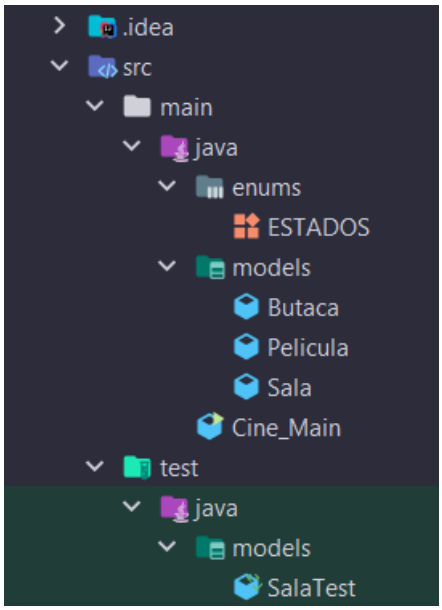
Documentación de un simulador de cine, hecho tanto en Kotlin como en Java.

CINE_KOTLIN /JAVA

Testeado con JUnit

JiaCheng Zhang

Estructura de directorios:



El proyecto se estructura en directorios, dentro de la carpeta java está el paquete de “models” donde se almacenan los modelos; el paquete “enums” donde se almacena el enum de estados.

La parte de test está en el directorio de “test”.

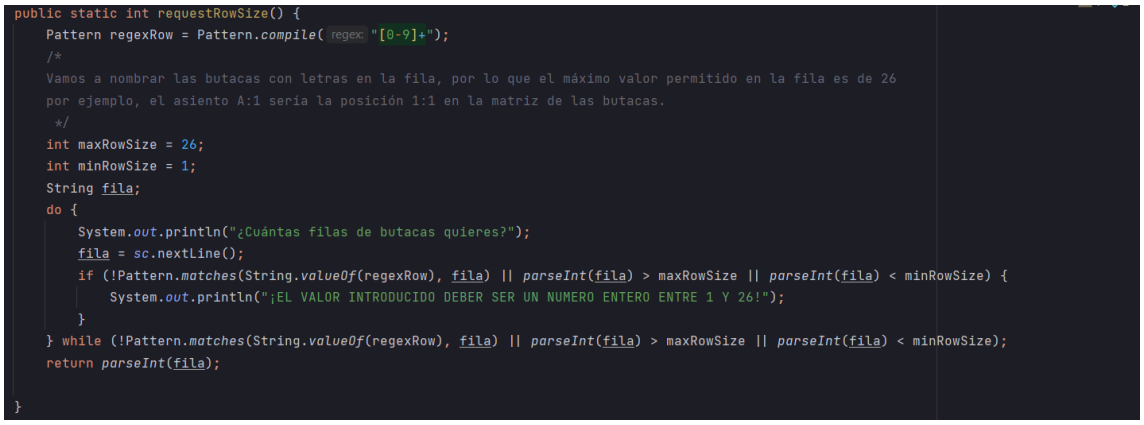
Explicación de los métodos y clases utilizadas.

Cine_Main.java × Pelicula.java × Sala.java × SalaTest.java × Butaca.java × ESTADOS.java ×

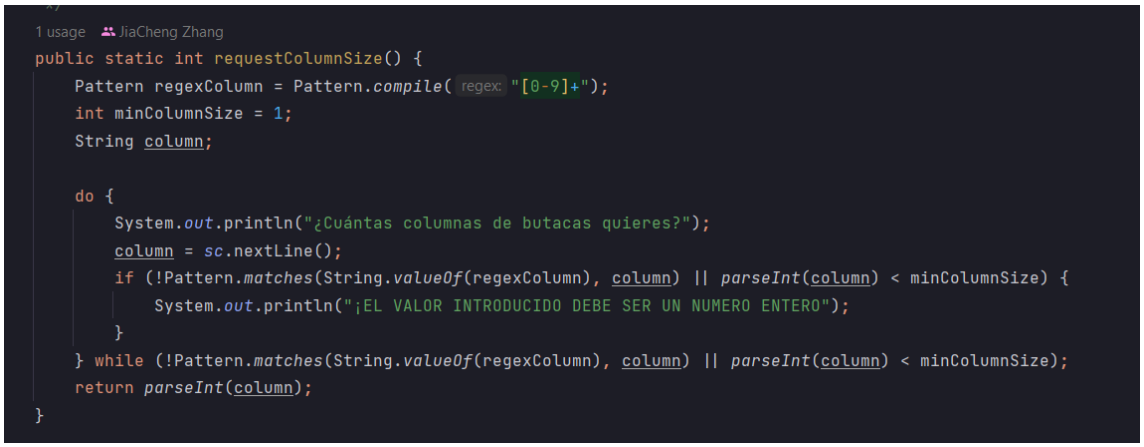
```
JiaCheng Zhang
public class Cine_Main {
    2 usages
    static boolean exit = false;
    JiaCheng Zhang
    public static void main(String[] args) {

        //pedimos los valores del tamaño de la matriz de asientos y el nombre de la sala con diferentes funciones simples.
        int row = Sala.requestRowSize();
        int column = Sala.requestColumnSize();
        String roomName = Sala.requestRoomName();

        //coloco los asientos con los valores que se han pedido por consola antes.
        Butaca[][] seatsMatrix = new Butaca[row][column];

        Sala.placeSeats(seatsMatrix);

        do {
            switch(Sala.selectOption()){
                //Función que reserva sitios
                case 1 -> Sala.processReservation(Sala.reverseSeat(seatsMatrix, column, row) ,seatsMatrix);
                //Función que formaliza la reserva
                case 2 -> Sala.processFormalization(Sala.formalizeReservation(seatsMatrix, column, row), seatsMatrix);
                //Función que cancela la reserva de un sitio que se elija.
                case 3 -> Sala.processCancellation(Sala.cancelReservation(seatsMatrix, column, row), seatsMatrix);
                //Función con la que se compran los asientos.
                case 4 -> Sala.processPucharse(Sala.buySeat(seatsMatrix, row, column), seatsMatrix);
                case 5 -> Sala.generateReport(roomName);
                case 6 -> exit = true;
            }
        }while(!exit);

```

En el archivo “Cine_Main” inicio la variable “exit”, que se usará mas tarde en el menú para salir del programa.

Se piden las dimensiones de la matriz de asientos y el nombre de la sala con 3 funciones distintas.



```java
public static int requestRowSize() {
    Pattern regexRow = Pattern.compile( regex: "[0-9]+");
    /*
    Vamos a nombrar las butacas con letras en la fila, por lo que el máximo valor permitido en la fila es de 26
    por ejemplo, el asiento A:1 sería la posición 1:1 en la matriz de las butacas.
     */
    int maxRowSize = 26;
    int minRowSize = 1;
    String fila;
    do {
        System.out.println("¿Cuántas filas de butacas quieres?");
        fila = sc.nextLine();
        if (!Pattern.matches(String.valueOf(regexRow), fila) || parseInt(fila) > maxRowSize || parseInt(fila) < minRowSize) {
            System.out.println("¡EL VALOR INTRODUCIDO DEBER SER UN NUMERO ENTERO ENTRE 1 Y 26!");
        }
    } while (!Pattern.matches(String.valueOf(regexRow), fila) || parseInt(fila) > maxRowSize || parseInt(fila) < minRowSize);
    return parseInt(fila);

}
```



```java
1 usage  JiaCheng Zhang
public static int requestColumnSize() {
    Pattern regexColumn = Pattern.compile( regex: "[0-9]+");
    int minColumnSize = 1;
    String column;

    do {
        System.out.println("¿Cuántas columnas de butacas quieres?");
        column = sc.nextLine();
        if (!Pattern.matches(String.valueOf(regexColumn), column) || parseInt(column) < minColumnSize) {
            System.out.println("¡EL VALOR INTRODUCIDO DEBE SER UN NUMERO ENTERO");
        }
    } while (!Pattern.matches(String.valueOf(regexColumn), column) || parseInt(column) < minColumnSize);
    return parseInt(column);
}
```

```java
1 usage  JiaCheng Zhang
public static String requestRoomName() {
    System.out.println("Introduce el nombre de la sala de cine:");
    var roomName = "";
    roomName = sc.nextLine();
    return roomName;
}
```

```
"C:\Program Files\BellSoft\LibericaJDK-17-Full\bin\java.exe" "-javaagent:C:\
¿Cuántas filas de butacas quieres?
15
¿Cuántas columnas de butacas quieres?
7
Introduce el nombre de la sala de cine:
nombre
**¡¡BIENVENIDO AL CINE!!**
Seleccione la opción que necesite

1 → RESERVAR ASIENTO (PALOMITAS DE REGALO)
2 → FORMALIZAR RESERVA (RESERVA REQUERIDA)
3 → CANCELAR RESERVA (RESERVA REQUERIDA)
4 → COMPRAR ASIENTO
5 → GENERAR INFORME DEL CINE
6 → SALIR
Opción seleccionada:
```

Tras introducir las dimensiones y el nombre con las funciones correspondientes, se nos muestra el menú con las diferentes opciones.

Reservar asiento.

```
do {
    switch(Sala.selectOption()){
        //Función que reserva sitios
        case 1 → Sala.processReservation(Sala.reverseSeat(seatsMatrix, column, row) ,seatsMatrix);
        //Función que formaliza la reserva
        case 2 → Sala.processFormalization(Sala.formalizeReservation(seatsMatrix, column, row), seatsMatrix);
        //Función que cancela la reserva de un sitio que se elija.
        case 3 → Sala.processCancellation(Sala.cancelReservation(seatsMatrix, column, row), seatsMatrix);
        //Función con la que se compran los asientos.
        case 4 → Sala.processPucharse(Sala.buySeat(seatsMatrix, row, column), seatsMatrix);
        case 5 → Sala.generateReport(roomName);
        case 6 → exit = true;
    }
}while(!exit);
```

```java
public static String reverseSeat(Butaca[][] seatsMatrix, int column, int row) {
    System.out.println();
    String reservedSeat;
    Pattern regex = Pattern.compile( regex: "[A-Z]:[0-9]+", Pattern.CASE_INSENSITIVE);

    printSeats(seatsMatrix);
    System.out.println("Hola! ¿Que asiento quieres reservar? Estas son las butacas disponibles.");
    System.out.println("Las filas se ordenan con las letras del abecedario y las columnas con números, un ejemplo de asiento seria B:4");
    System.out.println("que representaría el asiento de la segunda fila y la cuarta columna");
    System.out.println("Introduce el asiento: ");

    do {
        reservedSeat = sc.nextLine();
        if (!Pattern.matches(String.valueOf(regex), reservedSeat.toUpperCase())) {
            System.out.println("Debes el introducir la LETRA de la fila y el NÚMERO de la columna. Ejemplo: A:1 ");
        }
        if (Pattern.matches(String.valueOf(regex), reservedSeat.toUpperCase())) {
            if (!doesColumnExist(reservedSeat, column)) {
                System.out.println("La columna que has elegido no existe, elige otro asiento.");
            }
        }
        if (Pattern.matches(String.valueOf(regex), reservedSeat.toUpperCase())) {
            if (!doesRowExist(reservedSeat, row)) {
                System.out.println("La fila que has elegido no existe, elige otro asiento.");
            }
        }

    } while (!Pattern.matches(String.valueOf(regex), reservedSeat.toUpperCase()) || !doesColumnExist(reservedSeat, column));
    totalCash += 1.25;
    reservedSeatsCount++;
    System.out.println("El precio de la reserva es de 1.25€, cuando formalice la reserva se le cobrarán 4 euros para completar el precio de la entrada que es de 5.25");
    System.out.println("Se te ha cobrado $totalCash");
    return reservedSeat;
}
```

Nos lleva a la función “reserveSeat” que nos pide la entrada del asiento que queremos y la filtra con expresiones regulares y comprueba que la fila y la columna introducidas existan ya que las dimensiones de la matriz son

```java
 */
1 usage  JiaCheng Zhang
public static void processReservation(String reservation, Butaca[][] seatsMatrix) {
    String[] processedReservation = reservation.split( regex: ":");
    String selectedRow = processedReservation[0];
    String selectedColumn = processedReservation[1];
    int processedRow = rowLetterToNumber(selectedRow);
    changeSeatStatusToReserved(seatsMatrix, selectedColumn, processedRow);
    printSeats(seatsMatrix);

}
```

Una vez que nos aseguramos de que la entrada es correcta, nos lleva a la función “processReservation”, que nos separa la entrada con un “split”.

Cambiamos la letra de la fila al índice de la matriz con otra función.

Las otras

```java
public static int rowLetterToNumber(String selectedRow) {
    return switch (selectedRow.toUpperCase()) {
        case "A" → 0;
        case "B" → 1;
        case "C" → 2;
        case "D" → 3;
        case "E" → 4;
        case "F" → 5;
        case "G" → 6;
        case "H" → 7;
        case "I" → 8;
        case "J" → 9;
        case "K" → 10;
        case "L" → 11;
        case "M" → 12;
        case "N" → 13;
        case "O" → 14;
        case "P" → 15;
        case "Q" → 16;
        case "R" → 17;
        case "S" → 18;
        case "T" → 19;
        case "U" → 20;
        case "V" → 21;
        case "W" → 22;
        case "X" → 23;
        case "Y" → 24;
        case "Z" → 25;
        default → -1;
    };
```

Una vez cambiado esto, tenemos ya la posición donde se va a hacer la reserva y se cambia el asiento de libre a reservado en esa posición.

```java
public static Butaca changeSeatStatusToReserved(Butaca[][] seatsMatrix, String selectedColumn, int processedRow) {

    seatsMatrix[processedRow][parseInt(selectedColumn) - 1] = Butaca.RESERVED_SEAT;
    return RESERVED_SEAT;
}
```

El resto de los métodos del menú funcionan con la misma estructura de funciones. En el caso de las funciones de formalizar la reserva y cancelar la reserva, debe haberse hecho una reserva anteriormente.

Créditos:

Redacción del código:

- JiaCheng Zhang

Directora ejecutiva:

- Astrid44_ (“contesta tt 😊”)

Banda sonora:

- Astrid44_ (“contesta tt 😊”)

Encargada de proyecto:

- Astrid44_ (“contesta tt 😊”)

Protagonista principal:

- Astrid44_ (“contesta tt 😊”)