Module 2

Handson for module 1

MySql server starting issue (if anybody has)

GIT

Sonar Qube (start server needs elevation)

HTML

HyperText Markup Language

It is a markup language

There will be tags. <HTML> </HTML>

It is based on SGML Standard Generalized Markup Language

SGML had huge vocabulary and allowed us to create our own tags.

HTML is simple version of SGML

Why HTML is required?

We need HTML to create web pages.

Front end for any web application is a web page. So HTML cannot be avoided when we create web pages.

HTML programs are not compiled.

They are directly interpreted top down. Open the html file in browser. It is executed.

Internet Explorer (Edge)

Google Chrome

Safari

Opera

Netscape Navigator

Mozilla Firefox

Version 5 of HTML is used now a days.

Reminder: HTML5 features

Basic rules of any markup language:

1. Entire document should be inside a root element

In html page, the root element is <HTML> </HTML>

1. Every start tag must have end tag otherwise it is an empty tag

<BODY BGCOLOR=”blue”>

<HR/> <BR/>

</BODY>

1. Everything in HTML or markup is a text only. Values of attributes must be in double quotes

<P ALIGN=”LEFT”>This is a paragraph</P>

<TABLE BORDER=”2”> </TABLE>

In this example, TABLE is called as Element (tag). BORDER is called as Attribute.

1. Entities start with & and ends with ;

&nbsp;

How will you print a paragraph that displays following:

A<B

<p> A&lt;B</p>

A>B

<p> A&gt;</p>

A<=B

<p> A&le;</p>

Element

Attribute

Entity

<HEAD> Whatever we write in HEAD portion are loaded in memory even before the page is displayed

</HEAD>

<BODY> Whatever we want to display, we will put here

</BODY>

Lets write our first HTML program now.

<html>

<head>

<title>This is the title displayed in title bar of browser</title>

</head>

<body>

<h1>Largest heading</h1>

<h2>Sub title 2</h2>

<h3>Sub title 3</h3>

<p> This is a paragraph</p> <p>Another paragraph</p>

</body>

</html>

Identify the tags which can be container tags. (that can contain other tags inside)

Identify the empty tags.

Identify the block elements and inline elements.

Container tags:

Html

Head

Body

P

Span (inline)

Div

Table

Form

What is the difference between inline and block element?

Div is a block element

Span is an inline element

A block element is displayed in separate line

An inline element is displayed in the same line

Java\_Mock\_24 Requirement # 6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8584)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | s2 | 5 Joe,Incoming,STD,00:01:50,30-06-2018 David,Outgoing,Local,00:10:15,22-05-2018 John,Outgoing,STD,00:20:25,01-01-2018 Jane,Outgoing,ISD,00:15:20,03-08-2017 Choe,Outgoing,STD,00:30:03,31-05-2018 | Enter the number of calls: Contact  Type     Call Type  Date         Duration   Cost Joe      Incoming STD        30-06-2018   00:01:50   0.00 David    Outgoing Local      22-05-2018   00:10:15   5.54 John     Outgoing STD        01-01-2018   00:20:25   14.70 Jane     Outgoing ISD        03-08-2017   00:15:20   138.00 Choe     Outgoing STD        31-05-2018   00:30:03   21.64 | General |  | | 2 | s4 | 3 Wayne,Incoming,Local,00:05:35,10-07-2017 Joe,Incoming,STD,00:01:50,30-06-2018 John,Incoming,STD,00:01:50,02-05-2018 | Enter the number of calls: Contact  Type     Call Type  Date         Duration   Cost Wayne    Incoming Local      10-07-2017   00:05:35   0.00 Joe      Incoming STD        30-06-2018   00:01:50   0.00 John     Incoming STD        02-05-2018   00:01:50   0.00 | General |  | | 3 | s1 | 5 John,Incoming,STD,00:01:50,02-05-2018 Grace,Incoming,ISD,00:10:55,28-07-2018 Wayne,Incoming,Local,00:05:35,10-07-2017 Joe,Incoming,STD,00:01:50,30-06-2018 David,Outgoing,Local,00:10:15,22-05-2018 | Enter the number of calls: Contact  Type     Call Type  Date         Duration   Cost John     Incoming STD        02-05-2018   00:01:50   0.00 Grace    Incoming ISD        28-07-2018   00:10:55   49.13 Wayne    Incoming Local      10-07-2017   00:05:35   0.00 Joe      Incoming STD        30-06-2018   00:01:50   0.00 David    Outgoing Local      22-05-2018   00:10:15   5.54 | General |  | | 4 | s5 | 7 Grace,Incoming,ISD,00:10:55,28-07-2018 Wayne,Incoming,Local,00:05:35,10-07-2017 Joe,Incoming,Roaming,00:01:50,30-06-2018 David,Outgoing,Local,00:10:15,22-05-2018 John,Outgoing,STD,00:20:25,01-01-2018 Jane,Outgoing,ISD,00:15:20,03-08-2017 Choe,Outgoing,Roaming,00:30:03,31-05-2018 | Enter the number of calls: Contact  Type     Call Type  Date         Duration   Cost Grace    Incoming ISD        28-07-2018   00:10:55   49.13 Wayne    Incoming Local      10-07-2017   00:05:35   0.00 Joe      Incoming Roaming    30-06-2018   00:01:50   0.66 David    Outgoing Local      22-05-2018   00:10:15   5.54 John     Outgoing STD        01-01-2018   00:20:25   14.70 Jane     Outgoing ISD        03-08-2017   00:15:20   138.00 Choe     Outgoing Roaming    31-05-2018   00:30:03   21.64 | General |  | | 5 | s3 | 6 Joe,Incoming,STD,00:01:50,30-06-2018 David,Outgoing,Local,00:10:15,22-05-2018 John,Outgoing,STD,00:20:25,01-01-2018 Jane,Outgoing,ISD,00:15:20,03-08-2017 Choe,Outgoing,STD,00:30:03,31-05-2018 Chris,Outgoing,Local,00:04:30,21-06-2018 | Enter the number of calls: Contact  Type     Call Type  Date         Duration   Cost Joe      Incoming STD        30-06-2018   00:01:50   0.00 David    Outgoing Local      22-05-2018   00:10:15   5.54 John     Outgoing STD        01-01-2018   00:20:25   14.70 Jane     Outgoing ISD        03-08-2017   00:15:20   138.00 Choe     Outgoing STD        31-05-2018   00:30:03   21.64 Chris    Outgoing Local      21-06-2018   00:04:30   2.43 | General |  |   **CallLog - Requirement 4**  **Requirement 4:** In this requirement develop a feature in which you can search a List of Calls by date, and contact.  a) Create a Class Contact with the following attributes:   |  |  | | --- | --- | | **Member Field Name** | **Type** | | name | String | | mobileNumber | String | | mailId | String | | dob | java.util.Date |   The following methods are present in the Contact class   |  |  | | --- | --- | | **Method Name** | **Method Description** | | static List<Contact> prefill() | This method returns the List of prefilled Contact objects. |   Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a default constructor and a parameterized constructor to take in all attributes in the given order: **Contact( String name, String mobileNumber, String mailId, java.util.Date dob )**   b) Create a Class Call with the following attributes:    |  |  | | --- | --- | | **Member Field Name** | **Type** | | type | String | | callType | String | | cost | Double | | duration | java.util.Date | | date | java.util.Date | | contact | Contact |   Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a default constructor and a parameterized constructor to take in all attributes in the given order: **Call( String type, String callType, Double cost, java.util.Date duration, java.util.Date date, Contact contact )**  c) Create a class **CallBO** with the following methods,   |  |  | | --- | --- | | **Method Name** | **Description** | | public List<Call> findCall(List<Call> callList,java.util.Date date) | This method accepts a list of calls and date as arguments and returns a list of calls that matches with the given date. | | public List<Call> findCall(List<Call> callList,String contactName) | This method accepts a list of calls and contact name as arguments and returns a list of calls that matches with the given contact name(case -insensitive). |   The call details should be given as a comma-separated value in the below order, type, callType, cost, duration, date, contact's name **Format:** Date Format : "**dd/MM/yyyy**" Duration Format : "**HH:mm:ss**"  Print format: System.out.format("%-13s %-15s %-10s %-10s %-10s %-15s %s\n","Name","Mobile Number","Type", "Call Type","Cost" ,"Duration","Date");  **Note:**The call lists are displayed in the main method.             If any other choice is selected, display "**Invalid choice**"             If the search detail is not found, display "**No calls from <contactName>**"for search using contactName and             "**No calls on <date>**" for search using date,             Display two digit after the decimal point for Double Datatype.  **Sample Input & Output 1:**  Enter the number of calls: **6 Incoming,STD,0.25,00:01:50,20/06/2018,John Incoming,ISD,5.20,00:10:55,15/07/2018,Grace Outgoing,Local,2.00,00:10:15,10/02/2018,David Incoming,Local,0.00,00:05:35,10/07/2018,John Outgoing,STD,5.05,00:20:25,02/05/2018,John Incoming,STD,10.00,00:01:50,18/04/2018,Joe** Enter a search type: 1.By a name 2.By a call date **1** Enter the name: **john** Name          Mobile Number   Type              Call Type  Cost       Duration        Date John             9876543210           Incoming   STD               0.25       00:01:50         20/06/2018 John             9876543210           Incoming   Local            0.00       00:05:35         10/07/2018 John             9876543210           Outgoing   STD               5.05       00:20:25         02/05/2018  **Sample Input & Output 2:**  Enter the number of calls: **4 Incoming,STD,0.25,00:01:50,20/06/2018,John Incoming,ISD,5.20,00:10:55,15/07/2018,Grace Outgoing,STD,5.05,00:20:25,20/06/2018,Jane Outgoing,Local,1.50,00:04:30,01/08/2018,Chris** Enter a search type: 1.By a name 2.By a call date **2** Enter the call date: **20/06/2018** Name          Mobile Number   Type             Call Type  Cost       Duration        Date John             9876543210           Incoming   STD             0.25        00:01:50         20/06/2018 Jane            9765659820           Outgoing   STD             5.05        00:20:25         20/06/2018  **Test Cases**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8581)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | 1 | 6 Incoming,STD,0.25,00:01:50,20/06/2018,John Incoming,ISD,5.20,00:10:55,15/07/2018,Grace Outgoing,Local,2.00,00:10:15,10/02/2018,David Incoming,Local,0.00,00:05:35,10/07/2018,John Outgoing,STD,5.05,00:20:25,02/05/2018,John Incoming,STD,10.00,00:01:50,18/04/2018,Joe 1 john | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the name: Name          Mobile Number   Type       Call Type  Cost       Duration        Date John          9876543210      Incoming   STD        0.25       00:01:50        20/06/2018 John          9876543210      Incoming   Local      0.00       00:05:35        10/07/2018 John          9876543210      Outgoing   STD        5.05       00:20:25        02/05/2018 | General |  | | 2 | 6 | 0 1 john | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the name: No calls from john | General |  | | 3 | 5 | 4 Incoming,STD,10.00,00:01:50,18/04/2018,Joe Outgoing,ISD,55.75,00:15:20,20/06/2018,Jane Outgoing,STD,11.00,00:30:03,09/08/2018,Choe Incoming,STD,10.00,00:01:50,10/07/2018,Joe 0 | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Invalid choice | General |  | | 4 | 8 | 4 Outgoing,STD,5.05,00:20:25,02/05/2018,John Incoming,STD,0.25,00:01:50,20/06/2018,Wayne Outgoing,Local,2.00,00:10:15,10/07/2018,David Incoming,ISD,5.20,00:10:55,15/07/2018,Wayne 2 20/05/2018 | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the call date: No calls on 20/05/2018 | General |  | | 5 | 7 | 5 Incoming,ISD,5.20,00:10:55,15/07/2018,Grace Incoming,Local,0.00,00:05:35,10/07/2018,Wayne Outgoing,Local,2.00,00:10:15,10/07/2018,David Outgoing,STD,5.05,00:20:25,02/05/2018,John Incoming,STD,10.00,00:01:50,18/04/2018,Joe 1 choe | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the name: No calls from choe | General |  | | 6 | 3 | 10 Incoming,STD,0.25,00:01:50,20/06/2018,John Outgoing,Local,1.50,00:04:30,10/07/2018,Chris Incoming,ISD,5.20,00:10:55,15/07/2018,Grace Incoming,Local,0.00,00:05:35,10/07/2018,Wayne Outgoing,Local,2.00,00:10:15,10/07/2018,David Outgoing,STD,5.05,00:20:25,02/05/2018,John Incoming,STD,10.00,00:01:50,18/04/2018,Joe Outgoing,ISD,55.75,00:15:20,20/06/2018,Jane Outgoing,STD,11.00,00:30:03,09/08/2018,Choe Incoming,STD,10.00,00:01:50,10/07/2018,Joe 2 10/07/2018 | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the call date: Name          Mobile Number   Type       Call Type  Cost       Duration        Date Chris         9659659790      Outgoing   Local      1.50       00:04:30        10/07/2018 Wayne         9787621230      Incoming   Local      0.00       00:05:35        10/07/2018 David         9856985620      Outgoing   Local      2.00       00:10:15        10/07/2018 Joe           9879865898      Incoming   STD        10.00      00:01:50        10/07/2018 | General |  | | 7 | 2 | 4 Incoming,STD,0.25,00:01:50,20/06/2018,John Incoming,ISD,5.20,00:10:55,15/07/2018,Grace Outgoing,STD,5.05,00:20:25,20/06/2018,Jane Outgoing,Local,1.50,00:04:30,01/08/2018,Chris 2 20/06/2018 | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the call date: Name          Mobile Number   Type       Call Type  Cost       Duration        Date John          9876543210      Incoming   STD        0.25       00:01:50        20/06/2018 Jane          9765659820      Outgoing   STD        5.05       00:20:25        20/06/2018 | General |  | | 8 | 4 | 10 Incoming,STD,0.25,00:01:50,20/06/2018,Wayne Outgoing,Local,2.00,00:10:15,10/07/2018,David Outgoing,STD,5.05,00:20:25,02/05/2018,John Incoming,ISD,5.20,00:10:55,15/07/2018,Wayne Incoming,Local,0.00,00:05:35,10/07/2018,Wayne Incoming,STD,10.00,00:01:50,18/04/2018,Joe Outgoing,ISD,55.75,00:15:20,20/06/2018,Jane Outgoing,STD,11.00,00:30:03,09/08/2018,Choe Outgoing,Local,1.50,00:04:30,10/07/2018,Wayne Incoming,STD,10.00,00:01:50,10/07/2018,Joe 1 Wayne | Enter the number of calls: Enter a search type: 1.By a name 2.By a call date Enter the name: Name          Mobile Number   Type       Call Type  Cost       Duration        Date Wayne         9787621230      Incoming   STD        0.25       00:01:50        20/06/2018 Wayne         9787621230      Incoming   ISD        5.20       00:10:55        15/07/2018 Wayne         9787621230      Incoming   Local      0.00       00:05:35        10/07/2018 Wayne         9787621230      Outgoing   Local      1.50       00:04:30        10/07/2018 | General |  | |   **Call Log -Requirement 5**  Now in this requirement, we have to display a month wise report of the call history.   a)Create a class Call with following attributes,   |  |  | | --- | --- | | **Member Field Name** | **Type** | | contact | Contact | | type | String | | callType | String | | cost | Double | | duration | Date | | date | Date |   Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a default constructor and a parameterized constructor to take in all attributes in the given order:**public Call(Contact contact, String type, String callType, Double cost, Date duration, Date date)**   b) Create a class Contact with the following attributes,   |  |  | | --- | --- | | **Member Field Name** | **Type** | | name | String | | mobileNumber | String | | mailId | String | | dob | Date |   Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a default constructor and a parameterized constructor to take in all attributes in the given order:**public Contact(String name, String mobileNumber, String mailId, Date dob)**  c) Create the following method in the Call class,   |  |  | | --- | --- | | **Method Name** | **Description** | | public static Map<String,Integer> monthWiseCount(List<Call> callList) | This method takes list of calls asargument and return a tree map which has month name (in 3 letter format) as key and number of calls made in the month as value. |   d) Create the following method in the Contact class,   |  |  | | --- | --- | | **Method Name** | **Description** | | public static List<Contact> preFill() | This method return a list of prefilled Contacts, this will be given in the template |   The call details should be given as a comma-separated value in the below order, **contactName,type,cost,allType,duration,date**   Print format: System.out.printf("%-10s %s\n","Month", "Count");   **Sample Input and Output:**  Enter the number of calls: **5 Wayne,Incoming,Local,0,00:05:35,10-07-2018 Joe,Incoming,STD,10.00,00:01:50,30-06-2018 David,Outgoing,Local,2.00,00:10:15,22-05-2018 John,Outgoing,STD,5.05,00:20:25,01-01-2018 Jane,Outgoing,ISD,55.75,00:15:20,03-08-2018** Month      Count Aug        1 Jan        1 Jul        1 Jun        1 May        1  **Test Cases**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8585)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | s4 | 9 John,Incoming,STD,0.25,00:01:50,02-05-2018 Grace,Incoming,ISD,5.20,00:10:55,28-07-2018 Wayne,Incoming,Local,0,00:05:35,10-07-2018 Joe,Incoming,STD,10.00,00:01:50,30-06-2018 David,Outgoing,Local,2.00,00:10:15,22-05-2018 John,Outgoing,STD,5.05,00:20:25,01-01-2018 Jane,Outgoing,ISD,55.75,00:15:20,03-08-2018 Choe,Outgoing,STD,11.00,00:30:03,31-05-2018 Chris,Outgoing,Local,1.50,00:04:30,21-06-2018 | Enter the number of calls: Month      Count Aug        1 Jan        1 Jul        2 Jun        2 May        3 | General |  | | 2 | s3 | 5 David,Outgoing,Local,2.00,00:10:15,22-05-2018 John,Outgoing,STD,5.05,00:20:25,01-01-2018 Jane,Outgoing,ISD,55.75,00:15:20,03-08-2018 Choe,Outgoing,STD,11.00,00:30:03,31-05-2018 Chris,Outgoing,Local,1.50,00:04:30,21-06-2018 | Enter the number of calls: Month      Count Aug        1 Jan        1 Jun        1 May        2 | General |  | | 3 | s1 | 5 Wayne,Incoming,Local,0,00:05:35,10-07-2018 Joe,Incoming,STD,10.00,00:01:50,30-06-2018 David,Outgoing,Local,2.00,00:10:15,22-05-2018 John,Outgoing,STD,5.05,00:20:25,01-01-2018 Jane,Outgoing,ISD,55.75,00:15:20,03-08-2018 | Enter the number of calls: Month      Count Aug        1 Jan        1 Jul        1 Jun        1 May        1 | General |  | | 4 | s2 | 5 John,Incoming,STD,0.25,00:01:50,02-05-2018 Grace,Incoming,ISD,5.20,00:10:55,28-07-2018 Wayne,Incoming,Local,0,00:05:35,10-07-2018 Joe,Incoming,STD,10.00,00:01:50,30-06-2018 David,Outgoing,Local,2.00,00:10:15,22-05-2018 | Enter the number of calls: Month      Count Jul        2 Jun        1 May        2 | General |  | |   **CallLog - Requirement 6**  **Requirement 6:** In this requirement, Let's develop a feature to find the favourite contact from the Call Log based on total number of calls from and to the user.  Create a Class Contact with the following private attributes   |  |  | | --- | --- | | **Attributes** | **Datatype** | | name | String | | mobileNumber | String | | mailId | String | | dob | Date |   Create / Generate appropriate Getters & Setters Add a default constructor and a parameterized constructor to take in all attributes in the given order: **Contact ( String name, String mobileNumber, String mailId, java.util.Date dob )**  The following methods are present in the Contact class   |  |  | | --- | --- | | **Method Name** | **Method Description** | | static List<Contact> prefill() | This method returns the List of prefilled Contact objects. |   Create a Class Call with private attributes   |  |  | | --- | --- | | **Attributes** | **Datatype** | | type | String | | callType | String | | cost | Double | | duration | Date | | date | Date | | contact | Contact |   Create / Generate appropriate Getters & Setters Add a default constructor and a parameterized constructor to take in all attributes in the given order: **Call(String type, String callType, Double cost, Date duration,Date date, Contact contact)**  Create a Class CallLog with private attributes   |  |  | | --- | --- | | **Attributes** | **Datatype** | | callList | List<Call> |   Create / Generate appropriate Getters & Setters Add a default constructor and a parameterized constructor to take in all attributes in the given order: **CallLog(List<Call> callList)**  The following methods are present in the CallLog class   |  |  | | --- | --- | | **Method Name** | **Method Description** | | Contact getFavouriteContact() | This method returns a Contact that has maximum number of calls (both incoming and outgoing) to the user from list of Call objects in the invoking object. |   The call inputs are given in comma separated format in the below order, **type, callType, cost, duration, date, contactName**  Note : While creating CallLog object, create a empty list for callList attribute.            After calling appropriate methods, display the name of favourite contact in the main method.                  Use "dd/MM/yyyy" for date attribute.            Use "HH:mm:ss" for duration attribute.  **Sample Input/Output:**    Enter the number of calls:  **6**  **Incoming,STD,0.25,00:01:50,22/01/2018,John**  **Incoming,ISD,5.20,00:10:55,15/02/2018,Grace**  **Incoming,Local,0,00:05:35,05/02/2018,Wayne**  **Outgoing,STD,11.00,00:30:03,01/07/2018,John**  **Outgoing,Local,1.50,00:04:30,02/12/2017,John**  **Incoming,STD,10.00,00:01:50,02/05/2018,Joe**  The favourite contact is John  **Test Cases**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8583)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | t4 | 6 Incoming,Local,0,00:05:35,05/02/2018,Wayne Incoming,STD,10.00,00:01:50,02/05/2018,Joe Outgoing,Local,2.00,00:10:15,03/05/2018,David Incoming,ISD,5.20,00:10:55,15/02/2018,David Incoming,STD,0.25,00:01:50,22/01/2018,David Outgoing,Local,1.50,00:04:30,02/12/2017,Chris | Enter the number of calls: The favourite contact is David | General |  | | 2 | t2 | 7 Incoming,STD,0.25,00:01:50,22/01/2018,John Incoming,ISD,5.20,00:10:55,15/02/2018,Grace Incoming,Local,0,00:05:35,05/02/2018,John Incoming,STD,10.00,00:01:50,02/05/2018,Joe Outgoing,Local,2.00,00:10:15,03/05/2018,Joe Outgoing,STD,11.00,00:30:03,01/07/2018,Choe Outgoing,Local,1.50,00:04:30,02/12/2017,Joe | Enter the number of calls: The favourite contact is Joe | General |  | | 3 | t5 | 7 Incoming,STD,0.25,00:01:50,22/01/2018,John Incoming,ISD,5.20,00:10:55,15/02/2018,Grace Incoming,Local,0,00:05:35,05/02/2018,Wayne Incoming,STD,10.00,00:01:50,02/05/2018,Joe Outgoing,STD,5.05,00:20:25,28/02/2018,Wayne Outgoing,Local,1.50,00:04:30,02/12/2017,Chris Outgoing,STD,11.00,00:30:03,01/07/2018,Choe | Enter the number of calls: The favourite contact is Wayne | General |  | | 4 | t1 | 6 Incoming,STD,0.25,00:01:50,22/01/2018,John Incoming,ISD,5.20,00:10:55,15/02/2018,Grace Incoming,Local,0,00:05:35,05/02/2018,Wayne Outgoing,STD,11.00,00:30:03,01/07/2018,John Outgoing,Local,1.50,00:04:30,02/12/2017,John Incoming,STD,10.00,00:01:50,02/05/2018,Joe | Enter the number of calls: The favourite contact is John | General |  | | 5 | t3 | 5 Incoming,ISD,5.20,00:10:55,15/02/2018,Grace Outgoing,ISD,55.75,00:15:20,30/01/2018,Grace Outgoing,STD,11.00,00:30:03,01/07/2018,Grace Incoming,Local,0,00:05:35,05/02/2018,Grace Incoming,STD,0.25,00:01:50,22/01/2018,John | Enter the number of calls: The favourite contact is Grace | General |  | | |

Skeleton is in

02-may-mock24-req-6

Solution is pushed to git.

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class Contact {

private String name;

private String mobileNumber;

private String mailId;

private Date dob;

public Contact() {}

public Contact(String name, String mobileNumber, String mailId, Date dob) {

super();

this.name = name;

this.mobileNumber = mobileNumber;

this.mailId = mailId;

this.dob = dob;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getMobileNumber() {

return mobileNumber;

}

public void setMobileNumber(String mobileNumber) {

this.mobileNumber = mobileNumber;

}

public String getMailId() {

return mailId;

}

public void setMailId(String mailId) {

this.mailId = mailId;

}

public Date getDob() {

return dob;

}

public void setDob(Date dob) {

this.dob = dob;

}

public static List<Contact> prefill()throws ParseException{

List<Contact> contactList = new ArrayList<>();

SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");

contactList.add(new Contact("John","9876543210","john@gmail.com",sdf.parse("31/01/1990")));

contactList.add(new Contact("Grace","9898798652","grace@gmail.com",sdf.parse("27/02/1987")));

contactList.add(new Contact("Wayne","9787621230","wayne@gmail.com",sdf.parse("15/09/1988")));

contactList.add(new Contact("Joe","9879865898","joe@gmail.com",sdf.parse("23/07/1992")));

contactList.add(new Contact("David","9856985620","david@gmail.com",sdf.parse("24/05/1982")));

contactList.add(new Contact("Jane","9765659820","jane@gmail.com",sdf.parse("13/03/1993")));

contactList.add(new Contact("Choe","9595878580","choe@gmail.com",sdf.parse("26/09/1989")));

contactList.add(new Contact("Chris","9659659790","chris@gmail.com",sdf.parse("15/03/1993")));

return contactList;

}

}

import java.util.Date;

public class Call {

private String type;

private String callType;

private Double cost;

private Date duration;

private Date date;

private Contact contact;

public Call() {}

public Call(String type, String callType, Double cost, Date duration, Date date, Contact contact) {

super();

this.type = type;

this.callType = callType;

this.cost = cost;

this.duration = duration;

this.date = date;

this.contact = contact;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getCallType() {

return callType;

}

public void setCallType(String callType) {

this.callType = callType;

}

public Double getCost() {

return cost;

}

public void setCost(Double cost) {

this.cost = cost;

}

public Date getDuration() {

return duration;

}

public void setDuration(Date duration) {

this.duration = duration;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public Contact getContact() {

return contact;

}

public void setContact(Contact contact) {

this.contact = contact;

}

}

import java.util.ArrayList;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Optional;

public class CallLog {

private List<Call> callList;

public CallLog() {

this.callList=new ArrayList<Call>();

}

public CallLog(List<Call> callList) {

this.callList = callList;

}

public List<Call> getCallList() {

return callList;

}

public void setCallList(List<Call> callList) {

this.callList = callList;

}

Contact getFavouriteContact()

{

//write your code here

//what input we have? callList is the input

//what output we find? contact that has done more calls

Map<Contact, Integer> map=new LinkedHashMap<Contact, Integer>();

for(Call call : callList)

{

Contact contact = call.getContact();

//check if this contact already present in the map

Integer count = map.get(contact); //equals should be overriden

if(count==null)

count=0;

count++;

map.put(contact, count);

}

//find the largest count in the map

int max=0;

Contact maxContact=null;

for(Entry<Contact, Integer> e:map.entrySet())

{

int count=e.getValue();

if(count>max)

{

max=count;

maxContact=e.getKey();

}

}

return maxContact;

//using stream api

// Optional<Integer> result = map.values().stream()

// .max((a,b)->a-b);

//

// int count=0;

// if(result!=null)

// count=result.get();

//

// //count will contain the max count. we need to find key when value is count

//

// return null;

}

}

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class App {

public static void main(String[] args) throws ParseException {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the number of calls:");

int noOfCalls=sc.nextInt();

List<Call> callList=new ArrayList<>();

List<Contact> contactList = Contact.prefill();

for(int i=0;i<noOfCalls;i++)

{

String detail=sc.nextLine();

if(detail.equals(""))

detail=sc.nextLine();

// Incoming,STD,0.25,00:01:50,22/01/2018,John

String[] arr = detail.split(",");

Call call=new Call();

call.setType(arr[0]);

call.setCallType(arr[1]);

call.setCost(Double.parseDouble(arr[2]));

SimpleDateFormat sdf=new SimpleDateFormat("HH:mm:ss");

call.setDuration(sdf.parse(arr[3]));

sdf=new SimpleDateFormat("dd/MM/yyyy");

call.setDate(sdf.parse(arr[4]));

//find contact object from prefill using name

String contactName=arr[5];

for(Contact c : contactList)

{

if(c.getName().equals(contactName))

{

call.setContact(c);

break;

}

}

callList.add(call);

}

CallLog clog=new CallLog();

clog.setCallList(callList);

Contact contact = clog.getFavouriteContact();

System.out.println("The favourite contact is "+contact.getName());

}

}

Debugging:

1. Breakpoint

The program will pause at this point.

Breakpoint is allowed in any executable code. What is executable code, anything which is not a comment or declaration.

To add a breakpoint, double click on the left margin of the code in preferred line.

1. Intead of running the program, debug the program.

Got to Run menu, click debug

1. Now, the program pause at the breakpoint.

We need to proceed line by line and find out what is happening.

Options available for that:

Step into

Step over

Step out

Step into: steps into the method and continue inside that method

int i;

int j;

int k=sum(i,j);

Step over: goes to next line

If there is a method call, that entire method call is completed and moves next line

Step return/ out:

You are inside a method, you want to complete that method and return to the calling method.

Debugging windows: (name of the window is “Variables”)

Local variables

Challenge:

Java Mock 21 Req #6

**Food Aggregator - Requirement 6**

Requirement 6:  
In this requirement develop a feature to find the most valuable customer in a given month.  
  
Create a **User** class with the following private attributes

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| email | String |
| phoneNumber | String |
| location | String |
| purchaseList | List<Purchase> |

Create / Generate appropriate Getters & Setters

Add a default constructor and a parameterized constructor to take in all attributes in the given order:

**User ( String name, String email, String phoneNumber , String location ,List<Purchase> purchaseList )**  
  
The following methods are present in the User class

|  |  |
| --- | --- |
| **Method Name** | **description** |
| static List<User> prefill() | This method returns a List of User objects. |
| static User getValuableUser(List<User> userList,String month) | This method accepts List of User objects and month name as arguments. This method returns a user object that has the maximum cumulative purchase amount in the specified month. |

Create a **Purchase** class with the following private attributes

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| id | Integer |
| price | Double |
| couponCode | String |
| purchaseDate | Date |
| user | User |

Create / Generate appropriate Getters & Setters

Add a default constructor and a parameterized constructor to take in all attributes in the given order:

Purchase(Integer id, Double price, String couponCode, Date purchaseDate, User user)  
  
The inputs to the purchase are given the order below,  
**id, price, couponCode, purchaseDate, username**  
  
Create a driver class called Main. In the main method, obtain purchase details and display the valuable user of the specified month by calling appropriate methods.  
  
Note: The output should be in the format "The valuable user of the month [month name] is [valuable user name] (Refer Input/Output specification)  
             Use "dd-MM-yyyy" for the purchaseDate attribute.

**Sample Input/Output 1:**

Enter the number of purchases:

**4**

**1,750,FIRST,05-06-2018,Rob**

**2,1500,DEAL25,05-06-2018,Brandon**

**3,2500,FIRST,07-07-2018,Joe**

**4,800,DEAL25,08-06-2018,Rob**

Enter the month:

**June**

The valuable user of the month June is Rob 

**Sample Input/Output 2:**

Enter the number of purchases:

**5**

**120,1500,CHICKEN70,06-04-2018,Oliver**

**160,2500,CHICKEN70,08-04-2018,Harry**

**190,3000,FIRST,09-04-2018,Oliver**

**225,4500,BUYFIVE,08-04-2018,Oliver**

**280,500,BUYFIVE,12-04-2018,Rob**

Enter the month:

**April**

The valuable user of the month April is Oliver

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8561)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | t5 | 6 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-06-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 520,750,FIRST,15-07-2018,Danny 623,415,ORDER20,20-07-2018,Rob 752,1000,DEAL25,01-07-2018,Danny July | Enter the number of purchases: Enter the month: The valuable user of the month July is Danny | General |  | | 2 | t3 | 5 120,1500,CHICKEN70,06-05-2018,Oliver 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-06-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 280,500,BUYFIVE,03-03-2018,Rob March | Enter the number of purchases: Enter the month: The valuable user of the month March is Rob | General |  | | 3 | t1 | 4 1,750,FIRST,05-06-2018,Rob 2,1500,DEAL25,05-06-2018,Brandon 3,2500,FIRST,07-07-2018,Joe 4,800,DEAL25,08-06-2018,Rob June | Enter the number of purchases: Enter the month: The valuable user of the month June is Rob | General |  | | 4 | t6 | 6 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-05-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 520,750,FIRST,15-07-2018,Danny 1,750,FIRST,25-06-2018,Oliver 2,1500,DEAL25,27-06-2018,Harry June | Enter the number of purchases: Enter the month: The valuable user of the month June is Oliver | General |  | | 5 | t4 | 6 120,1500,CHICKEN70,06-05-2018,Oliver 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-06-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 280,500,BUYFIVE,03-03-2018,Rob 560,5000,FIRST,29-06-2018,Brandon June | Enter the number of purchases: Enter the month: The valuable user of the month June is Brandon | General |  | | 6 | t2 | 5 120,1500,CHICKEN70,06-04-2018,Oliver 160,2500,CHICKEN70,08-04-2018,Harry 190,3000,FIRST,09-04-2018,Oliver 225,4500,BUYFIVE,08-04-2018,Oliver 280,500,BUYFIVE,12-04-2018,Rob April | Enter the number of purchases: Enter the month: The valuable user of the month April is Oliver | General |  | |

Skeleton:

import java.util.ArrayList;

import java.util.List;

public class User {

private String name;

private String email;

private String phoneNumber;

private String location;

private List<Purchase> purchaseList;

public User()

{}

public User(String name, String email, String phoneNumber, String location, List<Purchase> purchaseList) {

super();

this.name = name;

this.email = email;

this.phoneNumber = phoneNumber;

this.location = location;

this.purchaseList = purchaseList;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public List<Purchase> getPurchaseList() {

return purchaseList;

}

public void setPurchaseList(List<Purchase> purchaseList) {

this.purchaseList = purchaseList;

}

public static List<User> prefill(){

List<User> userList=new ArrayList<>();

userList.add(new User("Harry","harry@gmail.com","9856231478","NewYork", new ArrayList<Purchase>()));

userList.add(new User("Joe","joe@gmail.com","8956231548","Chicago", new ArrayList<Purchase>()));

userList.add(new User("Oliver","oliver@gmail.com","7856124589","SanFrancisco", new ArrayList<Purchase>()));

userList.add(new User("Danny","danny@gmail.com","9382714568","Los Angeles", new ArrayList<Purchase>()));

userList.add(new User("Brandon","brandon@gmail.com","98596215488","Boston", new ArrayList<Purchase>()));

userList.add(new User("Rob","rob@gmail.com","9584596662","NewYork", new ArrayList<Purchase>()));

return userList;

}

static User getValuableUser(List<User> userList,String month)

{

//write code here

return null;

}

}

import java.util.Date;

public class Purchase {

private Integer id;

private Double price;

private String couponCode;

private Date purchaseDate;

private User user;

public Purchase() {}

public Purchase(Integer id, Double price, String couponCode, Date purchaseDate, User user) {

super();

this.id = id;

this.price = price;

this.couponCode = couponCode;

this.purchaseDate = purchaseDate;

this.user = user;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public String getCouponCode() {

return couponCode;

}

public void setCouponCode(String couponCode) {

this.couponCode = couponCode;

}

public Date getPurchaseDate() {

return purchaseDate;

}

public void setPurchaseDate(Date purchaseDate) {

this.purchaseDate = purchaseDate;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

}

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class App {

public static void main(String[] args) throws ParseException {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the number of purchases:");

int noOfPurchases=sc.nextInt();

List<Purchase> purchaseList=new ArrayList<Purchase>();

List<User> userList = User.prefill();

for(int i=0;i<noOfPurchases;i++)

{

String detail=sc.nextLine();

if(detail.equals(""))

detail=sc.nextLine();

String[] arr = detail.split(",");

// 1,750,FIRST,05-06-2018,Rob

Purchase purchase=new Purchase();

purchase.setId(Integer.valueOf(arr[0]));

purchase.setPrice(Double.valueOf(arr[1]));

purchase.setCouponCode(arr[2]);

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");

purchase.setPurchaseDate(sdf.parse(arr[3]));

String userName=arr[4];

//find user object using userName

for(User user: userList)

{

if(user.getName().equals(userName))

{

purchase.setUser(user);

user.getPurchaseList().add(purchase);

}

}

purchaseList.add(purchase);

}

System.out.println("Enter the month:");

String month=sc.next();

User result = User.getValuableUser(userList, month);

System.out.println("The valuable user of the month June is "+result.getName());

}

}

-----------------------------------------final code for User-------------

**static** User getValuableUser(List<User> userList,String month)

{

//write code here

Double max=0.0;

User valuableUser=**null**;

**for**(User user: userList)

{

List<Purchase> purchases = user.getPurchaseList();

Double total=0.0;

**for**(Purchase purchase : purchases)

{

String m=**new** SimpleDateFormat("MMMM").format(purchase.getPurchaseDate());

**if**(m.equals(month))

{

//consider this purchase

total+=purchase.getPrice();

}

}

**if**(total>max)

{

max=total;

valuableUser=user;

}

}

**return** valuableUser;

}