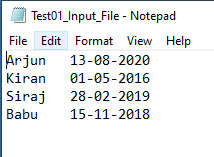
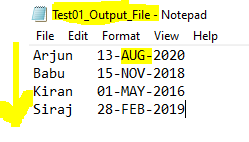
**CORE JAVA EXAM 01**

**Program 1:**

* **Test01\_Input\_File.txt** contains employee details (Employee Name and Joining Date).



* The date mentioned in the file would be in DD-MM-YYYY format.
* Write a program to read the Test01\_Input\_File from any of your local path and write the same details in **Test01\_Output\_File.txt** in below expected format and order.
* Expected Format:
  + The lines should be arranged in ascending order of employee name.
  + The Date Format should be changed to DD-MMM-YYYY (Example 01-JAN-2020)



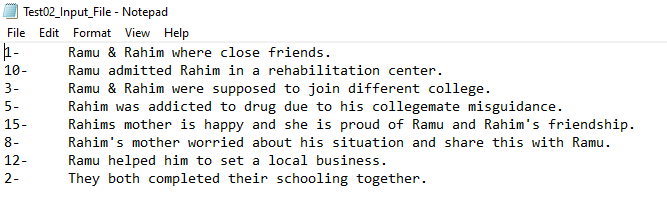
* Expected Cases:
  + If there is no file in the desired path, custom exception must be thrown stating, ‘No Input File to Process the Request’
  + If there are two employees are in same name, give high preference to employee with earlier joining date.
  + If there is any date format error, custom exception must be thrown stating,

‘Error in Data, kindly check the joining date for Arjun’

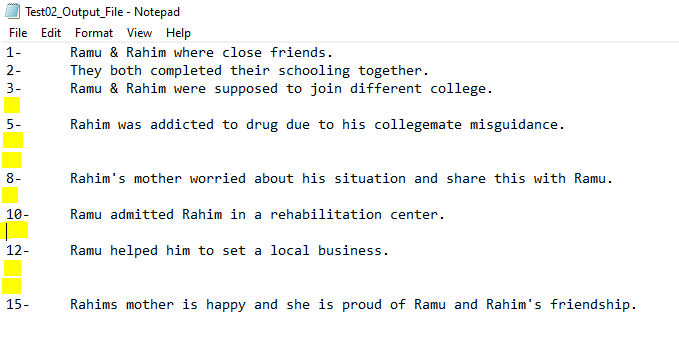
Example: 

**Program 2:**

* **Test02\_Input\_File.txt** contains a story with their lines jumbled.



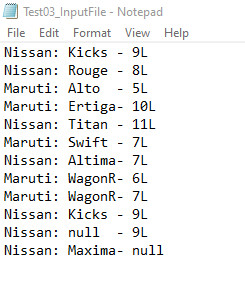
* Write a program to read the file from desired folder and write the same in **Test02\_Output\_File.txt** with their lines arranged in ascending order.
* Note: In case a numeric is missing, a blank line should be inserted.
* Expected output:



* Expected Cases:
  + If there is no file in the desired path, custom exception must be thrown stating, ‘No Input File to Process the Request’
  + If there is already a blank line in the input file, should ignore them in output file.

**Program 3:**

* **Test01\_Input\_File.txt** contains details of different car manufacturer and their model name and its market price.



* Write a Java program to group the Manufacturer and its Model and Market Price in a collection and arrange them in ascending order and print the collection in console.
* Expected:
  + Collections in Java must be used.
  + **All Models inside the collection should be listed in ascending order**.
  + If there are data for same model but different in price, only the model with high market price should get added to final result.
  + Null Values in Model should be ignored in final result.
  + Null Values in Market Price can be accepted and replace it with **Nil** in final result.

**RESULT:**



**Program 4:**

* Write a program to delete all **.txt** files from a desired folder and list the number of files deleted in console.

Expected Result:

* + Throw a custom defined exception if the folder is empty stating, ‘The Folder doesn’t contain any text files to trash.’
  + Throw a custom defined exception if the folder contains any other file formats stating, ‘There are some documents other than text files, backup them.’

**Program 5:**

* Write a program to add all digits of an integer and throw custom defined exception based on its result.
  + Naming conventions should be followed.
  + Add all digits and check if the sum is even or odd number.
  + Use try catch blocks.
  + Create 3 custom defined exception
    - OddNumberExcpetion
    - EvenNumberException
    - ZeroException
  + If the provided number is zero, then throw Zero Exception.
  + If the sum of digits of provided number is odd, then throw Odd Exception.
  + If the sum of digits of provided number is even, then throw Even Exception.