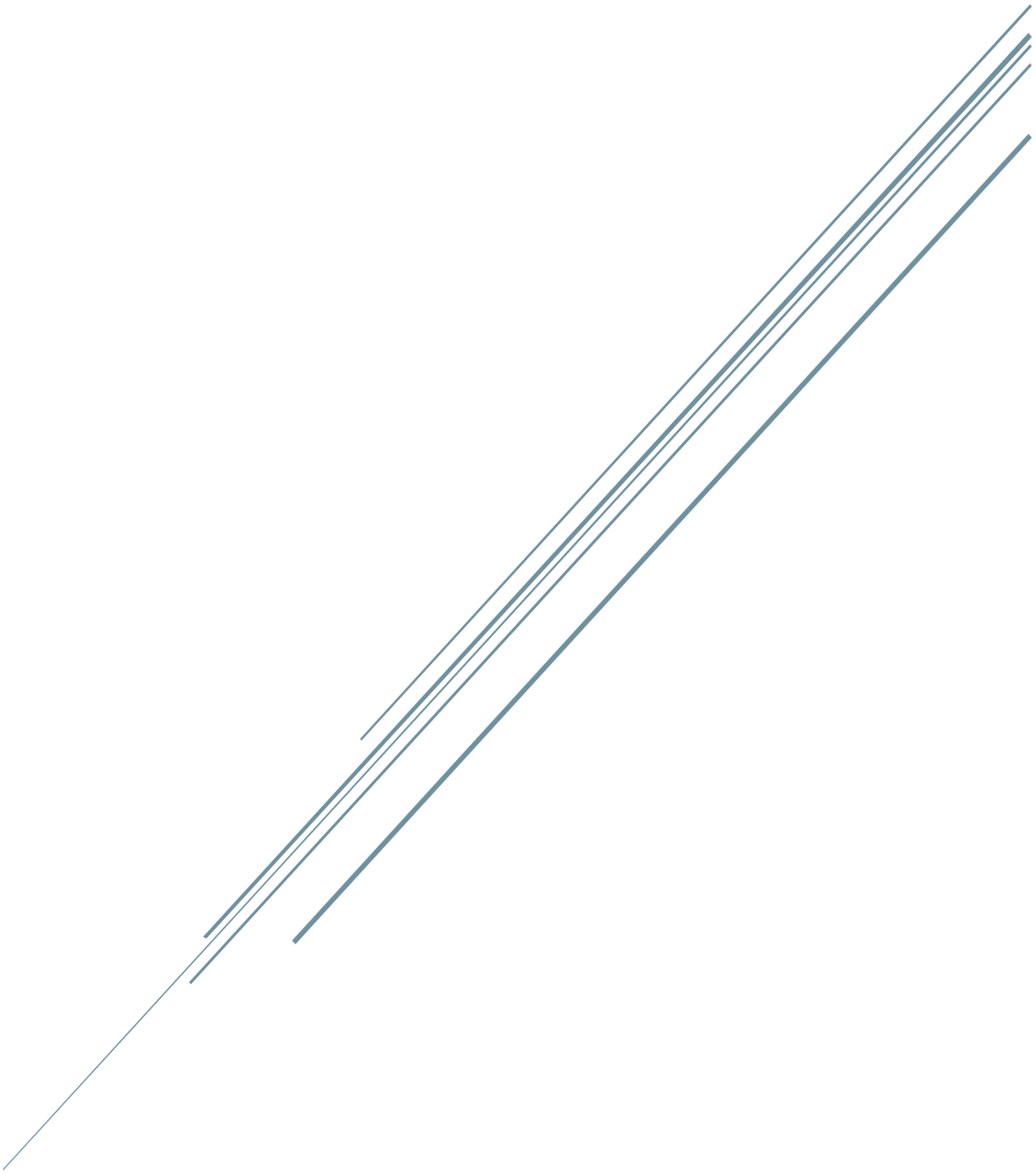
DATU EGITURAK ETA ALGORITMOAK: 2.PRAKTIKA

Egitura estekatuen laborategia



Unai Arrizabalaga / Leire Viseras

AURKIBIDEA

SARRERA

Oraingo laborategian egitura estekatuak erabiliz inplementatu ditugu aplikazioan erabiltzen diren metodo batzuk.

Aurreko laborategian eraikitako klaseak eta metodoak berrerabili ditugu, baina klase batzuetan, metodo eta datu egitura berriak gehituz.

Egindako aldaketa hauekin ikusiko dugu zein den berregin ditugun metodo berri hauen kostuak eta datu egitura honen eraginkortasuna ere bai.

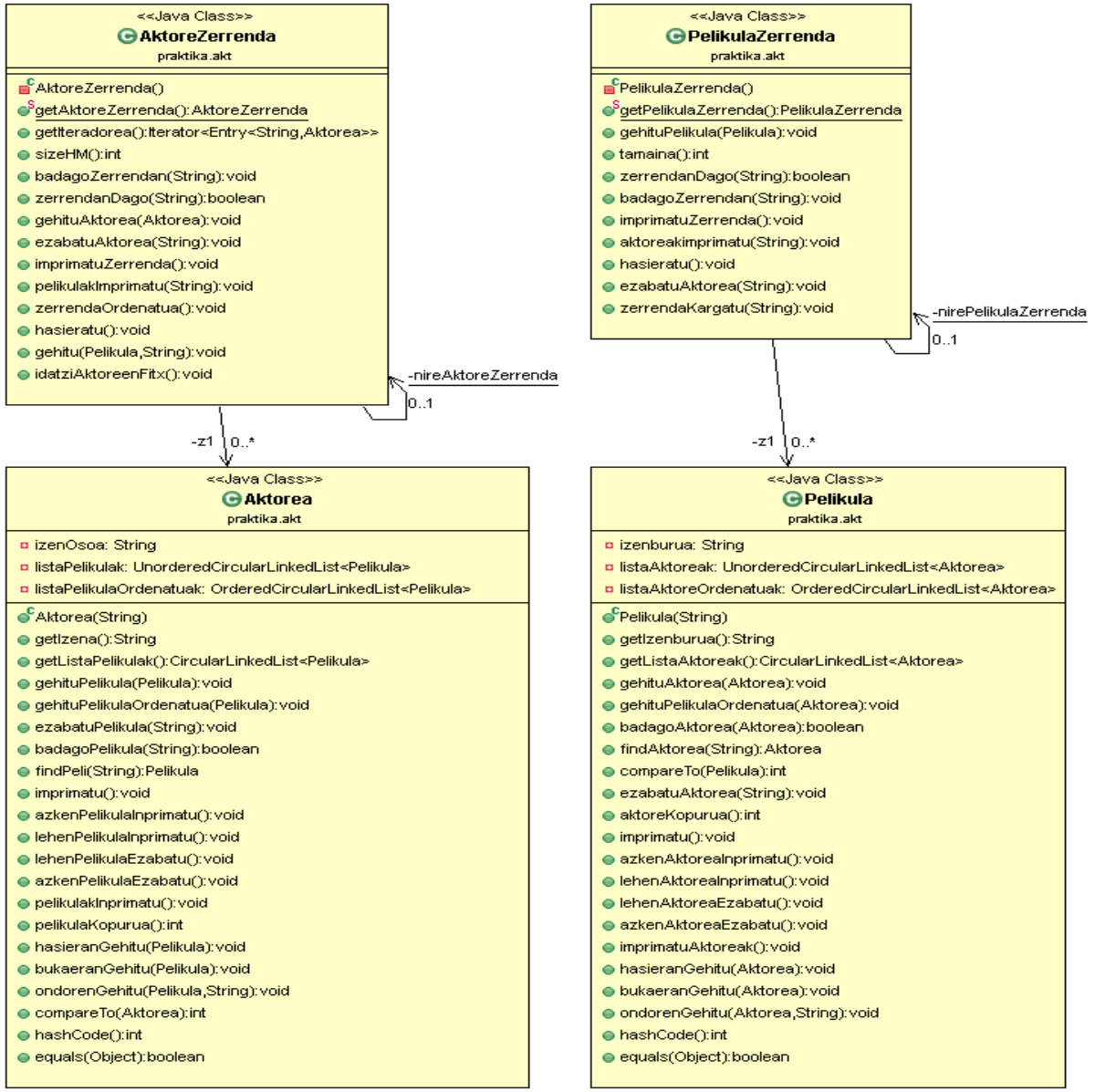
KLASEEN DISEINUA

Datuen kudeaketa eta eskatzen dizkiguten eragiketak aurrera eramateko, klase batzuk inplementatu ditugu. Atal honetan haien funtzionalitatea azalduko da.

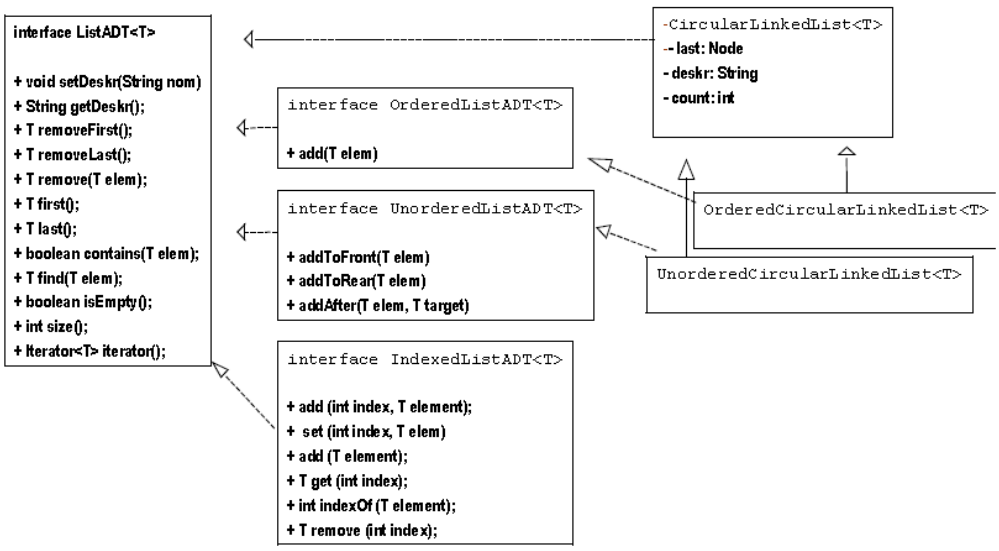
Proiektuan dauden “atal” desberdinak bereizteko atal bakoitza package batean sartu dugu. Gauzak argiago ikusten direlako pentsatzen dugulako horrela egin dugu inplementazioa baita ere.

Alde batetik aurreko praktikan inplementatutako klaseak ditugu (Programa nagusi bat, AktoreZerrenda, Aktorea, etab… , bestetik zerrenda estekatuekin erlazionatuta dauden klase guztiak (irakasleak eman zizkigun klaseak) eta amaitzeko proba programak; proba programa hauetan bi motetako LinkedList –en frogapena burutzen da Aktorea eta Pelikula klaseekin.

Package 1 (praktika.akt):



Package 2 (praktika2.eda):



IMPLEMENTATUTAKO KLASEAK

Praktika honetan aurreko praktikan inplementatu genituen klaseak berrerabili ditugu baina Aktorea eta Pelikula klaseek erabiltzen zituzten datu egiturak haien barruan zerrenda bat edukitzeko, orain aldatu egin ditugu.

- **MainP.java:** Klase nagusi gisa erabili duguna. Klase honetan probak kasuak ipini ditugu aplikazioa ondo dabilen edo ez ikusteko (aplikazio barruan inplementatutako metodoak).
- **AktoreZerrenda.java:** Aktoreen datuak gordetzeko erabili dugu. Zerrenda horrekin egin daitezkeen eragiketa nagusiak eta enuntziatuan eskatzen zirenak inplementatzeko erabilia.
- **Aktorea.java:** Aktore baten datuak gordetzeko erabilia, hau da, bere izena (*Abizena, Izena* formatuan) eta parte hartu dituen pelikulen zerrenda bat duen klasea da hau.
- **PelikulaZerrenda.java:** Pelikulen zerrenda bat da, AktoreZerrenda klasean esan dugun moduan, zerrenda honekin enuntziatuan eskatutako eragiketak eta zerrenda batekin egiten diren eragiketa nagusiak inplementatu dira.
- **Pelikula.java:** Pelikula bateko datuak gordetzeko erabilia. Pelikula batek haren izenburua eta pelikula horretan parte hartutako aktoreen zerrenda bat izango du.

Zerrenda estekatuak erabili ditugu ariketa hau burutzeko irakasleak emandako datu egitura batzuekin batera. Gure kasuan, ariketa ebazteko bi zerrenda estekatu mota erabili ditugu, hori bai, biak CircularLinkedList batetik hartutako metodoak inplementatzen dituzte. Bi zerrenda horiek UnorderedCircularLinkedList (ordenatu gabeko elementuak dituen zerrenda zirkularra) eta OrderedCircularLinkedList (elementuak ordena bat jarraituz dituen zerrenda zirkularra) dira.

Zergatik inplementatu ditugu klase biak? Haien artean dagoen desberdintasuna ikusteko. Hau da, ordenatu gabeko elementuak dituen zerrendak eta ordena bat jarraitzen duten elementuak dituen zerrendak burutzen dituzten eragiketek duten kostuen desberdintasuna ikusteko.

Hona hemen praktika honetarako behar izan ditugun klase berriak (irakasleak emandako klaseak):

- **ListADT** (interfazea)
- **IndexedListDT** (interfazea)
- **OrderedListADT**(interfazea)
- **UnorderedListADT**(interfazea)
- **CircularLinkedList**: ListADT interfazea inplementatzen du, beraz, haren metodoak dauzka. Klase honetan zerrenda estekatu batek egiten dituen hainbat metodo desberdin inplementatzen dira.
- **OrderedCircularLinkedList**: CircularLinkedList –ek dituen metodoak inplementatzen ditu, baina kasu honetan elementuak ordenaturik egongo dira. Metodo horietaz aparte elementua dagokion posizioan gehitzen duten beste metodo berri bat du.
- **UnorderedCircularLinkedList**: CircularLinkedList –ek dituen metodoak inplementatzen ditu. Horietaz aparte beste metodo batzuk inplementatuta daude klase honen barruan: bukaeran gehitu, hasieran gehitu, etab,…

KLASEEN ESPEZIFIKAZIOA

Aktore_Peli_ProgramaNagusia.java

Lehen esan bezala, klase hau bakarrik probak egikaritzeko erabili dugu. Beraz, klaseak duen metodo bakarrean, proba kasu desberdinak ipini ditugu enuntziatuan eskatzen ziren eragiketa bakoitzerako. Aplikazioa grafikoagoa egiteko klase nagusian menu txiki bat margotu dugu, aplikazioak dituen aukera desberdinak hobeto ikusteko eta erabiltzailearekiko interaktiboagoa egiteko.

Klase nagusi honek dituen metodo pribatuak menu -an dauden aukerei egiten die erreferentzia.

```
public class AktorePeli_ProgramaNagusia {

    private static Stopwatch timer;
    public static void main(String[] args) {
    private static void paintMenu(){
    private static void aukeraHartu(int aukera){
    private static void badagoZerrendan(){
    private static void gehituAktorea(){
    private static void aktorePelikulak(){
    private static void pelikulaAktoreak(){
    private static void ezabatuAktorea(){
    private static void idatziFitxategia(){
    private static void zerrendaOrdenatua(){
    private static void hasieratuZerrendak(){
}
```

PelikulaZerrenda.java

Klase honek bi atributu izango ditu. Lehenengoa PelikulaZerrenda klase bera izango da eta bigarrena pelikula zerrenda hori gordeko dituen HashMap -a.

Beste klaseekin gertatu den moduan, klase hau bakarrik behin hasieratuko da, eta zerrendan gordetzeko HashMap egitura erabili dugu egitura horrekin eragiketak egiterako orduan kostua ia konstantea delako.

Inplementatutako metodoak:

- **Eraikitzailea:** Pribatua izango da, hemen zerrenda hasieratuko dugu.
- **getPelikulaZerrenda():** PelikulaZerrenda klasea hasieratu gabe egon ezkero, metodo honekin hasieratuko dugu.
- **gehituPelikula():** Pelikula bat emanda pelikula hori zerrendan gehituko du.
- **tamaina():** Zerrendak duen pelikula kopurua esango digu.
- **zerrendanDago():** Pelikula baten izenburua emanda zerrendan dagoen edo ez esango digu boolean den aldagai bat erabilita.
- **badagoZerrendan():** Pelikula baten izenburua emanda zerrendan dagoen edo ez esango digu mezu bat pantailaratuz.
- **imprimatuZerrenda():** Zerrenda osoa pantailaratuko du.
- **aktoreakImprimatu():** Pelikula baten izenburua emanda pelikula horretan parte hartu duten aktoreen zerrenda inprimatu/pantailaratuko du.

- **hasieratu():** klase osoa hasieratu nahi badugu.
- **zerrendaKargatu():**.txt fitxategiaren irakurle gisa erabiliko den metodoa. () Metodoak path -a izango du parametro gisa, gure kasuan fitxategia dagoen path -a proiektuan barruan egongo da. Metodoak fitxategian dagoen informazioa (Pelikulen izenburua eta aktoreen izena) sailkatuko du eta datuetan bihurtuko du, hau da, PelikulaZerrenda eta AktoreZerrenda klaseak lortuz.

```
public class PelikulaZerrenda {

    //private ArrayList<Aktorea> zerrenda;

    private static PelikulaZerrenda nirePelikulaZerrenda= null;
    private Map<String,Pelikula> z1;

    private PelikulaZerrenda() {..

    public static synchronized PelikulaZerrenda getPelikulaZerrenda() ..
    //pelikula berria zerrendan gehituko du
    public void gehituPelikula(Pelikula p){..
    //pelikula zerrendaren tamaina
    public int tamaina() {..
    //pelikula baten izenburua emanda zerrendan dagoen edo ez esango digu
    public boolean zerrendanDago(String izenburua){..

    //pelikula baten bilaketa
    public void badagoZerrendan(String izenburua){..
    //pelikula zerrenda imprimatu
    public void imprimatuZerrenda(){..
    //pelikula batek dituen aktoreen zerrenda imprimatuko du
    public void aktoreakimprimatu(String izenburua){..
    public void hasieratu() {..
    public void ezabatuAktorea(String izenOsoa){..
    //fitxategitik datuak atera
        public void zerrendaKargatu(String nomF){..
}
```

AktoreZerrenda.java

Bi atributu izango ditu klase honek, behin bakarrik hasieratuko dugu; aktoreen zerrenda osoa behin bakarrik osorik bete behar delako, beraz, klasea bera (AktoreZerrenda) atributu estatiko gisa ipini dugu. Aktoreen datuak gordetzeko aukeratu dugun datu egitura HashMap bat izan da, honen gainean egiten diren eragiketak kostu ia konstantea bait dute.

AktoreZerrenda klaseak dituen metodoak:

- **Eraikitzailea:** kasu honetan pribatua izango da, klase hau behin bakarrik hasieratuko dugulako.
- **getAktoreZerrenda():** AktoreZerrenda hasieratu gabe egotekotan, metodo honekin hasieratu ahal izango da.
- **getIteradorea():** HashMap egituratik zehar ibiltzeko metodoa.
- **sizeHM():** metodo honek zerrendan dagoen aktore kopurua bueltatuko digu.
- **badagoZerrendan():** Aktore baten izen-abizena emanda zerrendan dagoen edo ez pantailaratuko digu.
- **zerrendanDago():** Aktore baten izen-abizena emanda zerrendan dagoen edo ez esango digu boolean den aldagai bat erabilita.
- **gehituAktorea():** Aktore objektua aldagaia pasatuz, aktorea zerrendan gehituko du. Aktore hori lehendik egotekotan mezu bat agertuko zaigu pantailan esanez aktore hori zerrendan zegoela jada.
- **ezabatuAktorea():** Aktore baten izen-abizena emanda, eta zerrendan egon ezkero, zerrendatik ezabatuko du; bestela mezu bat pantailaratuko da.
- **imprimatuZerrenda():** Aktoreen zerrenda inprimatuko da pantailan haien izen-abizenarekin.
- **pelikulakImprimatu():** Aktore baten izena emanda, aktore hori zerrendan zehar bilatuko da. Aktorea egotekotan, parte hartu dituen pelikulen zerrenda bat inprimatuko da, ez egotekotan, aldiz, mezu bat pantailaratuko da.
- **zerrendaOrdenatua():** Zerrenda abizenen arabera ordenatuta agertuko da pantailan.
- **gehitu():** Pasatzen diogun aktorearen izenarekin objektua lortu, eta bere zerrendan pelikula bat gehitzen dio. Beti aktorea listan jada dagoenean deituko zaio metodo honi.
- **hasieratu():** klase osoa hasieratu nahi badugu.
- **idatziAktoreenFitx():**testu fitxategi bat egiteko erabili dugu. metodo honekin aktoreen zerrenda duen .txt fitxategi bat idatziko da.

```
public class AktoreZerrenda {


    private Map<String,Aktorea> z1;
    private static AktoreZerrenda nireAktoreZerrenda= new AktoreZerrenda();

    private AktoreZerrenda() {

    public static synchronized AktoreZerrenda getAktoreZerrenda() {

    public Iterator<Entry<String, Aktorea>> getIteradorea(){
    //HashMap egitura
    public int sizeHM(){
    public void badagoZerrendan(String izenOsoa){
    public boolean zerrendanDago(String izenOsoa){
    public void gehituAktorea(Aktorea a){
    //aktore baten ezabaketa
    public void ezabatuAktorea(String izenOsoa){
    //aktoreen zerrenda inprimatu
    public void imprimatuZerrenda(){
    //aktore baten pelikula zerrenda pantailaratuko du
    public void pelikulakImprimatu(String izenOsoa){

    public void zerrendaOrdenatua(){
    public void hasieratu() {
    public void gehitu(Pelikula p, String izenOsoa){
    //aktoreen zerrenda fitxategi batean gorde
    public void idatziAktoreenFitx(){

}
```

Aktorea.java

Aktorea -k bi atributu izango ditu: lehenengoa izena gordetzeko eta bigarrena parte hartu duen pelikulen zerrenda bat izango da.

Pelikulen zerrenda kudeatzeko, kasu honetan, ArrayList egitura erabili dugu. Aktore batek ez dituelako 1000 pelikula izango eta zerrenda honekin egingo diren eragiketak ez dutelako denbora handirik hartuko.

Inplementatutako metodoak:

- **Eraikitzailea:** aktore berri bat sortzerakoan metodo hau erabilita hasieratuko da.
- **getIzena():** Aktore baten izena bueltatuko digu.
- **gehituPelikula():** Pelikula objektua emanda, aktore baten pelikulen zerrendan gehituko da pelikula hori.
- **pelikulakInprimatu():** Aktore horren pelikulak pantailaratuko dituen metodoa.
- **pelikulaKopurua():** Aktore horrek zenbat pelikuletan parte hartu duen esango digu.

```
public class Aktorea {
    //Atributuak
        private String izenOsoa;
        private UnorderedCircularLinkedList<Pelikula> listaPelikulak;
        private OrderedCircularLinkedList<Pelikula> listaPelikulaOrdenatuak;
        //Eraikitzailea

        public Aktorea (String izenOsoa){

        public String getIzena() {
        public CircularLinkedList<Pelikula> getListaPelikulak(){
        public void gehituPelikula(Pelikula peli){
        public void gehituPelikulaOrdenatua(Pelikula peli){
        public void ezabatuPelikula(String izenburua){
        public boolean badagoPelikula(String izenburua){
        public Pelikula findPeli(String izenburua){
        public void imprimatu(){
        public void azkenPelikulaInprimatu(){
        public void lehenPelikulaInprimatu(){
        public void lehenPelikulaEzabatu(){
        public void azkenPelikulaEzabatu(){
        public void pelikulakInprimatu() {

        public int pelikulaKopurua() {
        public void hasieranGehitu(Pelikula p){
        public void bukaeranGehitu(Pelikula p){
        public void ondorenGehitu(Pelikula p, String izen){
        public int compareTo(Aktorea akt) {

        public int hashCode() {
```

Pelikula.java

Pelikula klaseak bi atributu izango ditu. Alde batetik pelikularen izenburua eta bestetik pelikula horretan parte hartu duten aktoreen zerrenda bat.

Kasu honetan, eta Aktorea klasean egin dugun moduan, pelikulen zerrenda gordetzeko ArrayList egitura erabili dugu; kudeatu behar diren aktoreen kopurua ez delako oso altua izango.

Inplementatutako metodoak:

- **Eraikitzailea:** pelikula berri bat sortzerakoan metodo hau erabilita hasieratuko da.
- **getIzenburua():** Pelikula horren izenburua lortzeko.
- **gehituAktorea():** Parte hartu duten aktoreen zerrendan aktore berri bat gehitzeko.
- **aktoreKopurua():** Parte hartu duten aktoreen kopurua emango digun metodoa.

```
public class Pelikula {
    //Atributuak

        private String izenburua;
        private UnorderedCircularLinkedList<Aktorea> listaAktoreak ;
        private OrderedCircularLinkedList<Aktorea> listaAktoreOrdenatuak;

        public Pelikula(String izenburua){..

        public String getIzenburua() {..
        public CircularLinkedList<Aktorea> getListaAktoreak(){..
        public void gehituAktorea(Aktorea akt){..
        public void gehituPelikulaOrdenatua(Aktorea akt){..
        public boolean badagoAktorea(Aktorea a){..
        public Aktorea findAktorea(String izena){..
        public int compareTo(Pelikula peli) {..
        public void ezabatuAktorea(String izenOsoa){..
        public int aktoreKopurua() {..
        public void imprimatu(){..
        public void azkenAktoreaInprimatu(){..
        public void lehenAktoreaInprimatu(){..
        public void lehenAktoreaEzabatu(){..
        public void azkenAktoreaEzabatu(){..
        public void imprimatuAktoreak(){..
        public void hasieranGehitu(Aktorea p){..
        public void bukaeranGehitu(Aktorea p){..
        public void ondorenGehitu(Aktorea p, String izen){..
        public int hashCode() {..

        public boolean equals(Object obj) {..
}
```

Stopwatch.java

Eragiketen exekuzio denbora neurtzeko erabili dugun klasea.

- **Eraikitzailea:** Stopwatch hasieratzeko metodoa.
- **elapsedTime():** klasea hasieratu ondoren metodo honi deituko diogu eta eragiketa horretan eman duen denbora bueltatuko digu.

```
package eda.praktika1;

public class Stopwatch {

    private final long start;

   /** Create a stopwatch object. */
    public Stopwatch() {

   /**
     * Return elapsed time (in seconds) since this object was created.
     */
    public double elapsedTime() {
}
```

Aktore eta Pelikula klaseek dituzten metodo berriak zerrenda estekatuek inplementatzen dituzten metodoak dira, beraz, metodo berri bakoitzak zer egiten duen zerrenda estekatuen metodoak azaltzerakoan azalduko da.

Praktikaren atal berria egiteko erabili dugun datu egitura berria zerrenda estekatuak izan dira. Gure kasuan irakasleak pasatutako klase batzuk izan dira. Klase horiek guztiak zerrenda estekatu zirkularra bat inplementatzen dute. Interfaze orokor batetik metodoak heredatuko dituzte beste interfaze batzuk, baina azken finean, zerrenda estekatuarekin eragiketak egiteko metodoak inplementatuta dituzten klaseak hiru dira: CircularLinkedList, UnorderedCircularLinkedList eta OrderedCircularLinkedList.

CircularLinkedList.java

```
public class CircularLinkedList<T> implements ListADT<T> {
    // Atributuak
    protected Node<T> last; // azkenaren erreferentzia
    protected String deskr;  // deskribapena
    protected int count;
    public CircularLinkedList() {
    public Node<T> getLast(){
    public void setDeskr(String ize) {

    public String getDeskr() {

     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da
    public T removeFirst() {

     * Kostua O(n)izango da zerrenda osoa zeharkatu behar delako
    public T removeLast() {

     * Kostua O(n)izango da zerrenda zeharkatu behar delako, kasurik txarrenean zerrenda osoa zeharkatuko da
    public T remove(T elem) {
     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da
    public T first() {
     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da
    public T last() {
     * Kostua O(n)izango da zerrenda zeharkatu behar delako, kasurik txarrenean zerrenda osoa zeharkatuko da
    public boolean contains(T elem) {
     * Kostua O(n)izango da zerrenda zeharkatu behar delako, kasurik txarrenean zerrenda osoa zeharkatuko da
    public T find(T elem) {
     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da
    public boolean isEmpty()
     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da
    public int size()
```

```java
/** Return an iterator to the stack that iterates through the items . */
   public Iterator<T> iterator() { return new ListIterator(); }

   // an iterator, doesn't implement remove() since it's optional
   private class ListIterator implements Iterator<T> { ..


    public void imprimatu(){..

    public void adabegiakInprimatu() {..


    public String toString() {..
```

UnorderedCircularinkedList.java

```
public class UnorderedCircularLinkedList<T> extends CircularLinkedList<T> implements UnorderedListADT<T> {
     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da.
    public void addToFront(T elem) {
     * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. da.
    public void addToRear(T elem) {

     * Kostua konstantea O(n)izango da zerrenda osoa zeharkatu behar delako
    public void addAfter(T elem, T target) {

}
```

- **addToFront():** zerrendaren aurrekaldean gehitu elementua.
- **addToRear():** zerrendaren bukaeran gehitu elementua.
- **addAfter():** zerrendan dagoen elementu baten ondoren gehitu elementu berri bat.

OrderedCircularLinkedList.java

```
public class OrderedCircularLinkedList<T> extends CircularLinkedList<T> implements OrderedListADT<T> {

    public void add(T elem){


}
```

- **add():** elementu berria dagokion lekuan gehitu, hau da, ordena jarraituz.

CircularLinkedList metodoak:

- **removeFirst():** lehenengo elementua ezabatu.
- **removeLast():** azken elementua ezabatu.
- **remove():** zerrendako elementu bat ezabatu.
- **first():** zerrendako lehenengo elementua eman.
- **last():** zerrendako azken elementua eman.
- **contains():** bilatu zerrendan elementu bat.
- **find():** bilatu zerrendan elementu bat, zerrendan badago elementu hori bere informazioa bueltatu.
- **isEmpty():** zerrenda hutsik dagoen edo ez esan.
- **size():** zerrendan dagoen elementu kopurua bueltatu.

OINARRIZKO DATU EGITUREN DESKRIBAPENA

Aktoreak eta pelikulak kudeatuko dituen aplikazioan aldaketa batzuk egin ditugu, datu kopurua oso handia denez, aurreko praktikan erabilitako datu egitura batzuk errepikatu ditugu baina orain klase batzuen barruan (Aktore eta Pelikula) zeuden datuak kudeatzeko beste egitura batzuk erabili ditugu; Zerrenda estekatuak.

Alde batetik zerrenda ez ordenatu batekin egikaritu ditugu metodoak eta bestetik, bien arteko aldea ikusteko, zerrenda ordenatu batekin lan egin dugu. Bi zerrenda estekatuak zirkularrak dira.

Aurreko praktikan erabilitako datu egitura esanguratsuena:

> Aktoreen zerrenda eta pelikulen zerrenden gainean eragiketa desberdinak egiten dira (bilaketak, ezabaketak, elementu berrien gehitzea, zerrenda ordenatzea, etab…), eragiketa hauek modu ia konstantean egiteko **HashMap** egitura aukeratu dugu. Map horien "key" bezala aktoreen Abizena, Izena eta pelikulen Izenburua erabiliz.

Aplikazioak duen aukera bakoitza egikaritu ondoren, eragiketa horrekin emandako denbora pantailaratuko da.

Hona hemen, aplikazioak emandako denboraren adibide batzuk:

- Datuak kargatzen emandako denbora:

```
Datuak kargatuta! :), emandako denbora --> 0.147
```

- Aktore baten bilaketan emandako denbora:

```
1 - Aktore baten bilaketa
```

```
Sartu aktorearen Abizen, Izena
Sipke, Jeffrey
Sipke, Jeffrey aktorea badago zerrendan! :)
Sipke, Jeffrey aktorearen bilaketan emandako denbora --> 0.0
```

- Aktoreen zerrenda ordenatua lortzeko emandako denbora:

  …

```
Ziegler, Daniela
Zielcke, Marie
Zieminski, Derek
Zieser, Petra
Zimmer, Helene
Zimmerman, Phil (II)
Ziovas, Giorgos
Zischler, Hanns
Zmolek, Jill
Zoates, Toby
Zocco, Patrick
Zouroudis, Taso
Zublena, Zack
Zuckerman, Josh (I)
Zukowski, Kelsey
Zwinz, Marc
d'Arcy James, Brian
de Penguern, Artus
de la Campa, Veronica
de la Huerta, Paz
de la Torre, Antonio (I)
du Mont, Sky
von Dobsch?tz, Ulrich
von Krosigk, Sidonie
von Manteuffel, Felix
von Maydell, Sabine
Aktoreen zerrenda ordenatzeko emandako denbora --> 0.119
```

Erabilitako zerrenda estekatuak:

Azpian erakutsiko ditugu metodo esanguratsuenak eta haien exekuzio denborak.

UnorderedCircularLinkedList: Ordena gabeko zerrenda estekatu zirkularra.

- Datuak kargatzeko emandako denbora:

```
Datuak kargatzen...
Datuak kargatuta! :), emandako denbora --> 0.187
```

- Pelikula baten aktoreak inprimatzeko emandako denbora:

```
4
Sartu pelikularen izenburua
Hollywood Dreams
Kirkland, Sally (I)
Leo, Melissa (I)
Verafield, Tania (I)
Dunning, Douglas
Black, Karen (I)
Frederick, Tanna
Kramer, Kat
Norman, Zack
Roberts, Eric (I)
Jaglom, Sabrina
Feeney, F.X.
Jaglom, Simon Orson
Proctor, Phil
Proval, David
Bess, Mariah
Cassel, Seymour
de la Huerta, Paz
Evans, Alice (I)
Leontaritis, Alexander
Simons, Keaton
Kirk, Justin (I)
McClellan, Seth
Katzman, Gerry
Hunter, Alice
Dorr, Sabi
Noflin, Julius
Wit, Nadia
Astor, Tom (I)
Laurence, Ashley
Pohlson, Matthew
Baitz, Jon Robin
Cha, Essie
Smith, Paul (XX)
Kolarich, Kim
Robertson, Erick
Rahn, Michelle
Hale, Trevor (I)
Roske, Brent
Calhoun, Karl
Black, Celine (I)
Cwick, Tracy
Matson, Kathleen
Mukherjee, Tiarra
Naveau, Alex
Panagopoulou, Eleutheria
Pietromonaco, Christa
Simon, Rima
Solomou, Eleni
Jaglom, Henry
Jeffress, Jason
Karalekas, Nikos
Lewis, Don (XV)
Mae, Philip
Markoulakis, John
Hollywood Dreams pelikularen aktoreak inprimatzeko emandako denbora --> 0.013
```

- Aktore baten pelikulak inprimatzeko emandako denbora:

```
3
Sartu aktorearen Abizen, Izena
Duckworth, Barbara
Light: The Father Jay Samonie Documentary
LiNght: The Father Jay Samonie Documentary
Duckworth, Barbara aktorearen pelikulak inprimatzen emandako denbora --> 0.0
```

Proba programa batzuk egin ditugu Aktore eta Pelikula elementuekin. Horrela errazago ikusiko direlako zerrenda estekatuekin egiten diren eragiketak; elementua gehitu, ezabatu, bilatu, etab…

```
pelikula bat gehitzen emandako denbora --> 0.015
 Lista ..............

Bridget Jones I
Bridget Jones II
Bridget Jones III
Harry Potter I
Harry Potter II
Harry Potter III
Harry Potter IV
Harry Potter V
Harry Potter VI
Harry Potter VII
Zerrenda inprimatzen emandako denbora --> 0.015

Colin Firth aktoreak duen pelikula kopurua
10

ELEMENTU BAT LISTAN DAGOEN EDO EZ KONPROBATU
Ahora me ves zerrendan dago? false
Ahora me ves pelikula bilatzen emandako denbora --> 0.015
Ahora me ves zerrendan dago? null
Harry Potter III zerrendan dago? true
Harry Potter III pelikula bilatzen emandako denbora --> 0.015
Harry Potter III zerrendan dago? praktika.akt.Pelikula@e57e1894

 Azken elementua
 ==================================
Harry Potter VII
 ==================================

 Lehenengo elementua:
 ==================================
Bridget Jones I
 ==================================

LISTAKO LEHENENGO ELEMENTUA EZABATU
Lehenengo pelikula ezabatzen emandako denbora --> 0.015
Bridget Jones I zerrendan dago? false
Lista berria lehenengo elementua ezabatuta... ==================================Bridget Jones II
Bridget Jones III
Harry Potter I
Harry Potter II
Harry Potter III
Harry Potter IV
Harry Potter V
Harry Potter VI
Harry Potter VII
 ==================================
LISTAKO AZKEN ELEMENTUA EZABATU
Azken pelikula ezabatzen emandako denbora --> 0.015
Harry Potter VII zerrendan dago? false
Lista berria azkenengo elementua ezabatuta...
 ==================================
Bridget Jones II
Bridget Jones III
Harry Potter I
Harry Potter II
Harry Potter III
Harry Potter IV
Harry Potter V
Harry Potter VI
```

```
 ==================================

Elementu bat ezabatzen... elementua ez dago zerrendan
 ==================================
Elementu bat ezabatzen emandako denbora --> 0.015
 ==================================

Elementu bat ezabatzen... elementua badago zerrendan
Ezabatutako pelikulaBridget Jones III
Bridget Jones III ezabatzen emandako denbora --> 0.015
Bridget Jones III zerrendan dago? true
Elementu bat ezabatzen lortutako lista berria...
 ==================================
Bridget Jones III
Harry Potter I
Harry Potter II
Harry Potter III
Harry Potter IV
Harry Potter V
Harry Potter VI
 ==================================
```

OrderedCircularLinkedList: Ordena bat jarraitzen duen zerrenda estekatu zirkularra.

```
pelikula bat gehitzen emandako denbora --> 0.006
 Zerrenda ..............

Bridget Jones I
Bridget Jones II
Bridget Jones III
Harry Potter I
Harry Potter II
Harry Potter III
Harry Potter IV
Harry Potter V
Harry Potter VI
Harry Potter VII
Zerrenda inprimatzen emandako denbora --> 0.008

Colin Firth aktoreak duen pelikula kopurua
10

ELEMENTU BAT LISTAN DAGOEN EDO EZ KONPROBATU
Ahora me ves zerrendan dago? false
Ahora me ves pelikula bilatzen emandako denbora --> 0.008
Ahora me ves zerrendan dago? null
Harry Potter III zerrendan dago? true
Harry Potter III pelikula bilatzen emandako denbora --> 0.008
Harry Potter III zerrendan dago? praktika.akt.Pelikula@e57e1894

 Azken elementua
 ==================================
Harry Potter VII
 ==================================
```

```
 Lehenengo elementua:
 ================================
Bridget Jones I
 ================================

Elementu bat txertatu
Emandako denbora --> 0.008




pelikula bat gehitzen emandako denbora --> 0.007
 Zerrenda ..............

Bridget Jones I
Bridget Jones II
Bridget Jones III
Harry Potter I
Harry Potter II
Harry Potter III
Harry Potter IV
Harry Potter V
Harry Potter VI
Harry Potter VII
Zerrenda inprimatzen emandako denbora --> 0.01

Colin Firth aktoreak duen pelikula kopurua
10

ELEMENTU BAT LISTAN DAGOEN EDO EZ KONPROBATU
Ahora me ves zerrendan dago? false
Ahora me ves pelikula bilatzen emandako denbora --> 0.011
Ahora me ves zerrendan dago? null
Harry Potter III zerrendan dago? true
Harry Potter III pelikula bilatzen emandako denbora --> 0.011
Harry Potter III zerrendan dago? praktika.akt.Pelikula@e57e1894

 Azken elementua
 ================================
Harry Potter VII
 ================================

 Lehenengo elementua:
 ================================
Bridget Jones I
 ================================

Elementu bat txertatu
Emandako denbora --> 0.011
```

METODO NAGUSIEN DISEINU ETA IMPLEMENTAZIOA

Atal honetan praktika honetan inplementatutako metodo nagusien azalpen txiki bat eta haien kostua aurkeztuko da modu arrazoitu batean, gainera posibleak izango diren proba kasuak ere kontutan izango ditugu.

Gorago azaldu dugun moduan praktika honetan Aktorea eta Pelikula klaseek izan dituzte aldaketa gehienak, zerrenda estekatuak erabili bait ditugu Aktore eta Pelikula bakoitzaren barruan dauden zerrendak kudeatzeko. Hau aurrera eraman ahal izateko beste klase batzuetaz baliatu gara. CircularLinkedList, UnorderedCircularLinkedList eta OrderedCircularLinkedList klaseetaz, hain zuzen. Azalduko diren metodoak hauek izango dira:

CircularLinkedList.java

- Lehenengo elementuaren ezabaketa.
- Azkenengo elementuaren ezabaketa.
- Elementu baten ezabaketa.
- Elementu bat dagoen edo ez esatea (boolean baten bitartez).
- Elementu bat dagoen edo ez esatea (elementua badago elementua bera bueltatuko du, ez egotekotan null).

Lehenengo elementuaren ezabaketa removeFirst()

```
public T removeFirst() {
	/* Aurrebaldintza: zerrenda ez da hutsa
	Postbaldintza: zerrendako lehenengo elementua bueltatuko du
	 * eta listako lehen elementua kendu da
	 */
```

PROBA KASUAK

1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago

Inplementazioa:

```
T firstNode;
		if(last.next.equals(last)){
			firstNode = last.data;
			last = null;
		}else{
			firstNode = last.next.data;
			last.next = last.next.next;
		}

		count --;
		//listako lehen elementua bueltatzen du
		return(firstNode);
```

Kostua:
Kostua konstantea izango da, bakarrik egiaztapenak eta esleipenak egiten dira, horien kostua konstantea izanda. Horregatik algoritmoaren kostua O(1) izango da.

Azken elementuaren ezabaketa removeLast()

```
public T removeLast() {
            /* Aurrebaldintza: zerrenda ez da hutsa
            Postbaldintza: zerrendako azken elementua bueltatuko du
             * eta listako azken elementua kendu da
             */
```

PROBA KASUAK	
1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago

Inplementazioa:

```
T lastNode;
            if(last.next.equals(last)){
                  lastNode = last.data;
                  last = null;
            }else{
                  lastNode = last.data;
                  Node<T> current = last;
                  //last -en aurreko nodoa aurkitu beharra dago
                  while(current.next != last){
                        current = current.next;
                  }
                  current.next = last.next;
                  last = current;
            }

            count --;
            return(lastNode);

    }
```

Kostua:
Kostua kalkulatzeko, kasurik txarrenean ipiniko gara. Kasu honetan kasurik txarrena zerrenda osoa zeharkatu beharko dugula da azken elementuaren aurreko elementua aurkitu behar dugulako. Beraz n = zerrendan dagoen elementu kopurua izanik, kostua O(n) izango da.

Elementu baten ezabaketa remove()

```
public T remove(T elem) {
      // Aurrebaldintza: zerrenda ez da hutsa
      // Balio hori listan baldin badago, bere lehen agerpena ezabatuko dut. Kendutako objektuaren erreferentzia
        //  bueltatuko du (null ez baldin badago)
```

PROBA KASUAK

1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago

Inplementazioa:

```
T r = null;
            //elementu bakarreko zerrenda
            if(last.next.equals(last)){
                  if(last.data.equals(elem)){
                        r = last.data;
                        last = null;
                  }
            }else{
                  Node<T> current = last.next;

                  if(current.data.equals(elem)){
                        r = current.data;
                        last.next = current.next;
                  }else{
                        boolean jarraitu = true;
                        while(jarraitu && current != last){
                              if(current.next.data.equals(elem)){
                                    jarraitu = false;
                              }else{
                                    current = current.next;
                              }
                        }
                        if(!jarraitu){
                              r = current.next.data;
                              if( current.next == last){
                                    last = current;
                              }
                              current.next = current.next.next;
                        }

                  }
            }
      return(r);
    }
```

Kostua:

Kostua kalkulatzeko, kasurik txarrenean ipiniko gara. Kasu honetan kasurik txarrena zerrenda osoa zeharkatu beharko dugula da elementua aurkitu behar dugulako, posiblea litzateke elementua ez aurkitzea ere bai. Beraz n = zerrendan dagoen elementu kopurua izanik, kostua O(n) izango da.

Elementu bat dagoen edo ez esatea(boolean bidez) contains()

```
public boolean contains(T elem) {
      // Egiazkoa bueltatuko du aurkituz gero, eta false bestela
```

PROBA KASUAK

1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago

Inplementazioa:

```
boolean badago = false;
            Node<T> current = last;
            if(last != null){
                  if(current.data.equals(elem)){
                        badago = true;

                  }else{
                        while(!badago && current.next != last){
                              if(current.next.data.equals(elem)){
                                    badago = true;
                              }else{
                              current = current.next;
                              }
                        }
                  }
            }return(badago);

               }
```

Kostua:
Kostua kalkulatzeko, kasurik txarrenean ipiniko gara. Kasu honetan kasurik txarrena zerrenda osoa zeharkatu beharko dugula da elementua aurkitu behar dugulako, posiblea litzateke elementua ez aurkitzea ere bai. Beraz n = zerrendan dagoen elementu kopurua izanik, kostua O(n) izango da.

Elementu bat dagoen edo ez esatea (elementua badago elementua bera bueltatuko du, ez egotekotan null) find()

```
public T find(T elem) {
            //Aurre.:
      // Elementua bueltatuko du aurkituz gero, eta null bestela
```

PROBA KASUAK	
1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago

Inplementazioa:

```
T r = (T)new Object();
            Node<T> current = last;
            boolean badago = false;
            if(last != null){

                  if(current.data.equals(elem)){
                              badago = true;
                              r = current.data;

                  }else{
                        while(!badago && current.next != last){
                              if(current.next.data.equals(elem)){
                                    badago = true;
                                    r = current.next.data;
                              }else{
                                    current = current.next;
                              }
                        }
                  }
            }
            if(!badago){
                  r = null;
            }
            return(r);
      }
```

Kostua:
Kostua kalkulatzeko, kasurik txarrenean ipiniko gara. Kasu honetan kasurik txarrena zerrenda osoa zeharkatu beharko dugula da elementua aurkitu behar dugulako, posiblea litzateke elementua ez aurkitzea ere bai. Beraz n = zerrendan dagoen elementu kopurua izanik, kostua O(n) izango da.

UnorderedCircularLinkedList

- Aurrean gehitu.
- Azkeneko posizioan gehitu.
- Elementu baten atzean gehitu.

Aurrean gehitu addToFront()

```
public void addToFront(T elem) {
	// hasieran gehitu
```

PROBA KASUAK	
1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago
3. PROBA KASUA	Zerrenda hutsa izatea

Inplementazioa:

```
Node<T> elemN = new Node<T>(elem);
		if(isEmpty()){
			elemN.next = elemN;
			last = elemN;

		}else{
			elemN.next = last.next;
			last.next = elemN;
		}

		count++;
		System.out.println(elemN.data);
	}
```

Kostua:

Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. Da

Azkeneko posizioan gehitu addToRear()

```
public void addToRear(T elem) {
	// bukaeran gehitu
```

PROBA KASUAK	
1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago
3. PROBA KASUA	Zerrenda hutsa izatea

Inplementazioa:

```
Node<T> elemN = new Node<T>(elem);
		if(last == null){
			last = elemN;
			last.next = last;

		}else{
			elemN.next = last.next;
			last.next = elemN;
			last = elemN;
		}
		count++;
		//System.out.println(elemN.data);


	}
```

Kostua:

Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta esleipenen kostua kte. Da

Elementu baten atzean gehitu addAfter()

```
public void addAfter(T elem, T target) {
		//target elementua listan dagoela suposatu behar da
```

PROBA KASUAK	
1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago

Inplementazioa:

```
Node<T> current = last;
		boolean aurkitu = false;
		Node<T> berria = new Node<T>(elem);
		if(current.data.equals(target)){
			berria.next = current.next;
			current.next = berria;
			last = berria;
		}else{
			while(!aurkitu){
				if(current.data.equals(target)){
					aurkitu = true;
				}else{
					current = current.next;
				}
			}


			berria.next = current.next;
			current.next = berria;
		}
```

```
            count++;
      }
```

Kostua:
Kostua kalkulatzeko, kasurik txarrenean ipiniko gara. Kasu honetan kasurik txarrena zerrenda osoa zeharkatu beharko dugula da elementua aurkitu behar dugulako horren atzean gehitzeko. Beraz n = zerrendan dagoen elementu kopurua izanik, kostua O(n) izango da.

OrderedCircularLinkedList

- Elementua gehitu, elementu berria dagokion posizioan gehituko du.

Elementu bat gehitu add()

```
public void add(T elem){
            //elementua dagokion posizioan gehitu
```

PROBA KASUAK

1.PROBA KASUA	Zerrendan elementu bakarra dago
2. PROBA KASUA	Zerrendan elementu bat baino gehiago dago
3.PROBA KASUA	Zerrenda hutsik dago

Inplementazioa:

```
boolean jarraitu = true;
            Node<T> current = last;
            Node<T> berria = new Node<T>(elem);
            //lista hutsik badago
            if(isEmpty()){
                  berria.next = berria;
                  last = berria;
            }else{
                  //lista ez dago hutsik
                  while(jarraitu){
                        if(current.compareTo(elem)<0 &&
current.next.compareTo(elem)>0){
                              berria.next = current.next;
                              current.next = berria;
                              jarraitu = false;
                        }else{
                              current = current.next;
                        }
                  }
                  if(current == last){
                        last = berria;
                  }
            }
            count++;
      }
```

Kostua:
Kostua kalkulatzeko, kasurik txarrenean ipiniko gara. Kasu honetan kasurik txarrena zerrenda osoa zeharkatu beharko dugula da elementua aurkitu behar dugulako horren atzean gehitu ahal izateko. Beraz n = zerrendan dagoen elementu kopurua izanik, kostua O(n) izango da.

KODEA

AktorePeli_ProgramaNagusia.java

```
package praktika.akt;

import java.util.Scanner;

public class AktorePeli_ProgramaNagusia {

	private static Stopwatch timer;
	public static void main(String[] args) {
		hasieratuZerrendak();
		//datuak kargatu fitxategitik
		//fitxategia proiektuaren barruan dago, baina irakurtzeko
klase bat erabiliko da
		timer = new Stopwatch();
		System.out.println("Datuak kargatzen...");
		String nomF = "fitxategiak/listaHandiagoa.txt";

	PelikulaZerrenda.getPelikulaZerrenda().zerrendaKargatu(nomF);
		System.out.println("Datuak kargatuta! :), emandako denbora
--> "+timer.elapsedTime());

		//menu bat, apikazioak dauzkan aukera guztiak proban
jartzeko
		paintMenu();

		/*
		 * Probak *
		 */
		/*Aktoreen zerrendan aktore baten bilaketa, izena eta
abizena erabilita
		IzenAbizen formatua --> Abizena, Izena

		Proba kasu 1: Aktorea zerrendan dago
		Hudson, Annette;


		Proba kasu 2: Aktorea ez dago zerrendan
		Perez, Pepita;
		*/


		/*aktore baten pelikula zerrenda lortu
		Britt, Eva*/

		/*aktore zerrenda osoa inprimatu*/

		/*Pelikula bateko aktoreak bueltatu
		Pantailan agertuko dira formatu honekin --> Abizena, Izena
		Eager to Die
		*/

		/*pelikula zerrenda osoa imprimatu
		*/

		/*zerrendatik aktore bat ezabatu
		Proba kasu 1: aktorea ez dago zerrendan
		Ez, Dago
```

```
		Proba kasu 2: aktorea badago zerrendan
		Britt, Eva
		*/

		/*aktoreen zerrenda fitxategi batean gorde
		 * */


		/*aktoreen zerrenda ordenatua lortu --> Abizena, Izena
		*/


	}
	@SuppressWarnings("resource")
	private static void paintMenu(){
		//String aukera="";
		int select = 0;
		Scanner sc = new Scanner(System.in);
		while (select != 8){

	System.out.println("==========================================
==========");
			System.out.println("=======1-etik 7-rainoko aukeren
artean bat hartu=======");
			System.out.println("1 - Aktore baten bilaketa");
			System.out.println("2 - Aktore berri baten
txertaketa");
			System.out.println("3 - Aktore baten pelikulak
bueltatu");
			System.out.println("4 - Pelikula bateko aktoreak
bueltatu");
			System.out.println("5 - Aktore baten ezabaketa");
			System.out.println("6 - Aktoreen zerrenda fitxategi
batean gorde");
			System.out.println("7 - Aktoreen zerrenda ordenatua
(Abizena, Izena)");
			System.out.println("8 - Irten");

	System.out.println("==========================================
==========");
			select = Integer.parseInt(sc.nextLine());
			aukeraHartu(select);
		}
	}
	private static void aukeraHartu(int aukera){

		//Scanner sc = new Scanner(System.in);
		//aukera = sc.nextLine();


		switch (aukera) {
		case 1:
			//bilaketa
			badagoZerrendan();
			break;
		case 2:
			//txertaketa
			gehituAktorea();
			break;
		case 3:
			//pelikulak bueltatu
```

```
                        aktorePelikulak();
                        break;
                case 4:
                        //aktoreak bueltatu
                        pelikulaAktoreak();
                        break;
                case 5:
                        //ezabaketa
                        ezabatuAktorea();
                        break;
                case 6:
                        //fitxategia sortu
                        idatziFitxategia();
                        break;
                case 7:
                        //zerrenda ordenatua
                        zerrendaOrdenatua();
                        break;
                case 8:
                        System.out.println("Programa amaitu da :(");
                        System.exit(0);
                default:
                        System.out.println("EZ DAGO AUKERA HORI!");
                        //aukeraHartu();
                        break;
                }
        }
        @SuppressWarnings("resource")
        private static void badagoZerrendan(){

                String aIzena= "";
                System.out.println("Sartu aktorearen Abizen, Izena");
                Scanner sc = new Scanner(System.in);
                aIzena = sc.nextLine();
                timer = new Stopwatch();
                AktoreZerrenda.getAktoreZerrenda().badagoZerrendan(aIzena);
                System.out.println(aIzena+" aktorearen bilaketan emandako
denbora --> "+timer.elapsedTime());
        }
        @SuppressWarnings("resource")
        private static void gehituAktorea(){
                String aIzena= "";
                System.out.println("Sartu aktorearen Abizen, Izena");
                Scanner sc = new Scanner(System.in);
                aIzena = sc.nextLine();
                Aktorea a = new Aktorea(aIzena);
                timer = new Stopwatch();
                AktoreZerrenda.getAktoreZerrenda().gehituAktorea(a);
                System.out.println(aIzena+" gehitzeko emandako denbora -->
"+timer.elapsedTime());
        }
        @SuppressWarnings("resource")
        private static void aktorePelikulak(){
                String aIzena= "";
                System.out.println("Sartu aktorearen Abizen, Izena");
                Scanner sc = new Scanner(System.in);
                aIzena = sc.nextLine();
                timer = new Stopwatch();

        AktoreZerrenda.getAktoreZerrenda().pelikulakImprimatu(aIzena);
```

```
            System.out.println(aIzena+" aktorearen pelikulak
inprimatzen emandako denbora --> "+timer.elapsedTime());
      }
      @SuppressWarnings("resource")
      private static void pelikulaAktoreak(){
            String pIzena= "";
            System.out.println("Sartu pelikularen izenburua");
            Scanner sc = new Scanner(System.in);
            pIzena = sc.nextLine();
            timer = new Stopwatch();

      PelikulaZerrenda.getPelikulaZerrenda().aktoreakimprimatu(pIzena);
            System.out.println(pIzena+" pelikularen aktoreak
inprimatzeko emandako denbora --> "+timer.elapsedTime());
      }
      @SuppressWarnings("resource")
      private static void ezabatuAktorea(){
            String aIzena= "";
            System.out.println("Sartu aktorearen Abizen, Izena");
            Scanner sc = new Scanner(System.in);
            aIzena = sc.nextLine();
            timer = new Stopwatch();
            AktoreZerrenda.getAktoreZerrenda().ezabatuAktorea(aIzena);
            System.out.println(aIzena+" ezabatzeko emandako denbora -->
"+timer.elapsedTime());
      }
      private static void idatziFitxategia(){
            timer = new Stopwatch();
            AktoreZerrenda.getAktoreZerrenda().idatziAktoreenFitx();
            System.out.println("Fitxategia idazteko emandako denbora --
> "+timer.elapsedTime());
      }
      private static void zerrendaOrdenatua(){
            timer = new Stopwatch();
            AktoreZerrenda.getAktoreZerrenda().zerrendaOrdenatua();
            System.out.println("Aktoreen zerrenda ordenatzeko emandako
denbora --> "+timer.elapsedTime());
      }
      private static void hasieratuZerrendak(){
            AktoreZerrenda.getAktoreZerrenda().hasieratu();
            PelikulaZerrenda.getPelikulaZerrenda().hasieratu();
      }
}
```

AktoreZerrenda.java

```
package praktika.akt;


import java.io.FileWriter;
import java.io.IOException;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

public class AktoreZerrenda {


	private Map<String,Aktorea> z1;
	private static AktoreZerrenda nireAktoreZerrenda= new
AktoreZerrenda();

	private AktoreZerrenda() {

		z1 = new HashMap<String, Aktorea>();
	}

	public static synchronized AktoreZerrenda getAktoreZerrenda()
	{
		if (AktoreZerrenda.nireAktoreZerrenda==null){
			AktoreZerrenda.nireAktoreZerrenda=new
AktoreZerrenda();
		}
		return AktoreZerrenda.nireAktoreZerrenda;
	}

	public Iterator<Entry<String, Aktorea>> getIteradorea(){
		return z1.entrySet().iterator();
	}
	//HashMap egitura
	public int sizeHM(){
		return z1.size();
	}
	public void badagoZerrendan(String izenOsoa){
		if(zerrendanDago(izenOsoa)){
			System.out.println(izenOsoa+" aktorea badago
zerrendan! :)");
		}else{
			System.out.println(izenOsoa+" ez da aurkitu
zerrendan... :(");
		}

	}
	public boolean zerrendanDago(String izenOsoa){
		/*
		 * Aurre: Datu egitura ez dago hutsik
		 * Post: emaitza true izango da aktorea egon ezkero, false
ez badago
		 */
```

```
            return z1.containsKey(izenOsoa);
      }
      public void gehituAktorea(Aktorea a){
            //konprobatu aktorea zerrendan dagoen jada
            /*
             * Aurre:Datu egitura ez dago hutsik
             * Post: aktorea gehituko du zerrendan ez badago,
egotekotan
             *          mezu bat pantailaratuko du eta ez du gehituko
             *
             */

      if(!AktoreZerrenda.getAktoreZerrenda().zerrendanDago(a.getIzena()
)){
                  z1.put(a.getIzena(), a);
            }else{
                  System.out.println(a.getIzena()+" aktorea badago
zerrendan!!");
            }

      }
      //aktore baten ezabaketa
      public void ezabatuAktorea(String izenOsoa){
            /*
             * Aurre:Datu egitura ez dago hutsik
             * Post: aktorearen ezabatuko du zerrendan badago
             *          ez badago, mezu bat pantailaratuko da
             */

      if(AktoreZerrenda.getAktoreZerrenda().zerrendanDago(izenOsoa)){

      AktoreZerrenda.getAktoreZerrenda().z1.remove(izenOsoa);

      PelikulaZerrenda.getPelikulaZerrenda().ezabatuAktorea(izenOsoa);
                  System.out.println(izenOsoa+" aktorea ezabatu da.");
            }else{
                  System.out.println("Ezin da "+izenOsoa+" ezabatu,
zerrendan ez dagoelako");
            }

      }
      //aktoreen zerrenda inprimatu
      @SuppressWarnings("rawtypes")
      public void imprimatuZerrenda(){
            Iterator it = z1.entrySet().iterator();
            System.out.println("Aktoreen Zerrenda:");

      System.out.println("==========================================
==========");
            while (it.hasNext()) {
                Map.Entry a = (Map.Entry)it.next();
                System.out.println(a.getKey());
            }
      }
      //aktore baten pelikula zerrenda pantailaratuko du
      public void pelikulakImprimatu(String izenOsoa){
            /*
             * Aurre:Datu egitura ez dago hutsik
             * Post: aktorearen pelikulak pantailaratuko ditu
             *          aktorea egotekotan, ez badago mezu bat
             *          pantailaratuko du
```

```
		 *
		 */
		Aktorea a1 = z1.get(izenOsoa);
		if(a1 == null){
			System.out.println("Ez dago aktore hori!!");
		}else{
			a1.pelikulakInprimatu();
		}
	}

	public void zerrendaOrdenatua(){
		/*
		 * Aurre:Datu egitura ez dago hutsik
		 * Post: aktoreen zerrenda ordenatua pantailaratu
		 */
		List<Map.Entry<String, Aktorea>> l = new
LinkedList<Map.Entry<String, Aktorea>>(z1.entrySet());
		Collections.sort(l, new Comparator<Map.Entry<String,
Aktorea>>() {
			public int compare(Map.Entry<String, Aktorea> o1,
Map.Entry<String, Aktorea> o2){
				return o1.getValue().compareTo(o2.getValue());
			}
		});
		Map<String, Aktorea> sortMap = new LinkedHashMap<String,
Aktorea>();
		for(Map.Entry<String, Aktorea> a: l){
			sortMap.put(a.getKey(), a.getValue());
		}
		for(Map.Entry<String, Aktorea> ord: sortMap.entrySet()){
			System.out.println(ord.getValue().getIzena());
		}


	}
	public void hasieratu() {
		z1 = null;
		nireAktoreZerrenda = null;
	}
	public void gehitu(Pelikula p, String izenOsoa){

	AktoreZerrenda.getAktoreZerrenda().z1.get(izenOsoa).gehituPelikul
a(p);
	}
	//aktoreen zerrenda fitxategi batean gorde
		@SuppressWarnings("rawtypes")
	public void idatziAktoreenFitx(){
			FileWriter file;
			/*
			 * Aurre:Datu egitura ez dago hutsik
			 * Post: aktoreen zerrenda fitategian gordeta
			 */
			Iterator it =
AktoreZerrenda.getAktoreZerrenda().getIteradorea();
			try {
				file = new
FileWriter("fitxategiak/AktoreZerrenda.txt");

				while (it.hasNext()) {
				    Map.Entry a = (Map.Entry)it.next();
				    //System.out.println(a.getKey());
```

```
                        file.write(a.getKey()+"\r\n");
                    }
                    file.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }

}
```

PelikulaZerrenda.java

```
package praktika.akt;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Scanner;

public class PelikulaZerrenda {

	//private ArrayList<Aktorea> zerrenda;

	private static PelikulaZerrenda nirePelikulaZerrenda= null;
	private Map<String,Pelikula> z1;

	private PelikulaZerrenda() {
		//zerrenda=new ArrayList<Aktorea>();
		z1 = new HashMap<String, Pelikula>();
	}

	public static synchronized PelikulaZerrenda getPelikulaZerrenda()
	{
		if (PelikulaZerrenda.nirePelikulaZerrenda==null){
			PelikulaZerrenda.nirePelikulaZerrenda=new
PelikulaZerrenda();
		}
		return PelikulaZerrenda.nirePelikulaZerrenda;
	}
	//pelikula berria zerrendan gehituko du
	public void gehituPelikula(Pelikula p){

	if(!PelikulaZerrenda.getPelikulaZerrenda().zerrendanDago(p.getIze
nburua())){

	PelikulaZerrenda.getPelikulaZerrenda().z1.put(p.getIzenburua(),
p);
		}else{
			System.out.println(p.getIzenburua()+" pelikula badago
zerrendan");
		}
	}
	//pelikula zerrendaren tamaina
	public int tamaina() {
		return PelikulaZerrenda.getPelikulaZerrenda().z1.size();
	}
	//pelikula baten izenburua emanda zerrendan dagoen edo ez esango
digu
	public boolean zerrendanDago(String izenburua){
		return
PelikulaZerrenda.getPelikulaZerrenda().z1.containsKey(izenburua);
	}

	//pelikula baten bilaketa
	public void badagoZerrendan(String izenburua){

	if(!PelikulaZerrenda.getPelikulaZerrenda().zerrendanDago(izenburu
a)){
```

```
                  System.out.println(izenburua+" pelikula badago
zerrendan! :)");
            }else{
                  System.out.println(izenburua+" ez da aurkitu
zerrendan... :(");            }
      }
      //pelikula zerrenda imprimatu
      @SuppressWarnings("rawtypes")
      public void imprimatuZerrenda(){
            Iterator it = z1.entrySet().iterator();
            System.out.println("Pelikula Zerrenda:");

      System.out.println("=============================================
===========");
            while (it.hasNext()) {
                Map.Entry p = (Map.Entry)it.next();
                System.out.println(p.getKey());
            }

      }
      //pelikula batek dituen aktoreen zerrenda imprimatuko du
      public void aktoreakimprimatu(String izenburua){
            Pelikula p;

      if(PelikulaZerrenda.getPelikulaZerrenda().zerrendanDago(izenburua
)){
                  p =
PelikulaZerrenda.getPelikulaZerrenda().z1.get(izenburua);
                  p.imprimatuAktoreak();
            }else{
                  System.out.println("Ez da aurkitu zerrendan!!");
            }
      }
      public void hasieratu() {
            z1 = null;
            nirePelikulaZerrenda = null;
      }
      public void ezabatuAktorea(String izenOsoa){
            //bilatu aktore hori dagoen pelikula guztietan eta ezabatu
            for (Pelikula key : z1.values()) {
              key.ezabatuAktorea(izenOsoa);
          }
      }
      //fitxategitik datuak atera
            @SuppressWarnings({ "resource", "unused" })
            public void zerrendaKargatu(String nomF){
                  String [] peli = null;
                  String [] aktore = null;
                  Pelikula p = null;
                  Aktorea a = null;
                  //String
                  try{
                        Scanner entrada = new Scanner(new
FileReader(nomF));
                        FileReader f = new FileReader(nomF);
                        BufferedReader reader = new BufferedReader(f);
                        String linea;
                        while (entrada.hasNext()) {
                       // lerroko pelikularen izenburua lortu
                              linea = entrada.nextLine();
```

```
                              peli = reader.readLine().split("\\s---
>\\s");
                        //sortu pelikula objektua
                        p = new Pelikula(peli[0]);
                        //pelikula pelikulen katalogora gehitu

      PelikulaZerrenda.getPelikulaZerrenda().gehituPelikula(p);
                        aktore = peli[1].split("\\s&&&\\s");
                        for(int i = 0; i<aktore.length;i++){
                              //System.out.println(aktore[i]);
                              a = new Aktorea(aktore[i]);
                              p.gehituAktorea(a);


      if(!AktoreZerrenda.getAktoreZerrenda().zerrendanDago(aktore[i])){
                                    //ez badago zerrendan aktorea
zerrendara gehitzen dugu eta pelikula bere zerrendara gehitzen zaio
                                    a.gehituPelikula(p);

      AktoreZerrenda.getAktoreZerrenda().gehituAktorea(a);
                              }else{
                                    //aktorea badago zerrendan pelikula
gehitzen diogu bere zerrendari

      AktoreZerrenda.getAktoreZerrenda().gehitu(p, aktore[i]);

                              }
                        }

                         }
                         entrada.close();
                         }
                         catch(IOException e) {
                               //ezdago fitxategia
                               System.out.println("EZ DA FITXATEGIA
AURKITZEN");
                               //e.printStackTrace();
                               System.exit(0);
                               }
            }
}
```

StopWatch.java

```
/*************************************************************************
***
 *  Compilation:  javac Stopwatch.java
 *

*************************************************************************
**/

/**
 *  Stopwatch. This class is a data type for measuring
 *  the running time (wall clock) of a program.
 *
 *  For additional documentation, see
 *  <a href="http://introcs.cs.princeton.edu/32class">Section 3.2</a>
of
 *  <i>Introduction to Programming in Java: An Interdisciplinary
Approach</i>
 *  by Robert Sedgewick and Kevin Wayne.
 */

package praktika.akt;

public class Stopwatch {

    private final long start;

   /** Create a stopwatch object. */
    public Stopwatch() {
        start = System.currentTimeMillis();
    }

   /**
     * Return elapsed time (in seconds) since this object was created.
     */
    public double elapsedTime() {
        long now = System.currentTimeMillis();
        return (now - start) / 1000.0;
    }
}
```

ListADT.java

```
package praktika2.eda;

import java.util.Iterator;

public interface ListADT<T> {


public void setDeskr(String izena); // Listaren izena eguneratzen du

public String getDeskr(); // Listaren izena bueltatzen du


public T removeFirst(); // listako lehen elementua kendu da

public T removeLast(); // listako azken elementua kendu da

public T remove(T elem); // Balio hori listan baldin badago, bere lehen
agerpena ezabatuko dut. Kendutako objektuaren erreferentzia bueltatuko
du (null ez baldin badago)

public T first(); // listako lehen elementua ematen du

public T last(); // listako azken elementua ematen du

public boolean contains(T elem); // Egiazkoa bueltatuko du aurkituz
gero, eta false bestela

public T find(T elem); // Elementua bueltatuko du aurkituz gero, eta
null bestela

public boolean isEmpty(); // hutsa den esango du

public int size(); // Listako elementu-kopurua

public Iterator<T> iterator(); // Listako elementuen iteradorea


}
```

Node.java

```
package praktika2.eda;

	public class Node<T> implements Comparable<T>{
		public T data;                    // dato del nodo
		public Node<T> next;    // puntero al siguiente nodo de la
lista
		// ----------------------------------------------------------
-----

		public Node(T dd)              // constructor
		{
			data = dd;
			next = null;
		}
		@Override
		public int compareTo(T o) {
			// TODO Auto-generated method stub
			return this.compareTo(o);
		}
	}
```

IndexedListADT.java

```
/**
 * IndexedListADT defines the interface to an indexed list collection.
Elements
 * are referenced by contiguous numeric indexes.
 * @author Dr. Lewis
 * @author Dr. Chase
 * @version 1.0, 08/13/08
 */

package praktika2.eda;

public interface IndexedListADT<T> extends ListADT<T>
{
    /**
     * Inserts the specified element at the specified index.
     *
     * @param index   the index into the array to which the element is
to be
     *                inserted.
     * @param element the element to be inserted into the array
     */
    public void add (int index, T element);

    /**
     * Sets the element at the specified index.
     *
     * @param index   the index into the array to which the element is
to be set
     * @param element the element to be set into the list
     */
    public void set (int index, T element);

    /**
     * Adds the specified element to the rear of this list.
     *
     * @param element  the element to be added to the rear of the list
     */
    public void add (T element);

    /**
     * Returns a reference to the element at the specified index.
     *
     * @param index  the index to which the reference is to be retrieved
from
     * @return       the element at the specified index
     */
    public T get (int index);

    /**
     * Returns the index of the specified element.
     *
     * @param element  the element for the index is to be retrieved
     * @return         the integer index for this element
     */
    public int indexOf (T element);

    /**  Removes and returns the element at the specified index. */
    public T remove (int index);
}
```

OrderedListADT.java

```
package praktika2.eda;

public interface OrderedListADT<T>  extends ListADT<T> {

	public void add(T elem);
	// elementu bat gehitzen du listan (dagokion tokian)

}
```

UnorderedListADT.java

```
package praktika2.eda;

public interface UnorderedListADT<T> extends ListADT<T> {

	public void addToFront(T elem);
	// elementu bat gehituko du hasieran

	public void addToRear(T elem);
	// elementu bat gehituko du bukaeran

	public void addAfter(T elem, T target);
	// "elem" elementua "target" elementuaren ondoren gehitzen du
("target" listan zegoen)


}
```

Aktorea.java

```
package praktika.akt;


import praktika2.eda.CircularLinkedList;
import praktika2.eda.Node;
import praktika2.eda.OrderedCircularLinkedList;
import praktika2.eda.UnorderedCircularLinkedList;

public class Aktorea {
      //Atributuak
            private String izenOsoa;
            private UnorderedCircularLinkedList<Pelikula>
listaPelikulak;
            private OrderedCircularLinkedList<Pelikula>
listaPelikulaOrdenatuak;
            //Eraikitzailea

            public Aktorea (String izenOsoa){
                  this.izenOsoa=izenOsoa;
                  listaPelikulak  = new
UnorderedCircularLinkedList<Pelikula>();
                  listaPelikulaOrdenatuak= new
OrderedCircularLinkedList<Pelikula>();
            }

            public String getIzena() {
                  return izenOsoa;
            }
            public CircularLinkedList<Pelikula> getListaPelikulak(){
                  return listaPelikulak;
            }
            public void gehituPelikula(Pelikula peli){
                  if(!listaPelikulak.contains(peli)){
                        this.listaPelikulak.addToRear(peli);
                  }else{
                        System.out.println("Pelikula badago listan");
                  }
            }
            public void gehituPelikulaOrdenatua(Pelikula peli){
                  if(!listaPelikulaOrdenatuak.contains(peli)){
                        this.listaPelikulaOrdenatuak.add(peli);
                  }else{
                        System.out.println("Pelikula badago
zerrendan");
                  }
            }
            public void ezabatuPelikula(String izenburua){
                  Pelikula p = new Pelikula(izenburua);

                  p =listaPelikulak.remove(p);
                  if(p != null){
                        System.out.println("Ezabatutako
pelikula"+p.getIzenburua());
                  }
            }
            public boolean badagoPelikula(String izenburua){
                  Pelikula p = new Pelikula(izenburua);
                  return listaPelikulak.contains(p);
            }
```

```
	public Pelikula findPeli(String izenburua){
		Pelikula p = new Pelikula(izenburua);
		return listaPelikulak.find(p);
	}
	public void imprimatu(){
		System.out.println(this.getIzena());
	}
	public void azkenPelikulaInprimatu(){
		listaPelikulak.last().imprimatu();
	}
	public void lehenPelikulaInprimatu(){
		listaPelikulak.first().imprimatu();
	}
	public void lehenPelikulaEzabatu(){
		listaPelikulak.removeFirst();
	}
	public void azkenPelikulaEzabatu(){
		listaPelikulak.removeLast();
	}
	public void pelikulakInprimatu() {
		//listaPelikulak.adabegiakInprimatu();
		Node<Pelikula> current = listaPelikulak.getLast();
		if(!listaPelikulak.isEmpty()){
			if(listaPelikulak.getLast().next ==
listaPelikulak.getLast()){
				//elementu bakarreko lista
				current.data.imprimatu();
			}else{
				//elementu bat baino gehiago
				while(current.next!=
listaPelikulak.getLast()){
					current.next.data.imprimatu();
					current = current.next;
				}

	listaPelikulak.getLast().data.imprimatu();
			}
		}
	}

	public int pelikulaKopurua() {
		return listaPelikulak.size();
	}
	public void hasieranGehitu(Pelikula p){
		listaPelikulak.addToFront(p);
	}
	public void bukaeranGehitu(Pelikula p){
		listaPelikulak.addToRear(p);
	}
	public void ondorenGehitu(Pelikula p, String izen){
		Pelikula p2 = new Pelikula(izen);
		listaPelikulak.addAfter(p,p2);
	}
	public int compareTo(Aktorea akt) {
		return this.izenOsoa.compareTo(akt.izenOsoa);
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
```

```
                result = prime * result + ((izenOsoa == null) ? 0 :
izenOsoa.hashCode());
                return result;
        }

        @Override
        public boolean equals(Object obj) {
                if (this == obj)
                        return true;
                if (obj == null)
                        return false;
                if (!(obj instanceof Aktorea))
                        return false;
                Aktorea other = (Aktorea) obj;
                if (izenOsoa == null) {
                        if (other.izenOsoa != null)
                                return false;
                } else if (!izenOsoa.equals(other.izenOsoa))
                        return false;
                return true;
        }
}
```

Pelikula.java

```
package praktika.akt;



import praktika2.eda.CircularLinkedList;
import praktika2.eda.Node;
import praktika2.eda.OrderedCircularLinkedList;
import praktika2.eda.UnorderedCircularLinkedList;

public class Pelikula {
      //Atributuak

            private String izenburua;
            private UnorderedCircularLinkedList<Aktorea> listaAktoreak
;
            private OrderedCircularLinkedList<Aktorea>
listaAktoreOrdenatuak;

            public Pelikula(String izenburua){
                  this.izenburua=izenburua;
                  listaAktoreak = new
UnorderedCircularLinkedList<Aktorea>();
                  listaAktoreOrdenatuak = new
OrderedCircularLinkedList<Aktorea>();
            }

            public String getIzenburua() {
                  return this.izenburua;
            }
            public CircularLinkedList<Aktorea> getListaAktoreak(){
                  return listaAktoreak;
            }
            public void gehituAktorea(Aktorea akt){
                  if(!listaAktoreak.contains(akt)){
                        listaAktoreak.addToRear(akt);
```

```
                }else{
                        System.out.println("Aktorea badago listan");
                }
        }
        public void gehituAktoreOrdenatua(Aktorea akt){
                if(!listaAktoreOrdenatuak.contains(akt)){
                        this.listaAktoreOrdenatuak.add(akt);
                }else{
                        System.out.println("Aktorea badago zerrendan");
                }
        }
        public boolean badagoAktorea(Aktorea a){
                return listaAktoreak.contains(a);
        }
        public Aktorea findAktorea(String izena){
                Aktorea a = new Aktorea(izena);
                return listaAktoreak.find(a);
        }
        public int compareTo(Pelikula peli) {
                return this.izenburua.compareTo(peli.izenburua);
        }
        public void ezabatuAktorea(String izenOsoa){
                Aktorea a = new Aktorea(izenOsoa);
                //nodoaren .data konparatzen ditu
                a =listaAktoreak.remove(a);
                if(a != null){
                        System.out.println("Ezabatutako
pelikula"+a.getIzena());
                }
        }
        public int aktoreKopurua() {
                return listaAktoreak.size();
        }
        public void imprimatu(){
                System.out.println(this.getIzenburua());
        }
        public void azkenAktoreaInprimatu(){
                listaAktoreak.last().imprimatu();
        }
        public void lehenAktoreaInprimatu(){
                listaAktoreak.first().imprimatu();
        }
        public void lehenAktoreaEzabatu(){
                listaAktoreak.removeFirst();
        }
        public void azkenAktoreaEzabatu(){
                listaAktoreak.removeLast();
        }
        public void imprimatuAktoreak(){
                //listaAktoreak.adabegiakInprimatu();
                Node<Aktorea> current = listaAktoreak.getLast();
                if(!listaAktoreak.isEmpty()){
                        if(listaAktoreak.getLast().next ==
listaAktoreak.getLast()){
                                //elementu bakarreko lista
                                current.data.imprimatu();
                        }else{
                                //elementu bat baino gehiago
                                while(current.next!=
listaAktoreak.getLast()){
                                        current.next.data.imprimatu();
```

```
                                    current = current.next;

        }listaAktoreak.getLast().data.imprimatu();
                            }
                    }
              }
              public void hasieranGehitu(Aktorea p){
                    listaAktoreak.addToFront(p);
              }
              public void bukaeranGehitu(Aktorea p){
                    listaAktoreak.addToRear(p);
              }
              public void ondorenGehitu(Aktorea p, String izen){
                    Aktorea p2 = new Aktorea(izen);
                    listaAktoreak.addAfter(p,p2);
              }
              @Override
              public int hashCode() {
                    final int prime = 31;
                    int result = 1;
                    result = prime * result + ((izenburua == null) ? 0 :
izenburua.hashCode());
                    return result;
              }

              @Override
              public boolean equals(Object obj) {
                    if (this == obj)
                          return true;
                    if (obj == null)
                          return false;
                    if (!(obj instanceof Pelikula))
                          return false;
                    Pelikula other = (Pelikula) obj;
                    if (izenburua == null) {
                          if (other.izenburua != null)
                                return false;
                    } else if (!izenburua.equals(other.izenburua))
                          return false;
                    return true;
              }
}
```

CircularLinkedList.java

```
package praktika2.eda;

import java.util.Iterator;
import java.util.NoSuchElementException;

@SuppressWarnings("unused")
public class CircularLinkedList<T> implements ListADT<T> {
      // Atributuak
      protected Node<T> last; // azkenaren erreferentzia
      protected String deskr;  // deskribapena
      protected int count;
      public CircularLinkedList() {
            last = null;
            deskr = "";
            count = 0;
```

```
	}
	public Node<T> getLast(){
		return last;
	}
	public void setDeskr(String ize) {
	  deskr = ize;
	}

	public String getDeskr() {
	  return deskr;
	}

	/**
	 * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
	 */
	public T removeFirst() {
	/* Aurrebaldintza: zerrenda ez da hutsa
	Postbaldintza: zerrendako lehenengo elementua bueltatuko du
	 * eta listako lehen elementua kendu da
	 */
		T firstNode;
		if(last.next.equals(last)){
			firstNode = last.data;
			last = null;
		}else{
			firstNode = last.next.data;
			last.next = last.next.next;
		}

		count --;
		//listako lehen elementua bueltatzen du
		return(firstNode);

	}

	/**
	 * Kostua O(n)izango da zerrenda osoa zeharkatu behar delako
	 * n=zerrendan dauden elementu kopurua izanda
	 * esleipenenen kostua kte. da
	 */
	public T removeLast() {
		/* Aurrebaldintza: zerrenda ez da hutsa
		Postbaldintza: zerrendako azken elementua bueltatuko du
		 * eta listako azken elementua kendu da
		 */
		T lastNode;
		if(last.next.equals(last)){
			lastNode = last.data;
			last = null;
		}else{
			lastNode = last.data;
			Node<T> current = last;
			//last -en aurreko nodoa aurkitu beharra dago
			while(current.next != last){
				current = current.next;
			}
			current.next = last.next;
			last = current;
		}
```

```
            count --;
            return(lastNode);

    }

      /**
       * Kostua O(n)izango da zerrenda zeharkatu behar delako, kasurik
txarrenean zerrenda osoa zeharkatuko da
       * n=zerrendan dauden elementu kopurua izanda
       * esleipenenen kostua kte. da
       */
      public T remove(T elem) {
      // Aurrebaldintza: zerrenda ez da hutsa
      // Balio hori listan baldin badago, bere lehen agerpena ezabatuko
dut. Kendutako objektuaren erreferentzia
        //  bueltatuko du (null ez baldin badago)
            T r = null;
            //elementu bakarreko zerrenda
            if(last.next.equals(last)){
                  if(last.data.equals(elem)){
                        r = last.data;
                        last = null;
                  }
            }else{
                  Node<T> current = last.next;

                  if(current.data.equals(elem)){
                        r = current.data;
                        last.next = current.next;
                  }else{
                        boolean jarraitu = true;
                        while(jarraitu && current != last){
                              if(current.next.data.equals(elem)){
                                    jarraitu = false;
                              }else{
                                    current = current.next;
                              }
                        }
                        if(!jarraitu){
                              r = current.next.data;
                              if( current.next == last){
                                    last = current;
                              }
                              current.next = current.next.next;
                        }

                  }
            }
      return(r);
    }
      /**
       * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
       */
      public T first() {
      // listako lehen elementua ematen du
            if (isEmpty())
                return null;
            else return last.next.data;
      }
      /**
```

```
	 * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
	 */
	public T last() {
	// listako azken elementua ematen du
		if (isEmpty())
		    return null;
		else return last.data;
	}
	/**
	 * Kostua O(n)izango da zerrenda zeharkatu behar delako, kasurik
txarrenean zerrenda osoa zeharkatuko da
	 * n=zerrendan dauden elementu kopurua izanda
	 * esleipenenen kostua kte. da
	 */
	public boolean contains(T elem) {
	// Egiazkoa bueltatuko du aurkituz gero, eta false bestela
		boolean badago = false;
		Node<T> current = last;
		if(last != null){
			if(current.data.equals(elem)){
				badago = true;

			}else{
				while(!badago && current.next != last){
					if(current.next.data.equals(elem)){
						badago = true;
					}else{
					current = current.next;
					}
				}
			}
		}return(badago);

		}
	/**
	 * Kostua O(n)izango da zerrenda zeharkatu behar delako, kasurik
txarrenean zerrenda osoa zeharkatuko da
	 * n=zerrendan dauden elementu kopurua izanda
	 * esleipenenen kostua kte. da
	 */
	@SuppressWarnings("unchecked")
	public T find(T elem) {
		//Aurre.:
	// Elementua bueltatuko du aurkituz gero, eta null bestela
		T r = (T)new Object();
		Node<T> current = last;
		boolean badago = false;
		if(last != null){

			if(current.data.equals(elem)){
					badago = true;
					r = current.data;

			}else{
				while(!badago && current.next != last){
					if(current.next.data.equals(elem)){
						badago = true;
						r = current.next.data;
					}else{
						current = current.next;
```

```
                        }
                    }
                }
            }
            if(!badago){
                r = null;
            }
            return(r);
      }
      /**
       * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
       */
      public boolean isEmpty()
      { return last == null;};
      /**
       * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
       */
      public int size()
      { return count;};

      /** Return an iterator to the stack that iterates through the
items . */
         public Iterator<T> iterator() { return new ListIterator(); }

         // an iterator, doesn't implement remove() since it's optional
         private class ListIterator implements Iterator<T> {
               private Node<T> current;
               private ListIterator(){
                     current = last;
               }
               public boolean hasNext(){
                     return current != null;
               }
               public T next(){
                     if(hasNext()){
                           T next = current.data;
                           current = current.next;
                           return next;
                     }
                     //ez badago elementurik
                     throw new java.util.NoSuchElementException("linked
list.next");
               }

               public void remove(){
                     throw new UnsupportedOperationException("Linked
list iterator remove not supported");
               }
         } // private class


            public void imprimatu(){
                  Node<T> current = last.next;
                  if(!isEmpty()){
                        if(last.next == last){
                              //elementu bakarreko lista

      System.out.println(current.data.toString());
                        }else{
```

```
                              //elementu bat baino gehiago
                              while(current!= last){

      System.out.println(current.data.toString());
                                    current = current.next;
                              }
                        }
                  }
            }

            public void adabegiakInprimatu() {
                  System.out.println(this.toString());
            }


            @Override
            public String toString() {
                  String result = new String();
                  Iterator<T> it = iterator();
                  while (it.hasNext()) {
                        T elem = it.next();
                        result = result + "[" + elem.toString() + "]
\n";
                  }
                  return "SimpleLinkedList " + result + "]";
            }

}
```

OrderedCircularLinkedList.java

```
package praktika2.eda;

public class OrderedCircularLinkedList<T> extends CircularLinkedList<T>
implements OrderedListADT<T> {

      public void add(T elem){
            //elementua dagokion posizioan gehitu
            boolean jarraitu = true;
            Node<T> current = last;
            Node<T> berria = new Node<T>(elem);
            //lista hutsik badago
            if(isEmpty()){
                  berria.next = berria;
                  last = berria;
            }else{
                  //lista ez dago hutsik
                  while(jarraitu){
                        if(current.compareTo(elem)<0 &&
current.next.compareTo(elem)>0){
                              berria.next = current.next;
                              current.next = berria;
                              jarraitu = false;
                        }else{
                              current = current.next;
                        }
                  }
                  if(current == last){
                        last = berria;
                  }
```

```
			}
			count++;
		}


}
```

UnorderedCircularLinkedList.java

```
package praktika2.eda;

public class UnorderedCircularLinkedList<T> extends
CircularLinkedList<T> implements UnorderedListADT<T> {
	/**
	 * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
	 */
	public void addToFront(T elem) {
	// hasieran gehitu
			Node<T> elemN = new Node<T>(elem);
			if(isEmpty()){
				elemN.next = elemN;
				last = elemN;

			}else{
				elemN.next = last.next;
				last.next = elemN;
			}

			count++;
			System.out.println(elemN.data);
	}
	/**
	 * Kostua konstantea O(1) esleipenak bakarrik egiten direlako eta
esleipenen kostua kte. da
	 */
	public void addToRear(T elem) {
	// bukaeran gehitu
			Node<T> elemN = new Node<T>(elem);
			if(last == null){
				last = elemN;
				last.next = last;

			}else{
				elemN.next = last.next;
				last.next = elemN;
				last = elemN;
			}
			count++;
			//System.out.println(elemN.data);


	}

	/**
	 * Kostua konstantea O(n)izango da zerrenda osoa zeharkatu behar
delako
	 * n=zerrendan dauden elementu kopurua izanda
	 * esleipenenen kostua kte. da
	 */
```

```
	public void addAfter(T elem, T target) {
		//target elementua listan dagoela suposatu behar da
		Node<T> current = last;
		boolean aurkitu = false;
		Node<T> berria = new Node<T>(elem);
		if(current.data.equals(target)){
			berria.next = current.next;
			current.next = berria;
			last = berria;
		}else{
			while(!aurkitu){
				if(current.data.equals(target)){
					aurkitu = true;
				}else{
					current = current.next;
				}
			}


			berria.next = current.next;
			current.next = berria;
		}
		count++;
	}

}
```

ProbaAktoreaCircularLL.java

```
package praktika2.test;

import praktika.akt.Aktorea;
import praktika.akt.Pelikula;
import praktika.akt.Stopwatch;

public class ProbaAktoreaCircularLL {

	public static void main(String[] args) {
		Stopwatch timer = new Stopwatch();
		/*
		 * 1.PROBA KASUA: Elementu bat baino gehiago dituen
zerrenda
		 */
		System.out.println("1.PROBA KASUA: Elementu bat baino
gehiago dituen zerrenda ");
		Pelikula p1 = new Pelikula("Bridget Jones I");
		Pelikula p2 = new Pelikula("Bridget Jones II");
		Pelikula p3 = new Pelikula("Bridget Jones III");
		Pelikula p4 = new Pelikula("Harry Potter I");
		Pelikula p5 = new Pelikula("Harry Potter II");
		Pelikula p6 = new Pelikula("Harry Potter III");
		Pelikula p7 = new Pelikula("Harry Potter IV");
		Pelikula p8 = new Pelikula("Harry Potter V");
		Pelikula p9 = new Pelikula("Harry Potter VI");
		Pelikula p10 = new Pelikula("Harry Potter VII");

		Aktorea a = new Aktorea("Colin Firth");
		a.gehituPelikula(p1);
		System.out.println("pelikula bat gehitzen emandako denbora
--> " + timer.elapsedTime());
```

```
            a.gehituPelikula(p2);
            a.gehituPelikula(p3);
            a.gehituPelikula(p4);
            a.gehituPelikula(p5);
            a.gehituPelikula(p6);
            a.gehituPelikula(p7);
            a.gehituPelikula(p8);
            a.gehituPelikula(p9);
            a.gehituPelikula(p10);

            // elementuak inprimatu
            System.out.println(" Zerrenda ..............");
            System.out.println("");
            a.pelikulakInprimatu();
            System.out.println("Zerrenda inprimatzen emandako denbora -
-> " + timer.elapsedTime());

            // elementu kopurua
            System.out.println("");
            System.out.println(a.getIzena() + " aktoreak duen pelikula
kopurua");
            System.out.println(a.pelikulaKopurua());

            // elementua dagoen konprobatu
            /* elementua ez dago */
            System.out.println("");
            System.out.println("ELEMENTU BAT LISTAN DAGOEN EDO EZ
KONPROBATU");
            String ize1 = "Ahora me ves";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println(ize1 + " pelikula bilatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(ize1 + " zerrendan dago? " +
a.findPeli(ize1));

            ize1 = "Harry Potter III";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println(ize1 + " pelikula bilatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(ize1 + " zerrendan dago? " +
a.findPeli(ize1));

            // azken elementua
            System.out.println("");
            System.out.println(" Azken elementua ");
            System.out.println(" ================================");
            a.azkenPelikulaInprimatu();
            System.out.println(" ================================");
            // lehenengo elementua
            System.out.println("");
            System.out.println(" Lehenengo elementua: ");
            System.out.println(" ================================");
            a.lehenPelikulaInprimatu();
            System.out.println(" ================================");

            // ezabatu lehenengoa
            System.out.println("");
            System.out.println("LISTAKO LEHENENGO ELEMENTUA EZABATU");
            a.lehenPelikulaEzabatu();
```

```
            System.out.println("Lehenengo pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            ize1 = "Bridget Jones I";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println("Zerrenda berria lehenengo elementua
ezabatuta...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");

            // azkena ezabatu
            System.out.println("");
            System.out.println("LISTAKO AZKEN ELEMENTUA EZABATU");
            a.azkenPelikulaEzabatu();
            System.out.println("Azken pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            ize1 = "Harry Potter VII";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println("Zerrenda berria azkenengo elementua
ezabatuta...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");

            // elementu bat ezabatu
            /* P1: elementua ez dago zerrendan */
            System.out.println("");
            System.out.println("Elementu bat ezabatzen... elementua ez
dago zerrendan");
            System.out.println(" ================================");
            ize1 = "Probak";
            a.ezabatuPelikula(ize1);
            System.out.println("Elementu bat ezabatzen emandako denbora
--> " + timer.elapsedTime());
            System.out.println(" ================================");
            /* P2: elementua badago zerrendan */
            System.out.println("");
            ize1 = "Harry Potter VII";
            System.out.println("Elementu bat ezabatzen... elementua
badago zerrendan");
            a.ezabatuPelikula(ize1);
            System.out.println(ize1 + " ezabatzen emandako denbora -->
" + timer.elapsedTime());
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println("Elementu bat ezabatzen lortutako
zerrenda berria...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");

            // elementu baten ondoren gehitu
            System.out.println("");
            System.out.println("Harry Potter II elementuaren ondoren
gehitu...");
            a.ondorenGehitu(p1, "Harry Potter II");
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
```

```
            System.out.println("Elementu baten ondoren gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");

            // aurrean gehitu
            System.out.println("");
            System.out.println("Elementu bat aurrean gehitu");
            a.hasieranGehitu(p10);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat hasieran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");

            // bukaeran gehitu
            System.out.println("");
            System.out.println("Elementu bat bukaeran gehitu");
            a.bukaeranGehitu(p6);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat bukaeran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");

            /*
             * 2.PROBA KASUA: Elementu bakarra duen zerrenda
             */
            System.out.println("2.PROBA KASUA: Elementu bakarra duen
zerrenda ");
            a = null;
            a = new Aktorea("Angelina Jolie");
            p1 = new Pelikula("Malefica");
            a.gehituPelikula(p1);

            // elementuak inprimatu
            System.out.println(" Zerrenda ..............");
            System.out.println("");
            a.pelikulakInprimatu();
            System.out.println("Zerrenda inprimatzen emandako denbora -
-> " + timer.elapsedTime());

            // elementu kopurua
            System.out.println("");
            System.out.println(a.getIzena() + " aktoreak duen pelikula
kopurua");
            System.out.println(a.pelikulaKopurua());

            // elementua dagoen konprobatu
            /* elementua ez dago */
            System.out.println("");
            System.out.println("ELEMENTU BAT LISTAN DAGOEN EDO EZ
KONPROBATU");
            ize1 = "Ahora me ves";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
```

```
            System.out.println(izel + " pelikula bilatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(izel + " zerrendan dago? " +
a.findPeli(izel));

            izel = "Malefica";
            System.out.println(izel + " zerrendan dago? " +
a.badagoPelikula(izel));
            System.out.println(izel + " pelikula bilatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(izel + " zerrendan dago? " +
a.findPeli(izel));

            // azken elementua
            System.out.println("");
            System.out.println(" Azken elementua ");
            System.out.println(" ================================");
            a.azkenPelikulaInprimatu();;
            System.out.println(" ================================");
            // lehenengo elementua
            System.out.println("");
            System.out.println(" Lehenengo elementua: ");
            System.out.println(" ================================");
            a.lehenPelikulaInprimatu();;
            System.out.println(" ================================");

            // ezabatu lehenengoa
            System.out.println("");
            System.out.println("LISTAKO LEHENENGO ELEMENTUA EZABATU");
            a.lehenPelikulaEzabatu();
            System.out.println("Lehenengo pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            izel = "Malefica";
            System.out.println(izel + " zerrendan dago? " +
a.badagoPelikula(izel));
            System.out.println("Zerrenda berria lehenengo elementua
ezabatuta...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");
            a.gehituPelikula(p1);
            // azkena ezabatu
            System.out.println("");
            System.out.println("LISTAKO AZKEN ELEMENTUA EZABATU");
            a.azkenPelikulaEzabatu();;
            System.out.println("Azken pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            izel = "Malefica";
            System.out.println(izel + " zerrendan dago? " +
a.badagoPelikula(izel));
            System.out.println("Zerrenda berria azkenengo elementua
ezabatuta...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");
            a.gehituPelikula(p1);

            // elementu bat ezabatu
            /* P1: elementua ez dago zerrendan */
            System.out.println("");
```

```
            System.out.println("Elementu bat ezabatzen... elementua ez
dago zerrendan");
            System.out.println(" ==================================");
            ize1 = "Probak";
            a.ezabatuPelikula(ize1);
            System.out.println("Elementu bat ezabatzen emandako denbora
--> " + timer.elapsedTime());
            System.out.println(" ==================================");
            /* P2: elementua badago zerrendan */
            System.out.println("");
            ize1 = "Malefica";
            System.out.println("Elementu bat ezabatzen... elementua
badago zerrendan");
            a.ezabatuPelikula(ize1);
            System.out.println(ize1 + " ezabatzen emandako denbora -->
" + timer.elapsedTime());
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println("Elementu bat ezabatzen lortutako
zerrenda berria...");
            System.out.println(" ==================================");
            a.pelikulakInprimatu();
            System.out.println(" ==================================");
            a.gehituPelikula(p1);

            // elementu baten ondoren gehitu
            System.out.println("");
            System.out.println("Malefica elementuaren ondoren
gehitu...");
            a.ondorenGehitu(p10, "Malefica");
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu baten ondoren gehitzen
lortutako zerrenda berria...");
            System.out.println(" ==================================");
            a.pelikulakInprimatu();
            System.out.println(" ==================================");
            a.ezabatuPelikula("Harry Potter VII");

            // aurrean gehitu
            System.out.println("");
            System.out.println("Elementu bat aurrean gehitu");
            a.hasieranGehitu(p10);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat hasieran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ==================================");
            a.pelikulakInprimatu();
            System.out.println(" ==================================");
            a.ezabatuPelikula("Harry Potter VII");

            // bukaeran gehitu
            System.out.println("");
            System.out.println("Elementu bat bukaeran gehitu");
            a.bukaeranGehitu(p6);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat bukaeran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ==================================");
```

```
            a.pelikulakInprimatu();
            System.out.println(" ==================================");
            a.ezabatuPelikula("Harry Potter III");


            /*
             * 3.PROBA KASUA: Zerrenda hutsa
             */
            System.out.println("3.PROBA KASUA: Zerrenda hutsa ");
            a = null;
            a = new Aktorea("Richard Gere");
            p1 = new Pelikula("Pretty Woman");

            // aurrean gehitu
            System.out.println("");
            System.out.println("Elementu bat aurrean gehitu");
            a.hasieranGehitu(p1);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat hasieran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ==================================");
            a.pelikulakInprimatu();
            System.out.println(" ==================================");
            a.ezabatuPelikula("Pretty Woman");

            // bukaeran gehitu
            System.out.println("");
            System.out.println("Elementu bat bukaeran gehitu");
            a.bukaeranGehitu(p1);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat bukaeran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ==================================");
            a.pelikulakInprimatu();
            System.out.println(" ==================================");

      }

}
```

ProbaAktoreaOrderedCircularLL.java

```
package praktika2.test;

import praktika.akt.Aktorea;
import praktika.akt.Pelikula;
import praktika.akt.Stopwatch;

public class ProbaAktoreaOrderedCircularLL {

      public static void main(String[] args) {
            Stopwatch timer = new Stopwatch();
            /*
             * 1.PROBA KASUA: Elementu bat baino gehiago dituen
zerrenda
             */
            Pelikula p1 = new Pelikula("Bridget Jones I");
            Pelikula p2 = new Pelikula("Bridget Jones II");
```

```
            Pelikula p3 = new Pelikula("Bridget Jones III");
            Pelikula p4 = new Pelikula("Harry Potter I");
            Pelikula p5 = new Pelikula("Harry Potter II");
            Pelikula p6 = new Pelikula("Harry Potter III");
            Pelikula p7 = new Pelikula("Harry Potter IV");
            Pelikula p8 = new Pelikula("Harry Potter V");
            Pelikula p9 = new Pelikula("Harry Potter VI");
            Pelikula p10 = new Pelikula("Harry Potter VII");

            Aktorea a = new Aktorea("Colin Firth");
            a.gehituPelikula(p1);
            System.out.println("pelikula bat gehitzen emandako denbora
--> " + timer.elapsedTime());
            a.gehituPelikula(p2);
            a.gehituPelikula(p3);
            a.gehituPelikula(p4);
            a.gehituPelikula(p5);
            a.gehituPelikula(p6);
            a.gehituPelikula(p7);
            a.gehituPelikula(p8);
            a.gehituPelikula(p9);
            a.gehituPelikula(p10);

            // elementuak inprimatu
            System.out.println(" Zerrenda ..............");
            System.out.println("");
            a.pelikulakInprimatu();
            System.out.println("Zerrenda inprimatzen emandako denbora -
-> " + timer.elapsedTime());

            // elementu kopurua
            System.out.println("");
            System.out.println(a.getIzena() + " aktoreak duen pelikula
kopurua");
            System.out.println(a.pelikulaKopurua());

            // elementua dagoen konprobatu
            /* elementua ez dago */
            System.out.println("");
            System.out.println("ELEMENTU BAT LISTAN DAGOEN EDO EZ
KONPROBATU");
            String ize1 = "Ahora me ves";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println(ize1 + " pelikula bilatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(ize1 + " zerrendan dago? " +
a.findPeli(ize1));

            ize1 = "Harry Potter III";
            System.out.println(ize1 + " zerrendan dago? " +
a.badagoPelikula(ize1));
            System.out.println(ize1 + " pelikula bilatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(ize1 + " zerrendan dago? " +
a.findPeli(ize1));

            // azken elementua
            System.out.println("");
            System.out.println(" Azken elementua ");
            System.out.println(" ================================");
```

```
            a.getListaPelikulak().last().imprimatu();
            System.out.println(" ================================");

            // lehenengo elementua
            System.out.println("");
            System.out.println(" Lehenengo elementua: ");
            System.out.println(" ================================");
            a.getListaPelikulak().first().imprimatu();
            System.out.println(" ================================");

            // txertatu
            System.out.println("");
            System.out.println("Elementu bat txertatu");
            Pelikula proba=new Pelikula("Proba");
            a.gehituPelikulaOrdenatua(proba);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat txertatzean lortutako
zerrenda berria...");
            System.out.println(" ================================");
            a.pelikulakInprimatu();
            System.out.println(" ================================");
      }

}
```

ProbaPelikulaCircularLL.java

```
package praktika2.test;

import praktika.akt.Aktorea;
import praktika.akt.Pelikula;
import praktika.akt.Stopwatch;

public class ProbaPelikulaCircularLL {

      public static void main(String[] args) {
            Stopwatch timer = new Stopwatch();
            /*
             * 1.PROBA KASUA: Elementu bat baino gehiago dituen
zerrenda
             */
            Aktorea a1 = new Aktorea("George Clooney");
            Aktorea a2 = new Aktorea("Brad Pitt");
            Aktorea a3 = new Aktorea("Matt Damon");
            Aktorea a4 = new Aktorea("Scott Caan");
            Aktorea a5 = new Aktorea("Casey Affleck");
            Aktorea a6 = new Aktorea("Shaobo Qin");
            Aktorea a7 = new Aktorea("Bernie Mac");
            Aktorea a8 = new Aktorea("Don Cheadle");
            Aktorea a9 = new Aktorea("Carl Reiner");
            Aktorea a10 = new Aktorea("Eddie Jemison");
            Aktorea a11 = new Aktorea("Elliott Gould");
            Aktorea a12 = new Aktorea("Julia Roberts");
            Aktorea a13 = new Aktorea("Andy García");

            Pelikula p = new Pelikula("Ocean's Eleven");
            p.gehituAktorea(a1);
            System.out.println("pelikula bat gehitzen emandako denbora
--> " + timer.elapsedTime());
```

```
            p.gehituAktorea(a2);
            p.gehituAktorea(a3);
            p.gehituAktorea(a4);
            p.gehituAktorea(a5);
            p.gehituAktorea(a6);
            p.gehituAktorea(a7);
            p.gehituAktorea(a8);
            p.gehituAktorea(a9);
            p.gehituAktorea(a10);
            p.gehituAktorea(a11);
            p.gehituAktorea(a12);
            p.gehituAktorea(a13);

            // elementuak inprimatu
            System.out.println(" Zerrenda ...............");
            System.out.println("");
            p.imprimatuAktoreak();
            System.out.println("Zerrenda inprimatzen emandako denbora -
-> " + timer.elapsedTime());

            // elementu kopurua
            System.out.println("");
            System.out.println(p.getIzenburua() + " aktoreak duen
pelikula kopurua");
            System.out.println(p.aktoreKopurua());

            // elementua dagoen konprobatu
            /* elementua ez dago */
            System.out.println("");
            System.out.println("ELEMENTU BAT LISTAN DAGOEN EDO EZ
KONPROBATU");
            Aktorea a14 = new Aktorea("Carl Urban");
            ;
            System.out.println(a14.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a14));
            System.out.println(a14.getIzena() + " aktorea bilatzen
emandako denbora --> " + timer.elapsedTime());
            System.out.println(a14.getIzena() + " zerrendan dago? " +
p.findAktorea(a14.getIzena()));

            System.out.println(a7.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a7));
            System.out.println(a7.getIzena() + " pelikula bilatzen
emandako denbora --> " + timer.elapsedTime());
            System.out.println(a7.getIzena() + " zerrendan dago? " +
p.findAktorea(a7.getIzena()));

            // azken elementua
            System.out.println("");
            System.out.println(" Azken elementua ");
            System.out.println(" ================================");
            p.getListaAktoreak().last().imprimatu();
            System.out.println(" ================================");

            // lehenengo elementua
            System.out.println("");
            System.out.println(" Lehenengo elementua: ");
            System.out.println(" ================================");
            p.getListaAktoreak().first().imprimatu();
            System.out.println(" ================================");
```

```
            // ezabatu lehenengoa
            System.out.println("");
            System.out.println("LISTAKO LEHENENGO ELEMENTUA EZABATU");
            p.getListaAktoreak().removeFirst();
            System.out.println("Lehenengo pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a1));
            System.out.println("Lista berria lehenengo elementua
ezabatuta...");
            System.out.println(" ==================================");
            p.imprimatuAktoreak();
            System.out.println(" ==================================");

            // azkena ezabatu
            System.out.println("");
            System.out.println("LISTAKO AZKEN ELEMENTUA EZABATU");
            p.getListaAktoreak().removeLast();
            System.out.println("Azken pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(a13.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a13));
            System.out.println("Zerrenda berria azkenengo elementua
ezabatuta...");
            System.out.println(" ==================================");
            p.imprimatuAktoreak();
            System.out.println(" ==================================");

            // elementu bat ezabatu
            /* P1: elementua ez dago zerrendan */
            System.out.println("");
            System.out.println("Elementu bat ezabatzen... elementua ez
dago zerrendan");
            System.out.println(" ==================================");
            String proba = "Probak";
            p.ezabatuAktorea(proba);
            System.out.println("Elementu bat ezabatzen emandako denbora
--> " + timer.elapsedTime());
            System.out.println(" ==================================");
            /* P2: elementua badago zerrendan */
            System.out.println("");
            System.out.println("Elementu bat ezabatzen... elementua
badago zerrendan");
            p.ezabatuAktorea(a2.getIzena());
            System.out.println(a2.getIzena() + " ezabatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(a2.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a2));
            System.out.println("Elementu bat ezabatzen lortutako
zerrenda berria...");
            System.out.println(" ==================================");
            p.imprimatuAktoreak();
            System.out.println(" ==================================");

            // aurrean gehitu
            System.out.println("");
            System.out.println("Elementu bat aurrean gehitu");
            p.hasieranGehitu(a10);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
```

```
            System.out.println("Elementu bat hasieran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");

            // bukaeran gehitu
            System.out.println("");
            System.out.println("Elementu bat bukaeran gehitu");
            p.bukaeranGehitu(a6);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat bukaeran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");

            /*
             * 2.PROBA KASUA: Elementu bakarra duen zerrenda
             */
            System.out.println("2.PROBA KASUA: Elementu bakarra duen
zerrenda ");
            p = null;
            p = new Pelikula("300");
            a1 = new Aktorea("Gerard Butler");
            p.gehituAktorea(a1);

            // elementuak inprimatu
            System.out.println("Zerrenda ..............");
            System.out.println("");
            p.imprimatuAktoreak();
            System.out.println("Zerrenda inprimatzen emandako denbora -
-> " + timer.elapsedTime());

            // elementu kopurua
            System.out.println("");
            System.out.println(p.getIzenburua() + " aktoreak duen
pelikula kopurua");
            System.out.println(p.aktoreKopurua());

            // elementua dagoen konprobatu
            /* elementua ez dago */
            System.out.println("");
            System.out.println("ELEMENTU BAT LISTAN DAGOEN EDO EZ
KONPROBATU");
            Aktorea a302 = new Aktorea("Eva Green");
            System.out.println(a302.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a302));
            System.out.println(a302.getIzena() + " pelikula bilatzen
emandako denbora --> " + timer.elapsedTime());
            System.out.println(a302.getIzena() + " zerrendan dago? " +
p.findAktorea(a302.getIzena()));

            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a1));
            System.out.println(a1.getIzena() + " pelikula bilatzen
emandako denbora --> " + timer.elapsedTime());
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.findAktorea(a1.getIzena()));
```

```
            // azken elementua
            System.out.println("");
            System.out.println(" Azken elementua ");
            System.out.println(" ================================");
            p.getListaAktoreak().last().imprimatu();
            System.out.println(" ================================");
            // lehenengo elementua
            System.out.println("");
            System.out.println(" Lehenengo elementua: ");
            System.out.println(" ================================");
            p.getListaAktoreak().first().imprimatu();
            System.out.println(" ================================");

            // ezabatu lehenengoa
            System.out.println("");
            System.out.println("LISTAKO LEHENENGO ELEMENTUA EZABATU");
            p.getListaAktoreak().removeFirst();
            System.out.println("Lehenengo pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a1));
            System.out.println("Zerrenda berria lehenengo elementua
ezabatuta...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");
            p.gehituAktorea(a1);
            // azkena ezabatu
            System.out.println("");
            System.out.println("LISTAKO AZKEN ELEMENTUA EZABATU");
            p.getListaAktoreak().removeLast();
            System.out.println("Azken pelikula ezabatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a1));
            System.out.println("Zerrenda berria azkenengo elementua
ezabatuta...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");
            p.gehituAktorea(a1);

            // elementu bat ezabatu
            /* P1: elementua ez dago zerrendan */
            System.out.println("");
            System.out.println("Elementu bat ezabatzen... elementua ez
dago zerrendan");
            System.out.println(" ================================");
            p.ezabatuAktorea(a302.getIzena());
            System.out.println("Elementu bat ezabatzen emandako denbora
--> " + timer.elapsedTime());
            System.out.println(" ================================");
            /* P2: elementua badago zerrendan */
            System.out.println("");
            System.out.println("Elementu bat ezabatzen... elementua
badago zerrendan");
            p.ezabatuAktorea(a1.getIzena());
            System.out.println(a1.getIzena() + " ezabatzen emandako
denbora --> " + timer.elapsedTime());
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a1));
```

```
            System.out.println("Elementu bat ezabatzen lortutako
zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");
            p.gehituAktorea(a1);

            // aurrean gehitu
            System.out.println("");
            System.out.println("Elementu bat aurrean gehitu");
            p.hasieranGehitu(a10);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat hasieran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");
            p.ezabatuAktorea(a10.getIzena());

            // bukaeran gehitu
            System.out.println("");
            System.out.println("Elementu bat bukaeran gehitu");
            p.bukaeranGehitu(a6);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat bukaeran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");
            p.ezabatuAktorea(a6.getIzena());

            // elementu baten ondoren gehitu
            System.out.println("");
            System.out.println("Gerard Butler elementuaren ondoren
gehitu...");
            Aktorea aBerria=new Aktorea("Lena Headey");
            p.ondorenGehitu(aBerria,a1.getIzena());
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu baten ondoren gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");

            /*
             * 3.PROBA KASUA: Zerrenda hutsa
             */
            System.out.println("3.PROBA KASUA: Zerrenda hutsa ");
            p = null;
            p = new Pelikula("Silence");
            a1 = new Aktorea("Liam Neeson");

            // aurrean gehitu
            System.out.println("");
            System.out.println("Elementu bat aurrean gehitu");
            p.hasieranGehitu(a1);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
```

```
            System.out.println("Elementu bat hasieran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");
            p.ezabatuAktorea("Liam Neeson");

            // bukaeran gehitu
            System.out.println("");
            System.out.println("Elementu bat bukaeran gehitu");
            p.bukaeranGehitu(a1);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat bukaeran gehitzen
lortutako zerrenda berria...");
            System.out.println(" ================================");
            p.imprimatuAktoreak();
            System.out.println(" ================================");

      }

}
```

ProbaPelikulaOrderedCircularLL.java

```
package praktika2.test;

import praktika.akt.Aktorea;
import praktika.akt.Pelikula;
import praktika.akt.Stopwatch;

public class ProbaPelikulaOrderedCircularLL {

      public static void main(String[] args) {
            Stopwatch timer = new Stopwatch();
            /*
             * 1.PROBA KASUA: Elementu bat baino gehiago dituen
zerrenda
             */
            Aktorea a1 = new Aktorea("Bridget Jones I");
            Aktorea a2 = new Aktorea("Bridget Jones II");
            Aktorea a3 = new Aktorea("Bridget Jones III");
            Aktorea a4 = new Aktorea("Harry Potter I");
            Aktorea a5 = new Aktorea("Harry Potter II");
            Aktorea a6 = new Aktorea("Harry Potter III");
            Aktorea a7 = new Aktorea("Harry Potter IV");
            Aktorea a8 = new Aktorea("Harry Potter V");
            Aktorea a9 = new Aktorea("Harry Potter VI");
            Aktorea a10 = new Aktorea("Harry Potter VII");

            Pelikula p = new Pelikula("Colin Firth");
            p.gehituAktorea(a1);
            System.out.println("pelikula bat gehitzen emandako denbora
--> " + timer.elapsedTime());
            p.gehituAktorea(a2);
            p.gehituAktorea(a3);
            p.gehituAktorea(a4);
            p.gehituAktorea(a5);
            p.gehituAktorea(a6);
            p.gehituAktorea(a7);
```

```
            p.gehituAktorea(a8);
            p.gehituAktorea(a9);
            p.gehituAktorea(a10);

            // elementuak inprimatu
            System.out.println(" Zerrenda ..............");
            System.out.println("");
            p.imprimatuAktoreak();
            System.out.println("Zerrenda inprimatzen emandako denbora -
-> " + timer.elapsedTime());

            // elementu kopurua
            System.out.println("");
            System.out.println(p.getIzenburua() + " aktoreak duen
pelikula kopurua");
            System.out.println(p.aktoreKopurua());

            // elementua dagoen konprobatu
            /* elementua ez dago */
            System.out.println("");
            System.out.println("ELEMENTU BAT LISTAN DAGOEN EDO EZ
KONPROBATU");
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a1));
            System.out.println(a1.getIzena() + " pelikula bilatzen
emandako denbora --> " + timer.elapsedTime());
            System.out.println(a1.getIzena() + " zerrendan dago? " +
p.findAktorea(a1.getIzena()));

            Aktorea a15 = new Aktorea("Harry PoPoPoPotter VII");
            System.out.println(a15.getIzena() + " zerrendan dago? " +
p.badagoAktorea(a15));
            System.out.println(a15.getIzena() + " pelikula bilatzen
emandako denbora --> " + timer.elapsedTime());
            System.out.println(a15.getIzena() + " zerrendan dago? " +
p.findAktorea(a15.getIzena()));

            // azken elementua
            System.out.println("");
            System.out.println(" Azken elementua ");
            System.out.println(" ==================================");
            p.getListaAktoreak().last().imprimatu();
            System.out.println(" ==================================");

            // lehenengo elementua
            System.out.println("");
            System.out.println(" Lehenengo elementua: ");
            System.out.println(" ==================================");
            p.getListaAktoreak().first().imprimatu();
            System.out.println(" ==================================");

            // txertatu
            System.out.println("");
            System.out.println("Elementu bat txertatu");
            Aktorea proba=new Aktorea("Proba");
            p.gehituAktoreOrdenatua(proba);
            System.out.println("Emandako denbora --> " +
timer.elapsedTime());
            System.out.println("Elementu bat txertatzean lortutako
zerrenda berria...");
            System.out.println(" ==================================");
```

```
            p.imprimatuAktoreak();
            System.out.println(" ==================================");

        }

}
```

ONDORIOAK

Taldekideok praktika burutu ondoren batu gara eta egindako lanari buruz hitz egitean honako gogoeta eta ondorioak atera dira:

- Nodoekin tratatzen ikasi dugu.
- Gure kodea egiten hasi baino lehen zer egingo dugun eta nola egingo dugun pentsatzearen garrantzia ikusi dugu, batez ere datu kantitate handiekin lan egitean, hasieran pentsatzeko denbora hartu ezik, proiektuak kale egin dezakeela ikusi baitugu.
- Gure kideengan fidatzea ze garrantzitsua den ikasi dugu bakarka egitea oso neketsua izango bailitzateke proiektu hau eta lana banatzean lan karga asko txikitzea lortu dugu. Bakoitzak bere lana burutuz eta kideari behar izanez gero lagunduz proiektua lan eginda baina erraz aurrera eraman dugu.
- Egitura Estekatu Zirkularrak erabiltzen ikasi dugu, ordenatuak edo ez ordenatuak izanik.
- Guk sortutako algoritmoen kalkulua egiten ikasi dugu, algoritmoen kasu posible guztiak aztertuz.