][0] and runners[n][1] |
| getTimes | Method | Returns runners[n][3] |
| Fastestandslowesttime | Method | Returns runners[0][2] + Return runners[15][2] |
| Search | Method | Displays search menu, validates user pattern, searches and displays occurrences |
| output | Method | Prints sorted data to text file and to console |

|  |  |  |
| --- | --- | --- |
| **Variable Dictionary** | | |
| **Name** | **Type** | **Purpose** |
| **login Method** | | |
| password | String | Holds Password |
| attempts | int | Tracks login attempts |
| passwordAttempt | String | Captures user password attempt |
| **readFile Method** | | |
| File\_Path | String | Specify text file to target |
| runner | 1D String Array | Iterate through runners |
| line | 1D String Array | Capture and separate values in row of text file |
| temp | 1D String Array | Sort values in array |
| **menu** | | |
| menuOption | String | Capture user input |
| **search** | | |
| choice | String | Capture user input |
| occurrences | int | Track search occurrences |
| pattern | String | Compare to values in arrays |
| numberPattern | String | Compare to values in arrays |
| **output method** | | |
| runner | 1D String Array | Iterate through array |
| date | Date | Capture current date and time |



**Pseudocode**

Main

* Calls login method

Login method

* Store password
* Capture password attempt
* Check password attempt against password
* Print validated or invalid
* Call read File or lock out

readFile

* Find file from specified path
* Capture values
* Sort values and sort runners array based on times
* Call menu

Menu

* 1 names
* 2 times
* 3 fastest and slowest time
* 4 search
* 5 display data and output file
* 6 exit
* Menu recursion

getNames

* prints first names plus last names

Fastest and Slowest time

* prints the first runners full name and time (runners [0])
* prints the last runners full name and time (runners[15])

Search

* Search Menu
* 1.Names
  + Capture input pattern
  + Check pattern
  + Display occurrences
  + Display search menu
* 2.Times
  + Capture input pattern
  + Check pattern
  + Display occurrences
  + Display search menu
* 3. Back to main menu

Output

* Print message and date to file
* Print message and date to console
* Iterate through runners array
  + Prints each runner to text file
  + Prints each runner to console

# **Algorithms**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| login | passwordAttempt | password validation, call menu and readFile methods or exit | Security |
| do {  System.out.println("Enter Password: ");  String passwordAttempt = in.nextLine();  if (passwordAttempt.equals(password)) {  System.out.println("Password Validated");  loggedIn = true;  readFile();    // Reduce attempts if password incorrect  } else {  attempts--;  System.out.println("Your Password is incorrect");  System.out.println("You have: " + attempts + " attempts left.");  }  } while (attempts != 0); | | | |
| While attempts is greater than 0  1.Prompt user to enter password  1.1 If passwordAttempt == password  1.2 Print a welcome to system message  1.3 Call readFile and menu  else  2. Print an invalid password message  2.1 attempt - 1  3.Break loop and print locked out message if attempts == 0  4.System exit | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| readFile | File\_Path | Pushes file data to runners array | Data handling |
| while (in.hasNextLine()) {  // Creates 'runner' for each index of runners and adds each value in each line  for (String[] runner : runners) {  String[] line = in.nextLine().split(" ");  System.arraycopy(line, 0, runner, 0, line.length);  }} | | | |
| While text file has a row of values   1. Create 1D runner array to iterate through index of 2D array runners 2. Create 1D line array and capture each row of text file and push values separated by white space to it 3. Push values in 1D array to 2D array | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| readFile | runners | Sorts runner array | Sorts from lowest int to highest |
| for (int i = 0; i < runners.length; i++) {  // Sorts runners array from fastest to slowest  for (int j = 1; j < runners.length; j++) {  if (parseInt(runners[j-1][2]) > parseInt(runners[j][2])) {  String[] temp = runners[j-1];  runners[j-1] = runners[j];  runners[j] = temp;  }}} | | | |
| 1.loop all values in runners  2.If one value is greater than another  3.Store all values of runner with greatest time in temp[]  3.1Switch all values in runner with lowest time to greatest runner time position  3.2Swap stored runner to greatest value runner position | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| menu | menuOption | Validates menuOption | Input validation |
| do {  menuOption = in.nextLine();  if (!menuOption.matches("^[0-9]\\d\*$")) {  System.out.println("Invalid input. Please enter a Number");  menuOption = "";  } else if (parseInt(menuOption) > 6 || parseInt(menuOption) < 1) {  System.out.println("Please choose options 1-6");  menuOption = "";  }  } while (menuOption.equals("")); | | | |
| 1.0 While menuOption(pattern) is empty  1.1 Capture user input menuOption  1.2 If pattern is not a digit  1.3 Print enter a number  1.4 Reset pattern to empty  2.0 Else if pattern is outside the range of menu options  2.1 Print choose options 1-6  2.2 Reset pattern to empty | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| menu | menuOption | Calls choice method | navigation |
| switch (Integer.parseInt(menuOption)) {  case 1 -> getNames();  case 2 -> getTimes();  case 3 -> fastestandSlowest();  case 4 -> search();  case 5 -> output();  case 6 -> System.exit(0);  } | | | |
| If option = 1 call getNames  If option = 1 call getTimes  If option = 1 call minMax  If option = 1 call search  If option = 1 call output  If option = 1 exit | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| getNames | runners | Every full name | Display data |
| for (String[] runner : runners) {  System.out.println(runner[0] + " " + runner[1]);  } | | | |
| 1.Loop For each runner in runners  1.1Print runner[0] (first name) + space + runner[1] (last name) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| getTimes | runners | Print every time | Display data |
| for (String[] runner : runners) {  System.out.println(runner[2]);  } | | | |
| 1.Loop for every runner in runners  1.2Print runner[2] (time) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| search | choice | Validates menu choice | Input validation |
| do {  choice = in.nextLine();  if (!choice.matches("^[1-9]\\d\*$")) {  System.out.println("Invalid input. Please enter a Number");  choice = "";  } else if (parseInt(choice) > 3) {  System.out.println(" Make a selection between 1-3");  choice = "";  }  } while (choice.equals("")); | | | |
| 1.0 While choice(menu selection) is empty  1.1 Capture use input choice  1.2 If choice is not a digit  1.3 Print enter a number  1.4 Reset choice to empty  2.0 Else if choice outside range of menu options  2.1 Print choose 1-3  2.2 Reset choice to empty | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| search | pattern | Validates name search term | Input validation |
| do {  pattern = in.nextLine();  if (!pattern.matches("[a-zA-Z]+$")) {  System.out.println("Invalid input. Please enter a string.");  pattern = "";  }  } while (pattern.isEmpty()); | | | |
| 1.0 While pattern is empty  1.1 If pattern doesn’t match a string  1.2 Print invalid enter a string  1.3 Reset pattern to empty | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| search | pattern | Matched names | Display data |
| for (String[] runner : runners) {  if (runner[0].toLowerCase().contains(pattern.toLowerCase()) || runner[1].toLowerCase().contains(pattern.toLowerCase())) {  occurrences++;  System.out.println("Occurence " + occurrences + " : " + Arrays.toString(runner));  }  } | | | |
| 1.0 Loop all runners names  1.1 If pattern matches a first or last name  1.2 Print occurrence and matched data from runners | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| search(2) | numberPattern | Validates time search term | Input validation |
| do {  numberPattern = in.nextLine();  if (!numberPattern.matches("^[1-9]\\d\*$")) {  System.out.println("Invalid input. Please enter a Number");  numberPattern = "";  }  } while (numberPattern.equals("")); | | | |
| 1.0 While numberPattern is empty  1.1 Capture user input  1.2 If input is not a digit  1.3 Print invalid input, enter a number  1.4 Reset numberPattern to empty | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| search(2) | numberPattern | Time matches | Find and display data |
| for (String[] runner: runners) {  if (numberPattern.equals(runner[2])) {  occurrences++;  System.out.println("occurrence: " + occurrences + " : " + Arrays.toString(runner));  }  } | | | |
| 1.0 Loop for all times  1.1 If numberPattern matches a times  1.2 Occurrences + 1  1.3 Print Occurrence + runner data | | | |

# **Test Plan**

Unit tests will be conducted on the login method as well as integration between login and menu methods after the readFile method has been added. Menu will be called from readFile to ensure data is processed before the user can access it.

Remaining methods will be tested during a walkthrough of the program as an entire system. Since some of the methods simply output data we will only test that they output the correct data.

Before the password code can run a few changes need to be made.

* Import Scanner utility and add a global public static scanner to capture user input.
* Wrap the code in its own login() method.
* Change the ‘Int’ to ‘int’ and comment out the call to ‘Menu()’.
* The integration test for login will simply repeat the tests for login once **the readFile method is added after test 4 after the code is integrated into the main program.Tests 1-4 will be completed again and if no differences are found no additional comments will be made. Menu will be called from readFile instead of login.**

Since Integration testing requires the algorithm in the readFile method to be complete the test plan for the algorithm is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithms and Logic** | | | |
| **Method** | **Input** | **Output** | **Purpose** |
| readFile | runners | Sorts runner array | Sorts from lowest int to highest |
| for (int i = 0; i < runners.length; i++) {  // Sorts runners array from fastest to slowest  for (int j = 1; j < runners.length; j++) {  if (parseInt(runners[j-1][2]) > parseInt(runners[j][2])) {  String[] temp = runners[j-1];  runners[j-1] = runners[j];  runners[j] = temp;  }}} | | | |
| 1.loop all values in runners  2.If one value is greater than another  3.Store all values of runner with greatest time in temp[]  3.1Switch all values in runner with lowest time to greatest runner time position  3.2Swap stored runner to greatest value runner position | | | |

* This bubble sort compares each value (j-1; at position i) to the adjacent value (j; i + 1).
* By comparing (j-1) to (j) we can sort a value across the entire array in each pass.
* In the first pass (j-1) = 4 and it is > 1 so the values are swapped. (the position of the runner is actually swapped in our example not just the values)
* Since 4 is not greater than any other value the (j) loop continues to compare values until it finds another (j-1 > j).
* As we can see below 11 is the highest value so it maintains its position as (j-1) because it is being swapped with each adjacent value (j) in the first pass.
* The outer (i) loop continues to iterate for length specified for (i)
* Since all the values have been sorted in the first pass with this data no changes are made as the outer loop (i) continues and *this is why bubble sort is not the best algorithm to use for most cases as it is not as efficient as algorithms such as quicksort*

Test array:[[4], [1], [11], [5], [7], [8]] (We will remove the outer array in the tables to make it easier to read as its only function is to hold data as an index.)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **j** | **j-1** | **j-1> j?** | **temp** | **Array before** | **Expected after** | **Actual** | **Comment** |
| 4 | 1 | 4 | T | 4 | [ 4, 1, 11, 5, 7, 8 ] | [ 1, 4, 11, 5, 7, 8 ] |  |  |
| 1 | 11 | 4 | F | - | [ 1, 4, 11, 5, 7, 8 ] | [ 1, 4, 11, 5, 7, 8 ] |  |  |
| 1 | 5 | 11 | T | 11 | [ 1, 4, 11, 5, 7, 8 ] | [ 1, 4, 5, 11, 7, 8 ] |  |  |
| 1 | 7 | 11 | T | 11 | [ 1, 4, 5, 11, 7, 8 ] | [ 1, 4, 5, 7, 11, 8 ] |  |  |
| 1 | 8 | 11 | T | 11 | [1, 4, 5, 7, 11, 8 ] | [ 1, 4, 5, 7, 8, 11 ] |  |  |

We can test the functionality using a trace table (Test data may be replaced with actual runner test data after implementation)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref | input | Expected output | Actual output | Comment |
| Normal | [ 4, 1, 11, 5, 7, 8 ] | [1, 4, 5, 7, 8, 11 ] |  |  |
| Exceptional | [ 4, 1, 11, 5.5, 7, 8 ] | Error |  |  |
| Extreme | [ 4, 1, 11, 5, @, 8 ] | Error |  |  |
| Normal | [ 4, 1, 11, 5, 7, -8 ] | [ -8, 1, 4, 5, 7, 11 ] |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| **login** | | | | | | |
| Security 1  Extreme | Tests wrong type | Variables password and login | @!£$\ | Incorrect 2 attempts left |  |  |
| Security 2  Exceptional | Tests combination type | Variables password and login | Runners@ | Incorrect 1 attempts left |  |  |
| Security 3  Normal | Tests valid value  Checks if password is compared rather than string length | Variables password and login | runnersclyde | Locked out |  |  |
| Security 4  Normal | Testing login  Integration | Variables password and login | clyderunneers | Validated, menu shown |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| readFile | | | | | | |
| Error Handling | Test try-catch | FILE\_PATH | Correct path | Display menu |  |  |
| Functionality | Test if read files correctly | FILE\_PATH | Incorrect path | Error |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| **menu** | | | | | | |
| Functionality | Tests output | MenuChoice | 1 | All names displayed |  |  |
| Functionality | Tests output | MenuChoice | 2 | All times displayed |  |  |
| Functionality | Tests output | MenuChoice | 3 | Fastest Time: [Chantelle, Oliver, 68]  Slowest Time: [Al, Capone, 140] |  |  |
| Functionality | Tests output | MenuChoice | 4 | 1: Search Name  2: Search Time  3: Back |  |  |
| Functionality | Tests output | MenuChoice | 5 | Produced file and displayed sorted data in console |  |  |
| Functionality | Tests output | MenuChoice | 6 | Process finished with exit code 0 |  |  |
| Input validation | Tests output | MenuChoice | A | Invalid input. Please enter a Number |  |  |
| Input validation | Tests output | MenuChoice | 19 | Please choose options 1-6 |  |  |
| Input validation | Tests output | MenuChoice | 0 | Please choose options 1-6 |  |  |
| Input validation | Tests output | MenuChoice | -1 | Please choose options 1-6 |  |  |
| Input validation | Tests output | MenuChoice | @ | Invalid input. Please enter a Number |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| Search Menu | | | | | | |
| Input validation | Tests out of range | choice | 0 | Make a selection between 1-3 |  |  |
| Input validation | Tests out of range | choice | -1 | Make a selection between 1-3 |  |  |
| Input validation | Tests out of range | choice | 4 | Choose options 1-3 |  |  |
| Input validation | Tests wrong type | choice | A | Invalid input. Please enter a Number |  |  |
| Input validation | Tests wrong type | choice | 1.1 | Choose options 1-3 |  |  |
| Input validation | Tests wrong type | choice | @ | Invalid input. Please enter a Number |  |  |
| Functionality | Tests valid value | choice | 1 | Enter letters from first OR last name (Not case sensitive): |  |  |
| Functionality | Tests valid value | choice | 2 | Enter time in seconds: |  |  |
| Functionality | Tests valid value | choice | 3 | Displays main menu |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| **Name search (1)** | | | | | | |
| Input validation | Tests wrong type | pattern | 1 | Invalid input. Please enter a string. |  |  |
| Input validation | Tests wrong type | pattern | -1 | Invalid input. Please enter a string. |  |  |
| Input validation | Tests wrong type | pattern | @ | Invalid input. Please enter a string. |  |  |
| Input validation | Tests combination t type | pattern | a1 | Invalid input. Please enter a string. |  |  |
| Functionality | Tests valid value | pattern | al | First Name occurrence: 1 Name: Al Capone Times: 140 Total Occurrences: 1 |  |  |
| Functionality | Tests valid value | pattern | Al | First Name occurrence: 1 Name: Al Capone Times: 140 Total Occurrences: 1 |  |  |
| Functionality | Tests valid value | pattern | Al cap | Occurence 1 : [Al, Capone, 140]  Total Occurrences: 1 |  |  |
| Functionality | Tests valid value | pattern | cap | Occurence 1 : [Al, Capone, 140]  Total Occurrences: 1 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| Time Search (2) | | | | | | |
| Input validation | Tests wrong type | NumberPatterm | a | Invalid input. Please enter a Number |  |  |
| Input validation | Tests wrong type | NumberPatterm | A | Invalid input. Please enter a Number |  |  |
| Input validation | Tests combination type | NumberPatterm | a1 | Invalid input. Please enter a Number |  |  |
| Input validation | Tests combination type | NumberPatterm | 1a | Invalid input. Please enter a Number |  |  |
| Input validation | Tests wrong type | NumberPatterm | al | Invalid input. Please enter a Number |  |  |
| Input validation | Tests wrong type | NumberPatterm | @ | Invalid input. Please enter a Number |  |  |
| Functionality | Tests valid value | NumberPatterm | -1 | Total Occurrences: 0 |  |  |
| Functionality | Tests wrong type | NumberPatterm | -1.2 | Total Occurrences: 0 |  |  |
| Functionality | Tests wrong type | NumberPatterm | 1.2 | Total Occurrences: 0 |  |  |
| Functionality | Tests wrong type | NumberPatterm | 1.12 | Total Occurrences: 0 |  |  |
| Functionality | Tests wrong type | NumberPatterm | 12.1 | Total Occurrences: 0 |  |  |
| Functionality | Tests wrong type | NumberPatterm | 12.12 | Total Occurrences: 0 |  |  |
| Functionality | Tests valid value | NumberPatterm | 1 | Total Occurrences: 0 |  |  |
| Functionality | Tests valid value | NumberPatterm | 12 | Total Occurrences: 0: |  |  |
| Functionality | Tests valid value  Make sure single digits from double digit times are not compared | NumberPatterm | 9 | Total Occurrences: 0: |  |  |
| Functionality | Tests valid value | NumberPatterm | 90 | occurrence: 1 : [Peter, Black, 90]  occurrence: 2 : [Richard, Smith, 90]  Total Occurrences: 2 |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Testing Plan and Outcome Tables** | | | | | | |
| **Ref.** | **Description** | **Item(s)** | **Input** | **Expected Output** | **Actual Output** | **Comment** |
| **getNames** | | | | | | |
| Functionality | Call from menu | runners | Called | Prints full name |  |  |
|  | | | | | | |
| Functionality | Call from menu | runners | Called | Prints slowest and fastest time |  |  |
|  | | | | | | |
| Functionality | Call from menu | Runners, date, | Called | Prints sorted list to console and file |  |  |