**MobileBrand - Requirement 1**

**Requirement 1:**

Let’s start off by creating two **Mobile** objects and check whether they are equal.

1. Create a **Mobile** Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| referenceId | String |
| modelName | String |
| displaySize | Double |
| Price | Double |
| launchedDate | java.util.Date |

1. Mark all the attributes as private
2. Create / Generate appropriate Getters & Setters
3. Add a default constructor and a parameterized constructor to take in all attributes in the given order:   
   **Mobile ( String referenceId, String modelName, Double displaySize, Double price, java.util.Date launchedDate )**
4. When the “Mobile” object is printed, it should display the following details: **[Override the toString method]**  
   Print format:  
   Reference Id: "referenceId"  
   Model Name: "modelName"  
   Display Size: "displaySize"  
   Price: "price"  
   Launched Date: "launchedDate"
5. Two Mobiles are considered same if they have the same referenceId, and modelName. Implement the logic in the appropriate function. (Case – Insensitive) **[Override the equals method]**  
     
   The input format consists of Mobile details separated by a comma in the below order,  
   referenceId, modelName, displaySize, price, launchedDate

The Input to your program would be details of two Mobiles, you need to display their details as given in "5th point(refer above)" and compare the two Mobiles and display if the Mobiles are same or different.  
  
**Note:**There is an empty line between display statements. Print the empty lines in the main function.  
              Display one digit after the decimal point for the Double data type.  
  
  
**Sample Input and Output 1:**

Enter mobile 1 detail:

**#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017**

Enter mobile 2 detail:

**#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017**

Mobile 1

Reference Id: #SM 45 JJ6-001

Model Name: Galaxy J6

Display Size: 5.6

Price: 13990.0

Launched Date: 02-01-2017

Mobile 2

Reference Id: #SM 45 JJ6-001

Model Name: Galaxy J6

Display Size: 5.6

Price: 13990.0

Launched Date: 02-01-2017

Mobile 1 is same as Mobile 2

**Sample Input and Output 2:**

Enter mobile 1 detail:

**#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017**

Enter mobile 2 detail:

**#SM 45 JJ6-001,Galaxy J7,5.6,13990,02-01-2017**

Mobile 1

Reference Id: #SM 45 JJ6-001

Model Name: Galaxy J6

Display Size: 5.6

Price: 13990.0

Launched Date: 02-01-2017

Mobile 2

Reference Id: #SM 45 JJ6-001

Model Name: Galaxy J7

Display Size: 5.6

Price: 13990.0

Launched Date: 02-01-2017

Mobile 1 and Mobile 2 are different

**MobileBrand - Requirement 2**

**Requirement 2:**  
Now we are gonna start creating a MobileBrand and add Mobiles to it. Start with creating a MobileBrand and use menu-driven approach to add, remove, display details of the Mobiles in the MobileBrand.  
  
a)Create a Class **Mobile** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| referenceId | String |
| modelName | String |
| displaySize | Double |
| price | Double |
| launchedDate | java.util.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:  **Mobile( String referenceId, String modelName, Double displaySize, Double price, java.util.Date launchedDate )**  
  
b)Create a Class **MobileBrand** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| mobileList | List<Mobile> |

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: **MobileBrand( String name, List<Mobile> mobileList ).**In constructor pass the mobileList value as an empty list. Only one MobileBrand will be present at a time.  
  
c) Create the following static method in Mobile class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public static Mobile createMobile(String detail) | This method accepts a string which contains mobile details separated by commas. Split the details and create a mobile object from the details and return it. |

The mobile details should be given as a comma-separated value in the below order,  
**referenceId, modelName, displaySize, price, launchedDate**  
  
d) Create the following methods in MobileBrand class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void addMobileToMobileBrand(Mobile mobile) | This method accepts a mobile object and adds the mobile to the mobile list of the current MobileBrand. |
| public Boolean removeMobileFromMobileBrand(String referenceId) | This method will get the referenceId of the mobile and delete the mobile with the specified referenceId from the current MobileBrand. If a mobile with the given referenceId found, delete the mobile and return**true**. If a mobile with the referenceId is not found return **false**. |
| public void displayMobiles() | This method will display the mobile list in the current MobileBrand. If the mobile list is empty display "**No mobiles to show"**, else display "Mobiles in [mobileBrand name]" and display all the mobile details in the specified format. Where [mobileBrand name] specifies the name of the mobileBrand. |

After deletion, if true is returned print "**Mobile successfully deleted**", else print "**Mobile not found in the mobileBrand**".  
  
**Note:** The above print statements should be present in the main method.  
  
Print format: **System.out.format("%-15s %-15s %-12s %-8s %s\n","Reference Id","Model Name","Display Size", "Price","Launched Date");**  
  
**Display 1 digit after decimal point in Double.**  
  
**Sample Input and Output:**

Enter the name of the Mobile Brand:

**Samsung**

1.Add Mobile

2.Delete Mobile

3.Display Mobiles

4.Exit

Enter your choice:

**1**

**#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017**

Mobile successfully added

1.Add Mobile

2.Delete Mobile

3.Display Mobiles

4.Exit

Enter your choice:

**3**

Mobiles in Samsung

Reference Id    Model Name      Display Size Price    Launched Date

#SM 45 JJ6-001  Galaxy J6       5.6          13990.0  02-01-2017

1.Add Mobile

2.Delete Mobile

3.Display Mobiles

4.Exit

Enter your choice:

**2**

Enter the reference id of the mobile to be deleted:

**#SM 45 JJ6-001**

Mobile successfully deleted

1.Add Mobile

2.Delete Mobile

3.Display Mobiles

4.Exit

Enter your choice:

**2**

Enter the reference id of the mobile to be deleted:

**#SM 45 JJ6-001**

Mobile not found in the Mobile Brand

1.Add Mobile

2.Delete Mobile

3.Display Mobiles

4.Exit

Enter your choice:

**3**

No mobiles to show

1.Add Mobile

2.Delete Mobile

3.Display Mobiles

4.Exit

Enter your choice:

**4**

**Mobile Brand Requirement 3**

**Requirement 3:**  
   In this requirement, you need to validate the referenceId of the Mobile.  
  
a)Create a class **Main** with the following static methods:

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Boolean validateReferenceId(String referenceId) | Validate the referenceId based on the rules given below. Returns**true**ifreferenceId is valid else return **false** |

b) While validating a referenceId follow the below rules.  
  
1. The first part should start with '#' and then followed 2 to 3 UPPERCASE letters.  
2. The second part contains a two digit number. Sometimes second part will be neglated.  
3. Third part contains 2 to 6 UPPERCASE letters and numbers followed by a '-' and 2 to 4 digit number.  
4. Each part is separated by space.

Valid Format**:#BA 45 CX-20**  
                     **#BA CX-20**  
                               **#JIO PH201-20**  
  
**Note:** Print "**Reference Id is valid**" if referenceId is valid else print "**Reference Id is invalid**".  
            All the above print statements are present in the main method.  
  
**[All text in bold corresponds to input]**  
**Sample Input and Output 1:**  
  
Enter the reference Id to be validated:  
**#RM 87 AAA-062**  
Reference Id is valid  
  
**Sample Input and Output 2:**  
  
Enter the reference Id to be validated:  
**#MT 57 X-067**  
Reference Id is invalid

**MobileBrand - Requirement 4**

**Requirement 4:**  
In this requirement develop a feature in which you can search a List of Mobiles by modelName, and displaySize.  
  
a) Create a Class **Mobile** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| referenceId | String |
| modelName | String |
| displaySize | Double |
| price | Double |
| launchedDate | java.util.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:  
**Mobile( String referenceId, String modelName, Double displaySize, Double price, java.util.Date launchedDate )**  
  
b) Create a class **MobileBO**with the following methods,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public List<Mobile> findMobile(List<Mobile> mobileList,String modelName) | This method accepts a list of mobiles and modelName as arguments and returns a list of mobiles that match with the given modelName. |
| public List<Mobile> findMobile(List<Mobile> mobileList,Double displaySize) | This method accepts a list of mobiles and displaySize as arguments and returns a list of mobiles that match with the given displaySize. |

The mobile details should be given as a comma-separated value in the below order,  
**referenceId, modelName, displaySize, price, launchedDate**  
  
When the “mobile” object is printed, it should display the following details  
Print format:  
**System.out.format("%-15s %-15s %-12s %-8s %s\n","Reference Id","Model Name","Display Size", "Price",    "Launched Date")**  
  
**Note:**The mobile 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 such mobile is present**"  
            Display one digit after the decimal point for Double Datatype.  
  
**Sample Input & Output 1:**  
  
Enter the number of mobiles:  
**7**  
**#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017  
#SM 46 JJ6-002,Galaxy J6,5.7,13900,17-02-2018  
#SM 47 PRO-003,J3 pro,5,7490,19-06-2017  
#SM 48 DUO-004,J7 Duo,5.6,17000,23-09-2016  
#SM 49 PRO-005,J7 Pro,5.5,16900,15-12-2017  
#SM 50 ONN-006,On5,5,6990,29-07-2017  
#SM 51 NXT-007,J7 Nxt,5.5,9490,30-03-2018**  
Enter a search type:  
1.By Model Name  
2.By Display Size  
**1**  
Enter the Model Name:  
**Galaxy J6**  
Reference Id      Model Name      Display Size  Price      Launched Date  
#SM 45 JJ6-001  Galaxy J6            5.6                 13990.0   02-01-2017  
#SM 46 JJ6-002  Galaxy J6            5.7                 13900.0   17-02-2018  
  
**Sample Input & Output 2:**  
  
Enter the number of mobiles:  
**7  
#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017  
#SM 46 MAX-002,Galaxy On Max,5.7,13900,17-02-2018  
#SM 47 PRO-003,J3 pro,5,7490,19-06-2017  
#SM 48 DUO-004,J7 Duo,5.6,17000,23-09-2016  
#SM 49 PRO-005,J7 Pro,5.5,16900,15-12-2017  
#SM 50 ONN-006,On5,5,6990,29-07-2017  
#SM 51 NXT-007,J7 Nxt,5.7,9490,30-03-2018**  
Enter a search type:  
1.By Model Name  
2.By Display Size  
**2**  
Enter the Display Size:  
**5.7**  
Reference Id         Model Name       Display Size   Price      Launched Date  
#SM 46 MAX-002   Galaxy On Max   5.7                   13900.0  17-02-2018  
#SM 51 NXT-007   J7 Nxt                   5.7                   9490.0     30-03-2018

**MobileBrand - Requirement 5**

**Requirement 5:**  
  
In this requirement, you need to sort the list of mobiles based on price, and launchedDate.  
  
a) Create a Class **Mobile** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| referenceId | String |
| modelName | String |
| displaySize | Double |
| price | Double |
| launchedDate | java.util.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: Mobile( String referenceId, String modelName, Double displaySize, Double price, java.util.Date launchedDate )  
  
b) Create the following static methods in the Mobile class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Mobile createMobile(String detail) | This method accepts a String. The mobile detail separated by commas is passed as the argument. Split the details and create a mobile object and returns it. |

The mobile details should be given as a comma-separated value in the below order,  
referenceId, modelName, displaySize, price, launchedDate  
  
c) The Mobile class should implement the **Comparable** interface which sorts the Mobile list based on price. While comparing, all the price attributes in the list are unique.  
  
d) Create a class **LaunchedDateComparator** which implements Comparator interface and sort the Mobile list based on launchedDate. While comparing, all the launchedDate attributes in the list are unique.  
  
Get the number of Mobiles and mobile details and create a mobile list. Sort the Mobiles according to the given option and display the list.  
  
When the “mobile” object is printed, it should display the following details  
Print format:  
**System.out.format("%-15s %-15s %-12s %-8s %s\n","Reference Id","Model Name","Display Size","Price","Launched Date");**  
Display one digit after decimal point for Double datatype.  
  
**Sample Input and Output 1:**  
  
Enter the number of mobiles:  
**5  
#MT 52 PLY-017,G6 Play,5.7,11999,02-01-2016  
#RM 97 PRO-061,Note 5 Pro,5.9,14999,02-01-2016  
#RM 87 AAA-062,5A,5,5999,15-12-2017  
#RM 17 NOT-063,Note 5,5.9,12000,14-05-2018  
#SM 45 JJ6-001,Galaxy J6,5.6,13990,02-01-2017**  
Enter a type to sort:  
1.Sort by price  
2.Sort by launched date  
**1**  
Reference Id    Model Name      Display Size Price    Launched Date  
#RM 87 AAA-062  5A              5.0          5999.0   15-12-2017  
#MT 52 PLY-017  G6 Play         5.7          11999.0  02-01-2016  
#RM 17 NOT-063  Note 5          5.9          12000.0  14-05-2018  
#SM 45 JJ6-001  Galaxy J6       5.6          13990.0  02-01-2017  
#RM 97 PRO-061  Note 5 Pro      5.9          14999.0  02-01-2016  
  
**Sample Input and Output 2:**  
  
Enter the number of mobiles:  
**5  
#MT 56 PLA-057,Z2 Play,5.5,18999,29-08-2017  
#MT 57 XMM-067,X4,5.4,24999,11-12-2018  
#SM 50 ONN-006,On5,5,6990,29-07-2017  
#SM 51 NXT-007,J7 Nxt,5.5,9490,30-03-2018  
#RM 67 MIX-068,Mix 2,6,29999,14-12-2018**  
Enter a type to sort:  
1.Sort by price  
2.Sort by launched date  
**2**  
Reference Id    Model Name      Display Size Price    Launched Date  
#SM 50 ONN-006  On5             5.0          6990.0   29-07-2017  
#MT 56 PLA-057  Z2 Play         5.5          18999.0  29-08-2017  
#SM 51 NXT-007  J7 Nxt          5.5          9490.0   30-03-2018  
#MT 57 XMM-067  X4              5.4          24999.0  11-12-2018  
#RM 67 MIX-068  Mix 2           6.0          29999.0  14-12-2018

**MobileBrand - Requirement 6**

**Requirement 5:**  
  
In this requirement,given a list of mobiles, you need to find launched year wise count of mobiles in the company.  
  
a) Create a Class **Mobile** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| referenceId | String |
| modelName | String |
| displaySize | Double |
| price | Double |
| launchedDate | java.util.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: **Mobile( String referenceId, String modelName, Double displaySize, Double price, java.util.Date launchedDate )**  
  
b) Create the following static methods in the Mobile class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Mobile createMobile(String detail) | This method accepts a String. The mobile detail separated by commas is passed as the argument. Split the details and create a mobile object and returns it. |
| public static Map<Date,Integer> yearWiseCount(List<Mobile> mobileList) | This method accepts a list of mobiles and returns a treemap with the launched year as key and count of mobiles as values. |

The mobile details should be given as a comma-separated value in the below order,  
**referenceId, modelName, displaySize, price, launchedDate**  
  
When the “mobile” object is printed, it should display the following details  
Print format:  
**System.out.format("%-15s %s\n","Year","No. of Mobiles");**  
Display one digit after the decimal point for the Double datatype.  
  
**Sample Input and Output 1:**  
  
Enter the number of mobiles  
**5  
#SM 48 DUO-004,J7 Duo,5.6,17000,23-09-2016  
#SM 49 PRO-005,J7 Pro,5.5,16900,15-12-2017  
#MT 57 XMM-067,X4,5.4,24999,14-12-2018  
#RM 97 PRO-061,Note 5 Pro,5.9,14999,02-01-2016  
#RM 17 NOT-063,Note 5,5.9,11999,14-05-2018**  
Year            No. of Mobiles  
2016            2  
2017            1  
2018            2

Close