**ArrayList - Introduction**

We have been using an array to store a group of objects. But arrays are of fixed size and are difficult to use compared to collections. So we are gonna move into collections. The basic collection is a list. Now let us try out basic ArrayList.  
  
Create a class **Main** and in the main method get the usernames and store them in an ArrayList. After getting all the names, just display them in the same order.  
  
**Input and Output format:**  
Refer to sample Input and Output for formatting specifications.  
  
**Note:** **All Texts in bold corresponds to the input and rest are output**  
**Sample Input and Output 1:**  
  
Enter the username 1  
**John**  
Do you want to continue?(y/n)  
**y**  
Enter the username 2  
**Joe**  
Do you want to continue?(y/n)  
**n**  
The Names entered are:  
**John**  
**Joe**  
  
**Sample Input and Output 2:**  
  
Enter the username 1  
**Jack**  
Do you want to continue?(y/n)  
**y**  
Enter the username 2  
**Jason**  
Do you want to continue?(y/n)  
**n**  
The Names entered are:  
**Jack**  
**Jason**

import java.util.Scanner;

import java.util.List;

import java.util.ArrayList;

public class Main {

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

//write your code here

boolean result=false;

List<String> list = new ArrayList<String>();

int i=1;

Scanner scanner = new Scanner(System.in);

do{

System.out.println("Enter the username " + i++);

String name = scanner.nextLine();

list.add(name);

System.out.println("Do you want to continue?(y/n)");

String diss = scanner.nextLine();

result=diss.equals("y");

}while(result);

System.out.println("The Names entered are:");

for(String s:list){

System.out.println(s);

}

}

}

**Set Introduction**

In the program let’s try using a **Set**. The property of Set is, it doesn't allow duplicate elements and does not maintain order like a list. Understand it by going through and completing the problem.  
  
Write a program to get the username and store it in the set and display the unique number of the username in the set.  
  
Create a driver class called **Main**. In the Main method, obtain username input from the user.  
  
**Input and Output format:**  
Refer to sample Input and Output for formatting specifications.  
  
**Sample Input and Output:  
[All text in bold corresponds to the input and rest corresponds to output]**  
  
Enter the username  
**Ram**  
Do you want to continue?(Y/N)  
**Y**  
Enter the username  
**Christoper**  
Do you want to continue?(Y/N)  
**Y**  
Enter the username  
**Ahamed**  
Do you want to continue?(Y/N)  
**Y**  
Enter the username  
**Ahamed**  
Do you want to continue?(Y/N)  
**N**  
The unique number of usernames is 3

Import java.util.HashSet;

import java.util.Scanner;

import java.util.Set;

public class Main {

public static void main(String[] args){

//Your code here

String username, askForContinue;

Scanner sc = new Scanner(System.in);

Set<String> userSet = new HashSet<>();

do {

System.out.println("Enter the username");

username = sc.nextLine();

userSet.add(username);

System.out.println("Do you want to continue?(Y/N)");

askForContinue = sc.nextLine();

} while (askForContinue.equals("Y"));

System.out.println("The unique number of usernames is " + userSet.size());

}

}

**TreeMap()**

To assist Event organizers, you need to develop a console application that shows the number of tickets sold in a particular price category. Thus enabling them to increase or decrease seats allocated for different price levels and thereby boosting ticket sales. The list of booking details that contains customer and price details are given.   
  
Use TreeMap with price as key and number of seats booked as value.  
  
Create a driver class named **Main.**In the main method, obtain details and display the price along with the number of tickets in increasing order of price.

**Input Format:**  
The first line of the input corresponds to the number of events 'n'.  
The next 'n' line of inputs corresponds to the event details in CSV format (Customer Name, Ticket Price, No of Seats Booked).  
Refer to Sample Input and Output for formatting specifications.  
  
**Output Format:**  
The output consists of the number of tickets booked for a particular ticket price in increasing order of price.  
Use (**"%-15s %s\n","Ticket Price","Tickets Booked")**for the format.  
Refer to Sample Input and Output for formatting specifications.  
  
**Sample Input and Output 1:**  
**[All Texts in bold corresponds to the input and rest are output]**  
  
Enter the number of events:  
**3**  
Enter event details in CSV(Customer Name,Ticket Price,No of Seats Booked)  
**Pramod,100,5  
Anamika,200,10  
Priscilla,100,3**  
Ticket Price    Tickets Booked  
100                 8  
200                 10

import java.util.Map;

import java.util.Scanner;

import java.util.TreeMap;

public class Main {

public static void main(String[] args){

//fill your code here

String[] events;

String input;

Scanner sc = new Scanner(System.in);

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

int noOfEvent = sc.nextInt();

sc.nextLine();

TreeMap<Integer,Integer> map = new TreeMap<Integer,Integer>();

System.out.println("Enter event details in CSV(Customer Name,Ticket Price,No of Seats Booked)");

for(int i=0;i<noOfEvent;i++) {

input = sc.nextLine();

events = input.split(",");

String name = events[0];

Integer price = Integer.parseInt(events[1]);

Integer seatBooked = Integer.parseInt(events[2]);

if(map.containsKey(price)) {

Integer seat = map.get(price);

map.put(price, (seat+seatBooked));

} else {

map.put(price, seatBooked);

}

}

System.out.println("Ticket Price Tickets Booked");

for(Map.Entry m:map.entrySet()){

System.out.printf("%-15s %s\n",m.getKey(),m.getValue());

}

}

}

**Comparable Interface**

Let's get in touch with the comparable interface. Given the list of Address details, sort them based on Pincode. If two address has the same Pincode, then sort them based on address line 1. This sorting will help us for segregating users based on Pincode when certain details (City and state details) are unavailable.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

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

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| username | String |
| addressLine1 | String |
| addressLine2 | String |
| pinCode | Integer |

Include appropriate getters and setters  
Create default constructor and a parameterized constructor with arguments in order  
**Address(String username, String addressLine1, String addressLine2, Integer pinCode)**.  
  
The **Address**class implements the **comparable interface**. Compare pin code, If Pincode is the same then compare with addressLine1.  
  
Create a driver class named **Main** to test the above class. Obtain input from the console and sort the user list.

**Input Format:**  
The first line input corresponds to the number of users 'n'.  
The next 'n' line of inputs corresponds to the user details in CSV format(Username,AddressLine 1,AddressLine 2,PinCode).  
Refer to sample input for formatting specifications.  
  
**Output Format:**  
The output consists of user details in the CSV format in sorted order. Print the output in the main method.  
Refer to sample output for formatting specifications.

**Sample Input and Output 1:**  
**[All text in bold corresponds to the input and rest corresponds to the output]**  
  
Enter the number of users:  
**3**  
Enter user address in CSV(Username,AddressLine 1,AddressLine 2,PinCode)  
**Josh,Marina street,Chennai,646461  
Martin,Mullai nagar,Salem,640002  
Justin,Ambedkar road,Chennai,646461**  
User Details:  
Martin,Mullai nagar,Salem,640002  
Justin,Ambedkar road,Chennai,646461  
Josh,Marina street,Chennai,646461

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String[] args){

//fill code here

Scanner sc = new Scanner (System.in);

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

//int num = Integer.parseInt(sc.nextLine());

int num = sc.nextInt();

sc.nextLine();

System.out.println("Enter user address in CSV(Username,AddressLine 1,AddressLine 2,PinCode)");

//Address userAddress = new Address();

ArrayList<Address> address = new ArrayList<Address>();

for (int i = 0; i < num; i++) {

String info [] = sc.nextLine().split(",");

address.add(new Address(info[0],info[1],info[2],Integer.parseInt(info[3])));

}

Collections.sort(address);

System.out.println("User Details:");

for (Address addr : address)

{

System.out.println(addr.getUsername()+"," + addr.getAddressLine1()+"," + addr.getAddressLine2()+"," + addr.getPinCode());

}

}

}

public class Address implements Comparable<Address>{

private String username;

private String addressLine1;

private String addressLine2;

private int pinCode;

Address(){

}

Address(String username, String addressLine1, String addressLine2, Integer pinCode) {

this.username = username;

this.addressLine1 = addressLine1;

this.addressLine2 = addressLine2;

this.pinCode = pinCode;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getAddressLine1() {

return addressLine1;

}

public void setAddressLine1(String addressLine1) {

this.addressLine1 = addressLine1;

}

public String getAddressLine2() {

return addressLine2;

}

public void setAddressLine2(String addressLine2) {

this.addressLine2 = addressLine2;

}

public int getPinCode() {

return pinCode;

}

public void setPinCode(int pinCode) {

this.pinCode = pinCode;

}

public int compareTo(Address address){

if (pinCode == address.pinCode)

{

return (addressLine1.compareTo(address.addressLine1));

//return 0;

}

else if (pinCode > address.pinCode)

{

return 1;

}

else

{

return -1;

}

}

}

**reverse() method**

In the collection, sort() method sort the objects in the ascending order. Suppose if we want to sort the list of objects in the descending order, we can use of reverse() method. Write a program to implement the reverse() method along with sort() to sort the list of User objects in the descending order.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
So let's create a class **User** with the following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Data type** |
| name | String |
| mobileNumber | String |
| username | String |
| password | String |

Include appropriate getter/setter, default constructor and parameterized constructor.  
  
Override **toString()** and print the details in a tabular format.  
  
Implement Comparable and sort the user objects based on name and reverse it by using the reverse().  
  
Create a driver class **Main**and using the main method get the details, create a map and display the details.  
  
**Hint:** Sort the user details based on the name of the user.  
  
**Input format:**  
The first line of input consists of number of users n.  
The next n line of input consists of user details in the CSV format (name,mobileNumber,userName,password).  
  
**Output format:**  
Display the name and the mobile number of the user in the reverse order.  
Use "%-15s%-15s" to display details in tabular format.  
Refer to sample input and output for other further details and format of the output.  
  
**Sample Input and Output 1:  
[All Texts in bold corresponds to the input and rest are output]**  
  
Enter the number of users:  
**3**  
Enter the details of User 1  
**Jack,12345,Jack,Jack**  
Enter the details of User 2  
**Jane,13579,Jane,Jane**  
Enter the details of User 3  
**John,24680,John,John**  
The user details in reverse order:  
Name           Mobile number    
John           24680            
Jane           13579            
Jack           12345

public class User implements Comparable<User> {

//write your code here

private String name;

private String mobileNumber;

private String username;

private String password;

public User(String name, String mobileNumber, String username, String password) {

this.name = name;

this.mobileNumber = mobileNumber;

this.username = username;

this.password = password;

}

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 getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public int compareTo(User user){

return (this.name.compareTo(user.name));

}

@Override

public String toString() {

return String.format("%-15s%-15s",getName(),getMobileNumber());

}

}

------------------------------

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String args[]) {

//write your code here

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

Scanner sc = new Scanner(System.in);

ArrayList<User> UserList = new ArrayList<User>();

try {

int num = sc.nextInt();

sc.nextLine();

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

{

System.out.println("Enter the details of User " + (i+1));

String user\_info = sc.nextLine();

String user[] = user\_info.split(",");

UserList.add(new User(user[0], user[1], user[2], user[3]));

}

Collections.sort(UserList);

Collections.reverse(UserList);

}

catch (Exception e){

}

finally {

System.out.println("The user details in reverse order:");

System.out.printf("%-15s%-15s\n","Name", "Mobile number");

for (User user: UserList){

System.out.println(user.toString());

}

}

}

}

**Generic Classes**

Create a generic class **Item**with a data. Write two methods **set**and **get**, to set a value to the data variable and to get the value of the data variable respectively. From **Main**class create two object for the class Item of types Integer and String.  
  
Refer sample input output form input output format  
  
**Sample Input and Output:**  
Enter a integer :  
**15**  
Enter a string :  
**Hello World Generic class**  
Integer Value :15  
String Value :Hello World Generic class

public class Item <T> {

private T t;

public void set(T t) {

this.t = t;

}

public T get() {

return t;

}

}

-------------------

import java.util.Scanner;

import java.io.\*;

public class Main {

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

Scanner sc = new Scanner(System.in);

Item<Integer> integerItem = new Item<Integer>();

Item<String> stringItem = new Item<String>();

System.out.println("Enter a integer :");

integerItem.set(sc.nextInt());

sc.nextLine();

System.out.println("Enter a string :");

stringItem.set(sc.nextLine());

System.out.println("Integer Value :" + integerItem.get());

System.out.println("String Value :" + stringItem.get());

}

}

**Iterator class**

It is time to explore some unique classes and methods in Collections. The Iterator class is one such. You have created test data for Stall class with the name of stall starting with prefix 'test', It's time to remove those objects. While iterating a collection through a for loop or a for each loop, you cannot modify or remove an element. The Iterator class facilitates such functionalities. Hence while you iterate through a Collection using Iterator you can modify the elements. Let's implement it to delete test data.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
Create a class **Stall** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| detail | String |
| type | String |
| ownerName | String |

Include getters and setters.  
Create a default and Parameterized constructors.  
Format for the parameterized constructor is**Stall(String name, String detail, String type, String ownerName)**  
  
Create a driver class called **Main**. In the Main method, obtain stall input from the user and create a list of Stall details. Using the Iterator class iterate through the List and remove stalls with a name starting with 'test'.  
Display the list of details in tabular form.  
  
**Input format:**  
The first line consists of the number of stalls 'n'  
The next 'n' line consists of 'n' stall details given in CSV format. **(name, detail,type,ownerName)**  
  
**Output format:**  
The first line of output displays the heading of the stall details.   
Then the stall details without containing the prefix 'test' are displayed in tabular format  
Use (**"%-15s %-20s %-15s %s"**) for formatting   
Refer to the sample Input and Output for further details and for the formatting specifications.  
  
**Sample Input and Output:**  
**[All text in bold corresponds to the input and rest corresponds to the output]**  
  
Enter the number of stall details  
**5**  
Enter the stall 1 detail  
**test1,detail,type,johndoe**  
Enter the stall 2 detail  
**test2,detail1,type,janedoe**  
Enter the stall 3 detail  
**Food court,fruits and juice,food,Mahesh**  
Enter the stall 4 detail  
**Book,novels,sale,Rajesh**  
Enter the stall 5 detail  
**test,data,data,data**  
Name            Detail               Type            Owner name  
Food court      fruits and juice     food            Mahesh  
Book            novels               sale            Rajesh

**sort() a List of Objects**

 Write a program to take hall objects as input in the list and sort them in the order of their costPerDay using **sort()** method of the comparable interface. Then display them in tabular form.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names,attribute names and method names should be the same as specified in the problem statement.**  
  
Create a class **Hall** with the following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Data type** |
| name | String |
| contactNumber | String |
| costPerDay | Double |
| ownerName | String |

Include appropriate getter/setter, default and parameterized constructor.  
  
Override **toString()**and print the details in a tabular format. And implement comparable interface in the class.  
  
Create driver class **Main** and use the main method to get inputs, sort, and display.  
  
**Input Format:**  
The first line has the number of halls n.  
The next n lines have details of the hall in CSV format. **(name,contactNumber,costPerDay,ownerName).**  
  
**Output format:**  
Use **"%-15s%-15s%-15s%-15s"** to display the hall details in the sorted order based on the cost per day as in tabular form.  
Refer to sample input and output for other further details and format of the output.  
  
**Note: All Texts in bold corresponds to the input and rest are output.  
  
Sample Input and Output 1:**  
  
Enter the number of halls:  
**3**  
Enter the details of hall 1  
**SDH hall,12345,12000,Jane**  
Enter the details of hall 2  
**SRT hall,13579,20000,John**  
Enter the details of hall 3  
**XUV hall,24680,15000,Jack**  
Sorted Order (from the least expensive to the most):  
Name           Contact number Cost per day   Owner name       
SDH hall       12345          12000.0        Jane             
XUV hall       24680          15000.0        Jack             
SRT hall       13579          20000.0        John     
  
**Sample Input and Output 2:**  
  
Enter the number of halls:  
**4**  
Enter the details of hall 1  
**ABC hall,12345,13000,John**  
Enter the details of hall 2  
**STR hall,13579,25000,Jane**  
Enter the details of hall 3  
**DFG hall,24680,10000,Jack**  
Enter the details of hall 4  
**JKL hall,67890,20000,Joe**  
Sorted Order (from the least expensive to the most):  
Name           Contact number Cost per day   Owner name       
DFG hall       24680          10000.0        Jack             
ABC hall       12345          13000.0        John             
JKL hall       67890          20000.0        Joe              
STR hall       13579          25000.0        Jane           

**Seat Arrangement**

Having recapped with already learned collection concepts, it's time to get involved in complex collection concepts. We would have created a list of primitive datatype and objects. Let's start with List of List in this exercise. Create a structure of seats in a StageEvent given the details of the number of rows and columns. Assign Section chronologically starting with 'A' and number starting from 1.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names, and method names should be the same as specified in the problem statement.**  
  
Create a class called **Seat**with following private variables.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| section | Character |
| number | Integer |
| booked | Boolean |

Include getters and setters.  
Create a default and Parameterized constructors.  
The format for the parameterized constructor is **Seat(Character section, Integer number,Boolean booked).**  
  
The **Seat**class has the following methods

|  |  |
| --- | --- |
| **Method name** | **Description** |
| static List<List<Seat>> generateSeats(int rows,int seat) | This method gets the number of rows and seats per row as arguments and It returns a List of seat List. |
| static void book(List<List<Seat>> seat,String seats) | This method accepts the List of List of Seats and String containing seats to be booked. It changes the booked variable to true for seats to be booked. |

Create a driver class called **Main**. In the Main method, obtain input from the user and create a list of list of  Seats. obtain Seat details for booking and at last display the Booked seats.  
  
**Input format:**  
The first line corresponds to the number of rows  
The second line corresponds to the number of seats per row  
The third line consists of tickets to be booked in CSV format.  
  
**Output format:**  
Seats that are booked are represented by "**--**" whereas the unbooked seats are represented by the section and number  
  
**[All text in bold corresponds to the input and rest corresponds to the output]  
Sample Input/Output 1:**  
  
Enter the number of rows  
**5**  
Enter the number of seats per row  
**5**  
Enter the seats to book in CSV format  
**A1,B2,C4,D3**

Seats

-- A2 A3 A4 A5

B1 -- B3 B4 B5

C1 C2 C3 -- C5

D1 D2 -- D4 D5

E1 E2 E3 E4 E5 

**Email Search**

In your application let’s dive deep into Set and explore its inbuilt functions. In this problem experiment with containsAll() method. Write a program to search all the email addresses which are given as CSV format.  
  
Create a **Main** class. Obtain email addresses from the user and add them to a Set. At last, get a String that has multiple email addresses in CSV format. Print "**Email addresses are present**" if all email addresses are present in the Set, else print "**Email addresses are not present**".  
  
**Input and Output format:**  
Refer to sample Input and Output for formatting specifications.  
  
**Note:**All Texts in bold corresponds to the input and rest are output  
  
**Sample Input and Output 1:**  
Enter Email address  
**Merry@gmail.com**  
Do you want to Continue?(yes/no)  
**yes**  
Enter Email address  
**Peter@yahoo.com**  
Do you want to Continue?(yes/no)  
**yes**  
Enter Email address  
**Christian@hotmail.com**  
Do you want to Continue?(yes/no)  
**yes**  
Enter Email address  
**Merry@gmail.com**  
Do you want to Continue?(yes/no)  
**no**  
Enter the email addresses to be searched separated by comma  
**Merry@gmail.com,Peter@yahoo.com**  
Email addresses are present  
  
  
**Sample Input and Output 2:**  
Enter Email address  
**Manikandan@yahoo.com**  
Do you want to Continue?(yes/no)  
**yes**  
Enter Email address  
**bala@google.co.in**  
Do you want to Continue?(yes/no)  
**no**  
Enter the email addresses to be searched separated by comma  
**bala@google.co.in,jeryy@gmail.com**  
Email addresses are not present

**List of List**

We have already seen a problem in the list of lists. So let's try to use it in our application. While the users try to book the tickets for the events they should know the count of remaining tickets. Let's create a list of 5 days of the week each has a list of the count of remaining tickets for 4 shows. List<List<Integer>> is the general format and for the problem, dayList<showList<count>>, ie., store the count of ticket available for each show of a day in a list and then place these lists for each day of a week inside another list.  
  
The maximum number of tickets for a show is 100. So after getting the bulk booked tickets from the user, subtract and store the remaining count of tickets for the whole week in this list of lists.  
  
Create a driver class **Main** and use the main method to get the count of already booked tickets and create a list of the list to store the remaining count.  
  
**Note**:CSV input format is (show1,show2,show3,show4) for each day. And enter day to know remaining ticket count for the day.  
  
Refer sample input/output for other further details and format of the output.  
  
**Input Format:**  
The first five lines have the number of tickets booked in each day  
The next lines have the day in which the remaining ticket to be shown  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input/Output 1:**  
  
Enter the count of booked tickets:  
On Day 1  
**20,25,30,35**  
On Day 2  
**20,20,20,20**  
On Day 3  
**15,25,35,20**  
On Day 4  
**50,60,40,75**  
On Day 5  
**85,88,93,78**  
Enter the day to know its remaining ticket count:  
**5**  
Remaining tickets:[15, 12, 7, 22]  
Do you want to continue?(y/n)  
**y**  
Enter the day to know its remaining ticket count:  
**2**  
Remaining tickets:[80, 80, 80, 80]  
Do you want to continue?(y/n)  
**y**  
Enter the day to know its remaining ticket count:  
**4**  
Remaining tickets:[50, 40, 60, 25]  
Do you want to continue?(y/n)  
**n**

**Min() and Max()**

In our fair, we have decided to announce one lucky winner in many who attends the events. We want the lucky winner as the one who spends the most in our events. So write a program to find the minimum and maximum spenders from the list of visitors who attended our events along with the money they spent in the events. Use min and max functions.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names, and method names should be the same as specified in the problem statement.**

Create a class **TicketBooking** with following private attributes which implements the Comparable interface

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| customerName | String |
| price | Integer |

Include appropriate getters and setters  
Create default constructor and a parameterized constructor with arguments in order **TicketBooking(String customerName, Integer price)**and overrides the compare method.  
  
Create a driver class named **Main**to test the above class. Obtain input from the console , get a list of TicketBooking, and use **Collections.min()** and **Collections.max()** to find the customer who spent more and less amount for ticket booking.

**Input Format:**  
The first line input corresponds to the number of customers 'n'. n>0 else display "**Invalid Input**".  
The next 'n' line of inputs corresponds to the user details in CSV format (Customer Name, Price).  
Refer to sample input for formatting specifications.  
  
**Output Format:**  
The output consists of the minimum and maximum amount spent by the customer. If two or more customer price is the same, keep the 1st one's price.  
Refer to sample output for formatting specifications.

**[All Texts in bold corresponds to the input and rest are output]**  
**Sample Input/Output-1:**  
Enter the number of customers  
**4**  
Enter the booking price accordingly with customer name in CSV(Customer Name,Price)  
**Jenny,1200  
Maria,450  
Jaquilin,600  
Renita sarah,150**  
Renita sarah spends minimum amount of Rs.150  
Jenny spends maximum amount of Rs.1200

**Replica of a List**

User data is always important and backup has to be made for every now and then. First of all, we'll back up the User authorization data for practice. The List of user details is provided. create a replica of the given list and store it in a backup list. An exact replica of a collection can be created using the copy() method of the List API.  
Follow the instruction below and display the backup list.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names, and method names should be the same as specified in the problem statement.**

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

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| username | String |
| password | String |

Include appropriate getters and setters  
Create default constructor and a parameterized constructor with arguments in order **User(String username, String password).**  
  
Include following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| List<User> backUp(List<User>dest , List<User> source) | This method takes the source list(user list) and destination list(blank list) for back up. It returns a list with user details backed up. |

Create a driver class named **Main** to test the above class. In Main class create destination list of size as same source list with null values(without a null list it throws **IndexOutOfBoundsException**) and this has sent as destination list to the **backUp method**.

**Input Format:**  
The first line input corresponds to the number of users 'n'. n>0 else display "**Invalid Input**".  
The next 'n' line of inputs corresponds to the user details in CSV format(Username, Password).  
Refer to sample input for formatting specifications.  
  
**Output Format:**  
The output consists user details in the format of **System.out.format("%-20s %s\n","Username","Password")**;.  
Refer sample output for formatting specifications.

**[All Texts in bold corresponds to the input and rest are output]**  
**Sample Input/Output-1:**  
Enter number of users  
**3**  
Enter the user details in CSV(Username,password)  
**Daniel,merry  
Bisoph,qwertyuio!@12345  
Jaques,877878785565**  
Copy of user list:  
Username             Password  
Daniel                      merry  
Bisoph                     qwertyuio!@12345  
Jaques                    877878785565

**State map**

Let's have a different variant of multimap. Create a Map<String,Map<String,List<Address>>> with State name as key and a map as a value having City name as key and List of address as value. It should be understood that the address should have the state and city name as that of the key. At last obtain state and city as search terms and display the corresponding list of addresses.  
  
Create a class called **Address** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| addressLine1 | String |
| addressLine2 | String |
| city | String |
| state | String |
| pincode | Integer |

Include appropriate getters and setters.  
Include default and parameterized constructor for the class.  
Format for the Parameterized Constructor **Address(String addressLine1, String addressLine2, String city,  
String state, Integerpincode)**  
  
Create a driver class called Main. In the main method, obtain address details and create the map of above specification. Obtain state and city as search term and display the address that has the given city and state. If no such address is present, Print "**Searched city not found**" or "**Searched state not found**" accordingly.  
  
Note:   
**[Strictly adhere to the Object-Oriented Specifications given in the problem statement.  
All class names, attribute names and method names should be the same as specified in the problem statement.]**  
  
**Input format:**  
  
First line corresponds to number of address inputs 'n'  
next n lines consists of 'n' address details in CSV format  
n+1th line consists of state input  
n+2nd line consists of city input  
  
**Output format:**  
  
Address details are displayed in tabular format (Use "%-15s %-15s %-15s %-15s %s\n" for formatting Address details.)  
  
**[All text in bold corresponds to the input and rest corresponds to the output]  
Sample Input/Output 1:**  
  
Enter the number of address  
**4**  
Enter the address 1 detail  
**5/15 7th lane,RK nagar,Madurai,Tamil nadu,625001**  
Enter the address 2 detail  
**1/45 8th street,KK nagar,Chennai,Tamil nadu,600021**  
Enter the address 3 detail  
**3rd street,KRK nagar,Visak,Andhrapradesh,745601**  
Enter the address 4 detail  
**22nd lane,RR nagar,Chennai,Tamil nadu,600028**  
Enter the state to be searched  
**Tamil nadu**  
Enter the city to be searched  
**Madurai**  
Line 1          Line 2          City            State           Pincode  
5/15 7th lane   RKnagar        Madurai         Tamil nadu      625001

\

**Generic Methods**

Write a single generic method declaration that can be called with arguments of different types to print the elements of Integer, Double and Character arrays.  
  
**Input Output Format:**  
Input consists of a single integer corresponding to the number of elements in the arrays.  
Refer Sample Input Output for output format.  
  
**Sample Input and Output:**  
Enter a number :  
**6**  
Enter the elements of the integer array  
**1 2 3 4 5 6**  
Enter the elements of the double array  
**1.1 2.2 3.3 4.4 5.5 6.6**  
Enter the elements of the character array  
**a n c d e f**  
Integer array contains:  
1 2 3 4 5 6  
Double array contains:  
1.1 2.2 3.3 4.4 5.5 6.6  
Character array contains:  
a n c d e fTop of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form