import java.util.\*;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

class Personne implements Comparable<Personne>{

private String nom;

private String preNom;

private int age;

Personne(String nom, String prenom, int age){

this.nom = nom;

this.preNom = prenom;

this.age = age;

}

@Override

public boolean equals(Object obj) {

if (obj.getClass()!=this.getClass()) return false;

Personne autre = (Personne)obj;

if(!this.nom.equals(autre.nom)) return false;

if(!this.preNom.equals(autre.preNom)) return false;

if(this.age!=autre.age) return false;

return true;

}

// deux objets identiques DOIVENT avoir le même hashCode()

// il est préférable que deux objets différents aient deux hashCode() différents

@Override

public int hashCode() {

return (nom+preNom+age).hashCode();

}

public String getNomComplet(){

return (nom+" "+preNom)+", "+age+" ans.";

}

@Override

public int compareTo(Personne autre) {

return getNomComplet().compareTo(autre.getNomComplet());

}

@Override

public String toString(){

return getNomComplet();

}

}

public class Main {

public static void demoArrayList(){

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

list.add(2);

list.add(14);

list.add(20);

list.add(27);

list.add(212);

list.add(7640);

System.out.println("parcours 1");

for (int i = 0; i < list.size(); i++) {

System.out.println(list.get(i));

}

list.removeIf( x -> x%2==1 );

System.out.println("parcours 2");

for (int nombre : list) {

System.out.println(nombre);

}

ArrayList<Integer> list2 =

IntStream.range(0,100)

.mapToObj(x->2\*x)

.map(x->x\*x)

.collect(Collectors.toCollection(ArrayList::new));

for (int i = 0; i < list2.size(); i++) {

System.out.println("list2["+i+"] : "+list2.get(i)); //O(1)

}

System.out.println(list2.stream().filter(x->x==2).findFirst());

for (int i = 0; i < list2.size(); i++) {

if (list2.get(i)==2) System.out.println(list.get(i));

}

}

public static void demoLinkedList(){

LinkedList<String> list = new LinkedList<>();

list.add("AAAA");

list.add("BBBB");

list.add("BBBB");

list.add("BONJOUR");

list.add("Salut");

list.add("BBBB");

list.add("BBBB");

list.add("Info4");

/\*

list.addFirst();

list.addLast();

list.removeFirst();

list.removeLast();

\*/

//loza fa O(n^2)

for (int i = 0; i < list.size(); i++) {

System.out.println(list.get(i)); // ! i itérations

}

/// 1 + 2 + 3 + 4 ... + n = n (1 + n) /2 = (n^2)/2 + n/2 = O(n^2)

System.out.println("Suppression de tous les éléments 'TOUT EN MAJUSCULES'. ");

list.removeIf(x->x.toUpperCase()==x); //O(n)

for (String s:list) { //O(n)

System.out.println(s);

}

/// p:=tete;

/// while (p<>nil) do

/// begin

/// writeLn(p^.val);

/// p:=p^.suiv;

/// end;

}

private static void printIntSet(boolean t[]){

boolean first=true;

System.out.print('{');

for (int i = 0; i <t.length ; i++) {

if (t[i]==true){

if (first) {

System.out.print(i);

first=false;

}

else{

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

}

}

}

System.out.print('}');

System.out.println();

}

private static void demoHashSet(){

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

list.add("AAAA"); // O(1) ajout à la fin

list.add("BBBB");

list.add("BBBB");

list.add("BONJOUR");

list.add("Salut");

list.add("BBBB");

list.add("BBBB");

list.add("Info4");

if(list.contains("Salut salut!")) { //O(n) pour ArrayList

System.out.println("la liste contient le mot");

}

else System.out.println("la liste ne contient pas le mot");

boolean t[]=new boolean[100];

for (int i = 0; i <100; i++) {

t[i]=false;

}

for (int i = 0; i <100; i++) {

System.out.println("t["+i+"] = "+t[i]);

}

printIntSet(t);

//t = {} ensemble vide

t[0] = true;

t[5] = true;

t[10] = true;

t[25] = true;

t[7] = true;

t[14] = true;

t[3] = true;

t[24] = true;

//t = {0, 3, 5, 7, 10, 14, 24, 25}

printIntSet(t);

if (t[30] == true){ //contains ? O(1)

System.out.println("t contient 30");

}

HashSet<String> present = new HashSet<>();

present.add("Roots");

present.add("Lyssa");

present.add("Tovo");

present.add("Tsiry");

present.add("Missiah");

present.add("Michael");

present.add("Armelle");

present.add("Doela");

present.add("Tsiry");

System.out.println("Armelle présente : "+present.contains("Armelle")); //O(1)

for (String x:present) { // O(Taille du tableau)

System.out.println(x);

}

System.out.println("Doela : "+"Doela".hashCode()%10);

System.out.println("Tsiry : "+"Tsiry".hashCode()%10);

System.out.println("Armelle : "+"Armelle".hashCode()%10);

System.out.println("Michael : "+"Michael".hashCode()%10);

System.out.println("Missah : "+"Missiah".hashCode()%10);

System.out.println("Tovo : "+"Tovo".hashCode()%10); //collision si tableau de taille 10

System.out.println("Tovo présent : " +present.contains("Tovo")); //O(1)

System.out.println("tsyao : " +present.contains("tsyao")); //O(1)

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

Personne Zo = new Personne("RAJOELISON", "Zo",21);

Personne Anjaratiana = new Personne("RANDRIAMIHANTA", "Anjaratiana",21);

hashSet.add(Zo);

hashSet.add(Anjaratiana);

System.out.println(hashSet.contains(new Personne("RAJOELISON", "Zo",21))); //false ???

System.out.println(Anjaratiana.hashCode());

System.out.println(Zo.hashCode());

System.out.println(new Personne("RANDRIAMIHANTA", "Anjaratiana",21).hashCode());

}

public static void demoHashMap(){

HashMap<String,Integer> map= new HashMap<>(); //C# Dictionary<,>

map.put("Herisoa",1); /// C# .add(,) /// map["Herisoa"] = 1

map.put("Tsanta",2);

map.put("Ariniaina",3);

map.put("Norotiana",4);

map.put("Andoniaina",6);

map.put("Malala",7);

map.put("Santatriniaina",8);

map.put("Andriantsiory",9);

map.put("Doela",5);

System.out.println(map.get("Herisoa")); //O(1) //map["Herisoa"]

System.out.println(map.get("Andriantsiory")); //O(1)

System.out.println(map.get("Doela")); //O(1)

System.out.println(map.get("Andry")); //O(1)

System.out.println(map.containsKey("Andry")); //O(1)

/// Devoir : 1.a) Qui est arrivé en 4ème ?

/// 1.b) Test d'existence ? map.containsKey(...)

}

public static void demoTreeSet(){

TreeSet<String> present = new TreeSet<>();

present.add("Roots"); //O(log(n))

present.add("Lyssa");

present.add("Tovo");

present.add("Tsiry");

present.add("Missiah");

present.add("Michael");

present.add("Armelle");

present.add("Doela");

if (!present.contains("Tsiry")) present.add("Tsiry");

/// 2) Tsiry ajouté deux fois. Que se passe-t-il ? pas de doubons, code sans effet

present.contains("Missiah"); //O(log(n))

System.out.println("Roots".compareTo("Missiah"));

System.out.println("Doela".compareTo("Lyssa"));

for (String x:present) { //parcours Infixé (arbre binaire) O(n)

System.out.println(x);

}

TreeSet<Personne> personnes = new TreeSet<>();

Personne Zo = new Personne("RAJOELISON", "Zo",21);

Personne Anjaratiana = new Personne("RANDRIAMIHANTA", "Anjaratiana",21);

Personne Luis = new Personne("ZEFANIA", "Luis",22);

Personne Aconit = new Personne("RAKOTOMIANGAVY", "Antsompiainana",22);

Personne Luis2 = new Personne("ZEFANIA", "Luis",24);

Personne Tsiry = new Personne("RANDRIAMBAO", "Tsiry",22);

personnes.add(Zo);

personnes.add(Anjaratiana);

personnes.add(Luis);

personnes.add(Aconit);

personnes.add(Tsiry);

personnes.add(Luis2);

System.out.println(Luis.compareTo(Luis2));

for (Personne p:

personnes) {

System.out.println(p);

}

//Devoir : 3) TreeMap ?

}

public static void main(String[] args) {

demoHashMap();

}

}