# AKTOREAK ETA PELIKULAK KUDEATZEKO APLIKAZIOA



**EGILEAK:**

**Javier Sautua**

**Alain Carpio**

**Iker Ortiz**



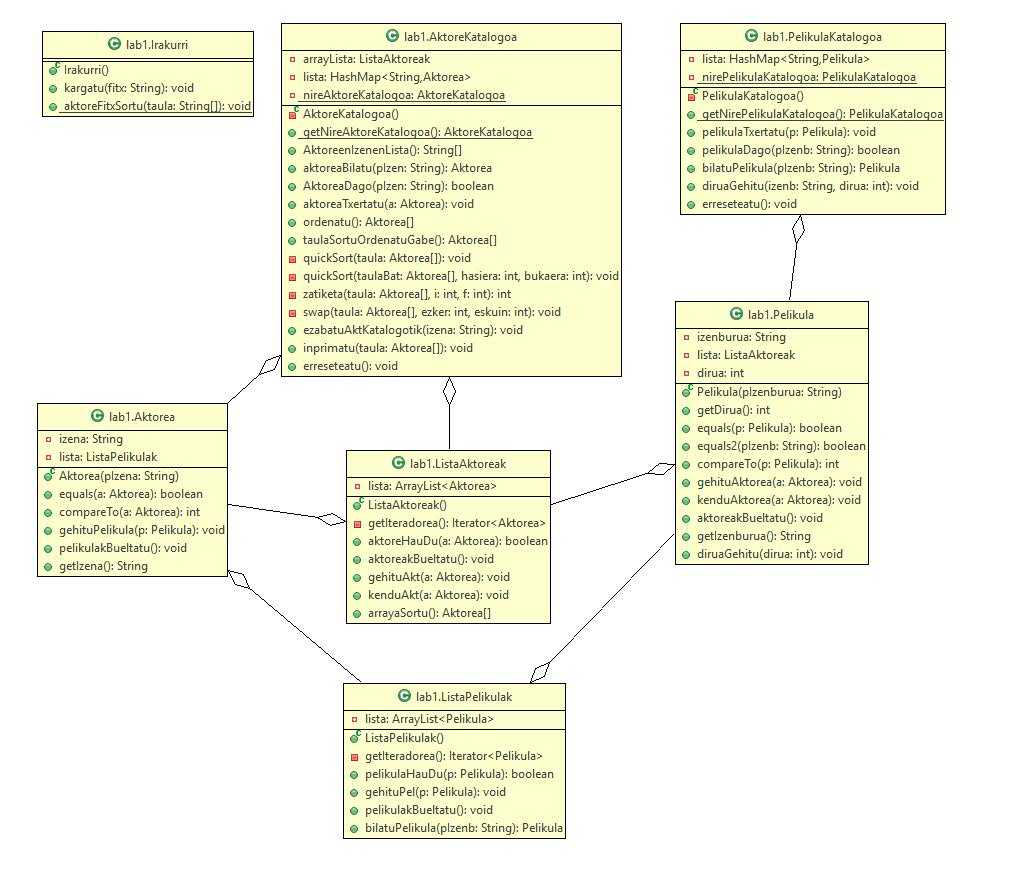
AURKIBIDEA

1. Problemaren deskribapena 3
2. Klase-digrama 3
3. Arazoen ebazpena 3
4. Algoritmoen diseinua eta inplementazioa 4
5. Kodea 5-20
   1. Klaseak 5-12
      1. Aktorea
      2. Pelikula
      3. ListaAktoreak
      4. ListaPelikulak
      5. AktoreKatalogoa
      6. PelikulaKatalogoa
      7. Irakurri
   2. Junit-ak 12-20
      1. AktoreaTest
      2. PelikulaTest
      3. ListaAktoreakTest
      4. ListaPelikulakTest
      5. AktoreKatalogoaTest
      6. PelikulaKatalogoaTest
      7. IrakurriTest
6. Proben emaitza enpirikoak eta denborak 21
7. Ondorioak 22

Problemaren deskribapena

Praktika honetan aktore- eta haien pelikulen zerrenda emanda fitxategi batean, fitxategi hori irakurtzea eta datu-egitura batean gorde eta gero, datu-base hori kudeatzea izan da gure helburua.

Klase-diagrama



Arazoen ebazpena

Hasiera batean pelikulen eta aktoreen katalogoa sortzeko, *ArrayList* datu-egitura [O(N) eraginkortasunezkoa] erabiltzea pentsatu genuen, baina *HashMap* egituraren eraginkortasuna [O(1)] ia konstantea zela ikusi genuenean, hori erabiltzea erabaki genuen.

Aktoreen lista ordenatzerako orduan, erabili dugun datu-egitura, *HashMap*-a, ezin denez ordenatu, *AktoreKatalogoa* klasean *ArrayList* lagungarri bat inplementatzea erabaki genuen.

*Quick-sort* ordenatze metodoa erabili dugu, beste metodoak ez bezala, denbora quasilineala hartzen duelako ordenatzeko [ O(n log n)]. Hala ere, jakina dugu ordenatu behar dugun lista ordenaturik badago, metodo honen abiadura nabarmenki murriztuko dela [O(n2)]. Jakinik emango diguten fitxategia oso handia izango dela, ordenatuta egoteko probabilitatea oso txikia izango da, horregatik *quick-sort* aukeratu dugu.

Algoritmoen diseinua eta inplementazioa

Kargatu metodoa:

fitxategian lerroak dauden bitartean

lerroko lehenengo elementua hartu (aktorea)

aktore berria sortu eta aktoreen katalogoan sartu

aktore horrek pelikulak dituen bitartean

pelikula katalogoan ez badago

sortu pelikula berria eta pelikulen katalogoan sartu

aktoreak bere pelikulen zerrendan ez badu

bere zerrenda propioan sartu pelikula

amaitu baldintza

bestela

aktoreak bere pelikulen zerrendan ez badu

bere zerrenda propioan sartu pelikula

amaitu baldintza

amaitu baldintza

amaitu loop

amaitu loop

Kodea

KLASEAK

AKTOREA

**public** **class** Aktorea **implements** Comparable<Aktorea> {

**private** String izena;

**private** ListaPelikulak lista;

**public** Aktorea(String pIzena){

**this**.izena=pIzena;

**this**.lista=**new** ListaPelikulak();

}

**public** **boolean** equals(Aktorea a){

**if** (**this**.izena.equals(a.izena)){

**return** **true**;

}

**else**{ **return** **false**;}

}

**public** **int** compareTo(Aktorea a){

**return** **this**.izena.compareTo(a.izena);

}

**public** **void** gehituPelikula(Pelikula p){

**if** (!**this**.lista.pelikulaHauDu(p)){

**this**.lista.gehituPel(p);

}

}

**public** **void** pelikulakBueltatu() {

**this**.lista.pelikulakBueltatu();

}

**public** String getIzena() {

**return** **this**.izena;

}

}

PELIKULA

**public** **class** Pelikula **implements** Comparable<Pelikula>{

**private** String izenburua;

**private** ListaAktoreak lista;

**private** **int** dirua;

**public** Pelikula(String pIzenburua){

**this**.izenburua=pIzenburua;

**this**.dirua=0;

**this**.lista=**new** ListaAktoreak();

}

**public** **int** getDirua(){

**return** **this**.dirua;

}

**public** **boolean** equals(Pelikula p){

**if** (**this**.izenburua.equals(p.izenburua)){

**return** **true**;

}

**else**{ **return** **false**;}

}

**public** **boolean** equals2(String pIzenb){

**if** (**this**.izenburua.equals(pIzenb)){

**return** **true**;

}

**else**{ **return** **false**;}

}

**public** **int** compareTo(Pelikula p){

**return** **this**.izenburua.compareTo(p.izenburua);

}

**public** **void** gehituAktorea(Aktorea a){

**if** (!**this**.lista.aktoreHauDu(a)){

**this**.lista.gehituAkt(a);

}

}

**public** **void** kenduAktorea(Aktorea a){

**if** (**this**.lista.aktoreHauDu(a)){

**this**.lista.kenduAkt(a);

}

}

**public** **void** aktoreakBueltatu() {

**this**.lista.aktoreakBueltatu();

}

**public** String getIzenburua() {

**return** **this**.izenburua;

}

**public** **void** diruaGehitu(**int** dirua) {

**this**.dirua=**this**.dirua+dirua;

}

}

LISTAAKTOREAK

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** ListaAktoreak {

**private** ArrayList<Aktorea> lista;

**public** ListaAktoreak(){

**this**.lista=**new** ArrayList<Aktorea>();

}

**private** Iterator<Aktorea> getIteradorea(){

**return** **this**.lista.iterator();

}

**public** **boolean** aktoreHauDu(Aktorea a) {

**boolean** baDago=**false**;

Aktorea aktore1=**null**;

Iterator<Aktorea>itr;

itr=**this**.getIteradorea();

**while**(itr.hasNext()&&!baDago){

aktore1=itr.next();

**if**(aktore1.equals(a)){

baDago=**true**;

}

}

**return** baDago;

}

**public** **void** aktoreakBueltatu() {

Aktorea aktore1=**null**;

Iterator<Aktorea>itr;

itr=**this**.getIteradorea();

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

aktore1=itr.next();

System.***out***.println(aktore1.getIzena());

}

}

**public** **void** gehituAkt(Aktorea a) {

**this**.lista.add(a);

}

**public** **void** kenduAkt(Aktorea a){

**this**.lista.remove(a);

}

**public** Aktorea[] arrayaSortu() {

Aktorea[] array = **this**.lista.toArray(**new** Aktorea[**this**.lista.size()]);

**return** array;

}

}

LISTAPELIKULAK

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** ListaPelikulak {

**private** ArrayList<Pelikula> lista;

**public** ListaPelikulak(){

**this**.lista=**new** ArrayList<Pelikula>();

}

**private** Iterator<Pelikula> getIteradorea(){

**return** **this**.lista.iterator();

}

**public** **boolean** pelikulaHauDu(Pelikula p) {

**boolean** baDago=**false**;

Pelikula pelikula1=**null**;

Iterator<Pelikula>itr;

itr=**this**.getIteradorea();

**while**(itr.hasNext()&&!baDago){

pelikula1=itr.next();

**if**(pelikula1.equals(p)){

baDago=**true**;

}

}

**return** baDago;

}

**public** **void** gehituPel(Pelikula p) {

**this**.lista.add(p);

}

**public** **void** pelikulakBueltatu() {

Pelikula pelikula1=**null**;

Iterator<Pelikula>itr;

itr=**this**.getIteradorea();

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

pelikula1=itr.next();

System.***out***.println(pelikula1.getIzenburua());

}

}

**public** Pelikula bilatuPelikula(String pIzenb){

**boolean** atera=**false**;

Pelikula pelikula1=**null**;

Iterator<Pelikula>itr;

itr=**this**.getIteradorea();

**while**(itr.hasNext()&&!atera){

pelikula1=itr.next();

**if**(pelikula1.equals2(pIzenb)){

atera=**true**;

}

}

**if** (!atera){

pelikula1=**null**;

}

**return** pelikula1;

}

}

AKTOREKATALOGOA

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

**public** **class** AktoreKatalogoa {

**private** ListaAktoreak arrayLista;

**private** HashMap<String,Aktorea> lista;

**private** **static** AktoreKatalogoa *nireAktoreKatalogoa*;

**private** AktoreKatalogoa(){

**this**.lista=**new** HashMap<String,Aktorea>();

**this**.arrayLista= **new** ListaAktoreak();

}

**public** **static** AktoreKatalogoa getNireAktoreKatalogoa(){

**if**(AktoreKatalogoa.*nireAktoreKatalogoa*==**null**){

AktoreKatalogoa.*nireAktoreKatalogoa*=**new** AktoreKatalogoa();

}

**return** *nireAktoreKatalogoa*;

}

**public** String[] AktoreenIzenenLista(){

**return** **this**.lista.keySet().toArray(**new** String[**this**.lista.size()]);

}

**public** Aktorea aktoreaBilatu(String pIzen){

Aktorea aktore1=**null**;

**if** (**this**.lista.containsKey(pIzen)){

aktore1=(Aktorea)**this**.lista.get(pIzen);

}

**return** aktore1;

}

**public** **boolean** AktoreaDago(String pIzen){

**return** **this**.lista.containsKey(pIzen);

}

**public** **void** aktoreaTxertatu(Aktorea a){

**if** (!**this**.lista.containsKey(a.getIzena())){

**this**.lista.put(a.getIzena(),a);

**this**.arrayLista.gehituAkt(a);

}

}

**public** Aktorea[] ordenatu(){

Aktorea[] taula=**this**.arrayLista.arrayaSortu();

**this**.quickSort(taula);

**return** taula;

}

**public** Aktorea[] taulaSortuOrdenatuGabe(){

Aktorea[] taula=**this**.arrayLista.arrayaSortu();

**return** taula;

}

**private** **void** quickSort(Aktorea[] taula){

quickSort(taula, 0, taula.length - 1);

}

**private** **void** quickSort(Aktorea[] taulaBat,**int** hasiera,**int** bukaera){

**if** (bukaera-hasiera>0){

**int** indizeaZatiketa=zatiketa(taulaBat,hasiera,bukaera);

quickSort(taulaBat,hasiera,indizeaZatiketa-1);

quickSort(taulaBat,indizeaZatiketa+1,bukaera);

}

}

**private** **int** zatiketa(Aktorea[] taula,**int** i,**int** f){

Aktorea lag=taula[i];

**int** ezker=i;

**int** eskuin=f;

**while**(ezker<eskuin){

**while**(taula[ezker].compareTo(lag)<=0 && ezker<eskuin)

ezker++;

**while**(taula[eskuin].compareTo(lag)>0)

eskuin--;

**if** (ezker<eskuin)

swap(taula,ezker,eskuin);

}

taula[i]=taula[eskuin];

taula[eskuin]=lag;

**return** eskuin;

}

**private** **void** swap(Aktorea[] taula, **int** ezker, **int** eskuin){

Aktorea a=taula[ezker];

taula[ezker]=taula[eskuin];

taula[eskuin]=a;

}

**public** **void** ezabatuAktKatalogotik(String izena){

Aktorea a=**this**.lista.get(izena);

**this**.lista.remove(izena);

**this**.arrayLista.kenduAkt(a);

}

**public** **void** inprimatu(Aktorea[] taula){ //Ordenatu metodoa probatzeko

**int** i=0;

**while** (i<taula.length){

System.***out***.println(taula[i].getIzena());

i++;

}

}

**public** **void** erreseteatu(){

**this**.lista.clear();

AktoreKatalogoa.*nireAktoreKatalogoa*=**null**;

}

}

PELIKULAKATALOGOA

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

**public** **class** PelikulaKatalogoa {

**private** HashMap<String,Pelikula> lista;

**private** **static** PelikulaKatalogoa *nirePelikulaKatalogoa*;

**private** PelikulaKatalogoa(){

**this**.lista=**new** HashMap<String,Pelikula>();

}

**public** **static** PelikulaKatalogoa getNirePelikulaKatalogoa(){

**if**(PelikulaKatalogoa.*nirePelikulaKatalogoa*==**null**){

PelikulaKatalogoa.*nirePelikulaKatalogoa*=**new** PelikulaKatalogoa();

}

**return** *nirePelikulaKatalogoa*;

}

**public** **void** pelikulaTxertatu(Pelikula p){

**this**.lista.put(p.getIzenburua(),p);

}

**public** **boolean** pelikulaDago(String pIzenb){

**return** **this**.lista.containsKey(pIzenb);

}

**public** Pelikula bilatuPelikula(String pIzenb){

**return** (Pelikula)**this**.lista.get(pIzenb);

}

**public** **void** diruaGehitu(String izenb, **int** dirua){

Pelikula p=**this**.bilatuPelikula(izenb);

**if** (p!=**null**){

p.diruaGehitu(dirua);

}

}

**public** **void** erreseteatu(){

**this**.lista.clear();

PelikulaKatalogoa.*nirePelikulaKatalogoa*=**null**;

}

}

IRAKURRI

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

**import** java.io.\*;

**import** javax.swing.JOptionPane;

**public** **class** Irakurri {

**public** Irakurri(){

}

**public** **void** kargatu(String fitx)**throws** Exception{

Aktorea aktore=**null**;

**int** i;

//int x = 0;

Pelikula peli=**null**;

**try**{

Scanner sarrera = **new** Scanner(**new** FileReader(fitx));

String lerroa;

**while** (sarrera.hasNext()) {

i=0;

lerroa = sarrera.nextLine();

String[] hitzak=lerroa.split(" ### ");

aktore= **new** Aktorea(hitzak[i]);

AktoreKatalogoa.*getNireAktoreKatalogoa*().aktoreaTxertatu(aktore);

i++;

**while**(i<hitzak.length){

**if** (!PelikulaKatalogoa.*getNirePelikulaKatalogoa*().pelikulaDago(hitzak[i])){

peli= **new** Pelikula(hitzak[i]); PelikulaKatalogoa.*getNirePelikulaKatalogoa*().pelikulaTxertatu(peli);

aktore.gehituPelikula(peli);

}

**else**{

peli=PelikulaKatalogoa.*getNirePelikulaKatalogoa*().bilatuPelikula(hitzak[i]);

aktore.gehituPelikula(peli);

}

peli.gehituAktorea(aktore);

i++;

}

//System.out.println(x++);

}

sarrera.close();

}

**catch**(IOException e) {e.printStackTrace();}

}

**public** **static** **void** aktoreFitxSortu(String[] taula) **throws** IOException{

File fitxIzena=**new** File("Aktoreak.txt");

**try**{

FileWriter fw=**new** FileWriter(fitxIzena);

BufferedWriter output=**new** BufferedWriter(fw);

**int** sz=taula.length;

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

output.write(taula[i]);

output.newLine();

}

output.close();

}

**catch**(Exception e) {

JOptionPane.*showMessageDialog*(**null**, "ezin da fitxategia sortu");

}

}

}

JUNIT-AK

AKTOREA TEST

**import** **static** org.junit.Assert.\*;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** AktoreaTest {

Aktorea a;

@Before

**public** **void** setUp() **throws** Exception {

a=**new** Aktorea("Kotsifakis, Hector");

}

@After

**public** **void** tearDown() **throws** Exception {

a=**null**;

}

@Test

**public** **void** testGehituPelikula() {

Pelikula p=**new** Pelikula("Me gusta cuando callas");

Pelikula p1=**new** Pelikula("Daniel & Ana");

a.pelikulakBueltatu();

a.gehituPelikula(p);

System.***out***.println("Pelikula gehitu da");

a.pelikulakBueltatu();

a.gehituPelikula(p1);

System.***out***.println("Orain 2 pelikula agertu behar dira");

a.pelikulakBueltatu();

a.gehituPelikula(p1);

System.***out***.println("Berriro 2 pelikula agertu behar dira");

a.pelikulakBueltatu();

}

@Test

**public** **void** testEquals(){

Aktorea a2=**new** Aktorea("Andreasson, Elisabeth");

*assertFalse*(a.equals(a2));

Aktorea a3=**new** Aktorea("Kotsifakis, Hector");

*assertTrue*(a.equals(a3));

}

@Test

**public** **void** testCompareTo(){

Aktorea a2=**new** Aktorea("Andreasson, Elisabeth");

*assertEquals*(10,a.compareTo(a2));

Aktorea a3=**new** Aktorea("Kotsifakis, Hector");

*assertEquals*(0,a.compareTo(a3));

*assertEquals*(-10,a2.compareTo(a));

}

}

PELIKULA TEST

**import** **static** org.junit.Assert.\*;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** PelikulaTest {

Pelikula p;

@Before

**public** **void** setUp() **throws** Exception {

p=**new** Pelikula("Me gusta cuando callas");

}

@After

**public** **void** tearDown() **throws** Exception {

p=**null**;

}

@Test

**public** **void** testEquals(){

Pelikula p2=**new** Pelikula("Daniel & Ana");

*assertFalse*(p.equals(p2));

Pelikula p3=**new** Pelikula("Me gusta cuando callas");

*assertTrue*(p.equals(p3));

}

@Test

**public** **void** testEquals2(){

*assertFalse*(p.equals2("Daniel & Ana"));

*assertTrue*(p.equals2("Me gusta cuando callas"));

}

@Test

**public** **void** testGehituAktorea() {

Aktorea a=**new** Aktorea("Kotsifakis, Hector");

Aktorea a1=**new** Aktorea("Andreasson, Elisabeth");

Aktorea a2=**new** Aktorea("McKechnie, Lee");

p.aktoreakBueltatu();

p.gehituAktorea(a);

System.***out***.println("Aktorea gehitu da");

p.aktoreakBueltatu();

p.gehituAktorea(a1);

System.***out***.println("Orain 2 aktore agertu behar dira");

p.aktoreakBueltatu();

p.gehituAktorea(a1);

System.***out***.println("Berriro 2 aktore agertu behar dira");

p.aktoreakBueltatu();

p.kenduAktorea(a1);

System.***out***.println("Kotsifakis, Hector agertu behar da");

p.aktoreakBueltatu();

p.kenduAktorea(a2);

System.***out***.println("Kotsifakis, Hector agertu behar da");

p.aktoreakBueltatu();

}

@Test

**public** **void** testCompareTo(){

Pelikula p2=**new** Pelikula("Daniel & Ana");

*assertEquals*(9,p.compareTo(p2));

Pelikula p3=**new** Pelikula("Me gusta cuando callas");

*assertEquals*(0,p.compareTo(p3));

*assertEquals*(-9,p2.compareTo(p));

}

}

LISTAAKTOREAK TEST

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** ListaAktoreak {

**private** ArrayList<Aktorea> lista;

**public** ListaAktoreak(){

**this**.lista=**new** ArrayList<Aktorea>();

}

**private** Iterator<Aktorea> getIteradorea(){

**return** **this**.lista.iterator();

}

**public** **boolean** aktoreHauDu(Aktorea a) {

**boolean** baDago=**false**;

Aktorea aktore1=**null**;

Iterator<Aktorea>itr;

itr=**this**.getIteradorea();

**while**(itr.hasNext()&&!baDago){

aktore1=itr.next();

**if**(aktore1.equals(a)){

baDago=**true**;

}

}

**return** baDago;

}

**public** **void** aktoreakBueltatu() {

Aktorea aktore1=**null**;

Iterator<Aktorea>itr;

itr=**this**.getIteradorea();

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

aktore1=itr.next();

System.***out***.println(aktore1.getIzena());

}

}

**public** **void** gehituAkt(Aktorea a) {

**this**.lista.add(a);

}

**public** **void** kenduAkt(Aktorea a){

**this**.lista.remove(a);

}

**public** Aktorea[] arrayaSortu() {

Aktorea[] array = **this**.lista.toArray(**new** Aktorea[**this**.lista.size()]);

**return** array;

}

}

LISTAPELIKULAK TEST

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** ListaPelikulak {

**private** ArrayList<Pelikula> lista;

**public** ListaPelikulak(){

**this**.lista=**new** ArrayList<Pelikula>();

}

**private** Iterator<Pelikula> getIteradorea(){

**return** **this**.lista.iterator();

}

**public** **boolean** pelikulaHauDu(Pelikula p) {

**boolean** baDago=**false**;

Pelikula pelikula1=**null**;

Iterator<Pelikula>itr;

itr=**this**.getIteradorea();

**while**(itr.hasNext()&&!baDago){

pelikula1=itr.next();

**if**(pelikula1.equals(p)){

baDago=**true**;

}

}

**return** baDago;

}

**public** **void** gehituPel(Pelikula p) {

**this**.lista.add(p);

}

**public** **void** pelikulakBueltatu() {

Pelikula pelikula1=**null**;

Iterator<Pelikula>itr;

itr=**this**.getIteradorea();

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

pelikula1=itr.next();

System.***out***.println(pelikula1.getIzenburua());

}

}

**public** Pelikula bilatuPelikula(String pIzenb){

**boolean** atera=**false**;

Pelikula pelikula1=**null**;

Iterator<Pelikula>itr;

itr=**this**.getIteradorea();

**while**(itr.hasNext()&&!atera){

pelikula1=itr.next();

**if**(pelikula1.equals2(pIzenb)){

atera=**true**;

}

}

**if** (!atera){

pelikula1=**null**;

}

**return** pelikula1;

}

}

AKTOREKATALOGOA TEST

**import** **static** org.junit.Assert.\*;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** AktoreKatalogoaTest {

AktoreKatalogoa ak1;

@Before

**public** **void** setUp() **throws** Exception {

ak1 = AktoreKatalogoa.*getNireAktoreKatalogoa*();

}

@After

**public** **void** tearDown() **throws** Exception {

ak1.erreseteatu();

}

@Test

**public** **void** testAktoreaBilatu() {

Aktorea a = **new** Aktorea("Kotsifakis, Hector");

ak1.aktoreaTxertatu(a);

*assertSame*(a,ak1.aktoreaBilatu("Kotsifakis, Hector"));

*assertNull*(ak1.aktoreaBilatu("Lilleb , Lena Granaas"));

}

@Test

**public** **void** testOrdenatu(){

Aktorea a1 = **new** Aktorea("Kotsifakis, Hector");

Aktorea a2 = **new** Aktorea("Kakurina, Alexandra");

Aktorea a3 = **new** Aktorea("Andreasson, Elisabeth");

ak1.aktoreaTxertatu(a1);

ak1.aktoreaTxertatu(a2);

ak1.aktoreaTxertatu(a3);

Aktorea[] ordenatuGabe = ak1.taulaSortuOrdenatuGabe();

ak1.inprimatu(ordenatuGabe);

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

Aktorea[] ordenatuta = ak1.ordenatu();

ak1.inprimatu(ordenatuta);

}

@Test

**public** **void** testEzabatuAktkatalogotik(){

Aktorea a1 = **new** Aktorea("Kotsifakis, Hector");

Aktorea a2 = **new** Aktorea("Kakurina, Alexandra");

Aktorea a3 = **new** Aktorea("Andreasson, Elisabeth");

ak1.aktoreaTxertatu(a1);

ak1.aktoreaTxertatu(a2);

ak1.aktoreaTxertatu(a3);

Aktorea[] taula1 = ak1.taulaSortuOrdenatuGabe();

ak1.inprimatu(taula1);

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

ak1.ezabatuAktKatalogotik(a1.getIzena());

Aktorea[] taula2 = ak1.taulaSortuOrdenatuGabe();

ak1.inprimatu(taula2);

}

}

PELIKULAKATALOGOA TEST

**import** **static** org.junit.Assert.\*;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** PelikulaKatalogoaTest {

PelikulaKatalogoa k1;

@Before

**public** **void** setUp() **throws** Exception {

k1=PelikulaKatalogoa.*getNirePelikulaKatalogoa*();

}

@After

**public** **void** tearDown() **throws** Exception {

k1=**null**;

}

@Test

**public** **void** testPelikulaDago() {

Pelikula p=**new** Pelikula("Me gusta cuando callas");

k1.pelikulaTxertatu(p);

*assertTrue*(k1.pelikulaDago("Me gusta cuando callas"));

*assertFalse*(k1.pelikulaDago("Daniel & Ana"));

k1.erreseteatu();

}

@Test

**public** **void** testBilatuPelikula() {

Pelikula p=**new** Pelikula("Me gusta cuando callas");

k1.pelikulaTxertatu(p);

*assertEquals*(p,k1.bilatuPelikula("Me gusta cuando callas"));

*assertNull*(k1.bilatuPelikula("Daniel & Ana"));

k1.erreseteatu();

}

@Test

**public** **void** testDiruaGehitu() {

Pelikula p=**new** Pelikula("Me gusta cuando callas");

k1.pelikulaTxertatu(p);

k1.diruaGehitu("Me gusta cuando callas",777);

*assertEquals*(777,p.getDirua());

k1.diruaGehitu("Daniel & Ana",888);

}

}

IRAKURRI TEST

**import** **static** org.junit.Assert.\*;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**public** **class** IrakurriTest {

**private** AktoreKatalogoa aktoreKat;

**private** Irakurri i;

@Before

**public** **void** setUp() **throws** Exception {

aktoreKat=AktoreKatalogoa.*getNireAktoreKatalogoa*();

}

@After

**public** **void** tearDown() **throws** Exception {

aktoreKat=**null**;

}

@Test

**public** **void** testKargatu() {

i = **new** Irakurri();

**try**{

i.kargatu("C://Users/Iker/Desktop/Program/DEA/src/lab1/aaa.txt");

//Hau ALDATU ordenagailuaren arabera

} **catch** (Exception e){

System.***out***.println(e.getMessage());

}

*assertTrue*(AktoreKatalogoa.*getNireAktoreKatalogoa*().AktoreaDago("Kotsifakis, Hector"));

//lehenengo aktorean 3 simbolo agertzen dira hasieran

*assertTrue*(AktoreKatalogoa.*getNireAktoreKatalogoa*().AktoreaDago("Aaberge, Tone Damli"));

*assertTrue*(AktoreKatalogoa.*getNireAktoreKatalogoa*().AktoreaDago("Holub, Carla"));

*assertTrue*(PelikulaKatalogoa.*getNirePelikulaKatalogoa*().pelikulaDago("Me gusta cuando callas"));

*assertTrue*(PelikulaKatalogoa.*getNirePelikulaKatalogoa*().pelikulaDago("Daniel & Ana"));

*assertTrue*(PelikulaKatalogoa.*getNirePelikulaKatalogoa*().pelikulaDago("Seven Mummies"));

//aktoreKat.inprimatu(aktoreKat.taulaSortuOrdenatuGabe());

}

@Test

**public** **void** testAktoreFitxSortu(){

i = **new** Irakurri();

**try**{

i.kargatu("C://Users/Iker/Desktop/Program/DEA/src/lab1/aaa.txt");

Irakurri.*aktoreFitxSortu*(aktoreKat.AktoreenIzenenLista());

} **catch** (Exception e){

System.***out***.println(e.getMessage());

}

}

@Test

**public** **void** testOrdenatu(){

i = **new** Irakurri();

**try**{

i.kargatu("C://Users/Iker/Desktop/Program/DEA/src/lab1/aaa.txt");

//Hau ALDATU ordenagailuaren arabera

} **catch** (Exception e){

System.***out***.println(e.getMessage());

}

Aktorea[] lista = aktoreKat.ordenatu();

//aktoreKat.inprimatu(lista);

}

}

Proben emaitza enpirikoak

Jarraian klase bakoitzaren metodo garrantzitsuen eraginkortasuna aztertuko dugu:

***Aktorea* klasea**

*gehituPelikula* = O(n)

*pelikulaBueltatu* = O(n)

***Pelikula* klasea**

*gehituAktorea* = O(n)

*kenduAktorea* = O(n)

*aktoreakBueltatu* = O(n)

***ListaAktoreak* klasea**

*aktoreHauDu*  = O(n)

*aktoreakBuelatu* = O(n)

*gehituAkt* = O(n)

*kenduAkt* = O(n)

*arrayaSortu* = O(n)

***ListaPelikulak* klasea**

*pelikulaHauDu* = O(n)

*gehituPel* = O(n)

*pelikulakBueltatu* = O(n)

*bilatuPelikula* = O(n)

***AktoreKatalogoa* klasea**

*aktoreenIzenenLista* = O(n)

*aktoreaBilatu* = O(1)

*aktoreaDago* = O(1)

*aktoreaTxertatu* = O(1) + O(n) => O(n)

*ordenatu* = O(n) + O(n\*log n) => O(n)

*quickSort* = O(n\*log n)

***PelikulaKatalogoa* klasea**

*pelikulaTxertatu* = O(1)

*pelikulaDago* = O(1)

*bilatuPelikula* = O(1)

*diruaGehitu* = O(1)

***Irakurri* klasea**

*kargatu* = O(n\*m\*(1+1++p))

n: Lerro/aktore kopurua

m: Aktore bakoitzaren pelikula kopurua

p: Pelikula bakoitzak duen aktore kopurua

*aktoreFitxSortu* = O(n)

https://i.gyazo.com/1753f699676f23697105168a02045076.pngDenborak

Fitxategia irakurtzea (1M+ aktore): 21 s

https://i.gyazo.com/02b4a44d6ba1a9693c3177b7ce7f58f4.png

Fitxategia irakurtzea eta aktoreen

katalogoa ordenatzea (1M+ aktore): 23 s

Ondorioak

Nahiz eta *HashMap* datu-egitura lauhilabetearen amaieran ikasiko dugun, lehenengo praktikan erabiltzeak hurrengoetan erabiltzeko gaitasuna emango digu, abiadura handiagoa duelako orain arte erabili ditugun datu.egiturekin alderatuta (Array eta ArrayList).