**ASSESSMENT(ECN-4138,Kajal)**

**Githublink:** https://github.com/kajal233/NewAssignment.git

**1 StudentMarks**

**package** com.generics;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** StudentMarks {

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

**int** totalNumberOfStudents;

**int** i;

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

System.***out***.println("Enter number of Students ");

totalNumberOfStudents=studentInput.nextInt();

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

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

**for**(i=0;i<totalNumberOfStudents;i++) {

marksList.add(studentInput.nextInt());//Storing marks to array list

}

// highest mark of the student

System.***out***.println("Highest marks : "+Collections.*max*(marksList));

//average marks of the student

**float** average=0;

**for**(i=0;i<totalNumberOfStudents;i++) {

average=average+marksList.get(i);

}

System.***out***.println("Average Marks : "+(average/totalNumberOfStudents));

//Displaying the marks of student

**for**(i=0;i<totalNumberOfStudents;i++) {

System.***out***.print((i+1)+"-"+marksList.get(i)+" ");

}

//third student mark

System.***out***.println("\n3rd Student marks : "+marksList.get(2));

//marks in sorted order

System.***out***.print("Sorted : ");

Collections.*sort*(marksList);

Iterator<Integer> iterator=marksList.iterator();

**int** flag=1;

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

System.***out***.print(flag+"-"+iterator.next()+" ");

flag=flag+1;

}

}

}

================================================================================== **2 – Cricket Score Analyzer**

**package** com.generics;

**import** java.util.Collections;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Scanner;

**class** ScoreAnalyzer{

**private** **int** runsData;

**public** ScoreAnalyzer() {

// **TODO** Auto-generated constructor stub

**super**();

}

**public** ScoreAnalyzer(**int** runsData) {

**super**();

**this**.runsData=runsData;

}

//Storing runs in list

List<Integer> runlist=**new** LinkedList<Integer>();

**public** **void** addRunsToList(**int** runsData){

runlist.add(runsData);

}

//Calculate run rate

**public** **float** calcRunRate() {

**float** runRate=0;

**for**(**int** i=0;i<5;i++) {

runRate=runRate+runlist.get(i);

}

**return** (runRate/50);

}

//Lowest run scored by player

**public** **int** lowestRunsScored() {

**int** lowestrun=Collections.*min*(runlist);

**return** lowestrun;

}

//Display all runs

**public** **void** displayRuns() {

**int** i=1;

**for**(Integer run:runlist){

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

i=i+1;

}

}

//Getter and Setter

**public** **int** getRunsData() {

**return** runsData;

}

**public** **void** setRunsData(**int** runsData) {

**this**.runsData = runsData;

}

}

**public** **class** TestScoreAnalyzer {

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

**int** i;

ScoreAnalyzer analyzer=**new** ScoreAnalyzer();

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

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

**for**(i=0;i<5;i++) {

**int** runInputByPlayer=input.nextInt();

analyzer.addRunsToList(runInputByPlayer);

}

//For displaying record

analyzer.displayRuns();

System.***out***.println("\nRunrate : "+analyzer.calcRunRate());

System.***out***.println("Lowest Runs : "+(analyzer.lowestRunsScored()));

System.***out***.println("Count of player who batted : "+analyzer.runlist.size());

}

}

**3 – ScoreCard**

**package** com.generics;

**import** java.util.Collections;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Scanner;

**public** **class** ScoreCard {

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

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

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

System.***out***.println("Enter runs scored");

**for**(**int** i=0;i<5;i++) {

String nameRun[]=input.next().split("-");

String name=nameRun[0];

**int** run=Integer.*parseInt*(nameRun[1]);

batsmanscore.put(name, run);

}

//Name of players who have batted

System.***out***.println("Players who batted ");

Iterator<String> iterator=batsmanscore.keySet().iterator();

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

String name=iterator.next();

Integer run=batsmanscore.get(name);

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

}

**int** runByDhoni=0;

**int** totalruns=0;

System.***out***.println("Scores by Players ");

Iterator<String> iterator1=batsmanscore.keySet().iterator();

**while**(iterator1.hasNext()) {

String name=iterator1.next();

Integer run=batsmanscore.get(name);

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

totalruns=totalruns+run;

**if**(name.equalsIgnoreCase("Dhoni")) {

runByDhoni=run;

}

}

System.***out***.println("Total Score : "+totalruns);

**int** maxRun=Collections.*max*(batsmanscore.values());

Iterator<String> iterator3=batsmanscore.keySet().iterator();

**while**(iterator3.hasNext()) {

String name=iterator3.next();

Integer run=batsmanscore.get(name);

**if**(run==maxRun) {

System.***out***.println("Name of Highest Scorer : "+name);

}

}

//Dhoni run

System.***out***.println("Runs Scored by Dhoni : "+runByDhoni);

}

}

**4 – Patient List - Comparable and Comparator**

**package** com.generics;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.TreeSet;

**class** Patient **implements** Comparable<Patient>{

**private** **int** patientId;

**private** String name;

**private** **int** age;

//Parameterized constructor

**public** Patient(**int** patientId,String name,**int** age) {

**super**();

**this**.patientId=patientId;

**this**.name=name;

**this**.age=age;

}

//Getters / Setters

**public** **int** getPatientId() {

**return** patientId;

}

**public** **void** setPatientId(**int** patientId) {

**this**.patientId = patientId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

//TreeSet

@Override

**public** **int** compareTo(Patient patient) {

// **TODO** Auto-generated method stub

**if**(**this**.patientId==patient.getPatientId()) {

**return** 0;

}

**else** **if**(**this**.patientId>patient.getPatientId()) {

**return** 1;

}

**else** {

**return** -1;

}

}

}

**public** **class** TestPatientList {

**public** **static** **int** getPatientAge(String name,TreeSet listofpatient) {

**int** age=0;

Iterator<Patient> itr=listofpatient.iterator();

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

Patient patient=itr.next();

**if**(patient.getName()==name) {

age=patient.getAge();

}

}

**return** age;

}

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

//List<String> li=new ArrayList<String>();

List<Patient> patientList=**new** ArrayList<Patient>();

patientList.add(**new** Patient(1, "Ram", 29));

patientList.add(**new** Patient(2, "Tom", 18));

patientList.add(**new** Patient(3, "Shyam", 23));

//Storing patient list by name sort

Collections.*sort*(patientList,**new** Comparator<Patient>(){

**public** **int** compare(Patient p1,Patient p2) {

**return** p1.getName().compareTo(p2.getName());

}

});

System.***out***.println("Sorting the patient by there name : ");

**for**(Patient patient:patientList) {

System.***out***.println("Patient id : "+patient.getPatientId()+", Patient name : "+patient.getName()+", Patient age : "+patient.getAge());

}

//Storing patient list by age sort

Collections.*sort*(patientList,**new** Comparator<Patient>(){

**public** **int** compare(Patient p1,Patient p2) {

**return** Integer.*valueOf*(p1.getAge()).compareTo(p2.getAge());

}

});

System.***out***.println("\nSorting the patient by there age : ");

Iterator<Patient> iterator= patientList.iterator();

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

Patient p=iterator.next();

System.***out***.println("Patient id : "+p.getPatientId()+", Patient name : "+p.getName()+", Patient age : "+p.getAge());

}

//Adding all object to tree set

TreeSet<Patient> listofpatientintreeset=**new** TreeSet<Patient>(patientList);

//Traversing patient list

System.***out***.println("\nTraversing patient list from tree set");

**for**(Patient patient1:listofpatientintreeset) {

System.***out***.println("Patient id : "+patient1.getPatientId()+", Patient name : "+patient1.getName()+", Patient age : "+patient1.getAge());

}

**int** age=*getPatientAge*("Ram",listofpatientintreeset);

System.***out***.println("\nAge of the given patient is : "+age);

}

}

========================================================================

**5 – Duplicate Patient - HashCode and Equals**

**package** com.generics;

**import** java.util.LinkedHashSet;

**import** java.util.Set;

**class** Patient{

**private** **int** patientId;

**private** String name;

**private** **int** age;

//Parameterized constructor

**public** Patient(**int** patientId,String name,**int** age) {

**this**.patientId=patientId;

**this**.name=name;

**this**.age=age;

}

//Getters / Setters

**public** **int** getPatientId() {

**return** patientId;

}

**public** **void** setPatientId(**int** patientId) {

**this**.patientId = patientId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**public** **class** TestPatientSet {

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

Set<Patient> patientset=**new** LinkedHashSet<Patient>();

patientset.add(**new** Patient(1, "Ram", 45));

patientset.add(**new** Patient(2, "Tom", 35));

patientset.add(**new** Patient(2, "Tom", 35));

System.***out***.println("Size of set is : "+patientset.size());

**for**(Patient patient:patientset) {

System.***out***.println("Id : "+patient.getPatientId()+" Patient Name : "+patient.getName()+" Patient age : "+patient.getAge());

}

}

}

=========================================================================

**6 – States**

**package** com.generics;

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.LinkedHashSet;

**import** java.util.List;

**import** java.util.Set;

**public** **class** States {

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

FileReader fileReader=**new** FileReader("D:/ExampleDirectory/abc.txt");

BufferedReader bufferedReader=**new** BufferedReader(fileReader);

Set<String> stateSet=**new** LinkedHashSet<String>();

**try** {

String reader=bufferedReader.readLine();

**while**(reader!=**null**) {

stateSet.add(reader);

reader=bufferedReader.readLine();

}

System.***out***.println("Total number of states : "+stateSet.size());

System.***out***.println("Removing state delhi : "+stateSet.remove("Delhi"));

System.***out***.println("After removing the Delhi : "+stateSet);

System.***out***.print("State which starts with k : ");

**for**(String state: stateSet) {

**if**(state.charAt(0)=='K' || state.charAt(0)=='k') {

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

}

}

//Sorting state in another set

List<String> stateList=**new** ArrayList<String>(stateSet);

Collections.*sort*(stateList);

Set<String> statesort=**new** LinkedHashSet<>(stateList);

System.***out***.println("\nStates in sorted order : "+statesort);

}

**finally** {

bufferedReader.close();

}

}

}

======================================================================