LEHENENGO PRAKTIKA

PELIKULA ETA AKTOREEN DATU BASEA

October 11, 2019

Partaideak:
Kerman Sanjuan
Ander San Juan
Josu Ferreras

Aurkibidea

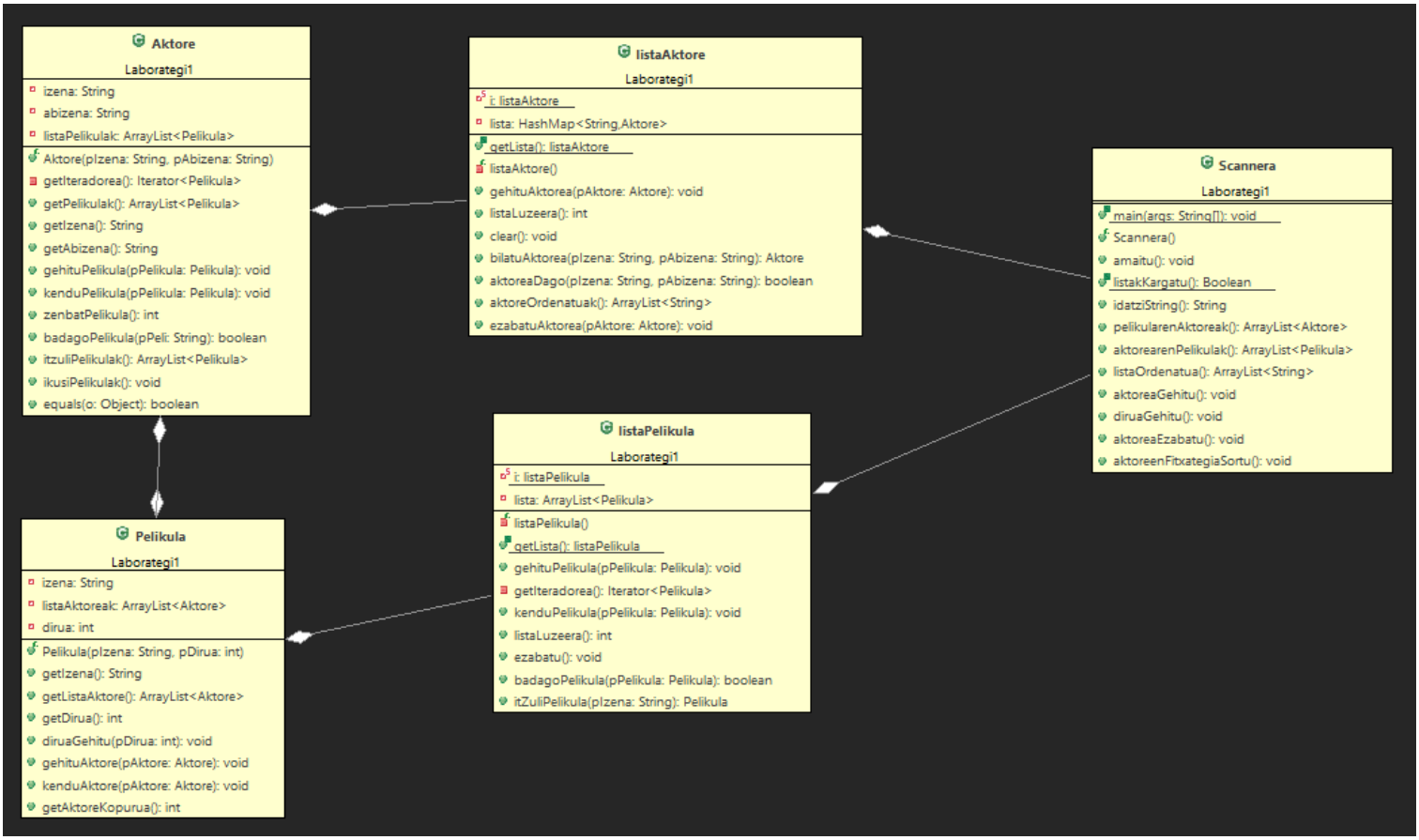
Sarrera

Laborategi honentan bi lista sortu ditugu, bata, ArrayList motakoa eta bestea, HashMap, izan ere, hasiera batean beste ArrayList izango zen lista hau, baina duplikatuak ekiditzeko bi aukera genituen: behin eta berriz gure lista irakurri ea gehitu nahi dugun elementua jadanik badagoen, edo hashMap bat erabili. Azkenean bigarren aukera hartu genuen, lehenengoa denbora asko behar zuen eta.

Laborategi honen funtza bi mota desberdineko listen kudeaketa, artxibo baten irakurketan eta artxibo baten sorreran zatzan.

Dokumentazioa

2.1 KLASEEN DISEINUA



2.2 DATU EGITURA NAGUSIEN AZALPENA

Aktore lista eta pelikula lista izango dira erabiliko ditugun datu egitura nagusiak, izan ere hauek gabe ia ezinezkoa izango litzateke pelikula eta aktore guztiak batera kudeatzea.

2.2.1 Pelikula lista

Pelikula bakoitza baten soilik agertzen denez fitxategian, ez dugu ea pelikula hori jadanik agertu den ala ez konprobatu behar. Hori dela eta pelikulez osatutako Arraylist bat erabiltzea aukeratu dugu.

2.2.2 Aktore lista

Pelikulak ez bezala, aktore bakoitza askotan ager daiteke fitxagian. Aktore berri bat gehitu nahi dugun bakoitzean aktore lista osoa konprobatu beharko bagenu, errepikapenak saihesteko, gure algoritmoa kostu konputazional handia edukiko zuen, horregatik HashMap bat erabiltzea aukeratu.

Dena den, bi klase hauek pelikula eta aktore guztiak dituzten listak dira. Hauetaz gain beste bi Arraylist erabiliko ditugu: -Pelikula bakoitza bere aktoreen lista edukiko du. -Aktore bakoitza bere pelikulen lista edukiko du.

2.3 AZTERTUTAKO AUKEREN DESKRIBAPEAN ETA HARTUTAKO SOLUZIOEN DESKRIBAPEN OROKORRA

2.3.1 Hasierako ideia

Gure idea, laborategiarekin hasi ginean lanean, bi arrayList sortzea zen (Biak EMAK). ArrayList batek pelikula guztiak eukiko zituen bere barnean. Modu berean, pelikula bakoitza bere izena eta pelikula osatzen zuten aktoreen lista. Beste ArrayListak berriz, Aktore guztien lista edukiko zuen, eta, aktore bakoitzak, bere izena eta berak parte artutako pelikulen lista edukiko zuen barne. Laborategiarekin aurrera joaten ginen bitartean, konturatu ginen nola, aktoreak errepikatuta egonda, eraginkortasuna geroz eta txikiagoa zela, hau dela eta, ArrayList izatetik hashMap batera aldatu genuen.

2.3.2 Amaierako ideia

hashMap-aren erabilerarik gauzak asko erraztu egin ziren, alde batetik, konpilazio eta exekuzio denborak asko murriztu ziren, eta beste alde batetik, bere metodo propiokin, konprobazioak eta metodoak egitea askoz errezagoa izan zen.

Gainera, Scanerra egiteko momentuan, 6 aukera desberdin inplementatu genituen, erabiltzaileak programaren kontrol osoa edukitzeko.

2.4 METODO GARRANTZITSUENEN AZALPENA

Atal honetan bakarrik gure programaren metodo garrantzitsuenen azalpenak agertzen dira, aipatzen ez diren metodoen eginkizuna oso argi interpretatzen da haien izenak irakurtzean (getter eta setter-ak, klaseen eraikitzaile gehienak, getIterator...).

2.4.1 Scannera klasean

main(): Programaren metodo nagusia da, haren exekuzioa gauzatzean nondik hasi behar den adierazten diguna.

Scannera(): Izen berdina duen klasearen eraikitzailea da, haren barnean programa honek eskaintzen dituen 7 aukerak "case" egiturarekin inplementatuta. Aukera bakoitzeko honekin erlazionatuta dagoen metodoari deia egiten dio; Adibidez, erabiltzaileak 3. kasua (Aktore bat gehitu aktoreen zerrendara) exekutatzea aginduz gero, aktoreaGehitu() metodoari deituko zaio.

amaitu(): Eskainitako aukeren bat exekutatuz gero erabiltzaileari beste edozein eginkizun egin nahi al duen galdetzen dio. Ezezko kasuan programaren exekuzioa bukatzen da.

listakKargatu(): Eskainitako fitxategiaren informazioa irakurtzeaz eta kudeatzeaz arduratzen da. Informazioaren kudeaketa aurkitutako pelikula guztiak zerrenda batean sartzean eta aurkitutako aktore guztiak beste zerrenda batean sartzean datza.

idatziString() eta idatziInt(): Metodo hauei beste metodo batzuetatik egiten zaie deia, erabiltzaileari zenbaki bat edo karaktere-kate bat idaztea eskatu behar zaion bakoitzean.

pelikularenAktoreak(): Pelikula baten izenburua sartuta, pelikula honetan parte hartu duten aktoreen ("Aktore" motatako objektuak) zerrenda itzultzen du. Sartutako pelikularen izenburua duen pelikularik ez aurkituz gero, errore mezu bat pantailaratuko da.

aktorearenPelikulak(): Aktore baten izena eta abizena sartuta, aktore horrek parte hartutako pelikulen ("Pelikula" motatako objektuak) zerrenda itzultzen du. Sartutako aktorearen izena eta abizena duen aktoreerik ez aurkituz gero, errore mezu bat pantailaratuko da.

listaOrdenatua(): Aktoreen izen eta abizenek osatzen duten String motatako elementuez osatutako alfabetikoki ordenatutako zerrenda bat bueltatzen du, ListaAktore klaseko

aktoreOrdenatuak() metodoari deituz.

aktoreaGehitu(): Aktore baten izena eta abizena sartuta, programak izen eta abizen berbera duen aktore bat bilatzen du aktoreen zerrendan. Horrelakorik ez aurkituz gero, aktore berri bat sortzen du datu horiekin, eta akoreen zerrendara gehitzen du.

diruaGehitu(): Erabiltzaileak emandako titulua duen pelikulari (existitzen bada) erabiltzailea berak idatzitako diru kantitatea gehitzen zaio pelikula aurretik zuen diruari, Pelikula klaseko diruaGehitu() metodoari deituz.

aktoreaEzabatu(): Erabiltzaileak emandako aktorearen izena eta abizenarekin datu berberak dituen aktorea ezabatzen du aktoreen zerrendatik. Horrelako izen eta abizena duen aktorerik ez bada aurkitzen, errore mezu bat pantailaratzen da.

aktoreenFitxategiaSortu(): Aukeratuttako helbidean eta aukeratutako izenarekin aktoreen zerrenda ordenatua (Scannera klaseko aktoreOrdenatuak() metodoa erabiliz) eduki bezala duen "txt" formatuko fitxategi bat sortzen du, helbide berberan izen berdineko fitxategirik ez dagoen ziurtatuz gero eta helbidea balioduna den ziurtatuz gero.

2.4.2 ListaPelikula klasean

gehituPelikula(): Pelikulen zerrendari Pelikula motatako objektu bat gehitzen zaio.

kenduPelikula(): Pelikulen zerrendatik pelikula motatako objektu bat kentzen da.

listaLuzeera(): Pelikulen zerrendak duen luzera bueltatzen du.

ezabatu(): Pelikulen zerrendatik elementu guztiak ezabatzen ditu.

badagoPelikula(): Pelikula motatako objektu bat emanda konprobatzen du elementu hori pelikulen zerrendan dagoen, "true" itzuliz baiezko kasuan.

itzuliPelikula(): String motatako pelikula baten izenburua emanda pelikulen zerrendan izenburu berdina duen pelikula bilatzen du, eta pelikula hori bueltatzen du aurkitzen badu.

2.4.3 Pelikulak klasean

diruaGehitu(): Pelikula baten "dirua" aldagaia inkrementatzen da sartutako kantitatean.

gehituAktore(): Pelikulak duen aktoreen zerrendari sartutako aktorea gehitzen zaio, aktore horrek zerrendan jadanik ez badago.

kenduAktore(): Pelikulak duen aktoreen zerrendatik sartutako aktorea ezabatzen da, aktore hori aurkituz gero.

getAktoreKopurua(): Aktoreen zerrendak duen luzera bueltatzen du.

2.4.4 ListaAktore klasean

gehituAktorea(): Aktoreen zerrendari Aktore motatako objektu bat gehitzen zaio.

listaLuzeera(): Aktoreen zerrendak duen luzera bueltatzen du.

clear(): Aktoreen zerrendatik elementu guztiak ezabatzen ditu.

bilatuAktorea(): Aktore baten izena eta abizena emanda, izen eta abizen berdinak dituen Aktorea bueltatzen du, aktore hori aurkituz gero.

aktoreaDago(): Aktoreen zerrendan sartutako izen eta abizen berdinak dituen aktorea dagoen ala ez bueltatzen du, aldagai boolear moduan.

aktoreOrdenatuak(): Aktoreen zerrendan dauden aktoreetatik haien izen eta abizenak osatzen duten String motatako zerrenda bat sortzen eta alfabetikoki ordenatzen du, ordenean prioritatea emanez zuriunei eta sistemak ezagutzen ez dituen letrei. Sortutako zerrenda bueltatzen du.

ezabatuAktorea(): Aktoreen zerrendan emandako izen eta abizena dituen aktorea aurkituz gero, ezabatzen du.

2.4.5 Aktore klasean

gehituPelikula(): Aktoreak duen pelikulen zerrendari sartutako pelikula gehitzen zaio, pelikula horrek zerrendan jadanik ez badago.

kenduPelikula(): Aktoreak duen pelikulen zerrendatik sartutako pelikula ezabatzen da, pelikula hori aurkituz gero.

zenbatPelikula(): Pelikulen zerrendak duen luzera bueltatzen du.

badagoPelikula(): Pelikula baten izenburua sartuta, pelikulen zerrendan izenburu berdina duen pelikula aurkituz gero egiazko balioa bueltatuko du aldagai boolear moduan.

itzuliPelikulak(): Pelikula motatako objektuez osatutako zerrenda bueltatzen du.

ikusiPelikulak(): Aktorearen pelikulen zerrendan dauden pelikula guztien izenburuak pantailaratzen ditu.

2.5 METODO NAGUSIEN DISEINUA ETA INPLEMENTAZIOA

2.5.1 ListaPelikulak klasean

2.5.1.1 gehituPelikula()

//Aurre: pPelikula ez-hutsa izatea

//Post: pPelikula gure listan egotea.

public void gehituPelikula(Pelikula pPelikula) [

lista.add (pPelikula)

] Programa honen kostea 0(1)- ekoa dela esan daiteke.

2.5.1.2 badagoPelikula()

//Aurre: Gure lista gutxienez elementu bat dauka eta pPelikula ez-nulua da.

//Post: True itzuliko du baldin pPelikula lista badago, bestela false

public boolean badagoPelikula(Pelikula pPelikula)[

itr = this.getIteradorea

bitartean (itr.HurrengoaDu) loop

p = itr.next

baldin (p.izena.Equals(pPelikula.izena)) orduan

return true

ambitartean

return false

]

Programa honen kostea O(n)-koa da, non N listaren luzeera den.

2.5.1.3 itzuliPelikula()

```
//Aurre: Gure lista ez dago hutsik
//Post: Izen bereko pelikula itzuliko du, ez badu aurkitzen, null itzuliko du.
public Pelikula itzuliPelikula(String pIzena)[
        aurkitua = false
        itr = this.getIteradorea
        Pelikula p = itr.next
        bitartean (itr.hurrengoaDu eta ezAurkitua) loop
                baldin (itr.izena = pIzena) orduan aurkitua = true
                bestela itr.next
        ambitartean
        baldin(aurkitua) return true
        bestela return false
]
```

Metodo honen kostuaa O(n)-koa da, non N listaren luzeera den.

2.5.2 Pelikulak klasean

2.5.2.1 gehituAktore()

public void gehituAktore(Aktore pAktore)[

baldin(listaHutsaDa) orduan listaAktoreak = new ArrayList<Aktorea>

bestela listariGehitu(pAktore)

]

Metodo honen kostua O(n)-koa da ArrayList-ak bere luzeera bikoiztu behar duenenan, baina hori N alditan behin gertatzen denez, kostua N/N edo O(1) da, non N listaren luzeera den.

2.5.3 ListaAktore klasean

2.5.3.1 gehituAktorea()

//post: pAktorea aktoreen listan gehitu da.

public void gehituAktore(Aktore pAktore)[]

baldin(listaAktorea == null) orduan

sortuAktoreenListaBerria

bestela

gehituAktorea(pAktore)] Metodo honen kostua O(1) da.

2.5.3.2 aktoreaDago()

//aurre: ez hutsa den lista bat.

//post: pAktorea aktorea badago, true itzuliko du, bestela false.

public boolean aktoreaDago(String pAbizena, String pIzena)[

return this.lista.badaukaKey(pAbizena+ " " + pAIzena)

] Metodo honen kostua O(1) da, hashMap bat delako.

2.5.3.3 aktoreOrdenatuak()

//aurre: Aktore lista ez huts bat

//post: lista ordenatuta alfabetikoki.

public ArrayList<String> aktoreOrdenatuak[

lista = new ArrayList<lista.keySet())

return mergeSort(lista)

]

Metodo honen kostua O(nLogN) da, non N listaren luzeera da, Collections.sort mergeSort motako ordenazio algoritmoa erabiltzen du eta.

2.5.3.4 ezabatuAktorea()

public void ezabatuAktorea(Aktore pAktore)[

this.lista.remove(pAktore.getAbizena+ " "+pAktore.getIzena)

]

Metodo honen kostua O(1) da, hashMap bat delako.

2.5.4 Aktore klasean

2.5.4.1 badagoPelikula()

```
//aurre: —
//post: Aktorea pelikula horretan parte hartu badu true itzuliko du, bestela false.
public boolean badagoPelikula(String pPeli)[
    itr = this.getIteradotea
    pelikula p
    bitartean(itr.hurrengoaDu)[
        p = itr
        baldin (p.izena.equals(pPeli) orduan return true
        bestela
            p = itr.next
        ambaldin
    ]
    return false
]
```

Metodo honen kostua O(n) da, kasurik txarrenean lista osoa errekorritu behar duelako.

2.5.4.2 ikusiPelikulak()

```
//aurre: —
//post: Aktorearen pelikulak itzultzen ditu.
public void ikusiPelikulak()[
    itr = this.getIteradorea
    baldin itr.duHurrengoa[
        Pelikula p
        bitartean (itr.duHurrengoa) loop
            p = itr.next
            printeatu(p.getIzena)
    ambaldin
    ]
]
```

Metodo honen kostua O(n) da, lista osoa igaro behar duelako.

2.6 PROGRAMAK DUEN 7 AUKEREN AZALPENA

```
C:\Users\elsan>cd \Users\elsan\Desktop

C:\Users\elsan\Desktop>java -Xmx1g -jar Laborategi1.jar
Pelikula eta aktoreak kargatzen daude, prozesu honek ez luke denbora luzerik hartu behar
1283334 aktore ezberdin daude.
238978 pelikula daude.
40 segundu behar izan dira.
Zer egin nahiko zenuke?
1. aukera: Aktore baten pelikula guztiak lortu
2. aukera: Aktoreen lista ordenatua lortu
3. aukera: Aktore berri bat gehitu
4. aukera: Pelikula baten aktoreak lortu
5. aukera: Pelikula bati dirua gehitu.
6. aukera: Aktore bat ezabatu
7. aukera: Fitxategi bat sortu aktoreen zerrenda ordenatuarekin
```

2.6.1 1. aukera: Aktore baten pelikula guztiak lortu

Aukera honek, aktorearenPelikulak() metodoari dei egiten dio.

//aurre: —

//post: Aktorearen pelikulen lista bueltatzea

abizena = idatziString()

izena=idatziString()

ArrayList<Pelikula> =listaAktore.getLista.bilatuAktore(abizena,izena).getPelikula

Metodo honek, aktore baten pelikulak itzuliko ditu, aktorea ez bada aurkitzen, exception bat aterako da, errore mezu batekin. Metodo honen kostua O(1) izango da, izan ere, HashMap baten bilatzen ari gara aktore bat eta horren kostua konstantea da.

2.6.2 2. aukera: Aktoreen lista ordenatua lortu

Izenak dioen moduan, metodo honek aktoreen zerrenda alfabetikoki ordenatuta itzuliko du.

//aurre: —

//post: Aktoreareen lista ordenatua itzuli

system.out.println("Modu efektiboan lortu da lista ordenatua")

return listaAktore.getlista().aktoreOrdenatuak()

Aurretik azalduta, baina berriro ere, merge-sort erabiliz alfabetikoki ordenatzen du. Metodo honek O(nlogn) kostua izango du, non n aktore listaren luzeera izango den.

2.6.3 3. aukera: Aktore berri bat gehitu

Modu erraz batean, gu idatzitako izen eta abizen baten bidez, aktorea jartzen du listan. Listan jadanik badago, ez du hutziko listan gehitzen.

//aurre: —

//post: Aktorea jadanik listan ez badago, aktorea listara gehitu. Bestela, ezer ez egin

system.out.println("Idatzi abizena")

abizena=idatziString()

system.out.println("Idatzi izena")

izena=idatziString()

baldin(!lista.aktoreadago(izena,abizena) {

    lista.gehituAktorea(new Aktore(abizena, izena))

}

Metodo honen kostua O(1) izango da, hashMap baten gehitzen gaudelako aktorea.

2.6.4 4. aukera: Pelikula baten aktoreak lortu

Lehenengo aukeraren antzekoa, baina honek, pelikula baten aktoreak itzuliko ditu.

//aurre: —

//post: Pelikularen aktore lista bueltatzea

system.out.println("Idatzi izena")

peliIzena=idatziString()

ArrayList<Aktore> lista=listaPelikula.itzuliPelikula(peliIzena).getlistaAktore()

system.out.println(lista.size() + " aktore ditu pelikulak")

return aktoreak

Metodo honen kostua O(n) izango da, non n Pelikula listaren luzeera den.

2.6.5 5. aukera: Pelikula bati dirua gehitu.

Hasieran, metodo honek, erabiltzaileak esandako pelikulari diru kantitate bat gehituko dio. Pelikularen izena ez badago edo diru kantitatea egokia ez bada, errore mezu bat aterako da, eta berriko eskatuko du.

//aurre:Nahi dugun pelikula listan egotea

//post: Lista horri diru hori gehitu izatea

system.out.println("Idatzi izena")

izena=idatziString()

baldin(listaPelikula.pelikulaDago(izena)

    system.out.println("Idatzi diru kopurua")

    dirua=idatziInt()

    listaPelikula.getPelikula(izena).diruaGehitu(dirua)

else

    system.out.println("Pelikula hori ez da existitzen")

metodo honi berriro dei egin

Metodo honen kostua O(n+n) izango da, beraz O(n), non n Pelikula listaren luzeera den.

2.6.6 6. aukera: Aktore bat ezabatu

Izenak dioen bezala, guk esandako aktorea listatik ezabatuko du.Aktorea ez badago, berriro izena eskatuko du.

//aurre: — (ez da beharrezkoa aktorea existitzea)

//post: Aktore hori listan ez egotea

system.out.println("Idatzi abizena")

abizena = idatziString()

system.out.println("Idatzi izena")

izena = idatziString()

baldin(listaAktore.aktoreaDago(abizena, izena)

    listaAktore.ezabatuAktorea(new Aktore(abizena,izena)

else

    system.out.println("Aktorea ez da existitzen, berriro sartu datuak")

Metodo honen kostua O(n) izango da, aktoreak HashMap baten daudelako eta aktore bat aurkitzeko denbora konstantea delako.

2.6.7 7. aukera: Fitxategi bat sortu aktoreen zerrenda ordenatuarekin

Aukera honetan Scannera klaseko aktoreenFitxategiaSortu() metodoari deitzen zaio.

Metodo honek dei egiten dio ListaAkore klaseko aktoreOrdenatuak() metodoari, aktoreen izen-abizenak alfabetikoki ordenatuta duen zerrenda bat lortuz. Hau egin eta gero "txt" formatuko fitxategi bat sortzen da erabiltzaileak eskainitako helbidean berak idatzitako izenarekin, helbidea

```
C:\\Users\\IZENA\\Desktop
```

formatuan eman behar da.

Behin fitxategia arazorik gabe sortu dela ziurtatuz gero, fitxategian idazten dira lehen lortutako zerrendak dituen datuak, bata albokoengandik bereizita. Azkenik mezu bat pantailaratzen da, fitxategia arazorik gabe sortu den ala ez esanez.

//aurre: Aktore lista ez hutsa izatea

//post: Fitxategi bat sortu izatea aktoreen izenekin (alfabetikoki ordenatua)

ArrayList<String> listaOrdenatua = listaAktore.aktoreaOrdenatuak()

system.out.println("Idatzi fitxategiaren izena")

izena = idatziString()

system.out.println("Idatzi non gorde behar den fitxategia")

helbidea = idatziString()

bitartean(aktoreak daude listan)

    fitxategian aktore berria idatzi

system.out.println("Fitxategia sortu da")

Metodo honen kostua O(n+n) izango da (n aktore ordenatuak lortzeko eta beste n bat lista oso irakurtzeko), beraz O(n). Klasean azaldu egin zen moduan, metodo honek, lista zaitu egiten du, ardatz "aleatorio" batekin, eta ezkerreko aldea ordenatu egiten du, ondoren, modu errekurtsiboan behin eta berriro deituz, listaren tamaina txikituz. Beste modu batean esanda, lista zatika ordenatu egiten du, gero batu zatiak, eta berriro egin.

2.7 PROBEN EMAITZA ENPIRIKOAK (DENBORAK)

2.7.1 Sarrera moduan

Metodo bakoitzak behar duen denbora kalkulatzeko, timer bat inplementatu diogu modu honetan:

long starTime = System.nanoTime() //Kodea hemendik aurrera.

long endTime = System.nanoTime()

long elapsed =endTime-startTime

System.out.println(timeElapsed / 1000000000 + " segundu behar izan dira. ");

2.7.2 Fitxategia irakurtzeko eta sortzeko

Gure programak, fitxategia irakurtzeko, eta hortik bi listak sortzeko behar duen denbora 31 segundukoa da.

2.7.3 Aktoreen lista alfabetikoki ordenatzeko behar duen denbora

Aktoreen lista hashMap bat izanda, metodo honek bakarrik segundu bat beharko du.

2.7.4 Fitxategi bat sortzeko denbora

Fitxategiaren izena eta bere helbidea pegatu dira, modu honetan, emaitza zehatzagoa izateko. Metodoak, guztira, 8 segundu behar izan ditu lista ordenatu eta fitxategia sortzeko.

2.7.5 Bilatu pelikula (Kasurin txarrenean)

Pelikula asko daudenez, kasurik txarrenean egonda, azken pelikula bilatuko dugu. Kasu honetan, ere segundu bat tardatu egin du (Borobilduta, kontagailua segundutan baitdago)

2.7.6 Gainontzeko metodoak

Hauek dira denbora gehien behar duten metodoak, besteak, gehien bat, atributu baten aldaketan edo bilaketa batean datzate, hau dela eta, denborak oso laburrak izaten dira (Segundu bat baino txikiagoak + hashMap-en erabilera), hau dela eta, ez ditugu kontuan izan.

2.8 PROBA KASUAK

Programaren exekuzioan gure aplikazioak kasu kritikoetan modu eraginkor batean erantzuteko eta arazoak ekiditzeko, metodo aipagarrien proba kasuen analisia egin behar da kodea idazten hasi baino lehen. Horretarako, metodo horietan kasu kritikoak aztertu behar ditugu.

2.8.1 Aktore klasea

2.8.1.1 gehituPelikula()

listaPelikula	Sarrera	listaPelikula (emaitza)
[]	a	[a]
[a]	b	[a,b]
[a]	a	[a,a]
[a,b,c,d]	e	[a,b,c,d,e]
[a,b,c,d]	a	[a,b,c,d,a]
[a,b,c,d]	c	[a,b,c,d,c]
[a,b,c,d]	d	[a,b,c,d,d]

2.8.1.2 kenduPekikula()

listaPelikula	Sarrera	listaPelikula (emaitza)
[]	a	[]
[a]	b	[a]
[a]	a	[]
[a,b,c,d]	e	[a,b,c,d]
[a,b,c,d]	a	[b,c,d]
[a,b,c,d]	c	[a,b,d]
[a,b,c,d]	d	[a,b,c]

2.8.1.3 badagoPelikula()

listaPelikula	Sarrera	Emaitza
[]	"Kima"	False
["Kima"]	"Brahmastram"	False
["Kima"]	"Kima"	True
["Kima","All City","Unforgiven","Brahmastram"]	"Geschichten aus dem Lepratal"	False
["Kima","All City","Unforgiven","Brahmastram"]	"Kima"	True
["Kima","All City","Unforgiven","Brahmastram"]	"Unforgiven"	True
["Kima","All City","Unforgiven","Brahmastram"]	"Brahmastram"	True
["Kima","All City","Unforgiven","Brahmastram"]	"unforgiven"	False
["Kima","All City","Unforgiven","Brahmastram"]	"Unforgiben"	False

2.8.2 Pelikula klasea

2.8.2.1 gehituAktore()

listaAktore	Sarrera	listaAktore (emaitza)
[]	a	[a]
[a]	b	[a,b]
[a]	a	[a,a]
[a,b,c,d]	e	[a,b,c,d,e]
[a,b,c,d]	a	[a,b,c,d,a]
[a,b,c,d]	c	[a,b,c,d,c]
[a,b,c,d]	d	[a,b,c,d,d]

2.8.2.2 kenduAktore()

listaAktore	Sarrera	listaAktore (emaitza)
[]	a	[]
[a]	b	[a]
[a]	a	[]
[a,b,c,d]	e	[a,b,c,d]
[a,b,c,d]	a	[b,c,d]
[a,b,c,d]	c	[a,b,d]
[a,b,c,d]	d	[a,b,c]

2.8.3 listaAktore klasea

2.8.3.1 gehituAktorea

listaAktore	Sarrera	listaAktore (emaitza)
[]	a	[a]
[a]	b	[a,b]
[a]	a	[a]
[a,b,c,d]	e	[a,b,c,d,e]
[a,b,c,d]	a	[a,b,c,d]
[a,b,c,d]	c	[a,b,c,d]
[a,b,c,d]	d	[a,b,c,d]

2.8.3.2 kenduAktore()

listaPelikula	Sarrera	listaAktore (emaitza)
[]	a	[]
[a]	b	[a]
[a]	a	[]
[a,b,c,d]	e	[a,b,c,d]
[a,b,c,d]	a	[b,c,d]
[a,b,c,d]	c	[a,b,d]
[a,b,c,d]	d	[a,b,c]

2.8.3.3 bilatuAktorea()

listaAktore	Sarrera	Emaitza
[]	a	False
[a]	b	False
[a]	a	True
[a,b,c]	d	False
[a,b,c,d]	a	True
[a,b,c,d]	c	True
[a,b,c,d]	d	True
[a,b,c,d]	e	False

2.8.4 listaPelikula klasea

2.8.4.1 gehituPelikula

listaPelikula	Sarrera	listaPelikula (emaitza)
[]	a	[a]
[a]	b	[a,b]
[a]	a	[a]
[a,b,c,d]	e	[a,b,c,d,e]
[a,b,c,d]	a	[a,b,c,d,a]
[a,b,c,d]	c	[a,b,c,d,c]
[a,b,c,d]	d	[a,b,c,d,d]

2.8.4.2 kenduPelikula()

listaPelikula	Sarrera	listaPelikula (emaitza)
[]	a	[]
[a]	b	[a]
[a]	a	[]
[a,b,c,d]	e	[a,b,c,d]
[a,b,c,d]	a	[b,c,d]
[a,b,c,d]	c	[a,b,d]
[a,b,c,d]	d	[a,b,c]

2.8.4.3 bilatuPelikula()

listaPelikula	Sarrera	Emaitza
[]	a	False
[a]	b	False
[a]	a	True
[a,b,c]	d	False
[a,b,c,d]	a	True
[a,b,c,d]	c	True
[a,b,c,d]	d	True
[a,b,c,d]	e	False

Kodea

3.1 SCANNERA

package Laborategi1; import java.io.File; import java.io.FileWriter; import java.io.BufferedWriter; import java.io.IOException; import java.util.ArrayList; import java.util.Scanner; import java.util.concurrent.TimeUnit; import java.util.*;

public class Scannera

public static void main(String[] args) throws Exception

Boolean kargatuta=false;

listaAktore.getLista().clear();

System.out.println("Pelikula eta aktoreak kargatzen daude, prozesu honek ez luke denbora luzerik hartu behar");

while (!kargatuta)

kargatuta=listakKargatu();

new Scannera();

//————————————————————————— ELEGIR LAS OPCIONES———-

```
public Scannera() throws Exception
        System.out.println("Zer egin nahiko zenuke?");
        System.out.println("1. aukera: Aktore baten pelikula guztiak lortu");
        System.out.println("2. aukera: Aktoreen lista ordenatua lortu");
        System.out.println("3. aukera: Aktore berri bat gehitu");
        System.out.println("4. aukera: Pelikula baten aktoreak lortu");
        System.out.println("5. aukera: Pelikula bati dirua gehitu.");
        System.out.println("6. aukera: Aktore bat ezabatu");
        System.out.println("7. aukera: Fitxategi bat sortu aktoreen zerrenda ordenatuarekin");
        int num = this.idatziInt();
        switch (num)

            case 1: //FUNCIONA. COMPROBADOS CASOS POSIBLES.
                //Comprobado que pasa si mete nombre erroneo.
                this.aktorearenPelikulak();
                amaitu();
                break; //para que despues de preguntar amaitu acabe
            case 2: // FUNCIONA
                this.listaOrdenatua();
                amaitu();
                break; //para que despues de preguntar amaitu acabe
            case 3: //FUNCIONA. COMPROBADOS CASOS POSIBLES
                this.aktoreaGehitu();
                amaitu();
```

```
            break; //para que despues de preguntar amaitu acabe
        case 4: //Comprobado que ocurre si el nombre es erroneo.
            this.pelikularenAktoreak();
            amaitu();
            break;
        case 5:
            this.diruaGehitu();
            amaitu();
            break;
        case 6:
            this.aktoreaEzabatu();
            amaitu();
            break;
        case 7:
            this.aktoreenFitxategiaSortu();
            amaitu();
            break;
        default:
            System.err.println ( "Errorea gertatu da sisteman" );
            this.amaitu();
            break;
```

public void amaitu() throws Exception //Metodo para cerrar lo que hay que hacer.

```
System.out.println("Saioa amaitu nahi duzu?");
System.out.println("BAI sartu 9");
System.out.println("EZ sartu 0");
Scanner irten = new Scanner(System.in);
try
        switch (irten.nextInt())
        case 9:
                System.out.println ("Saio amaituko da.");
                break;
        case 0:
                new Scannera();
                irten.close();
                break;
        default:
                System.err.println ( "Ez dago horrelako aukerarik" );
                irten.close();
                break;
       irten.close();
catch (InputMismatchException e)
        System.err.println("Ez duzu zenbakirik sartu, berriro egin");
        this.amaitu();
```

```
//————————————————————————————————————————- ESCANER——
        public static Boolean listakKargatu() throws Exception
                long startTime = System.nanoTime();
                String[] unekoAktore=null;
                String[] linea = null;
                File file = new File("C:\\Users\\elsan\\Desktop\\FilmsActors20162017.txt");
                Scanner sc = new Scanner(file);
                TimeUnit.SECONDS.sleep(1);
                while (sc.hasNextLine())
                        linea = sc.nextLine().replace("&&& S", "<").replace("> ", ">").split("[<>]+");
//la primera linea
                        String pelikulaIzena=linea[0].replace("-","");
                        Pelikula Pelikula1 =new Pelikula(pelikulaIzena,0);
                        for(int i=1;i<linea.length;i++)  //pongo i=1 porque el primer dato del
array es el nombre de la peli y no un actor
                                unekoAktore=linea[i].replace(", ",",").split(",");
                               if(unekoAktore.length==1)  //por algun motivo hay actores que
no tienen apellido (o estan mal formateados, yo que se) en la lista
                                        String izena = unekoAktore[0];
                                        Aktore Aktore1 = new Aktore("", izena);//eeeeeeeeeee
                                        if(listaAktore.getLista().aktoreaDago("", izena))
                                           Pelikula1.gehituAktore(listaAktore.getLista().bilatuAktorea("",
izena));
                                           listaAktore.getLista().bilatuAktorea("", izena). gehitu-
Pelikula(Pelikula1);
                                        else
                                           Pelikula1.gehituAktore(Aktore1);
```

```
                    Aktore1.gehituPelikula(Pelikula1);
                    listaAktore.getLista().gehituAktorea(Aktore1);

                else
                    String abizena = unekoAktore[0]; //el apellido
                    String izena = unekoAktore[1]; //el nombre
                    Aktore Aktore1 = new Aktore(abizena, izena);
                    if(listaAktore.getLista().aktoreaDago(abizena, izena))
                      Pelikula1.gehituAktore(listaAktore.getLista(). bilatuAktorea(abizena,izena));
                      listaAktore.getLista().bilatuAktorea(abizena, izena).gehituPelikula(Pelikula1);
                    else
                      Pelikula1.gehituAktore(Aktore1);
                      Aktore1.gehituPelikula(Pelikula1);
                      listaAktore.getLista().gehituAktorea(Aktore1);


            listaPelikula.getLista().gehituPelikula(Pelikula1);
        System.out.println(listaAktore.getLista().listaLuzeera() + " aktore ezberdin daude.");
        System.out.println(listaPelikula.getLista().listaLuzeera() + " pelikula daude.");
        long endTime = System.nanoTime();
        long timeElapsed = endTime - startTime;
```

```
                System.out.println(timeElapsed / 1000000000 + " segundu behar izan dira.
" );

                sc.close();

                return true;


        //————————————————————————————— METODOS DE LAS ACCIONES———————
————————————————

        public String idatziString()

                Scanner s = new Scanner(System.in);

                String gureString =null;

                gureString=s.nextLine();

                return gureString;


        public int idatziInt() throws Exception

                Scanner s = new Scanner(System.in);

                int gureInt = 1;

                try

                        gureInt=s.nextInt();

                catch( InputMismatchException e)

                        System.err.println("Ez duzu zenbakirik sartu.");

                        return 10;

               return gureInt;
```

```
public ArrayList<Aktore> pelikularenAktoreak()throws Exception
    System.out.println("Idatzi pelikularen izena ");
    String ad = this.idatziString();
    //System.out.println(listaPelikula.getLista().itZuliPelikula(ad) instanceof Pelikula);
    try
        ArrayList<Aktore> aktoreak= listaPelikula.getLista().itZuliPelikula(ad).getListaAktore();
        System.out.println(aktoreak.size() + " aktore ditu pelikula honek, izenen lista modu egokian lortu da.");
        return aktoreak;
    catch (NullPointerException e)
        System.err.println("Pelikularen izena ez da existitzen.");
        return null;
```

```
public ArrayList<Pelikula> aktorearenPelikulak() throws Exception
    try
        System.out.println("Idatzi aktorearen abizena");
        String abiz=this.idatziString();//usaremos este metodo apartir de ahora
        System.out.println("Idatzi aktorearen izena");
        String izen= this.idatziString();
        ArrayList<Pelikula> lista = listaAktore.getLista().bilatuAktorea(abiz,
izen). getPelikulak(); //Esto lo hago para que pueda decir que se ha hecho bien.
        System.out.println("Listak modu egokian itzuli dira.");
        return lista;
    catch (NullPointerException e)
        //Badakit ez dela gomendagarria pointer Exception-a modu honetan
tratatzea, baina noizbait erabiltzeko....
        System.err.println("Izena ez da existitzen.");
        return null;


public ArrayList<String> listaOrdenatua()
    ArrayList<String> lista =listaAktore.getLista().aktoreOrdenatuak();
    System.out.println("Modu efektiboan lortu da lista ordenatua.");
    return lista;
```

```
public void aktoreaGehitu()
    System.out.println("Idatzi aktorearen abizena");
    String abiz=this.idatziString();//usaremos este metodo apartir de ahora
    System.out.println("Idatzi aktorearen izena");
    String izen= this.idatziString();
    if( listaAktore.getLista().bilatuAktorea(abiz, izen) == null)
        listaAktore.getLista().gehituAktorea(new Aktore(abiz,izen));
            if(listaAktore.getLista().aktoreaDago(abiz, izen))  //Esto es si lo
ha añadido bien.
                System.out.println("Aktorea modu egokian gehitu da.");

    else
        System.err.println("Aktorea jadanik dago listan, orduan ez da gehitu.");
```

```
public void diruaGehitu() throws Exception
    //Caso de no mal metido el dinero, o la pelicula no existir, TRATADO:
    System.out.println("Idatzi pelikularen izena");
    String iZ=this.idatziString();
    if(listaPelikula.getLista().badagoPelikula(new Pelikula(iZ, 0)))
        System.out.println("Idatzi zenbat diru gehitu nahi duzun");
        try
            int dI=Integer.valueOf(this.idatziString());
            listaPelikula.getLista().itZuliPelikula(iZ).diruaGehitu(dI);
            System.out.println("Dirua modu egokian gehitu da, eguneko
dirua " + listaPelikula.getLista().itZuliPelikula(iZ).getDirua()+" eurokoa da." );
        catch(NumberFormatException e)
            System.err.println("Ez duzu dirua ondo sartu, berro egin");
            this.diruaGehitu();
    else
        System.err.println("Ez dago pelikula, berriro egin");
        this.diruaGehitu();
```

```
public void aktoreaEzabatu()
        System.out.println("Idatzi aktorearen abizena:");
        String abiz = this.idatziString();
        System.out.println("Idatzi aktorearen izena:");
        String izen = this.idatziString();
        if(listaAktore.getLista().aktoreaDago(abiz, izen))
                listaAktore.getLista().ezabatuAktorea(new Aktore(abiz,izen));
                System.out.println("Aktorea modu egokian ezabatu da.");
        else
                System.err.println("Aktorea ez dago, berriro idatzi datuak");
```

```
public void aktoreenFitxategiaSortu()
    ArrayList<String> lista=listaAktore.getLista().aktoreOrdenatuak();
    File fitxategia;
    try
        System.out.println();
        System.out.println("Idatzi sortuko den fitxategiaren izena:");
        String fitxIzena=this.idatziString();
        boolean sortuta=false;
        System.out.println("Idatzi sortuko den fitxategiaren helbidea:");
        String fitxHelbidea=this.idatziString();
        fitxategia=new File(fitxHelbidea+"
"+fitxIzena+".txt"); //Intenta crear el archivo
        while (!sortuta)
            sortuta=true;
            if (!fitxategia.createNewFile())  //Comprueba si hay algun fitx-
ategi con el mismo nombre en el mismo helbide
                sortuta=false;
                System.out.println("Jada existitzen da fitxategi bat izen
horrekin helbide honetan");
                System.out.println();
                System.out.println("Idatzi sortuko den fitxategiaren izena:");
                fitxIzena=this.idatziString();
                System.out.println("Idatzi sortuko den fitxategiaren hel-
bidea:");
                fitxHelbidea=this.idatziString();
                fitxategia=new File(fitxHelbidea+"
"+fitxIzena+".txt");
```

```
                    if (!fitxategia.isFile())  //Comprueba si se ha creado el fitxategi
                        throw new Exception();
                    BufferedWriter bw=new BufferedWriter(new FileWriter(fitxategia));
//El aldagai que escribe en el txt
                    Iterator<String> itr=lista.iterator();
                    String aktorea;
                   while (itr.hasNext())  //Escribe los actores en el txt
                        aktorea=itr.next();
                        if (itr.hasNext())
                            bw.write(aktorea+", ");
                        else
                            bw.write(aktorea+".");
                    bw.close();
                    System.out.println("Fitxategia sortu da");
                catch (Exception e)
                   System.err.println("Ezin izan da fitxategia sortu ");
```

3.2 AKTORE

```
package Laborategi1;
import java.util.ArrayList;
import java.util.Iterator;
public class Aktore
        private String izena;
        private String abizena;
        private ArrayList<Pelikula>listaPelikulak = new ArrayList<Pelikula>();
//Eraikitzailea
        public Aktore(String pAbizena,String pIzena)
                this.izena = pIzena;
                this.abizena = pAbizena;
        private Iterator<Pelikula> getIteradorea()
                return this.listaPelikulak.iterator();
//Getter metodoak.
        public ArrayList<Pelikula> getPelikulak()
                return this.listaPelikulak;
        public String getIzena()
                return this.izena;
        public String getAbizena()
                return this.abizena;
//Metodoak
        public void gehituPelikula (Pelikula pPelikula)
                listaPelikulak.add(pPelikula);
```

```
public void kenduPelikula(Pelikula pPelikula)
        listaPelikulak.remove(pPelikula);
public int zenbatPelikula()
        return listaPelikulak.size();
public boolean badagoPelikula(String pPeli)
        Iterator<Pelikula> itr = this.getIteradorea();
         Pelikula p;
         while (itr.hasNext() )
                p = itr.next();
                if (p.getIzena().equals(pPeli))
                        return true;
                        return false;
public ArrayList<Pelikula> itzuliPelikulak()
         return this.listaPelikulak;
public void ikusiPelikulak()
        Pelikula p =null;
         Iterator<Pelikula> itr = this.getIteradorea();
         if(itr.hasNext())
                while(itr.hasNext())
                        p=itr.next();
                       System.out.println(p.getIzena());
```

```
@Override
public boolean equals(Object o)
    Aktore person = (Aktore) o;
    if (o instanceof Aktore && this.getAbizena() == person.getAbizena()&& this.getIzena()==
person.getIzena())
        return true;
    else return false;
```

3.3 LISTAAKTORE

```
package Laborategi1;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
public class listaAktore
    private static listaAktore i;
    private HashMap<String,Aktore> lista = null;
    public static synchronized listaAktore getLista()
        if (listaAktore.i == null)
            listaAktore.i = new listaAktore();
        return i;
    // Eraikitzaile pribatua
    private listaAktore()
        lista = new HashMap<String,Aktore>();
    public void gehituAktorea(Aktore pAktore)
```

```
        this.lista.put(pAktore.getAbizena()+ " "+pAktore.getIzena(), pAktore);
public int listaLuzeera()
        return this.lista.size();
public void clear()
        this.lista.clear();
public Aktore bilatuAktorea(String pAbizena, String pIzena)
        return this.lista.get(pAbizena + " "+pIzena);
public boolean aktoreaDago(String pAbizena, String pIzena)
        return lista.containsKey(pAbizena+ " "+pIzena);
public void ezabatuAktorea(Aktore pAktore)
        this.lista.remove(pAktore.getAbizena()+" "+pAktore.getIzena());
public ArrayList<String> aktoreOrdenatuak()
        ArrayList<String> listaordenatua = new ArrayList<>(lista.keySet());
        return mergeSort(listaordenatua);
```

```
public ArrayList<String> mergeSort(ArrayList<String> osoa)
    ArrayList<String> ezk = new ArrayList<String>();
    ArrayList<String> esku = new ArrayList<String>();
    int erdia;
    if (osoa.size() == 1)  //Hau da, listak elementu bakarra badu.
        return osoa;
     else  //Bestela listan bitan zatituko dugu.
        erdia = osoa.size()/2;
        for (int i=0; i<erdia; i++)
            ezk.add(osoa.get(i));
        for (int i=erdia; i<osoa.size(); i++)
            esku.add(osoa.get(i));
//Alde bakoitzari merge-sort algoritmoa aplikatu modu errekurtsiboan.
    ezk = mergeSort(ezk);
    esku = mergeSort(esku);
// Batu emaitzak.
    merge(ezk, esku, osoa);
    return osoa;
```

```
private void merge(ArrayList<String> ezk, ArrayList<String> esku, ArrayList<String>
osoa)
    int ezkIndex = 0;
    int eskuIndex = 0;
    int osoaIndex = 0;
    //ezk edo esku luzeera duten bitartean, beti zatika joango gara hartzen eta
biak bat egiten.
    while (ezkIndex < ezk.size() && eskuIndex < esku.size())
        if ( (ezk.get(ezkIndex).compareTo(esku.get(eskuIndex))) < 0)
            osoa.set(osoaIndex, ezk.get(ezkIndex));
        ezkIndex++;
        else
            osoa.set(osoaIndex, esku.get(eskuIndex));
            eskuIndex++;
        osoaIndex++;
    ArrayList<String> gelditzenDena;
    int gelditzenDenaIndex;
    if (ezkIndex >= ezk.size())
        // Ezkerrko aldia guztiz erabili da.
        gelditzenDena = esku;
        gelditzenDenaIndex = eskuIndex;
    else
        //Eskuineko aldea guztiz erabili da.
        gelditzenDena = ezk;
        gelditzenDenaIndex = ezkIndex;
    // Kopiatu oraindik amaitu ez den aldea osorik.
```

```
        for (int i=gelditzenDenaIndex; i<gelditzenDena.size(); i++)
            osoa.set(osoaIndex, gelditzenDena.get(i));
            osoaIndex++;
```

3.4 PELIKULA

```
package Laborategi1;
import java.util.ArrayList;
public class Pelikula
    private String izena;
    private ArrayList<Aktore> listaAktoreak = new ArrayList<Aktore>();
    private int dirua;
    public Pelikula(String pIzena, int pDirua)
        this.izena = pIzena;
        this.listaAktoreak =null;
        this.dirua = pDirua;

    public String getIzena()
        return this.izena;

    public ArrayList<Aktore> getListaAktore()
        return this.listaAktoreak;

    public int getDirua()
        return this.dirua;
```

```
public void diruaGehitu(int pDirua)
        this.dirua = this.dirua + pDirua;

public void gehituAktore (Aktore pAktore)
        if(this.listaAktoreak ==null)
                this.listaAktoreak=new ArrayList<Aktore>();
        this.listaAktoreak.add(pAktore);

public void kenduAktore(Aktore pAktore)
        listaAktoreak.remove(pAktore);

public int getAktoreKopurua()
        return this.listaAktoreak.size();
```

3.5 LISTAPELIKULA

```
package Laborategi1;
import java.util.ArrayList;
import java.util.Iterator;
public class listaPelikula
        private static listaPelikula i = null;
        private ArrayList<Pelikula> lista;
        //eraikitzailea
        private listaPelikula()
                this.lista = new ArrayList<Pelikula>();
```

```
public static synchronized listaPelikula getLista()
        if ( listaPelikula.i == null)
                listaPelikula.i = new listaPelikula();
        return i;

public void gehituPelikula(Pelikula pPelikula)
        lista.add(pPelikula);

private Iterator<Pelikula> getIteradorea()
        return this.lista.iterator();

public void kenduPelikula(Pelikula pPelikula)
        lista.remove(pPelikula);

public int listaLuzeera()
        return lista.size();

public void ezabatu()
        lista.clear();
```

```
public boolean badagoPelikula (Pelikula pPelikula)
        Iterator<Pelikula> itr = this.getIteradorea();
        Pelikula p1=null;
        while (itr.hasNext() )
                // System.out.println(a++);
                p1 = itr.next();
                if (p1.getIzena().trim().equals(pPelikula.getIzena().trim()))
                        return true;
        return false;


public Pelikula itZuliPelikula(String pIzena)
        Iterator<Pelikula> itr = this.getIteradorea();
        Pelikula p1=null;
        boolean aurkitua = false;
        while (itr.hasNext() && !aurkitua)
                // System.out.println(a++);
                p1 = itr.next();
                if (p1.getIzena().trim().equals(pIzena.trim()))
                        aurkitua=true;
        if(!aurkitua)
                return null;
        else
                return p1;
```

3.6 AKTORETEST

```
package Laborategi1;
import static org.junit.jupiter.api.Assertions.*;
import java.util.ArrayList;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
class AktoreTest
    Aktore a1,a2,a3;
    Pelikula p1,p2,p3;
    @BeforeEach
    void setUp() throws Exception
        a1 = new Aktore("Sanjuan", "Kerman");
        a2 = new Aktore("Ruiz","Alba");
        a3 = new Aktore("Alonso","Luna");
        p1 = new Pelikula("as",0);
        p2 = new Pelikula("qw",0);
        p3 = new Pelikula("zx",0);
        @AfterEach      void tearDown() throws Exception
        a1 = null;
        a2 = null;
        a3 = null;
        p1 = null;
        p2 = null;
        p3 = null;
```

```
@Test
void testAktore()
    assertNotNull(a1);

@Test
void testGetIzena()
    assertEquals("Kerman", a1.getIzena());

@Test
void testGetAbizena()
    assertEquals("Sanjuan", a1.getAbizena());
    @Test
void testGehituPelikula()
    a1.gehituPelikula(p1);
    assertEquals(1, a1.zenbatPelikula());
    a1.gehituPelikula(p1);
    assertEquals(2, a1.zenbatPelikula());
     //ez dugunez implementatu pelikulak errepikatuta ez egotea, pelikula berdina bi aldiz gehi dezakegu
    a1.gehituPelikula(p2);
    assertEquals(3, a1.zenbatPelikula());
    a1.gehituPelikula(p3);
    assertEquals(4, a1.zenbatPelikula());
    //De este modo comprobamos tambien si "zenbat pelikula" funciona.
```

```
@Test
void testKenduPelikula()
        a1.kenduPelikula(p1);
        assertEquals(0,a1.zenbatPelikula());
        //pelikularik ez badagu eta pelikula bat kentzen saiatzen badugu
        a1.gehituPelikula(p1);
        a1.kenduPelikula(p1);
        assertEquals(0,a1.zenbatPelikula());
        //pelikula bakarra kendu
        a1.gehituPelikula(p1);
        a1.gehituPelikula(p2);
        a1.gehituPelikula(p3);
        a1.kenduPelikula(p1);
        assertEquals(2, a1.zenbatPelikula());
        //lehenengo pelikula kendu
        a1.gehituPelikula(p1);
        a1.kenduPelikula(p2);
        assertEquals(2, a1.zenbatPelikula());
        //"erdiko" pelikula bat kendu
        a1.gehituPelikula(p2);
        a1.kenduPelikula(p3);
        assertEquals(2, a1.zenbatPelikula());
        //azkeneko pelikula kendu
        a1.gehituPelikula(p3);
```

```
        Pelikula p4 = new Pelikula("ezdago",0);
        a1.kenduPelikula(p4);
        assertEquals(3, a1.zenbatPelikula());
        //listan ez dagoen pelikula kendu

@Test
void testBadagoPelikula()
        assertFalse(a1.badagoPelikula(p1.getIzena()));
        //pelikula bat bilatzen dugu baina ez daude pelikularik
        a1.gehituPelikula(p1);
        assertFalse(a1.badagoPelikula(p2.getIzena()));
        //pelikula bad dago baina ez da bilatzen ari garena
        a1.gehituPelikula(p2);
        a1.gehituPelikula(p3);
        assertTrue(a1.badagoPelikula(p1.getIzena()));
        //listaren lehenengo pelikula badago
        assertTrue(a1.badagoPelikula(p3.getIzena()));
        //listaren azkenengo pelikula
        assertFalse(a1.badagoPelikula("ez dago listan"));
```

```
@Test
void testItzuliPelikulak()
        ArrayList<Pelikula> listaP = a1.itzuliPelikulak();
        assertEquals(listaP.size(), 0);
        //Ez badaude pelikularik listaren luzeera 0 izango da.
        a1.gehituPelikula(p1);
        assertNotNull(a1.itzuliPelikulak());
        //Pelikulak egonda, lista ezin daiteke null izan.


@Test
void testIkusiPelikulak()
        //Hemen ezer ez.
```

3.7 LISTAAKTORETEST

```
package Laborategi1;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
class listaAktoreTest
    Aktore a1,a2,a3,a4;
    listaAktore lista,lista2;
    @BeforeEach
    void setUp() throws Exception
        lista = listaAktore.getLista();
        lista2 = listaAktore.getLista();
        a1 = new Aktore("Sanjuan", "Kerman");
        a2 = new Aktore("Ruiz","Alba");
        a3 = new Aktore("Alonso","Luna");
        a4 = new Aktore("Colate","Pacho");

    @AfterEach
    void tearDown() throws Exception
        listaAktore.getLista().clear();
        listaAktore.getLista().clear();
        a1 = null;
        a2 = null;
        a3 = null;
```

```
@Test
void testGetLista()
        assertNotNull(lista);


@Test
void testGehituAktorea()
        lista.gehituAktorea(a1);
        assertEquals(1, lista.listaLuzeera());
        lista.gehituAktorea(a1);
        assertEquals(1, lista.listaLuzeera());
        //ezin dira bi aktore berdin egon
        lista.gehituAktorea(a2);
        lista.gehituAktorea(a3);
        assertEquals(3, lista.listaLuzeera());
        lista.gehituAktorea(a4);
        assertEquals(4, lista.listaLuzeera());
        lista.gehituAktorea(a1);
        lista.gehituAktorea(a2);
        lista.gehituAktorea(a3);
        lista.gehituAktorea(a4);
        assertEquals(4, lista.listaLuzeera());
```

```
@Test
void testEzabatuAktorea()
        lista.ezabatuAktorea(a1);
        assertEquals(0,lista.listaLuzeera());
        lista.gehituAktorea(a1);
        lista.ezabatuAktorea(a1);
        assertEquals(0,lista.listaLuzeera());
        lista.gehituAktorea(a1);
        lista.gehituAktorea(a2);
        lista.ezabatuAktorea(a2);
        assertEquals(1,lista.listaLuzeera());
        lista.gehituAktorea(a2);
        lista.gehituAktorea(a3);
        lista.ezabatuAktorea(a4);
        assertEquals(3,lista.listaLuzeera());
        lista.ezabatuAktorea(a3);
        assertEquals(2,lista.listaLuzeera());

@Test
void testListaLuzeera()
        lista.gehituAktorea(a1);
        assertEquals(1, lista.listaLuzeera());
        lista.gehituAktorea(a2);
        lista.gehituAktorea(a3);
        assertEquals(3, lista.listaLuzeera());
```

```
@Test
void testAktoreaDago()
    assertFalse(lista.aktoreaDago(a1.getAbizena(),a1.getIzena()));
    lista.gehituAktorea(a1);
    assertTrue(lista.aktoreaDago(a1.getAbizena(),a1.getIzena()));
    lista.gehituAktorea(a2);
    lista.gehituAktorea(a3);
    assertTrue(lista.aktoreaDago(a2.getAbizena(),a2.getIzena()));
    assertTrue(lista.aktoreaDago(a3.getAbizena(),a3.getIzena()));
    assertFalse(lista.aktoreaDago(a4.getAbizena(),a4.getIzena()));
    @Test
void testAktoreOrdenatuak()
    //Bi lista sortuko ditugu orden diferentean, gero ordenatu eta berdinak izango beharko litzateke.
    a1 = new Aktore("Kerman", "A");
    a2 = new Aktore("Ander", "B");
    a3 = new Aktore("Josu", "C");
    lista.gehituAktorea(a1);
    lista.gehituAktorea(a2);
    lista.gehituAktorea(a3);
    lista.aktoreOrdenatuak();
    lista2.gehituAktorea(a3);
    lista2.gehituAktorea(a1);
    lista2.gehituAktorea(a2);
```

```
        lista2.aktoreOrdenatuak();
        assertEquals(lista, lista2);
```

3.8 PELIKULATEST

```
package Laborategi1;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
class PelikulaTest
    Aktore a1,a2,a3;
    Pelikula p1,p2,p3;
    @BeforeEach
    void setUp() throws Exception
        a1 = new Aktore("Sanjuan", "Kerman");
        a2 = new Aktore("Ruiz","Alba");
        a3 = new Aktore("Alonso","Luna");
        p1 = new Pelikula("as",0);
        p2 = new Pelikula("qw",0);
        p3 = new Pelikula("zx",0);
```

```
@AfterEach
void tearDown() throws Exception
        a1 = null;
        a2 = null;
        a3 = null;
        p1 = null;
        p2 = null;
        p3 = null;

@Test
void testPelikula()
        assertNotNull(p1);

@Test
void testGetIzena()
        assertEquals("as", p1.getIzena());

@Test
void testGetListaAktore()
        p1.gehituAktore(a1);
        assertNotNull(p1.getListaAktore());
        @Test
void testDiruaGehitu()
        p1.diruaGehitu(15);
        assertEquals(15,p1.getDirua());
```

@Test

void testGehituAktore()

p1.gehituAktore(a1);

assertEquals(1, p1.getAktoreKopurua());

p1.gehituAktore(a1); //ez dauka zentzu handirik, baina egon daiteke bi aktore izen abizen berdinarekin

assertEquals(2, p1.getAktoreKopurua()); //ez dugunez implementatu aktore (pelikula bakoitzaren aktoreak) errepikatuta ez egotea, aktore berdina bi aldiz gehi dezakegu

p1.gehituAktore(a2);

assertEquals(3, p1.getAktoreKopurua());

p1.gehituAktore(a3);

assertEquals(4, p1.getAktoreKopurua());

//De este modo comprobamos tambien si "zenbat pelikula" funciona.

@Test

void testKenduAktore()

p1.gehituAktore(a1);

p1.kenduAktore(a2);

assertEquals(1, p1.getAktoreKopurua()); //aktore bat dago baina ez da kentzen saiatzen duguna

p1.gehituAktore(a3);

p1.gehituAktore(a2);

Aktore a4= new Aktore("Colate","Pacho");

p1.kenduAktore(a4);

assertEquals(3, p1.getAktoreKopurua()); //hiru aktore daude eta beste laugarren bat kentzen saiatzen gara

```
@Test
void testGetAktoreKopurua()
    p1.gehituAktore(a1);
    p1.gehituAktore(a2);
    assertEquals(p1.getAktoreKopurua(),2);
    p1.kenduAktore(a1);
    assertEquals(1, p1.getAktoreKopurua());
```

3.9 LISTAPELIKULATEST

```
package Laborategi1;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
class listaPelikulaTest
    Pelikula p1,p2,p3,p4;
    listaPelikula l1;
    @BeforeEach
    void setUp() throws Exception
        p1 = new Pelikula("qw",0);
        p2 = new Pelikula("as",0);
        p3 = new Pelikula("zx",0);
        p4 = new Pelikula("ezdago",0);
        l1 = listaPelikula.getLista();
```

```
@AfterEach
void tearDown() throws Exception
        p1 = null;
        p2 = null;
        p3 = null;
        p4 = null;
        l1.ezabatu();

@Test
void testGetLista()
        assertNotNull(l1);

@Test
void testGehituPelikula()
        assertEquals(0, l1.listaLuzeera());
        l1.gehituPelikula(p1);
        assertEquals(1, l1.listaLuzeera());
        l1.gehituPelikula(p2);
        l1.gehituPelikula(p3);
        assertEquals(3, l1.listaLuzeera());
        l1.gehituPelikula(p1);
        assertEquals(4, l1.listaLuzeera()); //bi pelikula egon ahal dira izen berdinarekin
```

```
@Test
void testKenduPelikula()
    l1.kenduPelikula(p1);
    assertEquals(0, l1.listaLuzeera()); //ez dago pelikularik eta bad kentzen
dugu
    l1.gehituPelikula(p1);
    l1.kenduPelikula(p1);
    assertEquals(0, l1.listaLuzeera()); //pelikula bakarra dago eta hori kentzen
dugu
    l1.gehituPelikula(p1);
    l1.kenduPelikula(p2);
    assertEquals(1, l1.listaLuzeera()); //pelikula bakarra dago eta beste pelikula
bat kentzen dugu
    l1.gehituPelikula(p2);
    l1.kenduPelikula(p2);
    assertEquals(1, l1.listaLuzeera());
    l1.gehituPelikula(p2);
    l1.gehituPelikula(p3);
    l1.kenduPelikula(p3);
    assertEquals(2, l1.listaLuzeera());
    l1.gehituPelikula(p3);
    l1.kenduPelikula(p4);
    assertEquals(3, l1.listaLuzeera());
```

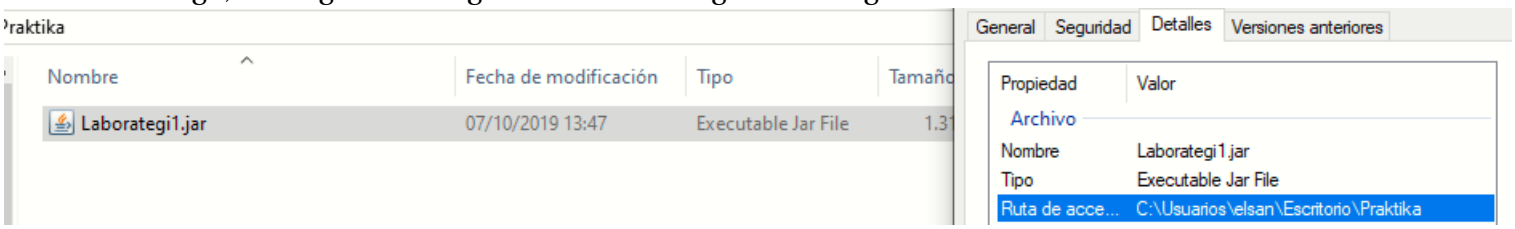
```
@Test
void testListaLuzeera()
        assertEquals(0, l1.listaLuzeera());
        l1.gehituPelikula(p1);
        assertEquals(1, l1.listaLuzeera());
        l1.gehituPelikula(p2);
        l1.gehituPelikula(p3);
        assertEquals(3, l1.listaLuzeera());


@Test
void testEzabatu()
        l1.gehituPelikula(p1);
        l1.gehituPelikula(p2);
        l1.gehituPelikula(p3);
        assertEquals(3, l1.listaLuzeera());
        l1.ezabatu();
        assertEquals(0, l1.listaLuzeera());
```

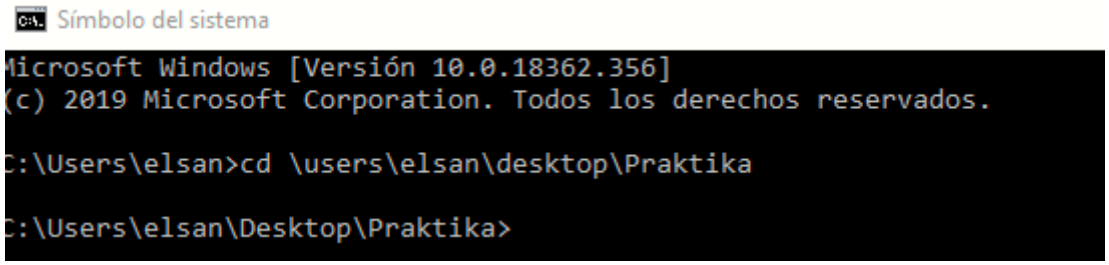
```
@Test
void testBadagoPelikula()
    assertFalse(l1.badagoPelikula(p1));
    l1.gehituPelikula(p1);
    assertTrue(l1.badagoPelikula(p1));
    l1.gehituPelikula(p2);
    l1.gehituPelikula(p3);
    assertTrue(l1.badagoPelikula(p2));
    assertTrue(l1.badagoPelikula(p3));
    assertFalse(l1.badagoPelikula(p4));
@Test
void testItZuliPelikula()
    assertNull(l1.itZuliPelikula("asdas")); //Hutsa denean ez du ezer aurkituko
    l1.gehituPelikula(p1);
    l1.gehituPelikula(p2);
    l1.gehituPelikula(p3);
    assertNotNull(l1.itZuliPelikula(p1.getIzena())); //Izen berarekin badago
pelikualaren bat, hori itzuliko du (Beti lehenengoa).
    assertNotNull(l1.itZuliPelikula(p2.getIzena()));//listaren erdian badago bi-
latzen dugun pelikula
    assertNotNull(l1.itZuliPelikula(p3.getIzena()));//listaren amaiera badago
bilatzen dugun pelikula
    assertNull(l1.itZuliPelikula(p4.getIzena()));
    //pelikula listan ez badago
```

Progama irekitzen

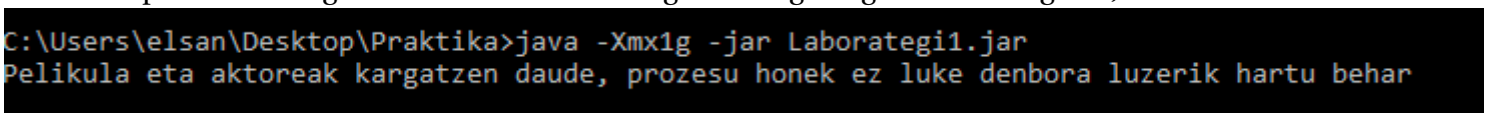
Lehenengo eta behin CMD kontsolan gure .jar fitxategia dagoen helbidea idatzi behar dugu, beraz gure fitxategiaren helbidea begiratuko dugu:



Eta ondoren CMD-n helbide hori aukeratuko dugu:



Ondoren fitxategia irekitzeko esango diogu, gure kasuan -Xmx1g komandoa erabili dugu programari 1GB-eko RAM memoria emateko (ez da beharrezkoa baina aktore eta pelikula asko gehitzen badira memoria geroz eta gehiago behar izango da):



Hori eginda programa erabiltzeko prest egongo gara.

Gure programak behar beste memoria ez badu honako errorea lortuko dugu, eta lehen esan dugun komandori esker, memoria gehiago emango beharko diogu.

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.util.Arrays.copyOfRange(Unknown Source)
        at java.lang.String.<init>(Unknown Source)
        at java.nio.HeapCharBuffer.toString(Unknown Source)
        at java.nio.CharBuffer.toString(Unknown Source)
        at java.util.regex.Matcher.toMatchResult(Unknown Source)
        at java.util.Scanner.match(Unknown Source)
        at java.util.Scanner.hasNextLine(Unknown Source)
        at Laborategi1.Scannera.listakKargatu(Scannera.java:121)
        at Laborategi1.Scannera.main(Scannera.java:20)
```

Amaiera

5.1 KONKLUSIOAK

Laborategi hau egin ostean, hainbat konklusiotara iritzi gara. Alde batetik, lehenengo laborategia izanda, hasieran pixkat kostatu zitzaigun, baina azkenean erritmoa lortu dugu. Beste alde batetik, datu-egitura berriak erabili ditugu, "bizitza errealeko" arazo bat konpontzeko, eta honek, begiak ireki dizkigu hurrengo laborategietan ideia berriak inplementatzeko eta beti arazoei beste buelta bat emateko. Azkenik, konklusio moduan, laborategi honen zailtasuna listen kudeaketan (elementuak gehitu eta kendu) zatzan, hau lortuta, bestelako arazoen soluzioa eta inplementazioa azkoz errezagoa izan da.

5.2 BIBLIOGRAFIA

- StackOverflow

- W3Schools

- CodeAcademy

- GeekForGeeks

- Oracle

- edu4java youtube kanala - Youtubeko bideoa.