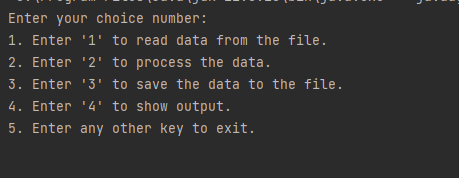
**DATA STRUCTURS AND ALGORITHMS:**

**Part 1:**

In the solution of this problem, we have 5 classes namely: “Person”, “Queue”, “Stack”, “Convert” and “Sort”. Person class holds the details of the input person, i.e. the name having each character in 8-bit format separated by semi-colons. Queue class is used to enqueue all the persons that are given in the input file to convert their names from 8-bit patterns to strings. Stack is used to achieve the conversion as described in the instruction manual. Finally, Sort class, as the name suggests is used to apply the insertion sort algorithm to sort the list of persons according to their Surname.

The flow of the program goes as:

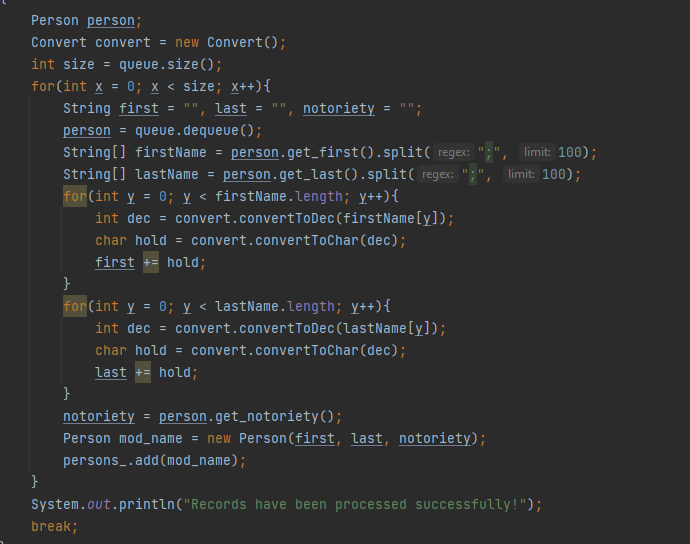
1. Menu Structure:



1. Reading from Input File:

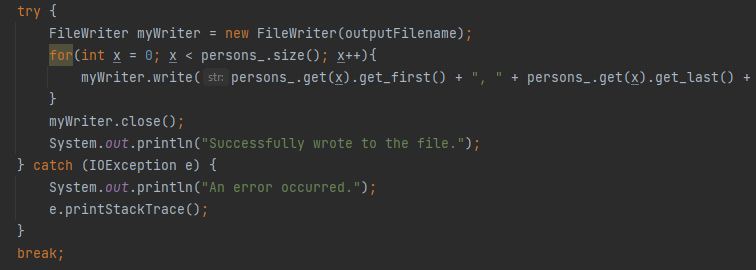


1. Converting the names:

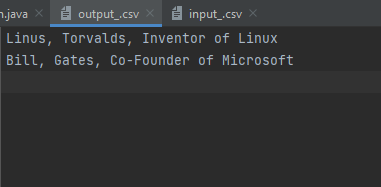


1. Output in the file:

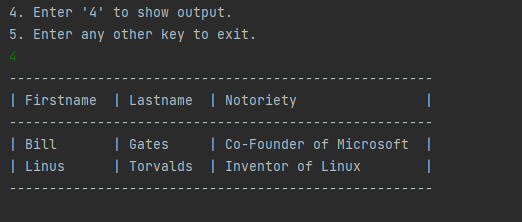
The code that does the functionality:



The output file looks like:



1. Output on console:



**Testing Data:**

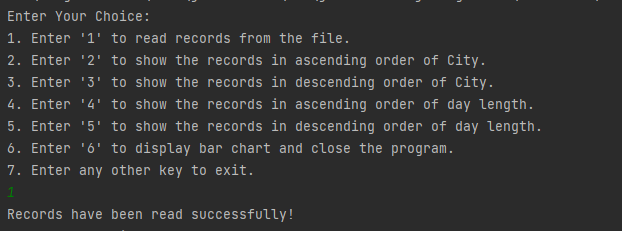
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case objective** | **Steps** | **Input data** | **Expected output** | **Actual output** | **Status(Pass/Fail)** |
| 1 | Convert binary to String | From queue, we extract the first name and last name, one by one to stack to calculate the total ascii value and get the character corresponding to that value | 01001100;01101001;01101110;01110101;01110011  ,01010100;01101111;01110010;01110110;01100001;01101100;01100100;01110011 | Linus, Torvalds | Linus, Torvalds | Pass |
| 2 | Convert Binary to String | Same as above | 01000010;01101001;01101100;01101100,01000111;01100001;01110100;01100101;01110011 | Bill, Gates | Bill Gates | Pass |
| 3 | Convert Binary to String | Same as above | 01000001;01101100;01100001;01101110,01010100;01110101;01110010;01101001;01101110;01100111 | Alan, Turing | Alan, Turing | Pass |
| 4 | Sort the names w.r.t last name | Apply insertion sort from the Sort class and pass the list of persons to get sorted index list | Linus Torvalds, Bill Gates, Alan Turing | Bill Gates, Linus Torvalds, Alan Turing | Bill Gates, Linus Torvalds, Alan Turing | Pass |

**Part 2:**

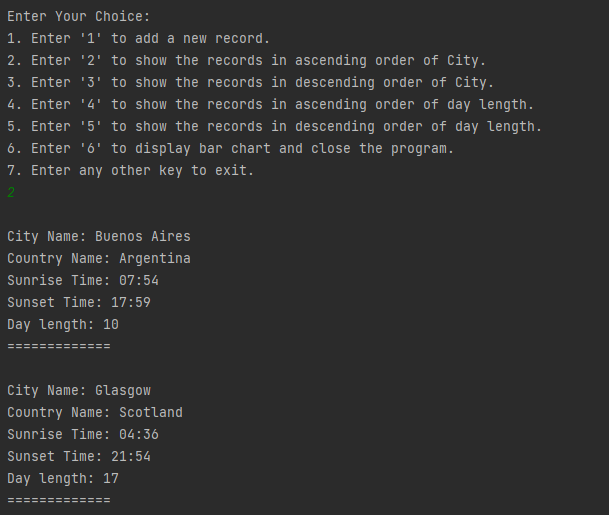
In the solution of this problem, we have 2 classes namely: “City” and “BST”. City class includes the details of the city provided and BST class arranges the cities according to the country in order to achieve sorting automatically. Furthermore, we have used JFX module to achieve the formation of bar charts.

The flow of the program goes as:

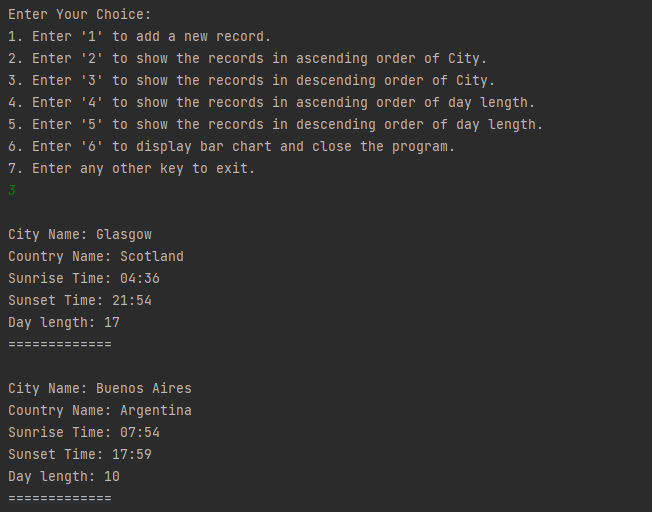
1. Reading the details of a city from the file and inserting to the tree:



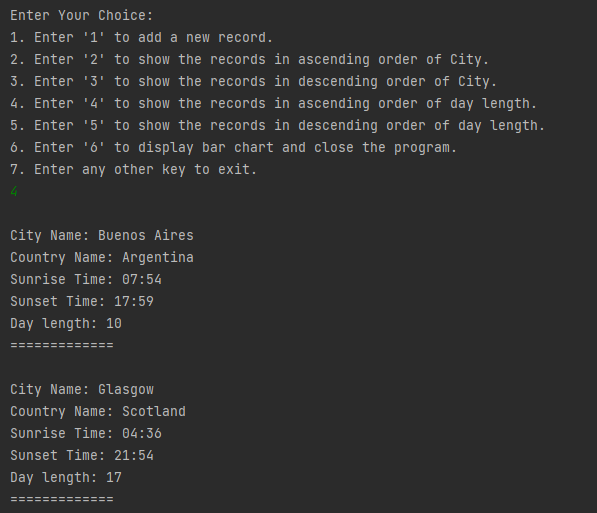
1. Show outputs as per the choice of the user:
2. City data in ascending order of Country:



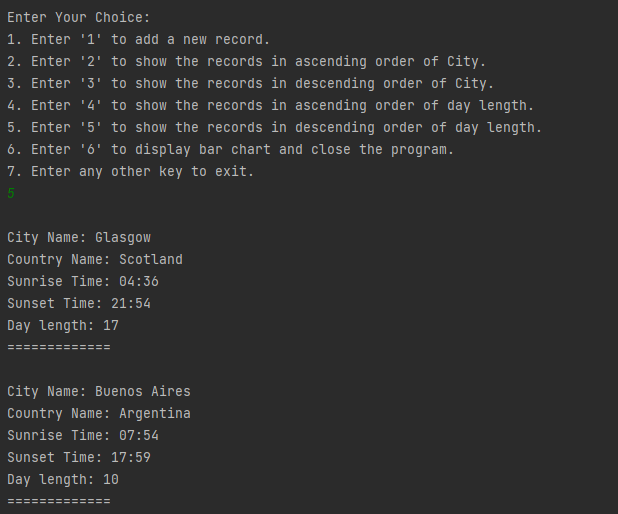
1. City data in descending order of Country:



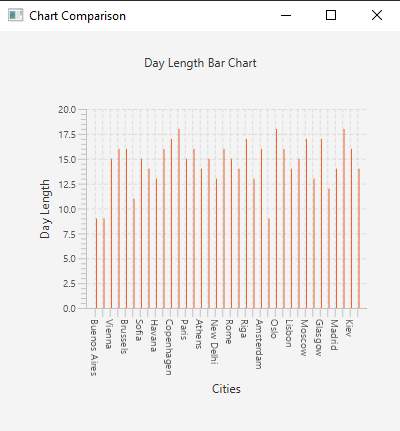
1. City data in ascending order of day length:



1. City data in descending order of day length:



1. Displaying bar chart:



**Testing Data:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case objective** | **Steps** | **Input data** | **Expected output** | **Actual output** | **Status(Pass/Fail)** |
| 1 | Show cities in ascending order of country | As data inserted in the binary search tree is on the basis of country, i.e. comparing country name, so we just print the tree in infix traversal to get sorted cities. | Minsk, Belarus; Warsaw, Poland; Budapest, Hungary; Vienna, Austria | Vienna, Austria, Minsk, Belarus; Budapest, Hungary; Warsaw, Poland | Vienna, Austria, Minsk, Belarus; Budapest, Hungary; Warsaw, Poland | Pass |
| 2 | Show cities in descending order of country | As data inserted in the binary search tree is on the basis of country, i.e. comparing country name, so we just print the tree by following “RNL” traversal to get sorted cities. | Minsk, Belarus; Warsaw, Poland; Budapest, Hungary; Vienna, Austria | Warsaw, Poland; Budapest, Hungary; Minsk, Belarus; Vienna, Austria | Warsaw, Poland; Budapest, Hungary; Minsk, Belarus; Vienna, Austria | Pass |
| 3 | Show cities in ascending order of day length | Extract all the cities from the tree to a list and apply bubbleSort in ascending order on the basis of day length | Minsk, Belarus; Warsaw, Poland; Budapest, Hungary; Vienna, Austria | Vienna, Austria; Budapest, Hungary; Minsk, Belarus; Warsaw, Poland | Vienna, Austria; Budapest, Hungary; Minsk, Belarus; Warsaw, Poland | Pass |
| 4 | Show cities in descending order of day length | Extract all the cities from the tree to a list and apply bubbleSort in descending order on the basis of day length | Minsk, Belarus; Warsaw, Poland; Budapest, Hungary; Vienna, Austria | Warsaw, Poland; Minsk, Belarus ; Budapest, Hungary; Vienna, Austria | Warsaw, Poland; Minsk, Belarus ; Budapest, Hungary; Vienna, Austria | Pass |