Asignatura: Programación II – Curso 2016/17 – 2er Semestre

Dpto. LSIIS. Unidad de Programación

Práctica 1: Cine

***Objetivo*:** El objetivo de esta práctica es la familiarización del alumno con la programación orientada a objetos en Java, incluyendo manejo de arrays de objetos, manejo de la clase genérica `ArrayList<E>` de la asignatura, organización en paquetes (modularidad) y ejecución de pruebas.

***Desarrollo del trabajo*:** La práctica se podrá realizar en grupos de dos alumnos (matriculados en el mismo semestre) o de forma individual. Esto se especificará mediante la anotación Java explicada en el apartado "6. Consideraciones de entrega". Esta práctica la deben hacer todos los alumnos (evaluación continua y sólo examen final).

***Código auxiliar*:** Todo este código se suministra en un archivo `CineAlumnos.zip` y debe ser utilizado sin modificación alguna.

***Autoevaluación*:** El alumno debe comprobar que su ejercicio no contiene ninguno de los errores explicados en el último apartado de este enunciado.

***Entrega*:** La práctica se entregará a través de la página web: http://triqui2.fi.upm.es/entrega/. Una vez realizada la primera entrega en grupo, el grupo no se podrá deshacer, es decir, los dos alumnos del grupo estarán obligados a realizar todas las entregas de esta práctica juntos. La misma norma se aplica cuando un alumno realiza la primera entrega de forma individual, es decir, ese alumno ya no podrá realizar y entregar la práctica en grupo. El alumno de un grupo que realice la primera entrega de la práctica, será el que tenga que hacer el resto de las entregas de la misma, si son necesarias. El historial de las entregas y la documentación de las pruebas automáticas que realiza el sistema de entrega se puede consultar en: http://telemaco.ls.fi.upm.es:30103

***Plazos:* El periodo de entrega finaliza el día 19-4-2017 a las 10:00 AM**. En el momento de realizar la entrega, la práctica será sometida a una serie de pruebas que deberá superar para que la entrega sea admitida. El alumno dispondrá de **un número máximo de 10 entregas**. Asimismo, por el hecho de que la práctica sea admitida, eso no implicará que la práctica esté aprobada.

Material a entregar: Se entregarán los tres ficheros separados `Cine.java`, `Sala.java`, y `Sesion.java` sin comprimir. El código debe compilar en la versión 1.8 del J2SE de Oracle/Sun.

***Evaluación*:** Las prácticas entregadas serán corregidas por un profesor, que revisará el código con el fin de identificar posibles errores de estilo o problemas de eficiencia. El peso de esta práctica en la nota final de la asignatura es el que indica la Guía de la Asignatura. Se mantendrá el mismo enunciado de la práctica para la convocatoria extraordinaria.

Detección Automática de Copias: Cada práctica entregada se comparará con el resto de prácticas entregadas en las distintas convocatorias del curso. Esto se realizará utilizando un sofisticado programa de detección de copias.

Consecuencias de haber copiado: Todos los alumnos involucrados en una copia, bien por copiar o por ser copiados, serán sancionados según las normas publicadas en la guía de aprendizaje de la asignatura.

1. Introducción

En esta práctica se pide implementar una aplicación para gestionar la venta *online* de entradas para un cine. Básicamente, esta aplicación debe permitir al usuario comprar una o más entradas para una cierta sesión de una película. Supondremos que el cine puede tener más de una sala y en cada sala sólo se proyecta una única película. Una película se proyectará en una sala en una o más sesiones. Una vez creado el cine, las salas del cine serán siempre las mismas. En cambio, las sesiones de una película podrán variar a lo largo del tiempo. Tras comprar las entradas, el usuario podrá recogerlas. Esto lo podrá hacer en cualquier momento y un número ilimitado de veces. Asimismo, en cualquier

momento que lo desee, el usuario podrá consultar el estado de ocupación de una sala en una cierta sesión sin necesidad de tener que iniciar el proceso de compra de entradas.

Si el usuario desea comprar entradas, podrá optar entre elegir él mismo las butacas indicando la fila y la columna de cada una, o quedarse con las butacas contiguas que le recomiende la aplicación. Una vez que haya seleccionado las butacas por un método u otro, la aplicación devolverá un identificador de venta, que posteriormente el usuario deberá proporcionar para poder recoger las entradas.

2. Arquitectura de la aplicación

El ejercicio a desarrollar por el alumno se realizará en un proyecto con los siguientes paquetes:

- *default*: `GestorVentaEntradas.java`,`TestVentaEntradas.java` `TestVentaEntradasSala.java` y `TestVentaEntradasSesion.java`
- *cine*: `Cine.java`, `Sala.java`, `Sesion.java` y `ButacasContiguas.java`
- *interfazusuario*: `VentanillaVirtualUsuario.java`, `LecturaTeclado.java` y `VentanillaVirtualAdministrador.java`
- *interfazusuario.menu*: `Menu.java`

Las clases a desarrollar son las que aparecen con fondo amarillo en la figura 1. El resto de las clases se proporcionan como código de apoyo. La clase GestorVentaEntradas incluye un main() que permite ejecutar las operaciones por medio de menús en la consola. Desde el main() se invoca a los métodos de la clase Cine para crear el cine, comprar entradas, recoger entradas, etc. Por otro lado, el main() utiliza los métodos de la clase VentanillaVirtualUsuario para mostrar en la consola los menús y pedir al usuario que introduzca los datos necesarios para realizar las operaciones.

Por otro lado, se proporciona un segundo main() en el fichero `TestVentaEntradas.java` a modo de muestra de cómo realizar pruebas específicas de cada método.

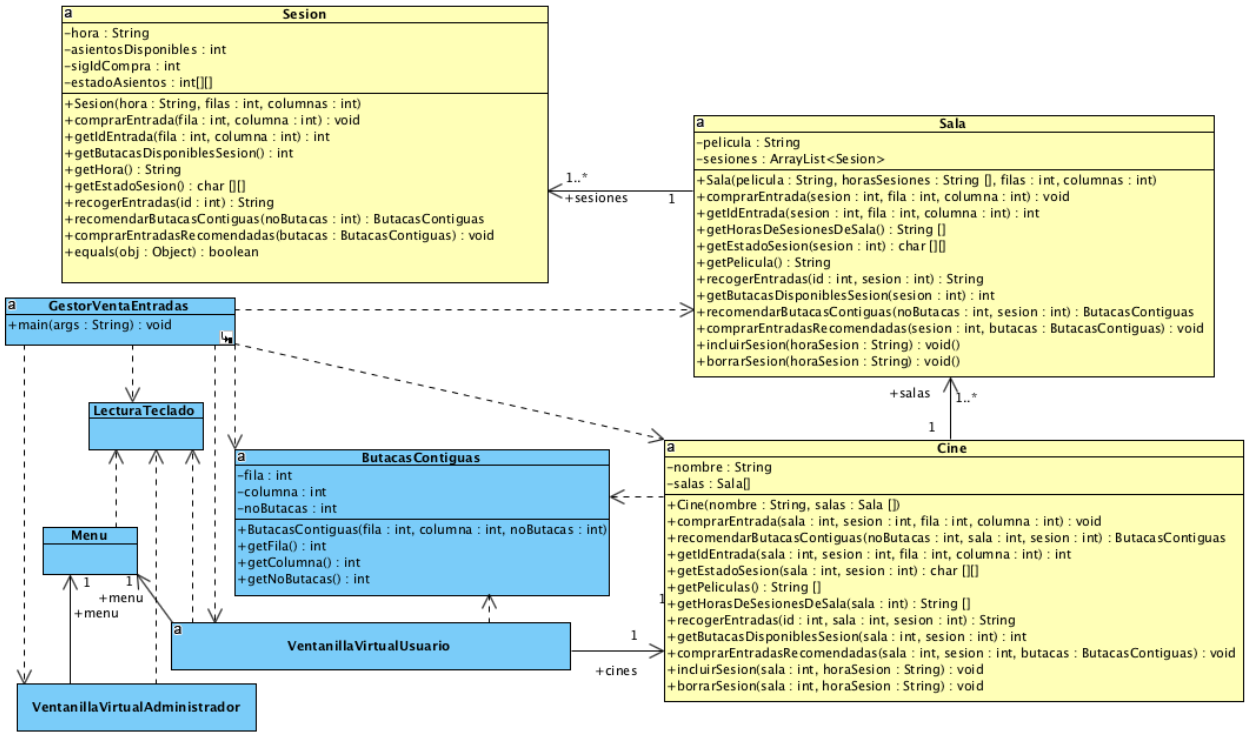

Figura 1. Diagrama de clases UML

3. Diseño detallado de las clases

Las salas deben identificarse mediante un número comprendido entre 1 y *N_SALAS*, y las sesiones mediante un número comprendido entre 1 y *N_SESIONES* (que puede ser diferente para cada sala). Las filas y las columnas se deben numerar empezando por el número 1 (la más cercana a la pantalla) y tendrán la siguiente disposición dentro de la sala:

PANTALLA

	1	2	3	4	5
1	O	O	O	O	O
2	O	O	O	O	O
3	O	O	O	O	O
4	O	O	O	O	O
5	O	O	#	O	O
6	#	#	O	#	O
7	O	#	O	#	O
8	O	#	O	O	#
9	O	O	#	O	O

Figura 2. Sala vista desde arriba

Atributos de la clase Cine:

nombre: nombre del cine.

salas: son las salas del cine representadas por un vector de objetos de tipo `Sala` asociados al cine.

Métodos de la clase Cine:

Cine: constructor de la clase `Cine` que recibe como argumento el nombre del cine y un vector de salas, y se encarga de inicializar los atributos del objeto.

comprarEntrada: método que compra una entrada con la fila y columna dadas para la sala y sesión dadas. La compra queda registrada en el objeto de tipo `Sesion` correspondiente.

getIdEntrada: método que devuelve el identificador de venta para una entrada especificada mediante su sala, sesión, fila y columna. El algoritmo para obtener este identificador se explica en la especificación del método con el mismo nombre en la clase `Sesion`.

getPeliculas: método que devuelve un vector de String con los títulos de las películas que se proyectan en el cine. En la posición 0 del vector se encontrará el título de la película proyectada en la sala 1, en la posición 1 la de la sala 2, y así sucesivamente.

getHorasDeSesionesDeSala: método que devuelve un vector de String con las horas de las sesiones asociadas a una sala. En la posición 0 del vector se encontrará la hora de la sesión 1, en la posición 1 la de la sesión 2, y así sucesivamente.

getEstadoSesion: método que devuelve una matriz de caracteres en la que se representa el estado de ocupación de una sala dada para una sesión dada. La matriz resultado tiene el mismo número de filas y columnas que la sala dada, y en cada posición (i,j) de la matriz debe aparecer un carácter '#' si la butaca ha sido comprada, y un carácter 'O' (O mayúscula) si la butaca está disponible.

getButacasDisponiblesSesion: método que devuelve el número de butacas disponibles en una sala dada para una sesión dada.

recogerEntradas: método que devuelve las *N* entradas asociadas a un identificador de venta dado para una sala y una sesión dadas. Las *N* entradas se devolverán dentro de un String con el siguiente formato:

"*nombre_del_cine@título_de_la_película@hora_de_la_sesión+fila1,columna1+fila2,columna2+ ... +filaN,columnaN+*"

Por ejemplo: "Cine Paraíso@Tiburón@22:00+3,4+3,5+"

Si el identificador de venta dado no existe en la sala y la sesión dadas, se devuelve null.

recomendarButacasContiguas: método que dados un número de butacas, una sala y una sesión, devuelve un objeto de tipo `ButacasContiguas` que contiene la fila y la columna de la butaca recomendaba con menor número de columna, y el número de butacas solicitadas. El algoritmo para obtener las butacas recomendadas se explica en la especificación del método con el mismo nombre en la clase `Sesion`.

comprarEntradasRecomendadas: método que dados un objeto de tipo `ButacasContiguas`, una sala y una sesión, registra la compra en el objeto de tipo `Sesion` correspondiente.

incluirSesion: método que añade una nueva sesión con la hora dada a la sala dada.

borrarSesion: método que borra la sesión con la hora dada de la sala dada.

Atributos de la clase Sala:

pelicula: título de la película.

sesiones: son las sesiones en las que se proyecta la película en esta sala representadas por un arraylist de objetos de tipo `Sesion`. El arraylist estará ordenado de menor a mayor por la hora de la sesión.

Métodos de la clase Sala:

Sala: constructor de la clase `Sala` que recibe como argumentos el título de la película, un vector de String con las horas de las sesiones de esta sala, y el número de filas y columnas de la sala. Con estos argumentos, inicializa los atributos del objeto.

getPelicula: método que devuelve el título de la película asociada a la propia sala.

comprarEntrada: método que compra una entrada con la fila y columna dadas para sesión dada de la propia sala. La compra queda registrada en el objeto de tipo `Sesion` correspondiente.

getIdEntrada: método que devuelve el identificador de venta para una entrada en la propia sala especificada mediante su sesión, fila y columna. El algoritmo para obtener este identificador se explica en la especificación del método con el mismo nombre en la clase `Sesion`.

getHorasDeSesionesDeSala: método que devuelve un vector de String con las horas de las sesiones asociadas a la propia sala. En la posición 0 del vector se encontrará la hora de la sesión 1, en la posición 1 la de la sesión 2, y así sucesivamente.

getEstadoSesion: método que devuelve una matriz de caracteres en la que se representa el estado de ocupación de la propia sala para una sesión dada. El contenido de esta matriz se especifica en el método con el mismo nombre de la clase `Sesion`.

getButacasDisponiblesSesion: método que devuelve el número de butacas disponibles en la propia sala para una sesión dada.

recogerEntradas: método que devuelve las *N* entradas asociadas a un identificador de venta dado para la propia sala y una sesión dada. Las *N* entradas se devolverán dentro de un String con el siguiente formato:

"*título_de_la_película@hora_de_la_sesión+fila1,columna1+fila2,columna2+ ... +filaN,columnaN+*"

Si el identificador de venta dado no existe en la propia sala y la sesión dada, se devuelve null.

recomendarButacasContiguas: método que dados un número de butacas y una sesión de la propia sala, devuelve un objeto de tipo `ButacasContiguas` que contiene la fila y la columna de la butaca recomendaba con menor número de columna, y el número de butacas solicitadas. El algoritmo para obtener las butacas recomendadas se explica en la especificación del método con el mismo nombre en la clase `Sesion`.

comprarEntradasRecomendadas: método que dados un objeto de tipo `ButacasContiguas` y una sesión de la propia sala, registra la compra en el objeto de tipo `Sesion` correspondiente.

incluirSesion: método que añade una nueva sesión con la hora dada a la propia sala. Se realizará la inserción de forma que se mantenga ordenado el arraylist de sesiones.

borrarSesion: método que borra la sesión con la hora dada de la propia sala.

Atributos de la clase Sesion:

hora: hora de la sesión en formato HH:MM.

estadoAsientos: es una matriz de enteros con las mismas dimensiones que la sala asociada a la propia sesión, en la que se representan las butacas compradas y las disponibles para la propia sesión. Si una butaca en la posición (i, j) está disponible, se guarda un valor 0, y en caso contrario, se guarda el identificador de venta asociado a la butaca. Como en una venta se puede comprar más de una butaca, podría suceder que en la matriz haya varios identificadores de ventas iguales en posiciones contiguas.

asientosDisponibles: es un entero que indica el número de asientos disponibles en la propia sesión.

sigIdCompra: es un entero que se incrementa cada vez que se realiza una venta de entradas para la propia sesión. Se utiliza para generar identificadores de venta diferentes para cada compra.

Métodos de la clase Sesion:

Sesion: constructor de la clase `Sesion` que recibe como argumentos la hora de proyección, y el número de filas y columnas de la sala asociada a la propia sesión. Con estos argumentos, inicializa los atributos del objeto. El atributo sigIdComprase inicializa a 1.

getHora: método que devuelve la hora asociada a la propia sesión.

equals: método que compara el objeto de tipo Sesion dado con la propia sesión, y devuelve cierto si son iguales y falso en caso contrario. Se considera que dos sesiones son iguales si son iguales sus atributos hora.

comprarEntrada: método que compra una entrada con la fila y columna dadas para la propia sesión. Para registrar la venta, se guarda el valor actual del atributo sigIdCompra en la posición

(fila-1, columna-1) del atributo estadoAsientos. A continuación, se incrementa en uno el atributo sigIdCompra.

getIdEntrada: método que devuelve el identificador de venta para una entrada en la propia sesión especificada mediante su fila y columna. El identificador que se devuelve se toma de la posición (fila-1, columna-1) del atributo estadoAsientos.

getEstadoSesion: método que devuelve una matriz de caracteres en la que se representa el estado de ocupación de la propia sesión. La matriz resultado tiene el mismo número de filas y columnas que la sala asociada a la propia sesión, y en cada posición (i, j) de la matriz debe aparecer un carácter '#' si la butaca ha sido comprada, y un carácter 'O' (O mayúscula) si la butaca está disponible. En la figura 2 se puede observar un ejemplo de una matriz de caracteres que refleja el estado de ocupación de una sala en una sesión.

getButacasDisponiblesSesion: método que devuelve el número de butacas disponibles en la propia sesión.

recogerEntradas: método que devuelve las *N* entradas asociadas a un identificador de venta dado para la propia sesión. Las *N* entradas se devolverán dentro de un String con el siguiente formato:

"*hora_de_la_sesión+fila1,columna1+fila2,columna2+ ... +filaN,columnaN+*"

Si el identificador de venta dado no existe en la propia sesión, se devuelve null.

recomendarButacasContiguas: método que dados un número N de butacas, devuelve un objeto de tipo `ButacasContiguas` que contiene la fila y la columna de la butaca recomendaba con menor número de columna, y el número de butacas solicitadas. El algoritmo para obtener las butacas recomendadas es el siguiente:

1. Se buscan las primeras N butacas contiguas libres empezando por la fila *(N_FILAS+1)/2+1* y acabando en la fila *N_FILAS*. Cada fila se recorre empezando por la columna *N_COLUMNAS* y acabando en la columna *1*.
2. Si en el paso anterior no se encuentran N butacas contiguas libres, se buscan las primeras N butacas contiguas libres empezando por la fila *(N_FILAS+1)/2* y acabando en la fila 1. Cada fila se recorre empezando por la columna *N_COLUMNAS* y acabando en la columna *1*.

Si no existen N butacas libres contiguas en la sala para la propia sesión, se devuelve null. Por ejemplo, si tomamos la sala de la figura 2 y buscamos 3 butacas contiguas libres, este algoritmo devolverá las butacas (4, 3), (4, 4) y (4, 5).

comprarEntradasRecomendadas: método que dado un objeto de tipo `ButacasContiguas`, registra la compra en la propia sesión guardando el valor actual del atributo sigIdCompra en las posiciones especificadas por el objeto dado como argumento. A continuación, se incrementa en uno el atributo sigIdCompra.

4. Código de apoyo

Los alumnos deben utilizar el código de apoyo que se encuentra en el archivo suministrado **CineAlumnos.zip**. Este archivo contiene:

- `autores.txt`: plantilla de anotación para identificar el ejercicio y los alumnos del grupo. Se insertará al comienzo de cada fichero.
- `src`: contiene el programa principal `GestorVentaEntradas.java` que proporciona una interfaz de usuario por consola para realizar todas las operaciones del cine: comprar entrada,

recoger entrada, consultar el estado de una sesión y comprar con recomendación de butacas. Este programa principal podrá ser utilizado por los alumnos para probar su solución de la práctica. Asimismo, también contiene otros tres programas principales `TestVentaEntradas.java`,`TestVentaEntradasSala.java` y `TestVentaEntradasSesion.java` que muestran cómo invocar directamente algunos de los métodos de la práctica para realizar pruebas más específicas para cada método.

- `src/cine`: contiene el paquete cine con la clase `ButacasContiguas.java`. En este paquete se deben incluir las clases a realizar por los alumnos: `Cine.java`, `Sala.java`, y `Sesion.java`.
- `src/interfazusuario`: contiene el paquete interfazusuario con las clases `VentanillaVirtualUsuario.java`, `VentanillaVirtualAdministrador.java` y `LecturaTeclado.java` que sirven de apoyo a la clase `GestorVentaEntradas.java`.
- `src/interfazusuario/menu`: contiene el paquete interfazusuario.menu con la clase `Menu.java` que sirve de apoyo a las clases `VentanillaVirtualUsuario.java` y `VentanillaVirtualAdministrador.java`.
- `src/comprobaratributos`: contiene el paquete comprobaratributos con las clases `TestAtributosCine`, `TestAtributosSala` y `TestAtributosSesion`. Estas clases implementan pruebas (JUnitTests) que permiten al alumno comprobar si ha declarado correctamente los atributos de sus clases. Para ejecutar estas pruebas, se debe seleccionar el paquete comprobaratributos en eclipse y luego Run As -> JUnit Test.

5. Recomendaciones

Se recomienda a los alumnos implementar las operaciones principales de la práctica en el siguiente orden:

1. Constructores de las clases (18%)
2. Incluir una nueva sesión en una sala dada (2%)
3. Consultar estado de la sesión y obtener las películas del cine (20%)
4. Comprar una entrada sin recomendación de butaca (26%)
5. Borrar una sesión de una sala dada (incluye equals() de la clase Sesion) (5%)
6. Recoger una entrada comprada sin recomendación de butaca (13%)
7. Comprar con recomendación de butacas (10%)
8. Recoger entradas compradas con recomendación de butaca (6%)

Junto a cada operación principal se indica su peso en la nota final de la práctica. Cada una de estas operaciones principales está asociada a varios métodos mencionados en la descripción de las clases.

6. Consideraciones de entrega

Se entregarán los tres ficheros separados `Cine.java`, `Sala.java` y `Sesion.java` sin comprimir. **Las clases incluidas en estos ficheros deben contener obligatoriamente las cabeceras de todos los métodos descritos en el apartado 3**. Si no se desea entregar algún método, este método deberá tener un cuerpo que asegure al menos la correcta compilación del programa `GestorVentaEntradas.java`. Eso se puede conseguir simplemente utilizando cuerpos vacíos o cuerpos con una única sentencia *return*.

Cada fichero debe comenzar con una anotación con los nombres de los alumnos del grupo según la plantilla suministrada en `autores.txt`:

```
@Programacion2 (
	nombreAutor1 = "nombre",
	apellidoAutor1 = "apellido1 apellido2",
	emailUPMAutor1 = "usr@alumnos.upm.es",
	nombreAutor2 = "",
	apellidoAutor2 = "",
	emailUPMAutor2 = ""
)
```

Al entregar la práctica, el sistema de entrega realizará una serie de pruebas automáticas. Si estas pruebas determinan que al menos las operaciones **constructores, incluir sesión, consulta del estado de una sesión, obtener las películas del cine y compra de una entrada sin recomendación** funcionan correctamente, se aceptará la entrega. En caso contrario, no se admitirá la entrega. Si sólo funcionan correctamente las operaciones mencionadas anteriormente, la máxima nota a la que se podrá aspirar es un 6,6 sobre 10.

7. Errores a evitar

Algunos errores que provocarán una calificación baja, de acuerdo con el baremo de calificación establecido, son (entre otros):

- Se declaran atributos públicos.
- Se realizan operaciones de entrada/salida en alguna de las clases implementadas por el alumno.

Otros errores que los profesores penalizarán también son los siguientes (entre otros):

- Atributos friendly o protected
- Getter o setter no previsto (accede a un atributo al que no debería acceder)
- Métodos auxiliares públicos
- Código duplicado
- Código innecesario o inalcanzable
- Llamadas a métodos innecesarias o redundantes
- Documentación deficiente (faltan comentarios significativos)
- Atributos innecesarios (de clase o de instancia)
- Identificadores no significativos
- No sigue el convenio de nombres de Oracle
- Código mal indentado
- Uso innecesario de if para asignar o devolver valores booleanos
- Bucles con ruptura mediante return, break o continue
- Recorridos completos de una estructura (array, arraylist, etc.) cuando basta recorrer una parte

8. Pautas de programación a tener en cuenta

Se deben seguir las pautas explicadas en el “Documento de buenas prácticas” que se encuentra publicado en el aula virtual de la asignatura (plataforma Moodle).