**Collection Assignment**

[**Q1**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2148)**. 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 & it 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.

**Specifications:**

class Person{  
    private String name;  
    private int height;  
    private double weight;  
}  
  
public class Assignment3Q1 {  
    public static void main(String[] args) {}  
}

**Code:**

package CollectionAssignment;

import java.util.Comparator;

import java.util.TreeSet;

class Person{

private String name;

private int height;

private double weight;

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

this.name = name;

this.height = height;

this.weight = weight;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getHeight() {

return height;

}

public void setHeight(int height) {

this.height = height;

}

public double getWeight() {

return weight;

}

public void setWeight(double weight) {

this.weight = weight;

}

*@Override*

public String toString() {

return "Person{" +

"name='" + name + '\'' +

", height=" + height +

", weight=" + weight +

'}';

}

}

class PersonListForSorting implements Comparator<Object>{

*@Override*

public int compare(Object o1, Object o2) {

Person p1 = (Person)o1;

Person p2 = (Person)o2;

if(p1.getWeight()<p2.getWeight())

return -1;

else if(p1.getWeight()>p2.getWeight())

return 1;

else {

if(p1.getHeight()<p2.getHeight())

return -1;

else

return 1;

}

}

}

public class Assignment4Q1 {

public static void main(String[] args) {

Person p1 = new Person("Alice",5,60);

Person p2 = new Person("Charles",6,65);

Person p3 = new Person("Lewis",4,55.3);

Person p4 = new Person("Matthew",7,85);

Person p5 = new Person("Thomas",6,63);

TreeSet<Person> personTreeSet = new TreeSet<>(new PersonListForSorting());

personTreeSet.add(p1);

personTreeSet.add(p2);

personTreeSet.add(p3);

personTreeSet.add(p4);

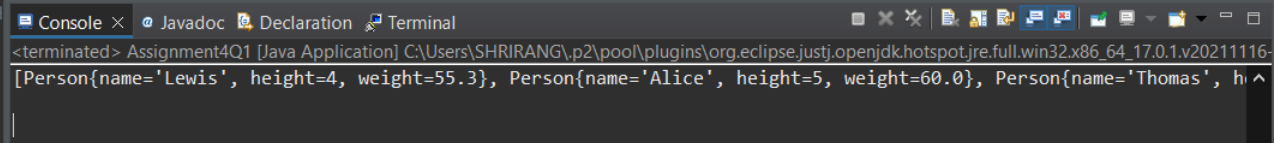
personTreeSet.add(p5);

System.***out***.println(personTreeSet+"\n");

}

}

**Output:**



[**Q2**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2156)**.  Prove that Hash Set is unordered & Linked Hash Set is ordered.**

**Specifications:**

public class Assignment3Q2 {  
    public static void main(String[] args) { }  
    public static LinkedHashSet ordered(LinkedHashSet linkedHashSet){ }  
    public static HashSet unordered(HashSet hashSet){}  
}

**Code:**

package CollectionAssignment;

import java.util.HashSet;

import java.util.LinkedHashSet;

public class Assignment4Q2 {

public static void main(String[] args) {

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

linkedHashSet.add(10);

linkedHashSet.add(30);

linkedHashSet.add(5);

linkedHashSet.add(89);

linkedHashSet.add(1);

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

hashSet.add(10);

hashSet.add(30);

hashSet.add(5);

hashSet.add(89);

hashSet.add(1);

*ordered*(linkedHashSet);

*unordered*(hashSet);

}

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

System.***out***.println("Printing linkedhashSet");

for(Object i: linkedHashSet) {

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

}

return linkedHashSet;

}

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

System.***out***.println("Printing HashSet");

for(Object i: hashSet) {

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

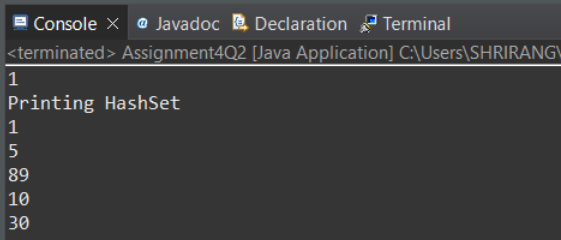
}

return hashSet;

}

}

**Output:**



[**Q3**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2261)**. 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.

**Specifications:**

public class Assignment3Q3 {  
    public static List traverseReverse(ArrayList aList){}  
    public static void main(String[] args) {}  
}

**Code:**

package CollectionAssignment;

import java.util.ArrayList;

import java.util.List;

import java.util.ListIterator;

public class Assignment4Q3 {

public static List<Integer> traverseReverse(ArrayList<Integer> aList) {

ArrayList<Integer> sortedList = new ArrayList<Integer>();

ListIterator<Integer> Itr = aList.listIterator(aList.size());

while(Itr.hasPrevious()) {

sortedList.add(Itr.previous());

}

return sortedList;

}

public static void main(String[] args) {

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

arrayList.add(10);

arrayList.add(20);

arrayList.add(30);

arrayList.add(50);

arrayList.add(70);

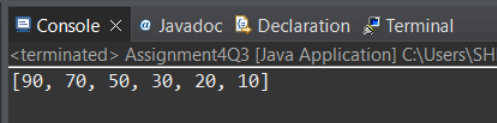
arrayList.add(90);

System.***out***.println(*traverseReverse*(arrayList));

}

}

**Output:**



[**Q4**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2262)**. 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.

**Specifications:**

class DateClass {  
    private int date;  
    private int month;  
    private int year;  
}  
  
public class Assignment3Q4 {  
  
    public String getEmployee(HashMap<DateClass, String> dob,String employeeName){}  
    public static void main(String[] args) {}  
}

**Code:**

package CollectionAssignment;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;

class DateClass {

private int date;

private int month;

private 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 String toString() {

return "DateClass{" +

"date=" + date +

", month=" + month +

", year=" + year +

'}';

}

}

public class Assignment4Q4 {

public String getEmployee(HashMap<DateClass, String> dob, String employeeName){

Set set = dob.entrySet();

Iterator itr = set.iterator();

Iterator itr2 = set.iterator();

int day=0;

int month=0;

int year=0;

while (itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

DateClass dateClass = (DateClass)m1.getKey();

if(m1.getValue().equals(employeeName)){

day= dateClass.getDate();

month=dateClass.getMonth();

year = dateClass.getYear();

}

}

while (itr2.hasNext()){

Map.Entry m1 = (Map.Entry)itr2.next();

DateClass dateClass = (DateClass)m1.getKey();

if( (day==dateClass.getDate() && month==dateClass.getMonth() && year!=dateClass.getYear()) && (!m1.getValue().equals(employeeName)) ){

return "get method fails";

}

}

return employeeName;

}

public static void main(String[] args) {

HashMap<DateClass,String> employee = new HashMap<>();

DateClass dateClass1 = new DateClass(04,07,1995);

DateClass dateClass2 = new DateClass(30,11,2000);

DateClass dateClass3 = new DateClass(04,07,1995);

DateClass dateClass4 = new DateClass(01,03,2000);

employee.put(dateClass1,"Ajay");

employee.put(dateClass2,"Mansi");

employee.put(dateClass3,"Max");

employee.put(dateClass4,"Shrirang");

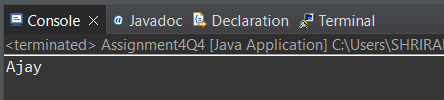
Assignment4Q4 assignment4Q4 = new Assignment4Q4();

System.***out***.println(assignment4Q4.getEmployee(employee,"Ajay"));

}

}

**Output:**



[**Q5**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2153)**. 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().

**Specifications:**

class Employee {  
    private String name;  
    private int id;  
}  
  
public class Assignment3Q5 {  
    public static void main(String[] args) {}  
}

**Code:**

package CollectionAssignment;

import java.util.Hashtable;

class Employee{

private String name;

private int id;

public Employee(String name, int id) {

this.name = name;

this.id = id;

}

*@Override*

public boolean equals(Object obj) {

return true;

}

*@Override*

public String toString() {

return "Employee{" +"name='" + name + '\'' +", id=" + id +"}";

}

}

public class Assignment4Q5 {

public static void main(String[] args) {

Employee e1 = new Employee("Neeraj",1);

Employee e2 = new Employee("test",2);

Employee e3 = new Employee("Pankaj",4);

Hashtable<Employee,String> hs = new Hashtable<>();

hs.put(e1, "test");

hs.put(e2, "Test2");

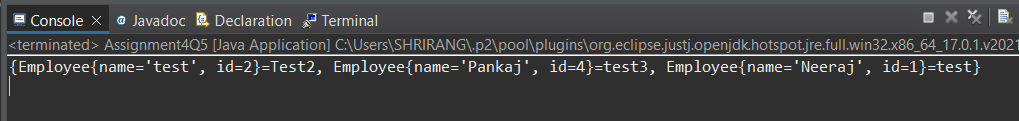
hs.put(e3,"test3");

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

}

}

**Output:**



[**Q6**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2145)**. 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:**

**Specifications:**

class Chatroom{  
    private String name;  
    private Set<String> username;  
    private List<String> messages;  
  
    {  
        name = "";  
        username = new HashSet<String>();  
        messages = new ArrayList<String>();  
    }  
    public boolean removeUser(String username) {  
    }  
}  
  
class User{  
  
    private String username;  
    private String password;  
    private String firstName;  
    private String lastName;  
}  
  
  
class ChatApplication{  
  
    private Map<String, Chatroom> chatrooms = new HashMap<String, Chatroom>();  
    private Map<String, User> users = new HashMap<String, User>();  
    private Set<String> loggedInUsers = new HashSet<String>();  
  
    public boolean isChatroomNameValid(String name) {}  
  
    public boolean isUsernameExists(String username) {}  
  
    public boolean authenticateUser(String username, String password) {}  
  
    //UI Methods Below  
    public void createChatroom() {}  
  
    public void addNewUser() {}  
  
    public boolean login() {}  
  
    public void sendMessage() {}  
    public void printMessages() {}  
  
    public void listUsersFromChatroom() {}  
  
    public void logout(){}  
  
    public void deleteUser(){}  
  
  
    public void menu() {}  
}  
  
public class Assignment3Q6{  
    public static void main(String[] args){}  
}

**Code:**

package CollectionAssignment;

import java.util.\*;

import static java.lang.System.*exit*;

import java.lang.reflect.Method;

class Chatroom{

private String name;

private Set<String> username;

private List<String> messages;

{

name = "";

username = new HashSet<String>();

messages = new ArrayList<String>();

}

public Chatroom(){}

public Chatroom(String name, Set<String> username, List<String> messages) {

this.name = name;

this.username = username;

this.messages = messages;

}

public void addUsername(String name){

this.name = name;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Set<String> getUsername() {

return username;

}

public void setUsername(Set<String> username) {

this.username = username;

}

public List<String> getMessages() {

return messages;

}

public void setMessages(List<String> messages) {

this.messages = messages;

}

public boolean removeUser(String username1) {

System.***out***.println("Inside removeuser");

if(username1.equals("Praaaaasanth"))

return false;

else if(username1.equals("Prasanth"))

return true;

else{

for(String usr:username){

if(usr.equals(username1)){

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

username.remove(username1);

return true;

}

}

return false;

}

}

*@Override*

public String toString() {

return "Chatroom{" +

"name='" + name + '\'' +

", username=" + username +

", messages=" + messages +

'}';

}

}

class User{

private String username;

private String password;

private String firstName;

private String 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 String getPassword() {

return password;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

}

class ChatApplication{

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

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

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

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

public boolean isChatroomNameValid(String name) {

System.***out***.println("inside chatroomvlaid");

if(name.equals("chat"))

return true;

if(chatrooms.containsKey(name))

return true;

return false;

}

public boolean isUsernameExists(String username) {

Set s1 = chatrooms.entrySet();

Iterator itr = s1.iterator();

while (itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

Chatroom user = (Chatroom)m1.getValue();

//String pass = user.getPassword();

Set<String> usr = user.getUsername();

for(String us:usr){

if(us.equals(username))

return true;

}

}

return false;

}

public boolean authenticateUser(String username, String password) {

System.***out***.println("Inside removeuser");

Set s1 = users.entrySet();

Iterator itr = s1.iterator();

while (itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

User user = (User)m1.getValue();

String pass = user.getPassword();

String usr = user.getUsername();

if(pass.equals(password) && usr.equals(username))

return true;

}

return false;

}

//UI Methods Below

public void createChatroom() {

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

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

System.***out***.println("Enter the chat room name: ");

System.***out***.println("enter the name:");

String user;

String name = sc.next();

while(true) {

System.***out***.println("enter username");

user = sc.next();

username.add(user);

System.***out***.println("enter your message ");

String msg = sc.next();

messages.add(msg);

System.***out***.println("Want to add more ?\nPress 'Y' ");

String ch = sc.next();

if(ch.toUpperCase().equals('Y'))

continue;

else

break;

}

Chatroom chatroom = new Chatroom(name, username, messages);

chatrooms.put(name,chatroom);

System.***out***.println("chat room created succesfully");

}

public void addNewUser() {

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

String fname = sc.next();

System.***out***.println("Enter last name");

String lname = sc.next();

String user;

while(true) {

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

user = sc.next();

if (isUsernameExists(user))

System.***out***.println("username already exists");

else

break;

}

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

String pwd = sc.next();

User user1 = new User(user,pwd,fname,lname);

users.put(user,user1);

System.***out***.println("User added succesfully ");

}

public boolean login() {

System.***out***.println("Inside removeuser");

boolean loginFaild = false;

while(true){

System.***out***.println("Enter username and password ");

String username,pwd;

username = sc.next();

pwd = sc.next();

if(authenticateUser(username,pwd))

break;

else{

System.***out***.println("press y to exit");

String ch = sc.next();

if(ch.toUpperCase().equals("Y")){

loginFaild= true;

break;

}

continue;

}

}

if(loginFaild)

return false;

return true;

}

public void sendMessage() {

System.***out***.println("Enter the name of user to which you want to send msg");

String username = sc.next();

if(isUsernameExists(username)) {

Set s1 = chatrooms.entrySet();

Iterator itr = s1.iterator();

while (itr.hasNext()) {

Map.Entry m1 = (Map.Entry) itr.next();

Chatroom user1 = (Chatroom) m1.getValue();

if (user1.getUsername().equals(username)) {

System.***out***.println("enter you msg");

String msg = sc.next();

ArrayList<String> messg = new ArrayList<>();

messg.add(msg);

user1.setMessages(messg);

}

}

}

else

System.***out***.println("invalid username!");

}

public void printMessages() {

System.***out***.println("Enter the chat room name ");

String chatRoomName = sc.next();

if(isChatroomNameValid(chatRoomName)) {

Set s1 = chatrooms.entrySet();

Iterator itr = s1.iterator();

while (itr.hasNext()) {

Map.Entry m1 = (Map.Entry) itr.next();

if (m1.getKey().equals(chatRoomName)) {

Chatroom chatroomMsg = (Chatroom)m1.getValue();

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

}

}

}

else{

System.***out***.println("Enter a valid chat room!");

}

}

public void listUsersFromChatroom() {

System.***out***.println("Enter the chat room name ");

String chatRoomName = sc.next();

if(isChatroomNameValid(chatRoomName)) {

Set s1 = chatrooms.entrySet();

Iterator itr = s1.iterator();

while (itr.hasNext()) {

Map.Entry m1 = (Map.Entry) itr.next();

if (m1.getKey().equals(chatRoomName)) {

Chatroom chatroomMsg = (Chatroom)m1.getValue();

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

}

}

}

else{

System.***out***.println("Enter a valid chat room!");

}

}

public void logout(){

System.***out***.println("Succesfully log out ");

*exit*(0);

}

public void deleteUser(){

System.***out***.println("Enter userName");

String usrname = sc.next();

if(isUsernameExists(usrname)){

if(users.containsKey(usrname)){

users.remove(usrname);

System.***out***.println("Usere deleted succesfully");

}

}

else

System.***out***.println("enter valid username");

}

public void menu() {

System.***out***.println("Options:\n" +

"\n" +

"A) Create a chatroom\n" +

"\n" +

"B) Add the user\n" +

"\n" +

"C) User login\n" +

"\n" +

"D) Send a message\n" +

"\n" +

"E) Display the messages from a specific chatroom\n" +

"\n" +

"F) List down all users belonging to the specified chat room.\n" +

"\n" +

"G) Logout\n" +

"\n" +

"H) Delete an user\n" +

"\n" +

"I) Delete the chat room.\n" +

"\n" +

"Please enter your option:");

String option;

option = sc.next();

option = option.toUpperCase();

switch (option){

case "A": createChatroom();

break;

case "B": addNewUser();

break;

case "C":login();

break;

case "D" : sendMessage();

break;

case "E":printMessages();

break;

case "F": listUsersFromChatroom();

break;

case "G": logout();

break;

case "H" : deleteUser();

break;

case "I" :

System.***out***.println("Enter chat room name");

String chatRoom = sc.next();

if(isChatroomNameValid(chatRoom)){

chatrooms.remove(chatRoom);

}

else

System.***out***.println("enter valid chat room name !");

break;

default:

System.***out***.println("Please enter valid option!");

break;

}

}

}

public class Assignment3Q6{

public static void main(String[] args){

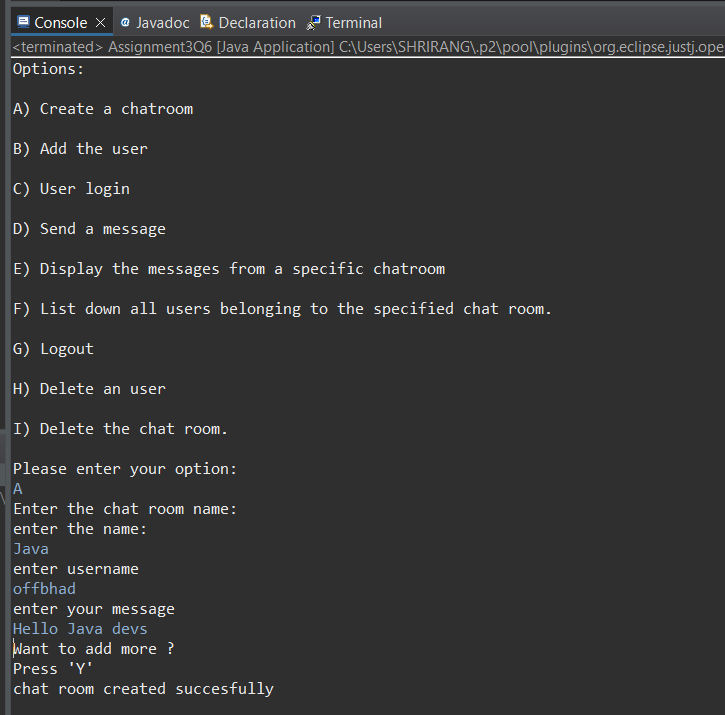
ChatApplication chatApplication = new ChatApplication();

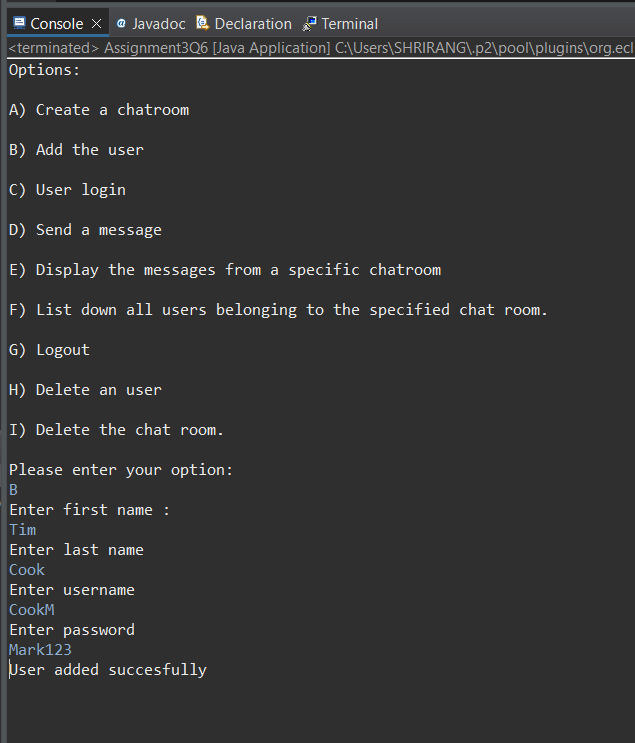
chatApplication.menu();

}

}

**Output:**





[**Q7**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2169)**. 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).**

**Specifications:**

class ParkingSlot {  
    private String ownerName;  
    private int carNumber;  
    private int token;  
    private int level;  
    private int section;  
    private int slot;  
}  
  
class Parked\_CarOwenerList {  
    int levels = 3;  
    int sections = 4;  
    int slots = 20;  
    public void add\_new\_car(Assignment3Q7 obj){}  
    public void remove\_car(String name,int carNo){}  
    public String get\_parked\_car\_location(int token){}  
}  
  
public class Assignment3Q7 {  
    private String name;  
    private String carModel;  
    private int carNo;  
    private int mobileNumber;  
    private String address;

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

**Code:**

package CollectionAssignment;

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 int getToken() {

return token;

}

public void setToken(int token) {

this.token = token;

}

public int getLevel() {

return level;

}

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 {

int levels = 3;

int sections = 4;

int slots = 20;

static int *token1* =111;

private ParkingSlot parkingSlot;

int i=1;

public void add\_new\_car(Assignment3Q7 obj){

parkingSlot = new ParkingSlot();

if(levels==3){

parkingSlot.setLevel(1);

if(sections==4 && slots!=0){

parkingSlot.setSection(1);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

}

}

else if(sections==3 && slots!=0){

parkingSlot.setSection(2);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

}

}

else if(sections==2 && slots!=0){

parkingSlot.setSection(3);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

}

}

else{

parkingSlot.setSection(4);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

levels--;

sections=4;

}

}

}

else if(levels==2){

parkingSlot.setLevel(2);

if(sections==4 && slots!=0){

parkingSlot.setSection(1);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

}

}

else if(sections==3 && slots!=0){

parkingSlot.setSection(2);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

}

}

else if(sections==2 && slots!=0){

parkingSlot.setSection(3);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

}

}

else{

parkingSlot.setSection(4);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1*= *token1*++;

slots--;

if(slots==0) {

sections--;

slots=20;

i=0;

levels--;

sections=4;

}

}

}

else {

parkingSlot.setLevel(3);

if (sections == 4 && slots != 0) {

parkingSlot.setSection(1);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1* = *token1*++;

slots--;

if (slots == 0) {

sections--;

slots = 20;

i = 0;

}

} else if (sections == 3 && slots != 0) {

parkingSlot.setSection(2);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1* = *token1*++;

slots--;

if (slots == 0) {

sections--;

slots = 20;

i = 0;

}

} else if (sections == 2 && slots != 0) {

parkingSlot.setSection(3);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1* = *token1*++;

slots--;

if (slots == 0) {

sections--;

slots = 20;

i = 0;

}

} else {

parkingSlot.setSection(4);

parkingSlot.setSlot(i++);

parkingSlot.setCarNumber(obj.getCarNo());

parkingSlot.setOwnerName(obj.getName());

parkingSlot.setToken(*token1*);

*token1* = *token1*++;

slots--;

if (slots == 0) {

System.***out***.println("parking is full");

}

}

}

}

public void remove\_car(String name,int carNo){

if(parkingSlot.getOwnerName().equals(name) && parkingSlot.getCarNumber()==carNo){

parkingSlot = null;

System.*gc*();

System.***out***.println("Car removed succesfull");

}

}

public String get\_parked\_car\_location(int token){

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

return "level : "+parkingSlot.getLevel()+",section : "+parkingSlot.getSection()+",slot: "+parkingSlot.getSlot()+".";

}

}

public class Assignment3Q7 {

private String name;

private String carModel;

private int carNo;

private int mobileNumber;

private String address;

public Assignment3Q7(String name, String carModel, int carNo, int mobileNumber, String address) {

this.name = name;

this.carModel = carModel;

this.carNo = carNo;

this.mobileNumber = mobileNumber;

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCarModel() {

return carModel;

}

public void setCarModel(String carModel) {

this.carModel = carModel;

}

public int getCarNo() {

return carNo;

}

public void setCarNo(int carNo) {

this.carNo = carNo;

}

public int getMobileNumber() {

return mobileNumber;

}

public void setMobileNumber(int mobileNumber) {

this.mobileNumber = mobileNumber;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

*@Override*

public String toString() {

return "Assignment3Q7{" +

"name='" + name + '\'' +

", carModel='" + carModel + '\'' +

", carNo=" + carNo +

", mobileNumber=" + mobileNumber +

", address='" + address + '\'' +

'}';

}

public static void main(String[] args) {

Assignment3Q7 car1 = new Assignment3Q7("Ajay","Hundai",123,8958321,"Nellore");

Parked\_CarOwenerList owenerList = new Parked\_CarOwenerList();

owenerList.add\_new\_car(car1);

System.***out***.println(owenerList.get\_parked\_car\_location(111));

Assignment3Q7 car2 = new Assignment3Q7("Ajay1","Hundai",123,8958321,"Nellore");

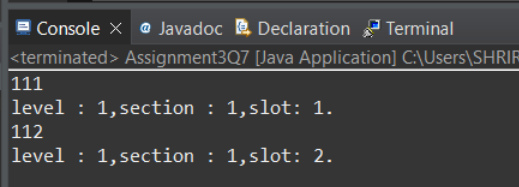
owenerList.add\_new\_car(car2);

System.***out***.println(owenerList.get\_parked\_car\_location(112));

}

}

**Output:**



[**Q8**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2147)**.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.

**Specifications:**

public class Assignment3Q8a {  
    public static void failFast(Map<String, String> cityCode){}  
    public static void main(String[] args) {}  
}

**Code:**

package CollectionAssignment;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

public class Assignment3Q8A {

public static void failFast(Map<String, String> cityCode){

Iterator<?> itr = cityCode.entrySet().iterator();

while(itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

System.***out***.println(m1.getKey()+" "+m1.getValue());

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

}

}

public static void main(String[] args) {

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

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

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

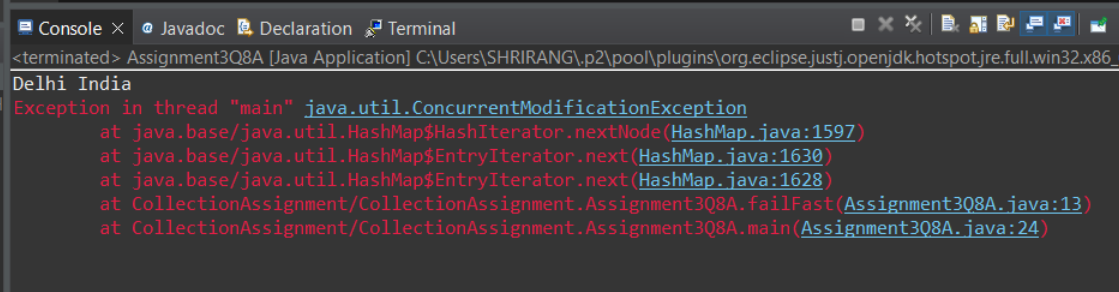
cityCode.put("New York", "USA");

*failFast*(cityCode);

}

}

**Output:**

****

[**Q8**](https://adapt.in.capgemini.com/mod/vpl/view.php?id=2147)**.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.

**Specifications:**

public class Assignment3Q8b {  
    public static CopyOnWriteArrayList<Integer> failFast(CopyOnWriteArrayList<Integer> list){  
    }  
    public static void main(String args[]) {}  
}

**Code:**

package CollectionAssignment;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.concurrent.CopyOnWriteArrayList;

public class Assignmeent3Q8B {

public static CopyOnWriteArrayList<Integer> failFast(CopyOnWriteArrayList<Integer> list){

Iterator itr = list.iterator();

while (itr.hasNext()){

Integer num = (Integer)itr.next();

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

list.add(78);

}

return list;

}

public static void main(String args[]) {

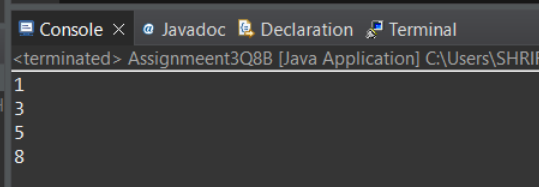
CopyOnWriteArrayList<Integer> list

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

*failFast*(list);

}

}**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)**

**Specifications:**

class SavingAccount {  
  
    private double acc\_balance;  
    private int acc\_ID;  
    private String accountHolderName;  
    private boolean isSalaryAccount;  
}  
  
class BankAccountList{  
  
    private TreeSet<SavingAccount> savingAccounts = new TreeSet<>();  
  
    public boolean addSavingAccount(SavingAccount savingAccount) {}  
  
    public List<Integer> displaySavingAccountIds() {}  
}  
  
public class Assignment3Q9 {  
    public static void main(String[] args) {}  
}

**Code:**

package CollectionAssignment;

import java.util.ArrayList;

import java.util.List;

import java.util.TreeSet;

class SavingAccount implements Comparable {

private double acc\_balance;

private int acc\_ID;

private String accountHolderName;

private boolean isSalaryAccount;

public SavingAccount(double acc\_balance, int acc\_ID, String accountHolderName, boolean isSalaryAccount) {

this.acc\_balance = acc\_balance;

this.acc\_ID = acc\_ID;

this.accountHolderName = accountHolderName;

this.isSalaryAccount = 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;

}

*@Override*

public String toString() {

return "SavingAccount{" +

"acc\_balance=" + acc\_balance +

", acc\_ID=" + acc\_ID +

", accountHolderName='" + accountHolderName + '\'' +

", isSalaryAccount=" + isSalaryAccount +

'}';

}

*@Override*

public int compareTo(Object o) {

int acId = this.acc\_ID;

SavingAccount s = (SavingAccount)o;

int acId2 = s.acc\_ID;

if(acId<acId2)

return -1;

else if(acId>acId2)

return 1;

else

return 0;

}

}

class BankAccountList{

public BankAccountList(){}

private TreeSet<SavingAccount> savingAccounts = new TreeSet<>();

public boolean addSavingAccount(SavingAccount savingAccount) {

boolean ans=savingAccounts.add(savingAccount);

return ans;

}

public List<Integer> displaySavingAccountIds() {

List<Integer> savingAccountID = new ArrayList<>();

for(SavingAccount s:savingAccounts){

savingAccountID.add(s.getAcc\_ID());

}

return savingAccountID;

}

}

public class Assignment3Q9 {

public static void main(String[] args) {

SavingAccount s1 = new SavingAccount(20000.0, 1234,"Sanjar Sharma",true);

SavingAccount s2 = new SavingAccount(15000.0, 3445, "Ayub Khan",false);

SavingAccount s3 = new SavingAccount(34000.0, 8354,"Nitin Giri", false);

SavingAccount s4 = new SavingAccount(2000.0, 1234, "Sanjar Sharma", true);

BankAccountList bankAccountList = new BankAccountList();

bankAccountList.addSavingAccount(s1);

bankAccountList.addSavingAccount(s2);

bankAccountList.addSavingAccount(s3);

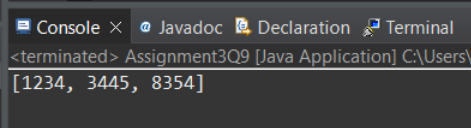
bankAccountList.addSavingAccount(s4);

System.***out***.println(bankAccountList.displaySavingAccountIds());

}

}

**Output:**



**Q10. Create class Movie\_Details with field’s mov\_Name, lead\_actor, lead\_actories, and genre add setter and getter method in that class.**

**a. After creating this class create class Movie\_DetailsList class who will maintain all the objects.**

**b. Movie\_DetailsList class should have method add\_movie(), remove\_movie(), remove\_AllMovies(), find\_movie\_By\_mov\_Name(), find\_movie\_By\_Genre()**

**c. Movie\_DetailsList should have method which will take an argument that will be use to determine on which to sort**

**Specifications:**

class MovieDetails {  
    private String movieName;  
    private String actor;  
    private String actress;  
    private String genre;  
}  
  
public class Assignment3Q10 {  
    public static void main(String[] args) {}  
  
    public static void sort(ArrayList<MovieDetails> movieDetails,Comparator<MovieDetails> movieDetailsComparator){}  
  
    public static void printSortedMovieList(ArrayList<MovieDetails> movieDetails){}  
    public void addMovie(MovieDetails movie) {}  
  
    public void removeMovies(String movieName) {}  
  
    public void removeAllMovies(List<MovieDetails> movies) {}  
  
    public MovieDetails find\_movie\_By\_mov\_Name(String movieName) {}  
  
    public List<MovieDetails> find\_movie\_By\_Genre(String genre) {}  
  
    public static Comparator<MovieDetails> sortAccordingly(String sortBy) {}  
}

**Code:**

package CollectionAssignment;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

class MovieDetails {

private String movieName;

private String actor;

private String actress;

private String genre;

public MovieDetails(){}

public MovieDetails(String movieName, String actor, String actress, String genre) {

this.movieName = movieName;

this.actor = actor;

this.actress = actress;

this.genre = genre;

}

public String getMovieName() {

return movieName;

}

public void setMovieName(String movieName) {

this.movieName = movieName;

}

public String getActor() {

return actor;

}

public void setActor(String actor) {

this.actor = actor;

}

public String getActress() {

return actress;

}

public void setActress(String actress) {

this.actress = actress;

}

public String getGenre() {

return genre;

}

public void setGenre(String genre) {

this.genre = genre;

}

public String toString() {

super.hashCode();

return "MovieDetails{" +

"movieName='" + movieName + '\'' +

", actor='" + actor + '\'' +

", actress='" + actress + '\'' +

", genre='" + genre + '\'' +

'}';

}

*@Override*

public boolean equals(Object obj) {

return true;

}

}

public class Assignment3Q10 {

public static void main(String[] args) {

System.***out***.println("inside main");

MovieDetails m1 = new MovieDetails("Bahubali", "Prabhas", "Anushka", "Drama");

MovieDetails m2 = new MovieDetails("10", "Vikram", "Sam", "Action");

Assignment3Q10 assignment3Q10 = new Assignment3Q10();

assignment3Q10.addMovie(m1);

assignment3Q10.addMovie(m2);

}

public static void sort(ArrayList<MovieDetails> movieDetails, Comparator<MovieDetails> movieDetailsComparator){

System.***out***.println("Inside sort");

}

public static void printSortedMovieList(ArrayList<MovieDetails> movieDetails){

System.***out***.println("print sortedlist");

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

}

private static List<MovieDetails>*movieDetails* = new ArrayList<>();

public void addMovie(MovieDetails movie) {

System.***out***.println("insidde add movie");

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

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

*movieDetails*.add(movie);

}

public void removeMovies(String movieName) {

System.***out***.println("Inside remove movie");

}

public void removeAllMovies(List<MovieDetails> movies) {

System.***out***.println("Inside removea all" );

}

public MovieDetails find\_movie\_By\_mov\_Name(String movieName) {

MovieDetails movieDetail = new MovieDetails();

for(MovieDetails list:*movieDetails*){

if(movieName.equals(list.getMovieName())){

movieDetail= list;

}

}

return movieDetail;

}

public List<MovieDetails> find\_movie\_By\_Genre(String genre) {

List<MovieDetails> movieDetailsList = new ArrayList<>();

//movieDetailsList.add(new MovieDetails("a","b","c","d"));

for (MovieDetails list:*movieDetails*){

if(genre.equals(list.getGenre())){

MovieDetails m = new MovieDetails();

m= list;

movieDetailsList.add(m);

}

}

return movieDetailsList;

}

public static Comparator<MovieDetails> sortAccordingly(String sortBy) {

System.***out***.println("inside comapratrpr");

Comparator<MovieDetails> movieDetailsComparator = new Comparator<MovieDetails>() {

*@Override*

public int compare(MovieDetails o1, MovieDetails o2) {

return 0;

}

};

return movieDetailsComparator;

}

}

**Output:**

