**Q1. Write a class Person having weight, height & name. Create multiple person objects & print them in the sorted order. In the sorting order first sort based upon their weight & if two persons have same weight them sort them based upon their height. Use TreeSet.**

**Description:-**

Create Person class with variables weight and height.Create multiple person objects & print them in the sorted order based upon weight first if the weights are equal then based upon height. Use TreeSet.

**import** java.util.Comparator;

**import** java.util.Set;

**import** java.util.TreeSet;

**class** Person{

String name;

**int** height;

**double** weight;

**public** Person(String name, **int** height, **double** weight) {

**this**.name = name;

**this**.height = height;

**this**.weight = weight;

}

@Override

**public** String toString() {

**return** "Name = "+**this**.name+"| Height = "+**this**.height+"| Weight = "+**this**.weight;

}

}

**class** HeightComparator **implements** Comparator<Person>{

@Override

**public** **int** compare(Person o1, Person o2) {

**if**(o1.weight>o2.weight){

**return** -1;

}

**else** **if**(o1.weight<o2.weight){

**return** 1;

}

**else** {

**return** Double.*compare*(o2.height, o1.height);

}

}

}

**public** **class** Assignment5Q1 {

**public** **static** **void** main(String[] args) {

Set<Person>set = **new** TreeSet<>(**new** HeightComparator());

set.add(**new** Person("Premal Miglani",155,55));

set.add(**new** Person("Mihir Gaikwad",176,75));

set.add(**new** Person("Prateek Sengar",168,60));

set.add(**new** Person("Prathmesh Asole",172,57));

set.add(**new** Person("Rajan Singh",163,64));

set.add(**new** Person("Amit Ray",162,57.5));

set.add(**new** Person("Ayush Shrivastava",160,57.5));

System.***out***.println("Printing in the sorted order based upon weight first, if the weights are equal then based upon height:");

**for** (Person person: set){

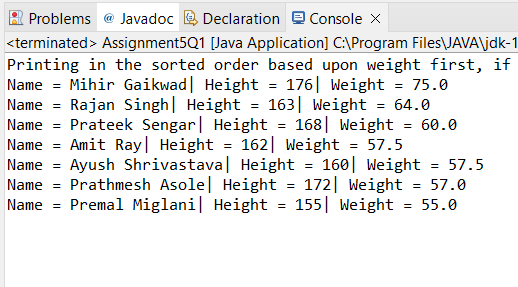
System.***out***.println(person);

}

}

}

**Output:**



**Q2.  Prove that Hash Set is unordered & Linked Hash Set is ordered.**

**Code:**

**import** java.util.HashSet;

**import** java.util.LinkedHashSet;

**import** java.util.Set;

**public** **class** Assignment5Q2 {

**public** **static** HashSet<Integer>unordered(HashSet<Integer>hashSet){

hashSet.add(3);

hashSet.add(4);

hashSet.add(1);

hashSet.add(5);

hashSet.add(2);

**return** hashSet;

}

**public** **static** LinkedHashSet<Integer>ordered(LinkedHashSet<Integer>linkedHashSet){

linkedHashSet.add(4);

linkedHashSet.add(5);

linkedHashSet.add(1);

linkedHashSet.add(2);

linkedHashSet.add(3);

**return** linkedHashSet;

}

**public** **static** **void** main(String[] args) {

Set<Integer>hashSet = **new** HashSet<>();

Set<Integer>linkedHashSet = **new** LinkedHashSet<>();

hashSet = *unordered*((HashSet<Integer>) hashSet);

linkedHashSet = *ordered*((LinkedHashSet<Integer>) linkedHashSet);

System.***out***.println("Elements in HashSet: ");

**for**(Object i:hashSet){

System.***out***.print(i+" ");

}

System.***out***.println("\nSee in HashSet the order in which the elements are inserted is not maintained ");

System.***out***.println("\nElements in LinkedHashSet: ");

**for**(**int** i:linkedHashSet){

System.***out***.print(i+" ");

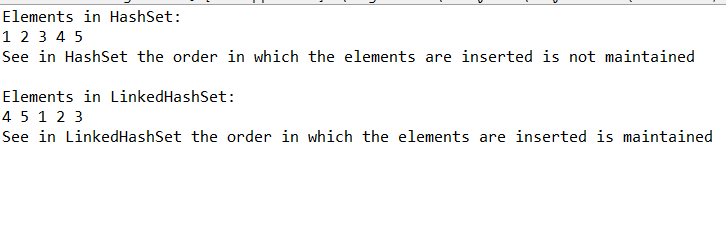
}

System.***out***.println("\nSee in LinkedHashSet the order in which the elements are inserted is maintained ");

}

}

**Output:**

****

**Q3. Create a ArrayList with few elements & print it in backward direction. Use ListIterator.**

**Description:**

Write a program which consists of ArrayList which has some elements and print them in reverse direction.

**Code:**

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Assignment5Q3 {

**public** **static** ArrayList traverseReverse(ArrayList aList){

Collections.*reverse*(aList);

**return** aList;

}

**public** **static** **void** main(String[] args) {

ArrayList<Integer>arrayList = **new** ArrayList<>();

arrayList.add(7);

arrayList.add(3);

arrayList.add(1);

arrayList.add(6);

arrayList.add(4);

arrayList.add(8);

arrayList.add(2);

arrayList.add(10);

arrayList.add(5);

arrayList.add(9);

System.***out***.println("Printing Original Array: ");

**for** (**int** i:arrayList){

System.***out***.print(i+" ");

}

System.***out***.println("\nPrinting Reverse Array: ");

arrayList = *traverseReverse*(arrayList);

**for** (**int** i:arrayList){

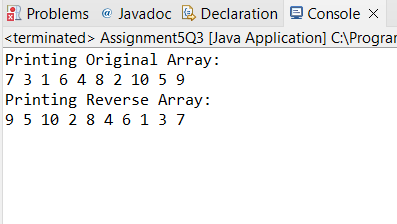
System.***out***.print(i+" ");

}

}

}

**Output:**

****

**Q4. Write a program using Hashtable or HashMap where Date of birth is a key & Employee name as value. Design the class Date is such a way where the get method fails if two employees have same day & month of birth but birth year is different**.

**Description:-**

Using hash table or hash map write a program where key is date of birth and employee name is value and also the condition in the question should be satisfied.

**Code:**

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Objects;

**class** DateClass{

**int** date;

**int** month;

**int** year;

**public** DateClass(**int** date, **int** month, **int** year) {

**this**.date = date;

**this**.month = month;

**this**.year = year;

}

**public** **int** getDate() {

**return** date;

}

**public** **void** setDate(**int** date) {

**this**.date = date;

}

**public** **int** getMonth() {

**return** month;

}

**public** **void** setMonth(**int** month) {

**this**.month = month;

}

**public** **int** getYear() {

**return** year;

}

**public** **void** setYear(**int** year) {

**this**.year = year;

}

@Override

**public** **boolean** equals(Object o) {

**if** (**this** == o) **return** **true**;

**if** (!(o **instanceof** DateClass)) **return** **false**;

DateClass dateClass = (DateClass) o;

**return** date == dateClass.date && month == dateClass.month || year == dateClass.year;

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(date, month, year);

}

@Override

**public** String toString() {

**return** "Date = "+**this**.date+" Month = "+**this**.month+" Year = "+**this**.year;

}

}

**public** **class** Assignment5Q4{

**public** **static** **void** main(String[] args) {

Map<DateClass,String> map = **new** HashMap<>();

DateClass d1 = **new** DateClass(29,9,2000);

DateClass d2 = **new** DateClass(21,1,2004);

DateClass d3 = **new** DateClass(29,9,2001);

DateClass d4 = **new** DateClass(15,7,1999);

map.putIfAbsent(d1,"Premal Miglani");

map.putIfAbsent(d2,"Ayush Sharma");

map.putIfAbsent(d3,"Aakash Pandey");

map.putIfAbsent(d4,"Rohan Mehra");

System.***out***.println("Key-value pair in HashMap: ");

**for**(Map.Entry<DateClass,String> m:map.entrySet()){

DateClass key = m.getKey();

String value = m.getValue();

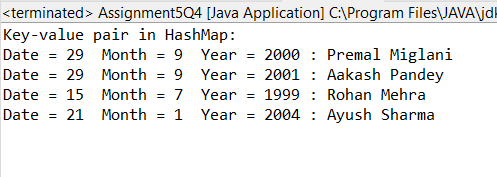
System.***out***.println(key+" : "+value);

}

}

}

**Output:**

****

**Q5. Write a user defined class say Employee that overrides equals() & hashCode() methods. Equals() always returns true & hashCode() always returns a fixed number. Make such a class as key of you Hashtable. Observe the behavior while calling put & get methods.**

**Description:-**

HashMap and HashSet use the hashcode value of an object to find out how the object would be stored in the collection, and subsequently hashcode is used to help locate the object in the collection. Hashing retrieval involves:

First, find out the right bucket using hashCode().

Secondly, search the bucket for the right element using equals().

**Code:**

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.Objects;

**class** employee{

String name;

**int** id;

**public** String getName() {

**return** name;

}

**public** **int** getId() {

**return** id;

}

**public** employee(**int** id, String name) {

**this**.name = name;

**this**.id = id;

}

@Override

**public** String toString() {

**return** "Id = "+**this**.id+" Name = "+**this**.name;

}

@Override

**public** **boolean** equals(Object o) {

**if** (**this** == o) **return** **true**;

**if** (!(o **instanceof** employee)) **return** **false**;

employee employee = (employee) o;

**return** id == employee.id;

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(id);

}

}

**public** **class** Assignment5Q5 {

**public** **static** **void** main(String[] args) {

Map<Integer,String>map = **new** HashMap<>();

employee e1 = **new** employee(1,"Premal Miglani");

employee e2 = **new** employee(1,"Mihir Gaikwad");

employee e3 = **new** employee(3,"Rohit Kore");

employee e4 = **new** employee(3,"Prathmesh Asole");

employee e5 = **new** employee(2,"Prateek Sengar");

map.put(e1.getId(), e1.getName());

map.put(e2.getId(), e2.getName());

map.put(e3.getId(), e3.getName());

map.put(e4.getId(), e4.getName());

map.put(e5.getId(), e5.getName());

System.***out***.println("Key-Value pair in map: ");

**for** (Map.Entry<Integer,String>m:map.entrySet()){

**int** key = m.getKey();

String value = m.getValue();

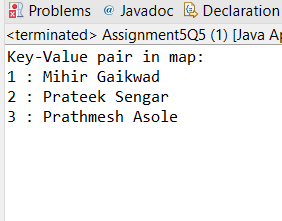
System.***out***.println(key+" : "+value);

}

}

}

**Output:**

****

**Q6. Implement the console based chatting using collections. Here are the options to be placed for to the user:**

**>java ChatApplication**

**Options:**

**A) Create a chatroom**

**B) Add the user**

**C) User login**

**D) Send a message**

**E) Display the messages from a specific chatroom**

**F) List down all users belonging to the specified chat room.**

**G) Logout**

**H) Delete an user**

**I) Delete the chat room.**

**Please enter your option:**

**Code:**

**import** java.util.\*;

**class** Chatroom{

String name;

Set<String>username;

List<String>messages;

**public** String getName() {

**return** name;

}

**public** Set<String>getUsername() {

**return** username;

}

**public** List<String>getMessages() {

**return** messages;

}

Chatroom(){

**this**.name = "";

**this**.username = **new** HashSet<String>();

**this**.messages = **new** ArrayList<String>();

}

@Override

**public** **boolean** equals(Object o) {

**if** (**this** == o) **return** **true**;

**if** (!(o **instanceof** Chatroom)) **return** **false**;

Chatroom chatroom = (Chatroom) o;

**return** username.equals(chatroom.username);

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(username);

}

**public** **boolean** removeUser(String username) {

**if**(**this**.username.contains(username)){

**this**.username.remove(username);

**return** **true**;

}

**return** **false**;

}

}

**class** User{

String username;

String password;

String firstName;

String lastName;

**public** User(){

username = "";

password = "";

firstName = "";

lastName = "";

}

**public** User(String username, String password, String firstName, String lastName) {

**this**.username = username;

**this**.password = password;

**this**.firstName = firstName;

**this**.lastName = lastName;

}

**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** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

@Override

**public** **boolean** equals(Object o) {

**if** (**this** == o) **return** **true**;

**if** (!(o **instanceof** User)) **return** **false**;

User user = (User) o;

**return** username.equals(user.username);

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(username);

}

}

**class** ChatApplication{

Map<String, Chatroom>chatrooms = **new** HashMap();

Map<String, User>users = **new** HashMap();

Set<String>loggedInUsers = **new** HashSet();

**int** chatroomCount = 0;

**int** userCount = 0;

**public** **boolean** isChatroomNameValid(String name) {

**for** (Map.Entry<String,Chatroom>mp: chatrooms.entrySet()){

Chatroom chatroom = mp.getValue();

**if**(chatroom.name.equals(name)){

**return** **true**;

}

}

**return** **false**;

}

**public** **boolean** isUsernameExists(String username) {

**for**(Map.Entry<String, User>mp: users.entrySet()){

User user = mp.getValue();

**if**(user.username.equals(username)){

**return** **true**;

}

}

**return** **false**;

}

**public** **boolean** authenticateUser(String username, String password) {

**for**(Map.Entry<String, User>mp: users.entrySet()){

User user = mp.getValue();

**if**(user.username.equals(username) &&user.password.equals(password)){

**return** **true**;

}

}

**return** **false**;

}

//UI Methods Below

**public** **void** createChatroom() {

Chatroom chatroom = **new** Chatroom();

chatroom.messages = **new** ArrayList<>();

chatroom.username = **new** HashSet<>();

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the chatroom name: ");

chatroom.name = sc.nextLine();

System.***out***.print("Enter the chatroom username: ");

chatroom.username.add(sc.nextLine());

System.***out***.print("Enter the chatroom message: ");

chatroom.messages.add(sc.nextLine());

System.***out***.println(chatroom.getName());

System.***out***.println(chatroom.getUsername());

System.***out***.println(chatroom.getMessages());

//Adding the message in map.

chatrooms.put(Integer.*toString*(chatroomCount+1),chatroom);

chatroomCount+=1;

}

**public** **void** addNewUser() {

User user = **new** User();

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the first name: ");

user.setFirstName(sc.nextLine());

System.***out***.print("Enter the last name: ");

user.setLastName(sc.nextLine());

System.***out***.print("Enter the username: ");

user.setUsername(sc.nextLine());

System.***out***.print("Enter the password: ");

user.setPassword(sc.nextLine());

//Adding user in map

users.put(Integer.*toString*(userCount+1),user);

userCount +=1;

}

**public** **boolean** login() {

User user = **new** User();

Scanner sc = **new** Scanner(System.***in***);

String username, password;

System.***out***.print("Enter the username: ");

username = sc.next();

System.***out***.println("Enter the password: ");

password = sc.next();

**if**(user.username.equals(username) &&user.password.equals(password)){

System.***out***.println("LOGIN SUCCESSFUL");

loggedInUsers.add(user.username);

**return** **true**;

}

**return** **false**;

}

**public** **void** sendMessage() {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the Message: ");

String message = sc.next();

System.***out***.println("Message has been sent successfully!");

System.***out***.println("Message is: "+message);

}

**public** **void** printMessages() {

**for** (Map.Entry<String,Chatroom>mp: chatrooms.entrySet()){

Chatroom chatroom = mp.getValue();

System.***out***.println("Username: "+chatroom.username);

System.***out***.println("Message: "+chatroom.messages);

}

}

**public** **void** listUsersFromChatroom() {

System.***out***.println("All the users from the chatroom");

**for**(Map.Entry<String ,Chatroom>mp: chatrooms.entrySet()){

Chatroom chatroom = mp.getValue();

System.***out***.println("username: "+chatroom.name);

}

}

**public** **void** logout(){

System.***out***.println("LOGOUT SUCCESSFULLY");

System.*exit*(0);

}

**public** **void** deleteUser(){

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the username to be deleted: ");

String username = sc.next();

**for** (Map.Entry<String,User>mp: users.entrySet()){

User user = mp.getValue();

String key = "null";

**if**(user.username.equals(username)){

key = mp.getKey();

**break**;

}

users.remove(key);

System.***out***.println("User removed successfully!");

}

}

**public** **void** deleteChatRoom(){

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter the chatroom username to be deleted: ");

String username = sc.next();

**for** (Map.Entry<String,Chatroom>mp: chatrooms.entrySet()){

Chatroom chatroom = mp.getValue();

String key = "null";

**if**(chatroom.username.equals(username)){

key = mp.getKey();

**break**;

}

users.remove(key);

System.***out***.println("User removed successfully!");

}

}

**public** **void** menu() {

Scanner sc = **new** Scanner(System.***in***);

**int** choice;

**do**{

System.***out***.println("1) Create a chatroom ");

System.***out***.println("2) Add the user ");

System.***out***.println("3) User login ");

System.***out***.println("4) Send a message ");

System.***out***.println("5) Display the messages from a specific chatroom ");

System.***out***.println("6) List down all users belonging to the specified chat room. ");

System.***out***.println("7) Delete an user ");

System.***out***.println("8) Delete the chat room. ");

System.***out***.println("9) Logout");

System.***out***.println("Please enter your option:");

choice = sc.nextInt();

**switch** (choice){

**case** 1: createChatroom();

System.***out***.println("ChatRoom Created Successfully!");

**break**;

**case** 2: addNewUser();

System.***out***.println("New User Added Successfuly!");

**break**;

**case** 3:

**if**(login()==**false**){

System.***out***.println("USERNAME OR PASSWORD IS INCORRECT PLEASE TRY AGAIN!");

}

**break**;

**case** 4: sendMessage();

**break**;

**case** 5: printMessages();

**break**;

**case** 6: listUsersFromChatroom();

**break**;

**case** 7: deleteUser();

**break**;

**case** 8: deleteChatRoom();

**break**;

**case** 9: logout();

**break**;

**default**:

System.***out***.println("PLEASE ENTER THE RIGHT CHOICE!!");

}

}**while**(choice!=9);

}

}

**public** **class** Assignment5Q6 {

**public** **static** **void** main(String[] args) {

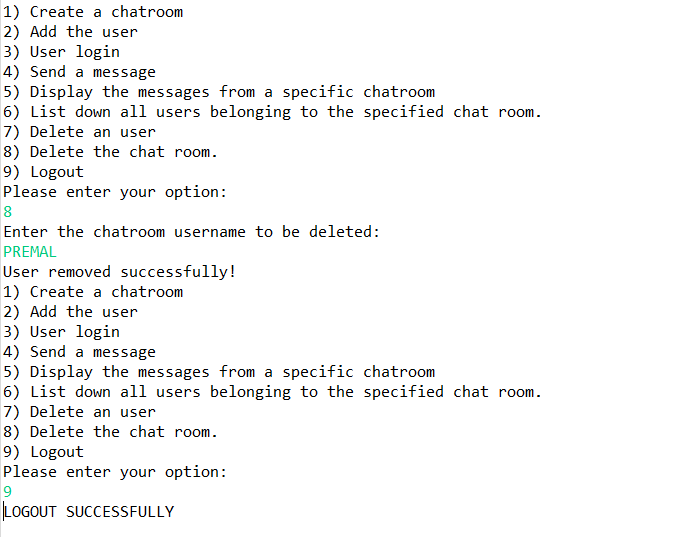
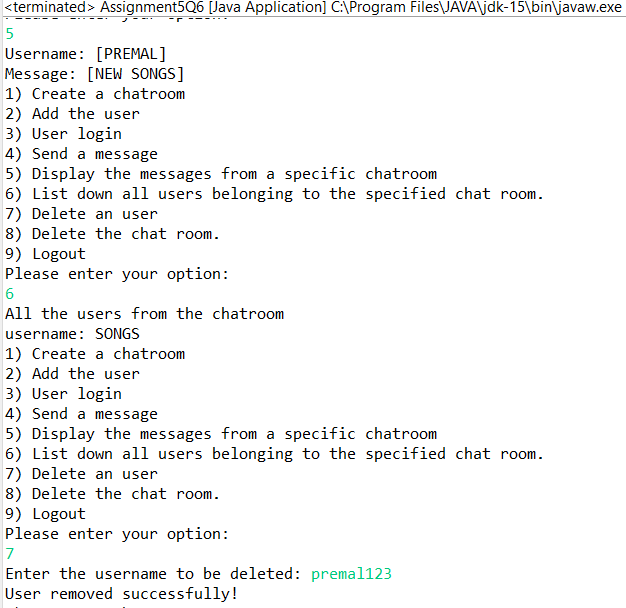
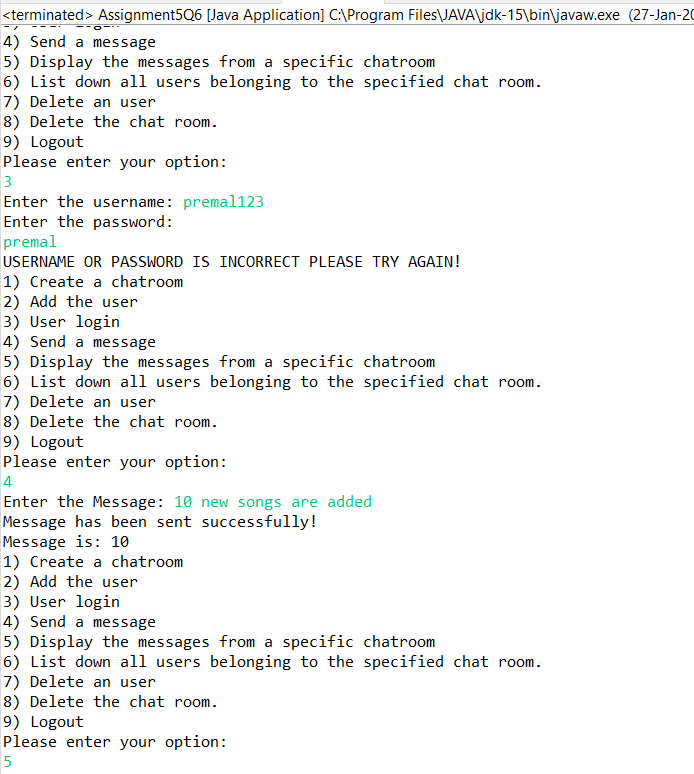
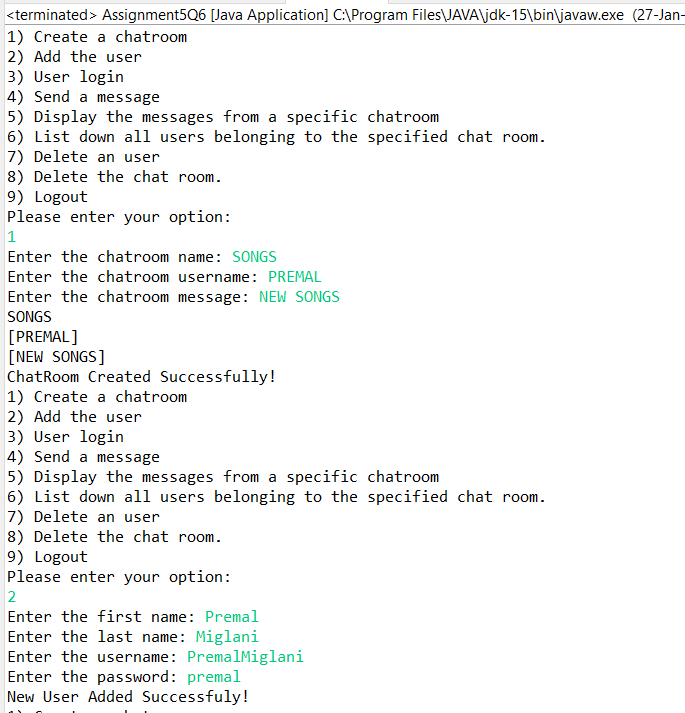
ChatApplication chatApplication = **new** ChatApplication();

chatApplication.menu();

}

}

**Output:**

****

**Q7. There is parking slot available in R-Mall with 3 floors; each floor has 4 sections and each section can maximum park 20 cars. You need to design one application which will maintain all car details in such way when a car owner arrives to collect his care your application should provide details including where it is located.**

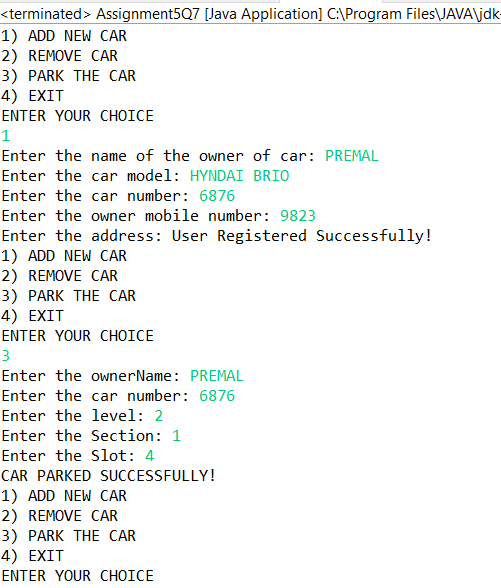
**a. Create class Parked\_CarOwner\_Details which will have field’s owerName, carModel, carNO, owerMobileNo, owerAddress with setter and getter methods.**

**b. Create class Parked\_CarOwenerList which will have method’s int add\_new\_car(Parked\_CarOwner\_Details obj), remove\_car(), get\_parked\_car\_location(token).**

**Code:**

import java.util.HashMap;  
import java.util.*Map*;  
import java.util.*Map*.*Entry*;  
import java.util.Scanner;  
  
class ParkingSlot {  
private String ownerName;  
private int carNumber;  
private int token;  
private int level;  
private int section;  
private int slot;  
  
public String getOwnerName() {  
return ownerName;  
 }  
  
public void setOwnerName(String ownerName) {  
this.ownerName = ownerName;  
 }  
  
public int getCarNumber() {  
return carNumber;  
 }  
  
public void setCarNumber(int carNumber) {  
this.carNumber = carNumber;  
 }  
  
  
public void setLevel(int level) {  
this.level = level;  
 }  
  
public int getSection() {  
return section;  
 }  
  
public void setSection(int section) {  
this.section = section;  
 }  
  
public int getSlot() {  
return slot;  
 }  
  
public void setSlot(int slot) {  
this.slot = slot;  
 }  
}  
  
class Parked\_CarOwenerList extends Assignment5Q7{  
int id = 1;  
int levels = 3;  
int sections = 4;  
int slots = 20;  
*Map*<Integer,Assignment5Q7>car = new HashMap<>();  
*Map*<Integer,ParkingSlot>parkingSlotMap = new HashMap<>();  
  
public void add\_new\_car(Assignment5Q7 obj){  
Scanner sc = new Scanner(System.in);  
System.out.print("Enter the name of the owner of car: ");  
obj.setName(sc.nextLine());  
System.out.print("Enter the car model: ");  
obj.setCarModel(sc.nextLine());  
System.out.print("Enter the car number: ");  
obj.setCarNo(sc.nextInt());  
System.out.print("Enter the owner mobile number: ");  
obj.setMobileNumber(sc.nextInt());  
System.out.print("Enter the address: ");  
obj.setAddress(sc.nextLine());  
  
car.put(id,obj);  
id++;  
System.out.println("User Registered Successfully!");  
 }  
public void remove\_car(String name,int carNo){  
int key = 0;  
for (*Map*.*Entry*<Integer, Assignment5Q7>mp: car.entrySet()){  
Assignment5Q7 obj = mp.getValue();  
if(obj.getName().equals(name)&&obj.getCarNo()==carNo){  
key = mp.getKey();  
break;  
 }  
 }  
car.remove(key);  
 }  
public String get\_parked\_car\_location(int token){  
Scanner sc = new Scanner(System.in);  
ParkingSlot parkingSlot = new ParkingSlot();  
  
System.out.print("Enter the ownerName: ");  
parkingSlot.setOwnerName(sc.nextLine());  
  
System.out.print("Enter the car number: ");  
parkingSlot.setCarNumber(sc.nextInt());  
  
System.out.print("Enter the level: ");  
parkingSlot.setLevel(sc.nextInt());  
  
System.out.print("Enter the Section: ");  
parkingSlot.setLevel(sc.nextInt());  
  
System.out.print("Enter the Slot: ");  
parkingSlot.setSlot(sc.nextInt());  
  
parkingSlotMap.put(token,parkingSlot);  
return "CAR PARKED SUCCESSFULLY!";  
 }  
}  
  
public class Assignment5Q7 {  
String name;  
String carModel;  
int carNo;  
int mobileNumber;  
String address;  
  
public String getName() {  
return name;  
 }  
  
public void setName(String name) {  
this.name = name;  
 }  
  
  
public void setCarModel(String carModel) {  
this.carModel = carModel;  
 }  
  
public int getCarNo() {  
return carNo;  
 }  
  
public void setCarNo(int carNo) {  
this.carNo = carNo;  
 }  
  
public void setMobileNumber(int mobileNumber) {  
this.mobileNumber = mobileNumber;  
 }  
  
public void setAddress(String address) {  
this.address = address;  
 }  
  
public void menu(){  
Parked\_CarOwenerList parked\_carOwenerList = new Parked\_CarOwenerList();  
Scanner sc = new Scanner(System.in);  
int choice;  
do{  
System.out.println("1) ADD NEW CAR");  
System.out.println("2) REMOVE CAR");  
System.out.println("3) PARK THE CAR");  
System.out.println("4) EXIT");  
System.out.println("ENTER YOUR CHOICE");  
choice = sc.nextInt();  
switch (choice){  
case 1: parked\_carOwenerList.add\_new\_car(new Assignment5Q7());  
break;  
case 2: String name;  
int number;  
System.out.print("Enter the owner name: ");  
name = sc.nextLine();  
System.out.print("Enter the car number: ");  
number = sc.nextInt();  
  
parked\_carOwenerList.remove\_car(name,number);  
break;  
case 3:  
System.out.println(parked\_carOwenerList.get\_parked\_car\_location(101));  
break;  
case 4:  
System.*exit*(0);  
  
default:  
System.out.println("PLEASE ENTER THE CORRECT CHOICE!!");  
 }  
 }while (choice!=4);  
 }  
public static void main(String[] args) {  
Assignment5Q7 obj = new Assignment5Q7();  
obj.menu();  
 }  
}

**Output:**

** **

**Q8.1.  Test fail-fast iterators within multithreaded environment. Note example of fail fast iterator is Vector, ArrayList, HashSet etc. And fail-safe is ConcurrentHashMap, CopyOnWriteArrayList etc.**

**Fail Fast:**

Iterators in java are used to iterate over the Collection objects.Fail-Fast iterators immediately throw ConcurrentModificationException if there is structural modification of the collection. Structural modification means adding, removing or updating any element from collection while a thread is iterating over that collection. Iterator on ArrayList, HashMap classes are some examples of fail-fast Iterator.

**Code:**

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**public** **class** Assignment5Q8 {

**public** **static** **void** main(String[] args) {

Map<String, String>cityCode = **new** HashMap<String, String>();

cityCode.put("Delhi", "India");

cityCode.put("Moscow", "Russia");

cityCode.put("New York", "USA");

Iterator iterator = cityCode.keySet().iterator();

**while** (iterator.hasNext()) {

System.***out***.println(cityCode.get(iterator.next()));

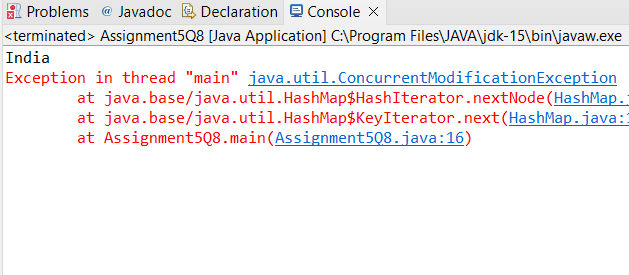
cityCode.put("Istanbul", "Turkey");

}

}

}

**Output:**



**Q8.2. Test fail-safe iterators within multithreaded environment. Note example of fail fast iterator is Vector, ArrayList, HashSet etc. And fail-safe is ConcurrentHashMap, CopyOnWriteArrayList etc.**

**Fail Safe:**

Fail-Safe iterators don’t throw any exceptions if a collection is structurally modified while iterating over it. This is because, they operate on the clone of the collection, not on the original collection and that’s why they are called fail-safe iterators. Iterator on CopyOnWriteArrayList, ConcurrentHashMap classes are examples of fail-safe Iterator.

**Code:**

**import** java.util.concurrent.CopyOnWriteArrayList;

**import** java.util.Iterator;

**public** **class** Assignment5Q8B {

**public** **static** **void** main(String[] args) {

CopyOnWriteArrayList<Integer>list

= **new** CopyOnWriteArrayList<Integer>(**new** Integer[] { 1, 3, 5, 8 });

Iterator itr = list.iterator();

**while** (itr.hasNext()) {

Integer no = (Integer)itr.next();

System.***out***.println(no);

**if** (no == 8)

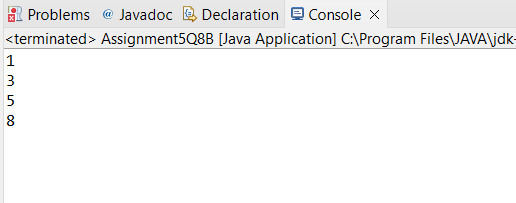
list.add(14);

}

}

}

**Output:**

****

[**Q9**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2150)**. Create a Class SavingAccount with field’s acc\_balance, acc\_ID, accountHoldername, isSalaryAccount. Also add setter and getter methods with business method like withdraw and deposit.**

**a. Create class BankAccountList which will maintain SavingAccount objects. Ensure that this class should not allow duplicates as well as data should be displayed in sorted order. (as per acc\_ID)**

**Code:**

**import** java.util.\*;

**class** savingaccount {

**private** **double** acc\_balance;

**private** **int** acc\_ID;

**private** String accountHolderName;

**private** **boolean** isSalaryAccount;

**public** **double** getAcc\_balance() {

**return** acc\_balance;

}

**public** **void** setAcc\_balance(**double** acc\_balance) {

**this**.acc\_balance = acc\_balance;

}

**public** **int** getAcc\_ID() {

**return** acc\_ID;

}

**public** **void** setAcc\_ID(**int** acc\_ID) {

**this**.acc\_ID = acc\_ID;

}

**public** String getAccountHolderName() {

**return** accountHolderName;

}

**public** **void** setAccountHolderName(String accountHolderName) {

**this**.accountHolderName = accountHolderName;

}

**public** **boolean** isSalaryAccount() {

**return** isSalaryAccount;

}

**public** **void** setSalaryAccount(**boolean** salaryAccount) {

isSalaryAccount = salaryAccount;

}

**public** **void** withDraw(**double** amount){

**if**(**this**.acc\_balance<amount){

System.***out***.println("YOUR ACCOUNT HAS INSUFFICIENT BALANCE");

}

**else**{

**this**.acc\_balance-=amount;

System.***out***.println("MONEY WITHDRAWN SUCCESSFULLY");

System.***out***.println("ACCOUNT BALANCE = "+**this**.acc\_balance);

}

}

**public** **void** Deposit(**double** amount){

**this**.acc\_balance += amount;

System.***out***.println("MONEY ADDED SUCCESSFULLY");

System.***out***.println("ACCOUNT BALANCE = "+**this**.acc\_balance);

}

}

**class** accountComparator **implements** Comparator<savingaccount>{

@Override

**public** **int** compare(savingaccount o1, savingaccount o2) {

**if**(o1.getAcc\_ID()>o2.getAcc\_ID()){

**return** -1;

}

**else** **if**(o1.getAcc\_ID()<o2.getAcc\_ID()){

**return** 1;

}

**else** {

**return** 0;

}

}

}

**class** BankAccountList{

TreeSet<savingaccount>savingAccounts = **new** TreeSet<>(**new** accountComparator());

**public** **boolean** addSavingAccount(savingaccount savingAccount) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter the Account Id: ");

savingAccount.setAcc\_ID(sc.nextInt());

System.***out***.println("Enter the Account holder name: ");

savingAccount.setAccountHolderName(sc.next());

System.***out***.println("Enter the Account Balance: ");

savingAccount.setAcc\_balance(sc.nextDouble());

System.***out***.println("Is Saving Account (Yes/No): ");

String accountType = sc.next();

**if**(accountType.toUpperCase().equals("YES")){

savingAccount.setSalaryAccount(**true**);

}

**else**{

savingAccount.setSalaryAccount(**false**);

}

savingAccounts.add(savingAccount);

System.***out***.println("ACCOUNT REGISTERED SUCCESSFULLY");

**return** **true**;

}

**public** List<Integer>displaySavingAccountIds() {

List<Integer>id = **new** ArrayList<>();

**for**(savingaccount savingaccount: savingAccounts){

id.add(savingaccount.getAcc\_ID());

}

**return** id;

}

}

**public** **class** Assignment5Q9 {

**public** **static** **void** main(String[] args) {

BankAccountList bankAccountList = **new** BankAccountList();

savingaccount savingaccount = **new** savingaccount();

Scanner sc = **new** Scanner(System.***in***);

**int** choice;

**do**{

System.***out***.println("1. ADD THE SAVING ACCOUNT");

System.***out***.println("2. DEPOSIT MONEY");

System.***out***.println("3. WITHDRAW MONEY");

System.***out***.println("4. DISPLAY THE SAVING ACCOUNT IDs");

System.***out***.println("ENTER YOUR CHOICE");

choice = sc.nextInt();

**switch** (choice){

**case** 1: bankAccountList.addSavingAccount(savingaccount);

**break**;

**case** 2:**double** amount;

System.***out***.println("Enter the amount to be deposit: ");

amount = sc.nextDouble();

savingaccount.Deposit(amount);

**break**;

**case** 3:

**double** amount1;

System.***out***.println("Enter the amount to be withdraw: ");

amount1 = sc.nextDouble();

savingaccount.withDraw(amount1);

**break**;

**case** 4: List<Integer>id = **new** ArrayList<>();

id = bankAccountList.displaySavingAccountIds();

System.***out***.println("Account Id: ");

**for**(Integer i: id){

System.***out***.println(i);

}

**break**;

**case** 5: System.*exit*(0);

**default**:

System.***out***.println("PLEASE ENTER THE VALID OPTION!");

}

}**while** (choice!=5);

}

}

**Output:**

