**Name format -** 1st\_april\_assignment\_1\_employee\_list

**Email -** nilay.das@hotmail.com

**Employees List**

Create a class ‘Employee’ with the following fields.

Int id;

String name;

String department;

Date dateOfJoining;

Int age;

Int salary;

All the fields should be declared as private and for each of the fields generate a pair of getter and setter as required.

The value for the ‘id’ field should be automatically generated by adding 1

To the id of the last employee.

Define a parameterised constructor function for initialising the fields

For each employee.

Implement the ‘Comparable’ interface into the class

and override the ‘int compareTo(Employee)’ to compare any 2 ‘Employee’

objects by salary’

Override the String toString() method to print the details of any ‘Employee’

Object in the following format

"%-15s %-30s %-30s %-10s %-10s\n"

Create a 2nd class ‘AgeComparator’ into which you implement the ‘Comparator’ interface to compare any 2 employees by age

And if the age are equal compare on dateOfJoining.

Create a 3rd class EmployeeBO for which you define a static method

printEmployees() that receives a list of ‘Employee’ objects and prints the employees’ details.

Create the last class ‘Main’ for which you define the main() method for receiving the inputs, printing a menu for with options to sort the list by salary/

By age and then by dateOfJoining.

Sample Input 1:

Input the number of employees :

3

Enter the details for employee 1

Rohini

Data Analysis

10/10/2000

45

90000

Enter the details for employee 2

Ranganathan

Production

09/10/2000

45

92000

Enter the details for employee 3

Pankaj

Marketing

02/02/2002

38

75000

Output 1

1.Sort employees by salary

2.Sort employees by age and by date of joining

Enter your choice

2

Employee ID Name Department Date Of Joining Age Salary

3 Pankaj Marketing 02/02/2002 38 75000

2 Ranganathan Production 09/10/2000 45 92000

1 Rohini Data Analysis 10/10/2000 45 90000

Sample Input 2:

Input the number of employees :

3

Enter the details for employee 1

Rohini

Data Analysis

10/10/2000

45

90000

Enter the details for employee 2

Ranganathan

Production

09/20/2000

45

92000

Enter the details for employee 3

Pankaj

Marketing

02/02/2002

38

70000

Output 2

1.Sort employees by salary

2.Sort employees by age and by date of joining

Enter your choice

1

Employee ID Name Department Date Of Joining Age Salary

3 Pankaj Marketing 02/02/2002 38 70000

1 Rohini Data Analysis 10/10/2000 45 90000

2 Ranganathan Production 09/20/2000 45 92000

1. Q2. **Password** **Validation**

Create a class ‘UserMainCode’ and for the class define a static method for validating the password received as parameter.

public static boolean checkPassword(String password){}.

A valid password should satisfy the following rules.

1. It should be minimum of 8 characters in length.
2. It should have at least one lower case letter,one upper case letter,

one digit and one special character

Create a ‘Main’ class and define th ‘main()’ method for input,output

Sample Input 1

Night#321

Output :Valid Password

Sample input 2

night#321

Output : Invalid Password

Q3.

Comparable - Display State

Write a Java program to get the country names and state names from the user seperated by a pipe symbol. Finally display all the countries and their states sorted in ascending order based on their names.  
  
Create a main class "Main.java"  
Create country class with below attributes,

* name - String
* stateList - List<State> (All state object for this country is stored in this list)

Add appropriate getter and setter methods for **Country** class  
Include a constructor accepting country name as a parameter  
Below are the methods in country class

* addState(String statename) - Add the new state to this country object
* getStateList() - Sort the state collection and return the list

Create **State** class with single attribute **name**  
Add appropriate getter and setter methods for State class  
Include a constructor with single argument state name  
Implement Comparable interface in the State class and implement the method compareTo()  
  
**Input and Output Format:**  
First input corresponds to the number of input elements and followed by country and state information in the format countryname|statename.  
Display the state name followed by two hyphen(-)  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :  
10  
India|Tamilnadu  
India|Kerala  
India|Odisha  
USA|Texas  
USA|Mississippi  
USA|Alaska  
India|Punjab  
Australia|Victoria  
Australia|Tasmania  
Australia|Queensland**  
Countries and States in ascending order  
Australia  
--Queensland  
--Tasmania  
--Victoria  
India  
--Kerala  
--Odisha  
--Punjab  
--Tamilnadu  
USA  
--Alaska  
--Mississippi  
--Texas  
**------------------------**

 Comparable - Contact Information Based on Mobile Number

Write a Java program to read all the contact information from the user and display the contact name and their mobile number sorted based on their mobile number (descending order). The contact details consist of name, email, mobile and address. Use Collections.sort() method for sorting.

Create a main class "Main.java"  
Create Contact class with below attributes  
name - String  
email - String  
mobile - Long  
address - String  
Add appropriate getter and setter methods for Contact class  
Include a constructor for Contact class with the arguments name, email, mobile and address  
Implment Comparable interface and implement the method compareTo() to perform sorting based on mobile number

Input and Output Format:  
First input corresponds to the number of contacts and followed by each contact information.  
Refer sample input and output for formatting specifications.

[All text in bold corresponds to input and the rest corresponds to output]  
Sample Input/Output :  
Enter number of contacts:  
2  
Enter contact 1 detail  
Enter Name  
Amar  
Enter Email  
[amar@gmail.com](https://mail.cognizant.com/owa/redir.aspx?REF=NrLMBWWMmGsQZAHZuxVxQ7lAECsTcwEMWskF1qMMzreYcME-PAbUCAFtYWlsdG86YW1hckBnbWFpbC5jb20.)  
Enter Mobile  
7200762700  
Enter Address  
Coimbatore  
Enter contact 2 detail  
Enter Name  
Thana  
Enter Email  
[thana@gmail.com](https://mail.cognizant.com/owa/redir.aspx?REF=8HVC8uTzE53c4S6TDvtTHpTH4LCMmiYDSzg4hZPlHNOYcME-PAbUCAFtYWlsdG86dGhhbmFAZ21haWwuY29t)  
Enter Mobile  
9566905846  
Enter Address  
Karur  
Contact list after sort by mobile number in descending order  
Thana-9566905846  
Amar-7200762700

4.

TreeMap - Letter Frequency

Write a Java program to calculate the character frequency in a sentence. The input consist of a single sentence and the output display a graphical chart displaying the freqency of each character by number of asterisk (\*). Display the character in the output in alphabetical order. Compute the frequence of all letters except space.  
  
Use TreeMap to store the characters and frequency since the tree map maintains the entries sorted based on their natural ordering.  
  
Create a main class "Main.java"  
Create a class **LetterSequence** and include below methods and attributes,  
Include a constructor to get the sentence as the input

|  |  |
| --- | --- |
| **Method/Attribute** | **Details** |
| public TreeMap<Character,Integer> computeFrequency() | Compute the frequency of each character in the sentence and store it in the TreeMap. Return the TreeMap after the computation. |
| public void displayLetterFrequency(TreeMap<Character,Integer> frequencyMap) | Iterate the tree map and get all the entries and print the information in a graphical view as shown the sample output |
| private String sentence | Input sentence is stored in this attribute |

**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Enter the input string  
**Refer sample input and output for formatting specifications**  
R : \*  
a : \*\*\*\*  
c : \*\*  
d : \*  
e : \*\*\*\*  
f : \*\*\*\*  
g : \*  
i : \*\*\*\*\*  
l : \*  
m : \*\*  
n : \*\*\*\*  
o : \*\*\*\*  
p : \*\*\*\*  
r : \*\*\*  
s : \*\*\*  
t : \*\*\*\*\*\*  
u : \*\*\*