TEMA 1. Programación básica

Práctica 1 Variables

Objetivo específico	Manejo de distintos tipos de variables y diferenciar entre ***var*** y ***val***
Recursos	Kotlinlang Kotlin
Actividades	Manejo de diferente tipo de variables
Producto(s)	Al final de la práctica deberá saber cuándo usar el tipo de dato ***var*** y cuando ***val***

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Observar el ejemplo

Aquí veremos una función llamada ***main*** donde se encontrarán dos variables una llamada "a" y otra "b". Nuestra variable a tiene un valor de 10 mientras que b tiene un 5. Al usar la palabra reservada **var** sabemos en automático que son variables y que más adelante si podemos cambiarles el valor a ambas variables. Debajo de las variables vemos ***print,*** con esta función podemos imprimir en consola que aparecerá en la parte inferior de nuestra página una vez que se corra nuestro código, debajo de ***print*** vemos ***println*** que, a diferencia del primero, este hará un salto de línea que sirve para dar mejor vista a lo que vayamos a imprimir en nuestra pantalla.

En este caso haremos una pequeña suma, después tendremos una resta, multiplicación, división y el resto de una división.

Como resultado de nuestro código tendremos:

```
Suma: 15
Resta: 5
Multiplicación: 50
División: 2
El módulo (resto): 0
```

PASO 3. Modificar la variable "b"

Ahora debajo de la línea `println(a + b)` modificaremos el valor de “b”, ponga el valor que usted desee. Ejemplo

```
fun main() {
    var a = 10
    var b = 5
    print("Suma: ")
    println(a + b)
    b = 8
    print("Resta: ")
    println(a - b)
}
```

Con esto vemos que podemos modificar nuestras variables las veces que se necesite, siempre y cuando sea del mismo tipo. Por ejemplo, en este caso nuestras variables son de tipo ***int*** y no podremos cambiarle el valor por un ***string*** tan fácil.

Ejercicio Extra.

No solo existe ***int*** como tipo de variable numérica, también existen otros tipos como ***long, float, double.*** Investigar e implementar ejemplos con estos tipos de variables ya que es común usar ***float*** o ***double*** por ejemplo al buscar una localidad en google maps necesitamos latitud y longitud que son de tipo ***double*** o ***float*** estos permiten decimales a diferencia de ***int*** o ***long***.

PASO 4. Variables alfanuméricas.

La variable ***string*** será la que más usemos como norma general, nos permite almacenar cualquier tipo de caracteres. Borraremos todo dentro de la función main y agregaremos los siguiente:

```
    val nombre = "Mi nombre es ..." //escriban su nombre en lugar de los ...
    val numeroFavorito: String = "Mi número favorito es el 3"
    print(nombre + "\n" + numeroFavorito)
```

Como verán está vez usamos ***val*** que significa que es una constante y que su valor no cambiará. En la tercera línea vemos que usamos el signo + y esto significa que estamos CONCATENANDO, que quiere decir la unión de dos o más cadenas de texto en una, que en lugar de escribir 2 veces print solo lo usamos una vez y con esto logramos imprimir las dos variables y ahorramos líneas en nuestro código. “\n” significa salto de línea.

Resultado:

```
Mi nombre es ...
Mi número favorito es el 3
```

PASO 5. Imprimir diferentes tipos de variables

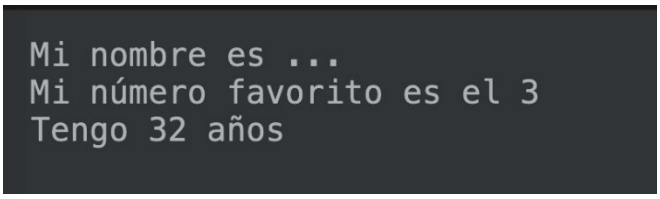
Con el paso anterior podemos guardar en una sola variable el valor de una variable ***numérica*** y el valor de otra variable ***string.*** Copien el siguiente código y peguen dentro de la función main en kotlinlang.

```
    val nombre = "Mi nombre es ..." //escriban su nombre en lugar de los ...
```

```
    val numeroFavorito: String = "Mi número favorito es el 3"
    var años = 32
    var edad = "Tengo $años años"
    print(nombre + "\n" + numeroFavorito + "\n" + edad)
```

En este caso agregamos dos líneas de código, tenemos una variable edad de tipo ***int*** y otra variable edad de tipo string. En la variable edad vemos el símbolo $ con esto estamos concatenando nuestra variable años con la variable edad, y después agregamos la variable edad a nuestro print e imprimimos para ver su valor.

Resultado:



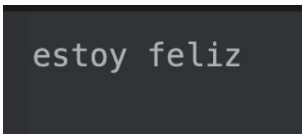
PASO 6. Variables de tipo boolean

Nos queda una última variable muy sencilla, pero a la vez muy práctica. Se tratan de los ***Booleanos***. Los Booleanos son variables que solo pueden ser verdaderas o falsas (true o false). Su uso es muy amplio sobre todo en las condiciones. Copiamos y pegamos el siguiente código dentro de la función main.

```
    var estoyFeliz: Boolean = true
    if (estoyFeliz) {
        print("estoy feliz")
    } else {
        print("estoy triste")
    }
```

Aquí tenemos la variable estoyFeliz con el valor true y abajo tenemos una condición if donde nos serán muy útiles estos tipos de variables. En la condición Si estoyFeliz es igual a true se imprimirá estoy feliz en pantalla, si no se imprimirá estoy triste. En este caso se imprimirá estoy feliz.

Resultado:



Práctica 2 Funciones

Objetivo específico	Manejo de funciones
Recursos	Kotlinlang Kotlin
Actividades	Declarar funciones con y sin parámetros de entrada y de salida
Producto(s)	Al final de la práctica deberá saber como crear una función donde se reciba información, se trabaje en ella y como resultado se devuelva parámetros.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Creamos una simple función

Copiamos y pegamos el siguiente código:

```
fun main(args: Array<String>) {
    showMyName()
    showMyLastName()
    showMyAge()
}
fun showMyName(){
    println("Me llamo Aris")
}
fun showMyLastName(){
    println("Mi Apellido es Guimerá")
}
fun showMyAge(){
    println("Tengo 24 años")
}
```

En este código vemos 3 funciones simples que nos sirve para simplificar código en la función main y así quede mejor leíble nuestra función. Con estás 3 funciones imprimimos nombre, apellido y edad. Resultado:

```
Me llamo Aris
Mi Apellido es Guimerá
Tengo 24 años
```

PASO 3. Creamos una función con parámetros de entrada

Estas funciones son iguales que las anteriores pero esta vez esperan parámetros como entrada, sin esa información la función no podrá trabajar y nos alertará nuestro código con un error. Los parámetros de entrada se declaran en la función dentro de los paréntesis con el nombre de la variable, después de dos puntos ":" el tipo de variable y si hay más parámetros se agregará una coma "," para separar los parámetros unos de otros.

```
fun main(args: Array<String>) {
    showMyInformation("Aris", "Guimerá", 24)
}
fun showMyInformation(name: String, lastName: String, age: Int){
    println("Me llamo $name $lastName y tengo $age años.")
}
```

Aquí vemos que nuestra función *showMyInformation* espera 3 parámetros de entrada, dos son de tipo string y uno de tipo int, a continuación se imprimirá el resultado.

```
Me llamo Aris Guimerá y tengo 24 años.
```

PASO 4. Creamos una función con parámetros de entrada y de salida

Nos queda ver como una función nos puede devolver un resultado, la única limitación es que solo se puede devolver un parámetro. Para declarar que se espera devolver un parámetro, después de los paréntesis de la función se agregan dos puntos ":" y se escribe el tipo de parámetro que se espera regresar. Copie el siguiente código y pegue en kotlinlang para ver el resultado.

```
fun main(args: Array<String>) {
    var result = add(5, 10)
    println(result)
}
fun add(firsNumber: Int, secondNumber: Int) : Int {
    return firsNumber + secondNumber
}
```

Aquí vemos como la función add recibe dos parámetros int, hace la suma y regresa el resultado como parámetro de salida int, dando como resultado la suma de los dos valores declarados en la variable result.

Resultado:

```
15
```

Práctica 3 Controles de flujo if

Objetivo específico	Las instrucciones condicionales nos permiten realizar lógica en función del resultado de una variable o condición. La condición if es de las más habituales y realizará una o varias funciones.
Recursos	Kotlinlang Kotlin
Actividades	Declarar la condicional if, if-else e if anidados
Producto(s)	Al final de la práctica deberá saber como crear una condicional if y entender que usos se les puede dar.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Creamos nuestra primera condicional if

Copiaremos el siguiente código.

```
fun main(args: Array<String>) {
    var result = add(5, 10)

    if(result > 10){
        println("El resultado es mayor que 10")
    }
}
```

```
fun add(firsNumber: Int, secondNumber: Int) : Int = firsNumber + secondNumber
```

Como podemos ver se agregó la línea if donde verificaremos que nuestra variable result sea mayor a 10, donde nuestra variable result es el resultado de una suma que se realizó en una función que nos regresó como parámetro un valor int, al comparar nuestro resultado en el if se imprimirá el mesnaje "El resultado es mayor que 10" solo si el resultado es mayor a 10.

PASO 3. Uso de operadores

No solo podemos usar operadores como <, >, = sino que podemos comparar String a través del doble igual ==. Por ejemplo, quiero saber si el nombre es igual a Luis lo podemos ver en el siguiente código.

```
var name = "Luis"

if(name == ("Luis")){
    println("Se llama Luis")
}
```

Como resultado tendremos que Se llama Luis.

PASO 4. If-else

En ocasiones hay casos que necesitamos más que un if, para esto usaremos la palabra else que actuará como una segunda condicional en caso de que no se cumpla la primera condición. Por ejemplo:

```
var name = "Mario"

if(name == ("Luis")){
     println("Se llama Luis")
}else{
    println("No se llama Luis")
}
```

El funcionamiento está muy claro, si no pasa por la primera condición este irá directo al else. Con esto evitamos escribir dos if uno comprobando si es el mismo nombre y el otro comprobando que no lo es, con esto hacemos que el código se vea mejor ordenado.

PASO 5. If anidado

Aunque no es la práctica más correcta y no deberíamos abusar, en determinadas ocasiones necesitamos más condiciones, y aunque podríamos recurrir a otras instrucciones, lo podemos hacer con if.

```
    if(animal == "dog"){
        println("Es un perro")
    }else if(animal == "cat"){
        println("Es un gato")
    }else if(animal == "bird"){
        println("Es un pájaro")
    }else{
        println("Es otro animal")
    }
```

Aquí vemos una condición para ver qué tipo de animal es el valor que este en nuestra variable animal, la primera condición es si es un perro, pero en la segunda condición ya tenemos un else if donde volvemos a comparar nuestra condición con cat para saber si hablamos de un gato. Podemos hacer esto n cantidad de veces y al final poner un else para decir que si no está en las condiciones anteriores podemos poner como resultado que hablamos de otro animal.

Extra.

También podemos usar más de una condición a la vez gracias a los operadores and && y or ||

Ejemplo:

```
//solo entrará si cumple ambas condiciones
    if(animal == "dog" && raza == "labrador"){
        println("Es un perro de raza labrador")
    }

//Entrará si es verdadera una de las condiciones
    if(animal == "dog" || animal == "gato"){
        println("Es un perro o un gato")
    }
```

Práctica 4 When

Objetivo específico	Optimizar una función if else cuando tiene varias condicionales. When es el sustituto de switch en otros lenguajes.
Recursos	Kotlinlang Kotlin
Actividades	Declarar la condicional when cuando hay varias acciones que dependen del resultado recibido
Producto(s)	Al final de la práctica deberá saber cómo crear una condicional when con varias condiciones por comprobar.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Crear función getMonth

Crearemos nuestra función getMonth. Copiaremos el siguiente código.

```
fun getMonth(month : Int){
    when (month) {
        1 -> print("Enero")
        2 -> print("Febrero")
        3 -> print("Marzo")
        4 -> print("Abril")
        5 -> print("Mayo")
        6 -> print("Junio")
        7 -> print("Julio")
        8 -> print("Agosto")
        9 -> print("Septiembre")
        10 -> print("Octubre")
        11 -> print("Noviembre")
        12 -> print("Diciembre")
        else -> {
            print("No corresponde a ningún mes del año")
        }
    }
}
```

Aquí podemos ver una función llamada getMonth que recibirá como parámetro de entrada un dato de tipo int llamada month. Después podemos ver la condicional when que entre paréntesis tenemos nuestra variable month que vamos a comparar todos los casos disponibles. Si concuerda con algún valor, automáticamente entrará por ahí y realizará la función oportuna, en este caso pintar el mes. Por lo contrario, entrará al else, dicho else no es obligatorio pero en caso de omitirlo simplemente no nos mostrará nada

PASO 3. Mandamos a llamar nuestra función

Creamos la función main donde llamaremos a nuestra función getMonth y mandaremos el parámetro de entrada que nos pide esta última función.

```
fun main() {
    getMonth(2)
}
```

Después corremos el código que así se debe ver.

```
fun getMonth(month : Int){
    when (month) {
        1 -> print("Enero")
        2 -> print("Febrero")
        3 -> print("Marzo")
        4 -> print("Abril")
        5 -> print("Mayo")
        6 -> print("Junio")
        7 -> print("Julio")
        8 -> print("Agosto")
        9 -> print("Septiembre")
        10 -> print("Octubre")
        11 -> print("Noviembre")
        12 -> print("Diciembre")
        else -> {
            print("No corresponde a ningún mes del año")
        }
    }
}

fun main(args: Array<String>) {
    getMonth(2)
}
```

Resultado.

```
Febrero
```

PASO 4. Separar varios valores

Podemos separar valores por comas “,” en caso de que algunos casos tengan el mismo resultados. Reemplazamos nuestra función getMonth.

```
fun getMonth(month : Int){
    when (month) {
        1,2,3 -> print("Primer trimestre del año")
        4,5,6 -> print("segundo trimestre del año")
        7,8,9 -> print("tercer trimestre del año")
        10,11,12 -> print("cuarto trimestre del año")
    }
}
```

Resultado.

```
Primer trimestre del año
```

Podemos simplificar un poco más el anterior when con dos puntos seguidos “..”

```
fun getMonth(month : Int){
    when (month) {
        in 1..6 -> print("Primer semestre")
        in 7..12 -> print("segundo semestre")
    }
}
```

Resultado.

```
Primer semestre
```

Práctica 5 For

Objetivo específico	Crear ciclos for con la intención de entender para que nos pueden ayudar.
Recursos	Kotlinlang Kotlin
Actividades	Declarar un loop for
Producto(s)	Al final de la práctica deberá saber cómo crear un ciclo for que sea capaz de iterar y arrojar un resultado

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Crear un loop for

Crearemos un for donde se espera que se imprima del número 1 al número 5. Copiamos el siguiente código y lo pegamos en kotlinlang.

```
fun main(){
    for (i in 1..5) {
        println(i)
    }
}
```

En este for tenemos nuestra variable i que es la que tendrá el valor del rango entre el 1 al 5, la palabra **in** significa que se repetirá este ciclo las 5 veces que vemos en el código 1..5. Como resultado tenemos lo siguiente:

```
1
2
3
4
5
```

PASO 3. Crear un for donde recorramos un arreglo

Podemos hacer un ciclo donde se repita la longitud que tenga algún arreglo, esto sirve para cuando deseamos buscar algo dentro de una lista y no sabemos de qué tamaño es. Tenemos nuestro arreglo que son los días de la semana de lunes a viernes y queremos imprimir toda la lista.

```
fun main(){
    var array = listOf("Lunes", "Martes", "Miércoles", "Jueves", "Viernes")
    for (i in array) {
        println(i)
```

```
    }
}
```

Como podemos ver el for se ciclará las n veces que sea la longitud de nuestro arreglo. Como resultado tendremos.

```
Lunes
Martes
Miércoles
Jueves
Viernes
```

Práctica 6 Arrays

Objetivo específico	Entender cómo funciona un array y que podemos hacer con ellos.
Recursos	Kotlinlang Kotlin
Actividades	Declarar un array que contenga cierta información y poder manipular dicho array.
Producto(s)	Al final de la práctica deberá saber como crear un array y como obtener algún valor de dicho array.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Crear nuestro primer array

Crearemos nuestro primer array, contendrá los meses del año, para declarar un array usaremos la palabra arrayOf(). Dentro de la función main declararemos nuestro array.

```
val months = arrayOf("Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
"Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "diciembre")
```

PASO 3. Obtener algún valor dentro del array.

Si queremos obtener el valor que está en la 8va posición usaremos get. Hay que tener en cuenta que nuestro arreglo inicia desde la posición 0 y no desde la 1ra posición, por lo cuál si queremos obtener el 8vo valor necesitamos restarle 1 para realmente obtener el 8vo, de lo contrario obtendremos el 9no, por ejemplo:

```
                    0       1       2     3      4       5      6       7
val months = arrayOf("Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
"Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "diciembre")

println(months.get(7))
```

Resultado:

```
Agosto
```

PASO 4. Obtener el tamaño del array

Para evitar traer un valor que está fuera del tamaño de nuestro array, que esto nos podría traer errores en nuestro código, podemos saber el tamaño con el atributo size. Haremos un ejemplo donde imprimiremos el tamaño de nuestro arreglo y además imprimiremos el valor número 11 del arreglo solo sí

el arreglo es de longitud menor o igual a 12.

```
fun main(args: Array<String>) {
    val months = arrayOf("Enero", "Febrero", "Marzo", "Abril", "Mayo",
"Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre",
"diciembre")
    println("El array es de longitud ${months.size}")
    if(months.size <= 12){
        println(months.get(11))
    }else{
        println("No hay más parámetros")
    }
}
```

Para concatenar un atributo de una variable se usan {} dentro se pondrá la variable con el atributo.

Resultado.

```
El array es de longitud 12
diciembre
```

PASO 5. Recorrer todo el array

Para recorrer un array hasta el final usaremos un ciclo for donde vamos a imprimir todos los valores del arreglo. En un futuro esto nos puede ayudar a buscar algún valor en específico. Usaremos índices en nuestro array para traernos la posición exacta de los valores aún si no conocemos la longitud exacta de este. Copia el siguiente código y pega en kotlinlang.

```
fun main(args: Array<String>) {
    val months = arrayOf("Enero", "Febrero", "Marzo", "Abril", "Mayo",
"Junio")

    for (posicion in months.indices){
        println("La posición $posicion contiene el valor
${months.get(posicion)}")
}
```

Resultado.

```
La posición 0 contiene el valor Enero
La posición 1 contiene el valor Febrero
La posición 2 contiene el valor Marzo
La posición 3 contiene el valor Abril
La posición 4 contiene el valor Mayo
La posición 5 contiene el valor Junio
```

Práctica 7 Listas inmutables

Objetivo específico	Entender cómo funciona una lista inmutable y que podemos hacer con ellos.
Recursos	Kotlinlang Kotlin
Actividades	Declarar una lista inmutable que contenga cierta información y poder acceder después a sus elementos.
Producto(s)	Al final de la práctica deberá saber cómo crear una lista inmutable con valores guardados y poder leerlos después.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Crear un listOf

Para crea una lista inmutable podemos llamar a la función listOf y pasar como parámetro los datos a almacenar, debemos indicar el tipo de datos que almacena luego de List. Una vez creada la lista no podemos modificar ni agregar datos.

```
var lista1: List<String> = listOf("lunes", "martes", "miercoles", "jueves",
"viernes", "sábado", "domingo")
```

PASO 3. Leer un listOf

Como sabemos lo único que podemos hacer con un listOf es acceder a sus elementos. Para esto lo podemos recorrer con un for.

```
for(elemento in lista1)
   print("$elemento ")
```

Como resultado obtenemos.

```
lunes martes miercoles jueves viernes sábado domingo
```

También podemos acceder a un elemento en específico.

```
println(lista1[0])
```

Resultado

```
lunes
```

Práctica 8 Listas mutables

Objetivo específico	Entender cómo funciona una lista mutable y diferenciar con la lista inmutable.
Recursos	Kotlinlang Kotlin
Actividades	Declarar una lista mutable que contenga cierta información y poder editar sus elementos.
Producto(s)	Al final de la práctica deberá saber cómo crear una lista mutable con elementos guardados y poder editarlos después.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Crear un mutableListOf

Una lista mutable además de poder acceder a los elementos también podemos ir agregando más elementos o eliminarlos, el único inconveniente es que es más ineficiente con la memoria. Para crear un mutableList llamamos a la función mutableListOf y pasar como parámetro los datos a almacenar, debemos indicar el tipo de datos que almacena luego de MutableList.

```
val edades: MutableList<Int> = mutableListOf(23, 67, 12, 35, 12)
```

PASO 3. Agregar un nuevo elemento.

Para agregar un nuevo elemento al final de la lista llamamos al método add.

```
edades.add(50)
```

Para agregar un nuevo elemento en cualquier posición dentro de la lista podemos usar el método con dos parámetros, primero la posición y segundo el valor. En este caso agregaremos el valor 17 en la primera posición de la lista.

```
edades.add(0, 17)
```

Agregamos un for para imprimir la lista.

```
for(elemento in edades)
   print("$elemento ")
```

Resultado.

```
17 23 67 12 35 12 50
```

PASO 4. Buscar elementos mediante una condición.

Vamos a determinar cuántas personas son mayores de edad en nuestra lista de edades. Para esto necesitaremos el método count para recorrer toda la lista para saber cuántas edades son iguales o superiores a 18. Copiar el siguiente código y pegar en kotlinlang

```
print("Cantidad de personas mayores de edad:")
val cant = edades.count { it >= 18 }
println(cant)
```

Como vemos en el código guardamos el resultado directamente en la variable cant. Entre corchetes tenemos nuestra variable it donde comparará que cada elemento que se está recorriendo en la lista cumpla con la condición que sea igual o mayor a 18, después imprimimos el resultado.

```
Cantidad de personas mayores de edad: 4
```

PASO 5. Eliminar un elemento de la lista mediante una condición.

Para eliminar un elemento usaremos el método removeAll. Borraremos todas las edades que sean igual a 12.

```
edades.removeAll { it == 12 }
println("Lista de edades después de borrar las que tienen 12 años")
println(edades)
```

Esta vez usamos la condición directamente sobre nuestra lista. Como resultado la lista quedará de la siguiente manera.

```
Lista de edades después de borrar las que tienen 12 años
[17, 23, 67, 35, 50]
```

Práctica 9 Búsqueda de una película

Objetivo específico	Hacer una búsqueda dentro de un catálogo de películas
Recursos	Kotlinlang Kotlin
Actividades	Crear una función donde recibamos como primer parámetro de entrada una lista inmutable y como segundo parámetro una variable de tipo **string** que será la película que deseamos buscar. Como resultado imprimiremos si se encontró la película, caso contrario imprimiremos que no se encontró la película.
Producto(s)	Al final de la práctica se tendrá un buscador de películas donde se le informará al usuario si se encontró la película o no.

PASO 1. Abrir kotlinlang.

Abrir el siguiente enlace Kotlinlang

PASO 2. Crear nuestro buscador

Para hacer nuestro buscador debemos crear una lista inmutable ***listOf*** donde almacenaremos los nombres de las películas que tendremos en nuestro catálogo, mínimo debe contener 5 películas. Crearemos otra variable de tipo ***String*** donde pondremos el nombre de la película que deseamos buscar.

Después crearemos nuestra función con dos parámetros de entrada y un parámetro de salida, donde vamos a comparar la película que estamos deseando buscar con el catálogo que tenemos. Como resultado de esta función regresaremos un parámetro ***booleano*** que nos devuelva ***true*** en caso de que se haya encontrado la película y ***false*** si no se encontró.

Como resultado final usaremos ***print*** para imprimir en consola si encontramos la película o no. Como puntos extra imprimiremos el nombre de la película que buscamos.

Resultados esperados:

```
La casa de papel se encuentra disponible
```

```
Lo sentimos no se encontró la película Titanic
```