

P1:



ACTORS AND MOVIES

Fecha: 9-X-2015

Participante: Andrea Siles Zeballos

Índice de contenido

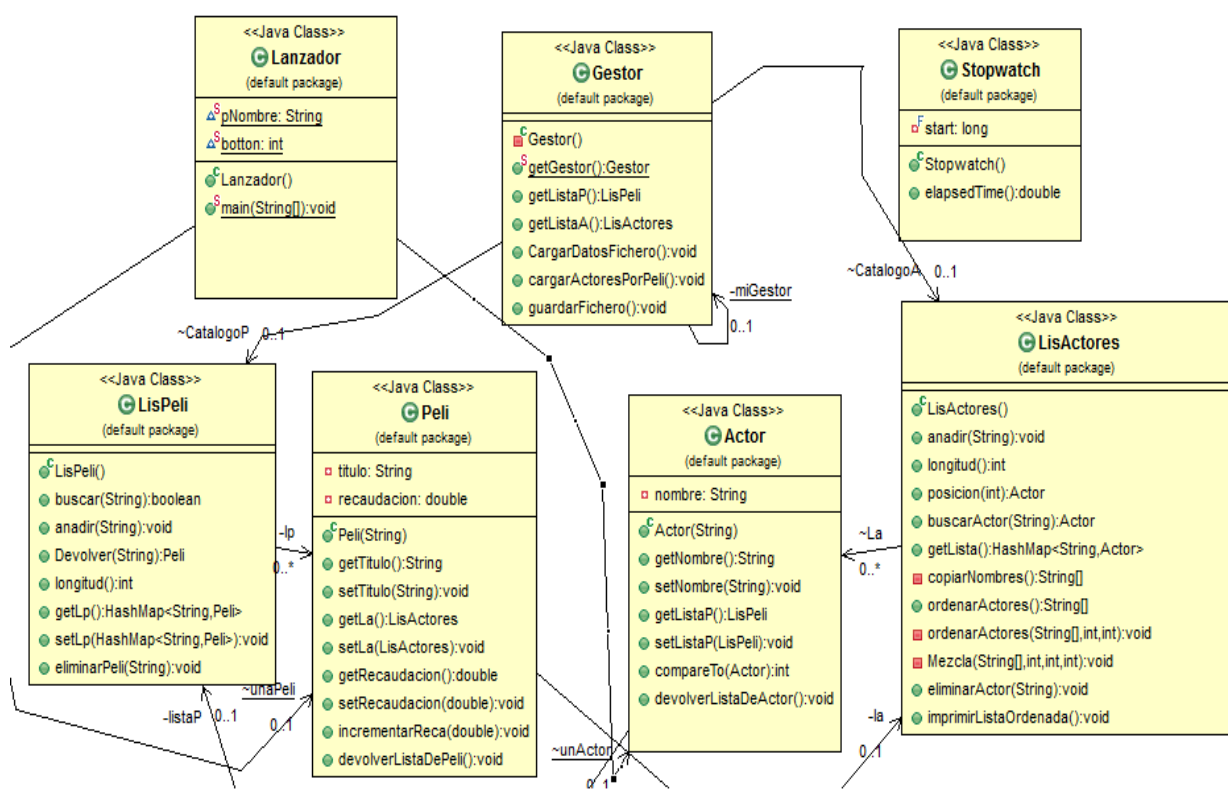
1	Introducción	3
2	Diseño de las clases.....	3
3	Descripción de las estructuras de datos principales	3
4	Diseño e implementación de los métodos principales	3
	4.1. Método buscarActor	
	4.2. Método ordenarActores	
	4.3. Método añadir	
	4.4. Método eliminar	
	3
5	Código.....	4
6	Conclusiones	4

1 Introducción

En la actualidad una gran cantidad de datos se procesan a mayor velocidad a medida que pasan los años, gracias a las nuevas tecnologías, avances en la algorítmica...etc. Esto es debido a la necesidad de almacenar datos cada vez mayores y a la obtención de ellos desde cualquier lugar del mundo y en un tiempo mínimo con el fin de desarrollar las tareas de forma más cómoda.

Esta Práctica se basa en gestionar un gran volumen de datos en el menor tiempo posible, utilizando algoritmos y estructuras de datos más eficientes.

2 Diseño de las clases



3 Descripción de las estructuras de datos principales

He utilizado la siguiente estructura de datos: Arrays y HashMap que corresponde a las listas: `ListaPeli` y `LisActores` como se ve en el diagrama UML.

Hash: es una estructura de datos que asocia *llaves* o *claves* con *valores*. La operación principal que soporta de manera eficiente es la *búsqueda*: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una función hash en un hash, un número que

identifica la posición (*casilla ocubeta*) donde la tabla hash localiza el valor deseado. Las tablas hash proveen tiempo constante de búsqueda promedio $O(1)$, en casos particularmente malos el tiempo de búsqueda puede llegar a $O(n)$.

Array: El array lo he utilizado para aplicar el método de ordenación MergeSort(), uso el array como variable local para devolver una lista ordenada.

un Array es un arreglo de elementos que crece o mengua dinámicamente conforme los elementos se agregan o se eliminan. Es un arreglo de tamaño fijo, pero cuyo tamaño se fija cuando se asigna por primera vez.

4 Diseño e implementación de los métodos principales

4.1 Método buscarActor

public Actor buscarActor(String A) {

Precondición: La lista estará con al menos un actor, podemos pensar que la lista este vacia tambien.

Postcondición: se devolverá el actor cuyo nombre es pasado por parámetro
casos de prueba:

Lista Inicial	"andrea"	Resultdo
[]	"andrea"	"lista vacia"
["pepe", "mario"]	"andrea"	Null
["ana", "karen", "joe"]	"andrea"	Null
["andrea", "karen", "joe"]	"andrea"	Actor entero("andrea", Actor)
["karen", "andrea", "joe"]	"andrea"	Actor entero("andrea", Actor)
["joe", "karen", "andrea"]	"andrea"	Actor entero("andrea", Actor)

Coste: el algoritmo, en el caso peor, puede que el coste sea $O(n)$, Pero al ser hashMap nos presentan algoritmos de búsqueda muy rápidas y normalmente su coste suele ser $O(1)$.

4.2 Método ordenarActores

public String[] ordenarActores() {

Precondición: La lista contendrá al menos un actor, y estará desordenada

Postcondición: el array estará ordenado alfabéticamente

He utilizado el algoritmo de ordenación mergeSort() por [John Von Neumann](#)

1. Si la longitud de la lista es 0 ó 1, entonces ya está ordenada. En otro caso:
2. Dividir la lista desordenada en dos sublistas de aproximadamente la mitad del tamaño.
3. Ordenar cada sublista [recursivamente](#) aplicando el ordenamiento por mezcla.
4. [Mezclar](#) las dos sublistas en una sola lista ordenada.

Casos Prueba:

Lista Inicial	Lista Final
[a]	[a]
[b,a,c,d,e]	[a,b,c,d,e]
[e,b,d,c,a]	[a,b,c,d,e]

La primera partición tendrá un coste $O(n)$, luego $n/2 \dots n/4 \dots$ etc

En el caso peor, medio y mejor este algoritmo tendrá un coste $O(n \log n)$

4.3 Método añadir

public void anadir(String pNombre) {

Precondición:

Postcondición: nuestra lista tendrá un elemento mas

Casos Prueba:

Lista Inicial	elemento	ListaFinal
[]	"andrea"	["andrea"]
["pepe", "mario"]	"andrea"	["pepe", "mario", "andrea"]
["andrea", "karen", "joe"]	"andrea"	["andrea", "karen", "joe"]

En el caso peor, mejor y medio al usar una hashMap es coste es $O(1)$

4.4 Método eliminar

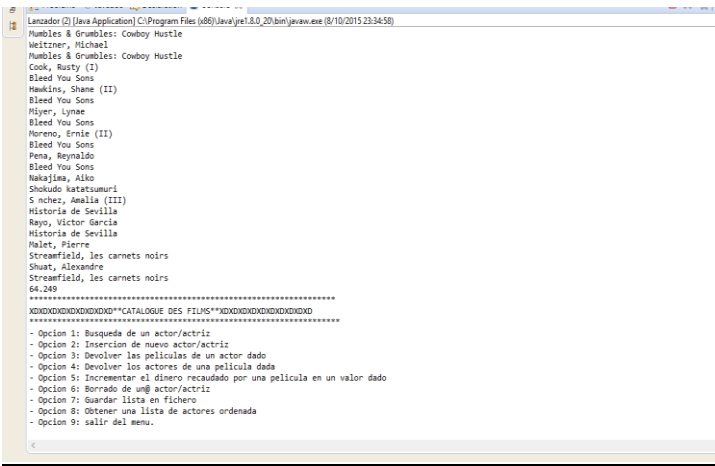
public void eliminarActor(String nombre) {

Postcondición: nuestra lista tendra un elemento menos

Lista Inicial	elemento	ListaFinal
[]	"andrea"	[]
["pepe", "mario"]	"andrea"	["pepe", "mario"]
["andrea", "karen", "joe"]	"andrea"	["karen", "joe"]
["karen", "andrea", "joe"]	"andrea"	["karen", "joe"]
["joe", "karen", "andrea"]	"andrea"	["karen", "joe"]

En el caso peor, mejor y medio al usar una hashMap es coste es $O(1)$

5 Código

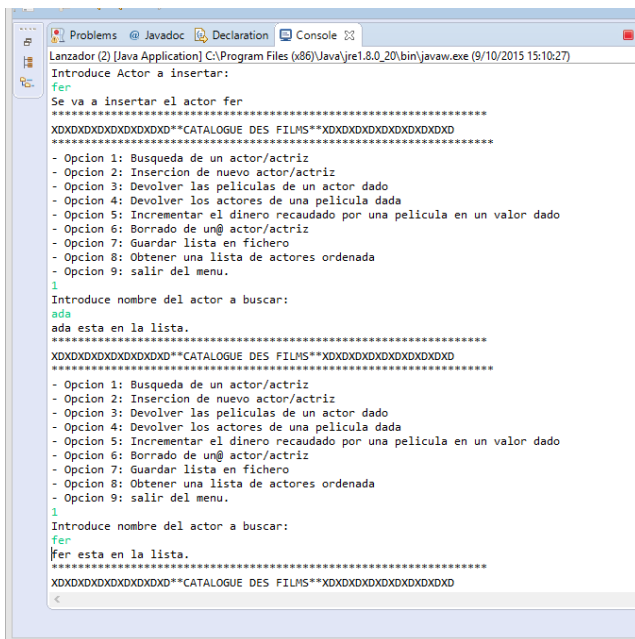


#####

```
public void anadir(String pNombre) {  
    if(!this.La.containsKey(pNombre))  
        this.La.put(pNombre, new Actor(pNombre));  
}
```

#####

Pruebas



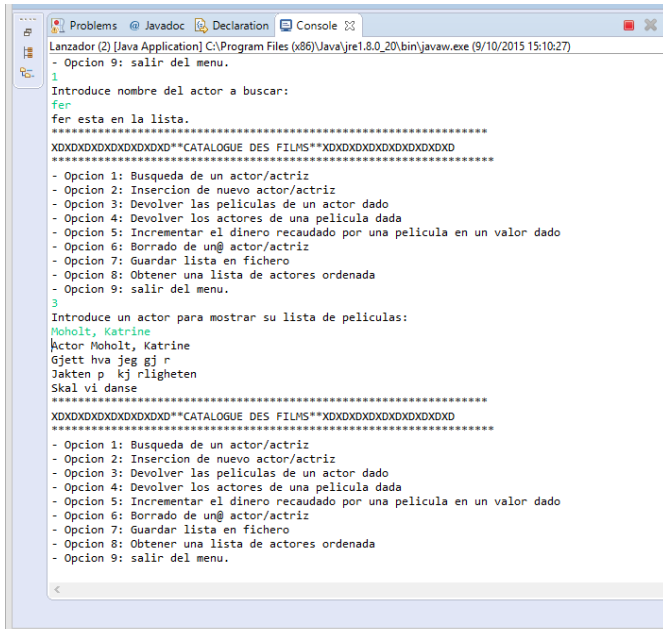
```
Problems Javadoc Declaration Console  
Lanzador (2) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_20\bin\javaw.exe (9/10/2015 15:10:27)  
Introduce Actor a insertar:  
fer  
Se va a insertar el actor fer  
*****  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
*****  
- Opcion 1: Busqueda de un actor/actriz  
- Opcion 2: Insercion de nuevo actor/actriz  
- Opcion 3: Devolver las peliculas de un actor dado  
- Opcion 4: Devolver los actores de una pelicula dada  
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado  
- Opcion 6: Borrado de un@ actor/actriz  
- Opcion 7: Guardar lista en fichero  
- Opcion 8: Obtener una lista de actores ordenada  
- Opcion 9: salir del menu.  
1  
Introduce nombre del actor a buscar:  
ada  
ada esta en la lista.  
*****  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
*****  
- Opcion 1: Busqueda de un actor/actriz  
- Opcion 2: Insercion de nuevo actor/actriz  
- Opcion 3: Devolver las peliculas de un actor dado  
- Opcion 4: Devolver los actores de una pelicula dada  
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado  
- Opcion 6: Borrado de un@ actor/actriz  
- Opcion 7: Guardar lista en fichero  
- Opcion 8: Obtener una lista de actores ordenada  
- Opcion 9: salir del menu.  
1  
Introduce nombre del actor a buscar:  
fer  
fer esta en la lista.  
*****  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
*****  
<
```


#####

```
public void devolverListaDeActor() {  
  
    for (Entry<String, Peli> unaPeli : this.listaP.getLp().entrySet()) {  
        String clave = unaPeli.getKey();  
        System.out.println(clave);  
    }  
}
```


#####

Pruebas



```
Lanzador (2) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_20\bin\javaw.exe (9/10/2015 15:10:27)
- Opcion 9: salir del menu.
1
Introduce nombre del actor a buscar:
fer
fer esta en la lista.
*****
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*****
- Opcion 1: Busqueda de un actor/actriz
- Opcion 2: Insercion de nuevo actor/actriz
- Opcion 3: Devolver las peliculas de un actor dado
- Opcion 4: Devolver los actores de una pelicula dada
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado
- Opcion 6: Borrado de un actor/actriz
- Opcion 7: Guardar lista en fichero
- Opcion 8: Obtener una lista de actores ordenada
- Opcion 9: salir del menu.
3
Introduce un actor para mostrar su lista de peliculas:
Moholt, Katrine
Actor Moholt, Katrine
Gjett hva jeg gj r
Jakten p kj rligheten
Skal vi danse
*****
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*****
- Opcion 1: Busqueda de un actor/actriz
- Opcion 2: Insercion de nuevo actor/actriz
- Opcion 3: Devolver las peliculas de un actor dado
- Opcion 4: Devolver los actores de una pelicula dada
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado
- Opcion 6: Borrado de un actor/actriz
- Opcion 7: Guardar lista en fichero
- Opcion 8: Obtener una lista de actores ordenada
- Opcion 9: salir del menu.
```

```
#####
#####
```

```
public void devolverListaDePeli() {
```

```
    for (Entry<String, Actor> unActor : this.getLa().getLista().entrySet()) {
        String clave = unActor.getKey();
        System.out.println(clave);
    }
```

```
#####
#####
```

Pruebas

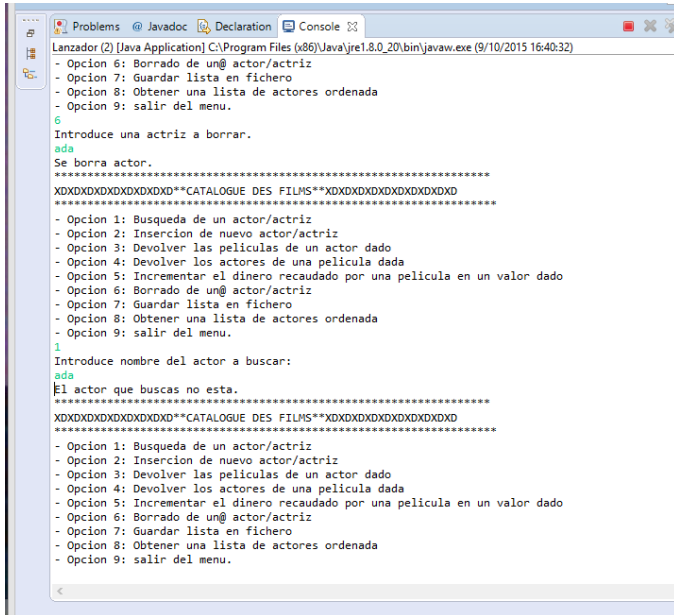
#####

```
public void eliminarActor(String nombre) {  
    this.La.remove(nombre);  
}
```

#####

#####

prueba

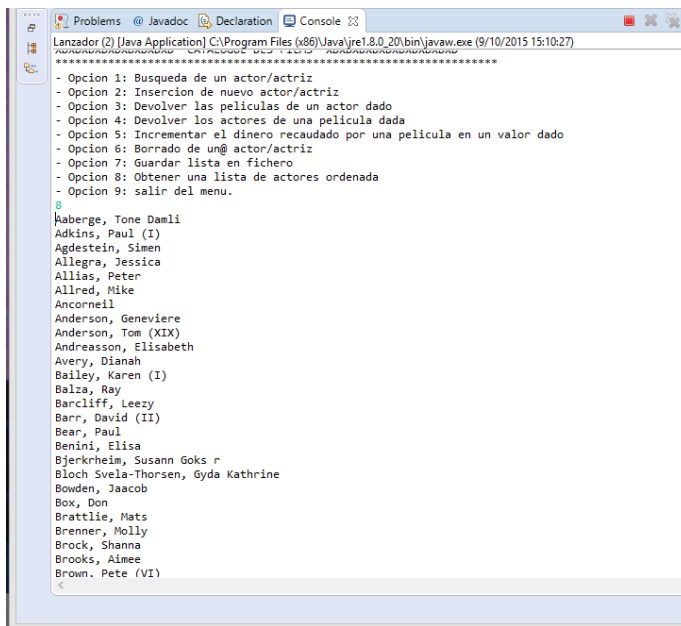


```
Lanzador (2) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_20\bin\javaw.exe (9/10/2015 16:40:32)  
- Opcion 6: Borrado de un@ actor/actriz  
- Opcion 7: Guardar lista en fichero  
- Opcion 8: Obtener una lista de actores ordenada  
- Opcion 9: salir del menu.  
6  
Introduce una actriz a borrar.  
ada  
Se borra actor.  
*****  
XXXXXXXXXXXXXXXXCATALOGUE DES FILMS**XXXXXXXXXXXXXXXX  
*****  
- Opcion 1: Busqueda de un actor/actriz  
- Opcion 2: Insercion de nuevo actor/actriz  
- Opcion 3: Devolver las peliculas de un actor dado  
- Opcion 4: Devolver los actores de una pelicula dada  
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado  
- Opcion 6: Borrado de un@ actor/actriz  
- Opcion 7: Guardar lista en fichero  
- Opcion 8: Obtener una lista de actores ordenada  
- Opcion 9: salir del menu.  
1  
Introduce nombre del actor a buscar:  
ada  
El actor que buscas no esta.  
*****  
XXXXXXXXXXXXXXXXCATALOGUE DES FILMS**XXXXXXXXXXXXXXXX  
*****  
- Opcion 1: Busqueda de un actor/actriz  
- Opcion 2: Insercion de nuevo actor/actriz  
- Opcion 3: Devolver las peliculas de un actor dado  
- Opcion 4: Devolver los actores de una pelicula dada  
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado  
- Opcion 6: Borrado de un@ actor/actriz  
- Opcion 7: Guardar lista en fichero  
- Opcion 8: Obtener una lista de actores ordenada  
- Opcion 9: salir del menu.
```

#####

#####

Prueba Ordenar



```
#####
#####
```

```
public String[] ordenarActores() {
    String[] actores = this.copiarNombres();
    ordenarActores(actores, 0, actores.length - 1);
    return actores;
}
```

```
private void ordenarActores(String[] actores, int inicio, int fin) {
    if (inicio < fin) {
        ordenarActores(actores, inicio, ((inicio + fin) / 2));
        ordenarActores(actores, ((inicio + fin) / 2) + 1, fin);
        Mezcla(actores, inicio, ((inicio + fin) / 2), fin);
    }
}
```

```
private void Mezcla(String[] actores, int inicio, int centro, int fin) {
    String[] laMezcla = new String[fin - inicio + 1];

    int izq = inicio;
    int der = centro + 1;
    int k = 0;
    while (izq <= centro && der <= fin) {
        if (actores[izq].compareTo(actores[der]) <= 0) {
            laMezcla[k] = actores[izq];
            k++;
            izq++;
        } else {
```

```

        laMezcla[k] = actores[der];
        k++;
        der++;
    }
}
if (izq > centro) {
    while (der <= fin) {
        laMezcla[k] = actores[der];
        k++;
        der++;
    }
} else {
    while (izq <= centro) {
        laMezcla[k] = actores[izq];
        k++;
        izq++;
    }
}
for (int j = inicio; j <= fin; j++) {
    actores[j] = laMezcla[j - inicio];
}
}

```

```

#####
#####

```

```

public void guardarFichero() throws IOException {
    Gestor.getGestor().CargarDatosFichero();
    String[] ListaO = Gestor.getGestor().getListaA().ordenarActores();
    try {

        PrintWriter pw = new PrintWriter("src/fichero/listag.txt");

        for (int j = 0; j < ListaO.length; j++) {
            String unActor = ListaO[j];
            LisPeli lp = Gestor.getGestor().getListaA()
                .buscarActor(unActor).getListaP();
            pw.println(unActor);
            for (Entry<String, Peli> unaPeli : lp.getLp().entrySet()) {
                String clave = unaPeli.getKey();
                pw.println("\t\t" + clave);
            }
        }

        pw.close();

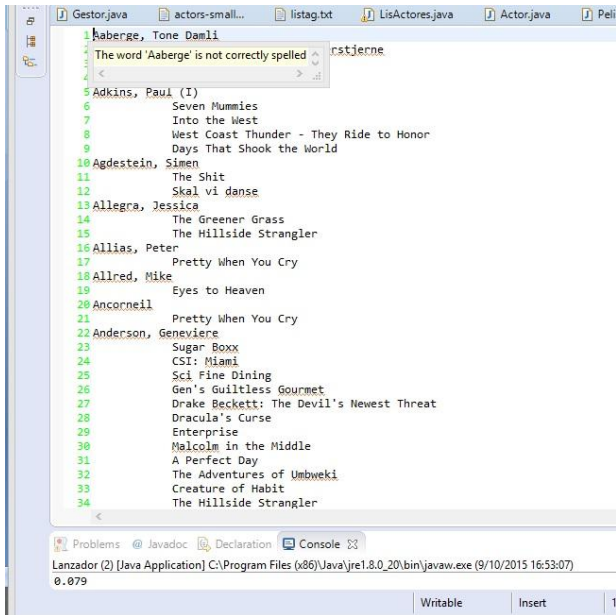
    } catch (Exception e) {
        e.printStackTrace();
    } finally {

```

```
}
```

```
}
```

```
#####  
#####
```



```
#####  
#####
```

```
public Actor buscarActor(String A)
```

```
{  
    Actor unActor = null;  
    if (!this.La.isEmpty()) {  
  
        unActor = this.La.get(A);  
    }  
    return unActor;  
}
```

```
#####
```

```

Problems @ Javadoc Declaration Console
Lanzador (2) [Java Application] C:\Program Files (x86)\Java\jre1.8.0_20\bin\javaw.exe (9/10/2015 15:10:27)
- Opcion 4: Devolver los actores de una pelicula dada
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado
- Opcion 6: Borrado de un actor/actriz
- Opcion 7: Guardar lista en fichero
- Opcion 8: Obtener una lista de actores ordenada
- Opcion 9: salir del menu.
1
Introduce nombre del actor a buscar:
Noen, Anita
Noen, Anita esta en la lista.
*****
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*****
- Opcion 1: Busqueda de un actor/actriz
- Opcion 2: Insercion de nuevo actor/actriz
- Opcion 3: Devolver las peliculas de un actor dado
- Opcion 4: Devolver los actores de una pelicula dada
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado
- Opcion 6: Borrado de un actor/actriz
- Opcion 7: Guardar lista en fichero
- Opcion 8: Obtener una lista de actores ordenada
- Opcion 9: salir del menu.
1
Introduce nombre del actor a buscar:
rachel
El actor que buscas no esta.
*****
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*****
- Opcion 1: Busqueda de un actor/actriz
- Opcion 2: Insercion de nuevo actor/actriz
- Opcion 3: Devolver las peliculas de un actor dado
- Opcion 4: Devolver los actores de una pelicula dada
- Opcion 5: Incrementar el dinero recaudado por una pelicula en un valor dado
- Opcion 6: Borrado de un actor/actriz
- Opcion 7: Guardar lista en fichero
- Opcion 8: Obtener una lista de actores ordenada
#####

```

6 Conclusiones

El uso del HashMap me ha permitido la reducción de líneas de código, . Gran parte de las gestiones de las diversas listas, con que cuenta la aplicación, se realiza en tiempo constante $O(1)$ como buscar, añadir , borrar, entre algún otro.

Con visión de unos posibles cambios para hacer aun mas rápida la ejecución, podríamos usar estructuras de datos Treemap, que añade los elementos de forma ordenada, ya que esta estructura está compuesta por arboles binarios y casi todas sus operaciones son en tiempo logarítmicos

Dificultades que he tenido ha sido en el diseño ,en elegir la estructura que iba a usar, y en plantearme como iba hacer el método de ordenación.