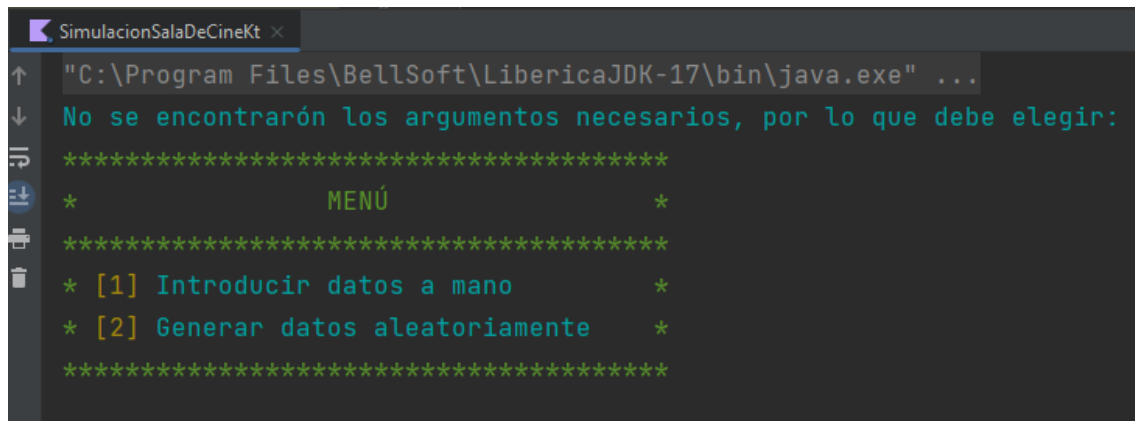


## Contenido

Inicialización de la “salaDeCine” y la matriz de “butacas” .....	2
1.Comprar entrada:.....	5
2.Reservar entrada:.....	5
3.Formalizar entrada:.....	6
4.Anular reserva/compra de entrada: .....	6
5.Conseguir informe de la sala:.....	6
6.Conseguir recaudación total en caja: .....	7
Los Tests: .....	7

Nota: Estaré comentando el programa tal y como lo hice en Kotlin, en el caso de que haya algo distinto en Java lo destacaré, todo lo demás es seguro afirmar que se mantuvo como mínimo, el razonamiento igual. Además, muchas veces he reiniciado el programa para hacer este escrito, por eso ciertas fotos pueden discrepar entre ellas.

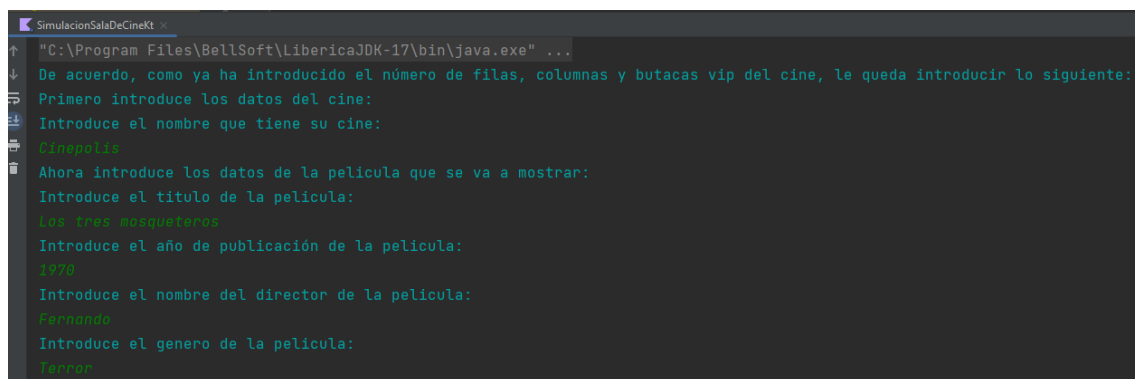
## Inicialización de la “salaDeCine” y la matriz de “butacas”



```
"C:\Program Files\BellSoft\LibericaJDK-17\bin\java.exe" ...  
No se encontraron los argumentos necesarios, por lo que debe elegir:  
*****  
*                MENÚ                *  
*****  
* [1] Introducir datos a mano          *  
* [2] Generar datos aleatoriamente     *  
*****
```

En esta primera imagen, lo que se nos muestra, son dos partes del programa.

La primera línea representa que a la hora de iniciar el programa, o bien no había argumentos de entrada válidos, o los necesarios, de manera que nos pasa a la siguiente parte. Debo destacar dos cosas de los argumentos antes de proceder. Primero que nada como funciona, en el caso de que los argumentos estén incompletos (no tengan por ejemplo: “-filas 5 -columnas 7 -butacasVip 6”), o sean algunos sino todos los valores numéricos incorrectos, nos pasará al menú representado en la imagen, en caso de que si sean correctos, nos mandará a rellenar los datos del cine restantes a mano, como en la imagen de abajo:



```
"C:\Program Files\BellSoft\LibericaJDK-17\bin\java.exe" ...  
De acuerdo, como ya ha introducido el número de filas, columnas y butacas vip del cine, le queda introducir lo siguiente:  
Primero introduce los datos del cine:  
Introduce el nombre que tiene su cine:  
Cinepolis  
Ahora introduce los datos de la película que se va a mostrar:  
Introduce el título de la película:  
Los tres mosqueteros  
Introduce el año de publicación de la película:  
1970  
Introduce el nombre del director de la película:  
Fernando  
Introduce el género de la película:  
Terror
```

Lo segundo a comentar de los argumentos, es que están implementados en la versión de Kotlin, pero no en la de Java.

Proseguimos con el menú que te permite seleccionar como construir el cine, puedes introducir todos los datos a mano:

```
SimulacionSalaDeCineKt x
C:\Program Files (x86)\Java\jdk-17\bin\java.exe ...
No se encontrarón los argumentos necesarios, por lo que debe elegir:
*****
*                               *
*           MENÚ                 *
*                               *
*****
* [1] Introducir datos a mano     *
* [2] Generar datos aleatoriamente *
*****
1
Primero introduce los datos del cine:
Introduce el nombre que tiene su cine:
Cinepolis
Introduce el número de filas que tiene su cine:
4
Introduce el número de columnas que tiene su cine:
4
Introduce el número de butacas vip que tiene su cine::
4
Ahora introduce los datos de la película que se va a mostrar:
Introduce el título de la película:
Don Quijote
Introduce el año de publicación de la película:
1950
Introduce el nombre del director de la película:
Ramón
Introduce el genero de la película:
Aventuras
```

Siendo un conjunto de funciones las que te permiten escribir los datos, y que te impiden escribirlos de manera incorrecta, por ejemplo:

```
*****
*                               *
*           MENÚ                 *
*                               *
*****
* [1] Introducir datos a mano     *
* [2] Generar datos aleatoriamente *
*****
hola
La opción introducida no es válida, vuelve a probar:
```

Cabe destacar, que en todas las partes del programa donde hace falta introducir algo por teclado, está implementado el sistema necesario para filtrar las entradas.

En el caso de que seleccionemos la segunda opción, todos los datos se generarán aleatoriamente y punto.

Una vez introduzcamos los datos, o estos se generen automáticamente, la “salaDeCine” se creará, junto a la “película”, a través de las clases Factory creadas para ello. Adicionalmente, cabe destacar que una vez se haya creado la “salaDeCine”, aparecerá inicializada a nulos la matriz con las butacas del cine, y que a está se la añadirán “butacas” a través de una función en Java y del método init en Kotlin.

Las “butacas” tendrán un identificador (calculado según su posición de fila y columna), un tipoButaca (normal o vip, que son un tipo enum, con un valor de lo que cuesta la butaca asociado a ese enum) y un estado (libre, reservado o comprado). Una vez colocadas “butacas” en todas las posiciones de la matriz, inicializadas como tipo normal, se llama a otra función que transformará aleatoriamente a algunas de ellas a tipo vip, según el número de butacasVip que introdujéramos anteriormente.

Sin embargo, en Java lo hice distinto, las “butacas” tienen un identificador, un estado, y un booleano “esVip” que marcará true en caso positivo y false en caso negativo, de manera que no hay como tal un tipo de butaca, y por ende el dinero que cuesta cada una se guarda de otra manera. Además en Java para llenar la matriz con “butacas”, llama a una función que introduce las butacas y que de paso hace un 2/5 de las veces la butaca recién creada, vip.

En ambas versiones del programa tenemos una Factory de “butacas”. Tras inicializar la matriz con las butacas, los identificadores (por ejemplo: A1) de estás se representarán con colores, verde si están libres, amarillas si están reservadas y rojas si están ocupadas, adiconamente la letra del identificador cambiará a un color especial, el rosa, en el caso de que esa butaca sea vip:

La sala del cine es la siguiente:

A1	A2	A3	A4	A5	A6	A7	A8	A9
B1	B2	B3	B4	B5	B6	B7	B8	B9
C1	C2	C3	C4	C5	C6	C7	C8	C9
D1	D2	D3	D4	D5	D6	D7	D8	D9
E1	E2	E3	E4	E5	E6	E7	E8	E9

Lo siguiente del programa es el menú principal con sus diversas opciones:

```
Seleccione la opción que desea:
*****
*                               MENÚ                               *
*****
* [1] Comprar entrada                                             *
* [2] Reservar entrada                                           *
* [3] Formalizar reserva de entrada                             *
* [4] Anular reserva o compra de entrada                       *
* [5] Conseguir informe de la sala                             *
* [6] Conseguir recaudación total de la caja                   *
* [0] Salir                                                       *
*****
```

## 1.Comprar entrada:

Primero, representa la sala de cine para que sepas que sillas están ocupadas, etc. Después pide al usuario que introduzca el identificador de la butaca a comprar, o “stop” para salir del menú de compra. En caso de que una butaca este comprada, te impedirán comprarla. Finalmente proporciona un mensaje destacando el número de butacas que fueron compradas.

La función que sirve para marcar una butaca como comprada, lo que hace es cambiar el estado de la butaca en cuestión a “comprada”.

```
Introduzca el identificador de la butaca que quieras comprar ( "A1", por ejemplo ) o escribir "stop" para dejar de comprar butacas:
A6
De acuerdo, has comprado la butaca A6
Ademas, cabe destacar que la butaca: A6, es VIP
Introduzca el identificador de la butaca que quieras comprar ( "A1", por ejemplo ) o escribir "stop" para dejar de comprar butacas:
A6
La butaca seleccionada ya está ocupada
Introduzca el identificador de la butaca que quieras comprar ( "A1", por ejemplo ) o escribir "stop" para dejar de comprar butacas:
stop
Has comprado un total de 1 butacas
```

## 2.Reservar entrada:

Primero, representa la sala de cine para que sepas que sillas están ocupadas, etc. Después pide al usuario que introduzca el identificador de la butaca a reservar, o “stop” para salir del menú de reserva. En caso de que una butaca este comprada o reservada, te impedirán reservarla. Finalmente proporciona un mensaje destacando el número de butacas que fueron reservadas.

La función que sirve para marcar una butaca como reservada, lo que hace es cambiar el estado de la butaca en cuestión a “reservada”.

```
Introduzca el identificador de la butaca que quieras reservar ( "A1", por ejemplo ) o escribir "stop" para dejar de reservar butacas:
B6
De acuerdo, has reservado la butaca B6
Ademas, cabe destacar que la butaca: B6, es VIP
Introduzca el identificador de la butaca que quieras reservar ( "A1", por ejemplo ) o escribir "stop" para dejar de reservar butacas:
B6
La butaca seleccionada ya está ocupada
Introduzca el identificador de la butaca que quieras reservar ( "A1", por ejemplo ) o escribir "stop" para dejar de reservar butacas:
stop
Has reservado un total de 1 butacas
```

### 3. Formalizar entrada:

Primero, representa la sala de cine para que sepas que sillas están ocupadas, etc. Después pide al usuario que introduzca el identificador de la butaca a formalizar, o “stop” para salir del menú de formalización. En caso de que una butaca no este reservada, te impedirán formalizarla. Finalmente proporciona un mensaje destacando el número de butacas que fueron formalizadas.

La función que sirve para marcar una butaca como formalizada/comprada, lo que hace es cambiar el estado de la butaca en cuestión a “comprada”.

```
Introduzca el identificador de la butaca que quieras formalizar de reservas a compras ( "A1", por ejemplo ) o escribir "stop" para dejarlo:
A6
De acuerdo, has formalizado la butaca B6
Ademas, cabe destacar que la butaca: B6, es VIP
Introduzca el identificador de la butaca que quieras formalizar de reservas a compras ( "A1", por ejemplo ) o escribir "stop" para dejarlo:
A6
Esa butaca no estaba reservada, por lo que no puedes formalizarla
Introduzca el identificador de la butaca que quieras formalizar de reservas a compras ( "A1", por ejemplo ) o escribir "stop" para dejarlo:
stop
Has formalizado un total de 1 butacas
```

### 4. Anular reserva/compra de entrada:

Primero, representa la sala de cine para que sepas que sillas están ocupadas, etc. Después pide al usuario que introduzca el identificador de la butaca a anular reserva/compra, o “stop” para salir del menú de anulación. En caso de que una butaca no este reservada o comprada, te impedirán anular la reserva/compra. Finalmente proporciona un mensaje destacando el número de butacas cuya reserva/compra fue anulada.

La función que sirve para marcar una butaca como anulada/libre, lo que hace es cambiar el estado de la butaca en cuestión a “libre”.

Por ahora, las funciones comentadas, tenían el mismo sistema para localizar la butaca cuyo indentificador escribió el usuario, se recorre la matriz comparando identificadores, si se encuentra una igualdad, se devuelve la fila y columna donde ocurrió la igualdad, y en caso contrario tras recorrer toda la matiz, se devuelve como valor de fila y columna un -1. En kotlin, para devolver esos datos, los pase como un Pair<Int, Int>, mientras que en java o lo devolvía como un mensaje vacio (equivalente a recibir un -1 en fila y columna en Koltin) o con en patrón “fila-columna”. Según recibiera una información un otra le pasaría un mensaje al usuario para comentar que se elaboro la acción correctamente, o para decirle que no se encontró esa butaca.

### 5. Conseguir informe de la sala:

Primero muestra los datos del cine y la película que se representará, después, para variar, cambia la forma con la que muestra la matriz de butacas a una versión simplificada, que no especifica cuando es vip o no, y que en vez de mostrar el identificador, muestra una “l” si es “libre”, “r” si está “reservada” y “c” si está “comprada”, cuenta con colores. Por ultimo, enseña el número de butacas que hay de cada tipo.

6. Conseguir recaudación total en caja:

En un mensaje se nos da el dinero total en caja con dos decimales (usando un `String.format()`), como se calcula el dinero es muy simple, recorreremos la matriz y cuando un butaca este marcada como “comprada”, sumaremos lo que cuesta esa butaca al total, lo que cuesta la butaca dependerá de si es normal o vip. En clase cuando comentamos los precios, fueron distintos de las dos veces, por ende no puedo confirmar que tenga los valores de la butacas igual que el resto.

El dinero total en caja en la actualidad es de: 29,35€

## Los Tests:

Lo último que queda por comentar son los tests. Lo primero es lo primero, y es mostrar que todos los tests pasan:

## Kotlin:

salaCineTests x

✓ Tests passed: 32 of 32 tests – 84 ms

✓ salaCineTests (SalaDeCineTests)	84 ms	"C:\Program Files\Bells
✓ reservarEntradaTest()	16 ms	l l l l l
✓ calcularDineroEnCajaTest()	1 ms	l l l l l
✓ generoNoValidoTest()	25 ms	l l l l l
✓ directorNoValidoTest()	2 ms	l l l l l
✓ hallarIdentificadorButacaTest()		l l l l l
✓ crearSalaDeCineTest()		kotlin.Unit
✓ tituloValidoTest()		l l l l l
✓ numeroButacasVipNoValidoTest()	5 ms	l l l l l
✓ generoValidoTest()		l l l l l
✓ anularEntradaTest()		l l l l l
✓ nombreNoValidoTest()	2 ms	l l l l l
✓ opcionIntroducirDatosValidaTest()		Estás en el cine s, en
✓ butacaValidaTest()	2 ms	l l l l l
✓ nombreValidoTest()	1 ms	l l l l l
✓ crearPeliculaTest()		l l l l l
✓ comprarEntradaTest()	1 ms	l l l l l
✓ filaColumnaNoValidaTest()	3 ms	Hay un total de 25 buta
✓ representarButacasTest()		Hay un total de 0 butac
✓ filaColumnaValidaTest()	2 ms	Hay un total de 0 butac
✓ opcionValidaTest()		
✓ contarNumeroDeButacasPorTipoTest()	1 ms	kotlin.Unit
✓ informeDeLaSalaDeCineTest()	1 ms	Estás en el cine s, en
✓ numeroButacasVipValidoTest()	1 ms	l l l l l
✓ opcionIntroducirDatosNoValidaTest()	4 ms	l l l l l
✓ crearButacaTest()		l l l l l
✓ directorValidoTest()	1 ms	l l l l l
✓ buscarButacaPorIdentificadorTest()	2 ms	l l l l l
✓ opcionNoValidaTest()	6 ms	Hay un total de 25 buta
✓ añoPublicacionNoValidoTest()	3 ms	Hay un total de 0 butac
✓ añoPublicacionValidoTest()	1 ms	Hay un total de 0 butac



Java:

SalaDeCineTest x

✓ Tests passed: 30 of 30 tests - 100 ms

✓ SalaDeCineTest (SalaDeCineTests) 100 ms

✓ reservarEntradaTest()	32 ms	✓	✓	✓	✓	✓
✓ calcularDineroEnCajaTest()	1 ms	✓	✓	✓	✓	✓
✓ generoNoValidoTest()	4 ms	✓	✓	✓	✓	✓
✓ directorNoValidoTest()	3 ms	✓	✓	✓	✓	✓
✓ hallarIdentificadorButacaTest()	1 ms	✓	✓	✓	✓	✓
✓ crearSalaDeCineTest()	4 ms	✓	✓	✓	✓	✓
✓ tituloValidoTest()	2 ms	✓	✓	✓	✓	✓
✓ generoValidoTest()	1 ms	✓	✓	✓	✓	✓
✓ anularEntradaTest()	1 ms	✓	✓	✓	✓	✓
✓ nombreNoValidoTest()	1 ms	✓	✓	✓	✓	✓
✓ opcionIntroducirDatosValidaTest()	1 ms	✓	✓	✓	✓	✓
✓ butacaValidaTest()	2 ms	✓	✓	✓	✓	✓
✓ nombreValidoTest()	1 ms	✓	✓	✓	✓	✓
✓ crearPeliculaTest()	1 ms	✓	✓	✓	✓	✓
✓ comprarEntradaTest()	1 ms	✓	✓	✓	✓	✓
✓ filaColumnaNoValidaTest()	2 ms	✓	✓	✓	✓	✓
✓ representarButacasTest()	5 ms	✓	✓	✓	✓	✓
✓ filaColumnaValidaTest()		✓	✓	✓	✓	✓
✓ opcionValidaTest()	2 ms	✓	✓	✓	✓	✓
✓ contarNumeroDeButacasPorTipoTest()	5 ms					
✓ informeDeLaSalaDeCineTest()	15 ms					
✓ opcionIntroducirDatosNoValidaTest()	2 ms					
✓ crearButacaTest()	3 ms					
✓ directorValidoTest()						
✓ buscarButacaPorIdentificadorTest()	1 ms					
✓ opcionNoValidaTest()	2 ms					
✓ añoPublicacionNoValidoTest()	3 ms					
✓ añoPublicacionValidoTest()	1 ms					
✓ butacaNoValidaTest()	1 ms					
✓ tituloNoValidoTest()	2 ms					

Process finished with

En Kotlin, hay dos test más por temas de haber implementado los argumentos y la función de validar el número de butacas vip, cosas que la versión de Java no tiene.

Los test elaborados en ambas versiones son los mismos, tenemos todos los tests de las funciones que sirven para validar la entrada de datos como, las filas o columnas, los identificadores de las butacas, etc. De esos tests tenemos una versión que es con datos que son válidos, y otra con datos inválidos y comprobar que el mensaje de error que mostraría al usuario es el correcto.

Tenemos los tests de las funciones crear de los Factory (menos los crear aleatoriamente), donde comparamos la creación de un objeto a través del constructor del mismo y a través del método crear del Factory.

También están los tests de las funciones de comprar/reservar/... entrada, donde comparamos el estado de una butaca tras hacerla cambiar con la función correspondiente.

Las funciones de representar la matriz, informe de la sala, y ese tipo de funciones que dan un mensaje o texto en general también tienen test, pero no se si es correcto lo que hice, ya que la verdad no me lo parece.

Tenemos tests de funciones, que buscan y dan una posición de fila y columna según el identificador de la butaca, otra que entrega el número de butacas libres, reservadas, y compradas, de como hallar el identificador que tendría una butaca según su fila y columna, y calcular la recaudación total de la sala de cine.