UNIVERSIDAD DE MÁLAGA	**PROGRAMACIÓN ORIENTADA A OBJETOS** 3 de diciembre de 2014
APELLIDOS, Nombre	GRUPO
	MÁQUINA

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- El ejercicio se almacenará en el directorio **C:\POO-ISALUD**. En caso de que no exista deberá crearse, y si ya existiese, deberá borrarse todo su contenido antes de comenzar.
- Al inicio del contenido de cada fichero deberá indicarse **el nombre del alumno, titulación, grupo y código del equipo** que está utilizando**.**
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, el alumno *no debe pararse en él indefinidamente*. Puede abordar otros.
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización. La suma de los apartados puede no coincidir con la puntuación total del ejercicio.
- **Está permitido:**
  - Consultar la documentación de la API.
- **No está permitido**:
  - Utilizar otra documentación electrónica o impresa.
  - Intercambiar documentación con otros compañeros.
  - Utilizar soportes de almacenamiento, ni ningún tipo de dispositivo móvil.
- Una vez terminado el ejercicio se debe subir un fichero comprimido (sólo con los **fuentes** que se hayan realizado) a la tarea creada en el campus virtual para ello.

Se desea desarrollar una aplicación que simule la confección de carteleras de un cine con diversas salas de proyección. Para ello se deberá crear un proyecto **`prCartelera`** *con las clases que siguen, todas ellas en el paquete* **`cartelera`**. *En el campus virtual se encuentran disponibles la clase* `Sesion`*, el fichero "*`cartelera.txt`*" y un programa Java de pruebas.*

1. (0,*25 puntos*) La clase `CarteleraException` que se utilizará para tratar las situaciones excepcionales que se consideren convenientes. Se tratará de excepciones no comprobadas (*unchecked*).

2. La clase `Sesion`, disponible en el campus virtual, representa instantes de tiempo, incluyendo hora y minutos. Esta clase dispone de un constructor para crear sesiones a partir de dos enteros, un método para calcular la diferencia en minutos entre dos sesiones, y métodos habituales para acceder a sus características, para definir la igualdad, un orden natural y una representación textual.

3. (*1,75 puntos*) La clase `Sala`, que mantenga información sobre el nombre de una sala de cine (de tipo `String`), la película que se proyecta (de tipo `String`) y su duración en minutos (de tipo `int`). La clase dispondrá de:

3.1. Un constructor que cree instancias de la clase a partir de un nombre de sala, un título de película y una duración, considerando que la duración no puede ser negativa.

3.2. Métodos para obtener el nombre de la sala, el título de la pelicula y la duración. Y métodos para modificar el título de la película y la duración.

3.3. Una noción de igualdad donde dos salas son iguales cuando coinciden sus nombres y sus títulos de pelicula (independientemente de mayúsculas y minúsculas).

3.4. Un orden natural donde una sala se considera menor que otra cuando su nombre es anterior lexicográficamente, independientemente de mayúsculas y minúsculas. Y, en caso de igualdad, cuando el título de la película que se proyecta en una sea menor lexicográficamente que el título que se proyecta en la otra (de nuevo, independientemente de mayúsculas o minúsculas).

3.5. La representación textual de una sala vendrá dada por:

```
Nombre de la sala > Título de la película (duración min)
```

Por ejemplo:

```
Sala 4 > Los juegos del hambre: Sinsajo (109 min)
```

4. (*1,00 puntos*) Defínase un orden alternativo para los objetos de la clase `Sala`, que determine que una sala es menor que otra cuando los títulos de las películas que se proyectan en cada una de ellas lo sean lexicográficamente (independientemente de mayúsculas y minúsculas). En caso de igualdad, se utilizará el orden lexicográfico del nombre de las salas (sin distinguir entre mayúsculas y minúsculas).

5. (*6 puntos*) La clase `Cartelera`, que debe mantener información sobre la cartelera de un multicine. Así, la clase mantendrá una estructura que sea capaz de asociar a cada sala (de tipo `Sala`), una colección con las sesiones en que se proyecta la película. La clase incluirá:

5.1. *(0,25 puntos)* Un constructor sin argumentos que permita inicializar la cartelera a una estructura vacía.

5.2. *(1,00 puntos)* Un método que permita añadir a una sala (proporcionada como primer argumento) una nueva sesión (proporcionado como segundo argumento):

```
void nuevaSesion(Sala sala, Sesion sesion)
```

5.3. *(1,50 puntos)* Dos métodos para leer información de la cartelera almacenada en un fichero, o a través de un objeto `Scanner`.

```
void leerCartelera(String fichero) throws FileNotFoundException

void leerCartelera(Scanner sc)
```

En ambos casos, se supondrá que la información estará organizada por líneas con el siguiente formato (consúltese en el fichero "`cartelera.txt`"):

```
Nombre sala>Título película(duracion)-hh:mm-hh:mm-hh:mm-…-hh:mm
```

En caso de producirse algún error de formato, deberá lanzarse una excepción `CarteleraException`.

5.4. *(1,25 puntos)* Un método que devuelva `true` cuando las sesiones de todas las salas son consistentes, considerando que lo son cuando las sesiones de cada sala están separadas lo suficiente para que la película programada pueda proyectarse (es decir, el tiempo entre una sesión y la siguiente es siempre superior a la duración de la película):

```
boolean esConsistente()
```

Si la sala no aparece en la parrilla se deberá lanzar una excepción.

5.5. (*0,75 puntos*) Un método que devuelva un conjunto ordenado con todas las sesiones de una película, cuyo título se pasa como argumento:

```
Set<Sesion> todasSesiones(String tituloPelícula)
```

Obsérvese que una película puede proyectarse en varias salas. Si la película no se proyecta en ninguna sala, debe devolverse el conjunto vacío.

5.6. (*0,75 puntos*) Métodos para mostrar la parrilla de programación sobre un fichero o sobre un objeto `PrintWriter`:

```
void mostrarCartelera(String fichero)

void mostrarCartelera(PrintWriter pw)
```

La información a mostrar en el fichero o transmitir sobre el flujo de salida vendrá dado (por ejemplo) por líneas con el siguiente formato:

```
Sala 1 > Trash: ladrones de esperanza (115 min)
    [15:30][17:50][20:10][22:25]
Sala 2 > Los pingüinos de Madagascar (92 min)
    [16:00][18:00][20:05][22:00]
Sala 3 > Dos tontos todavía más tontos (110 min)
    [17:40][19:50]
Sala 4 > Los juegos del hambre: Sinsajo (125 min)
    [16:10][18:40][21:10]
Sala 5 > Los juegos del hambre: Sinsajo (125 min)
    [17:30][19:55][22:10]
```

Para cada sala, después de su nombre aparecerá " > ", a continuación el título de la pelicula y su duracion seguida de la expresión "min" entre paréntesis. En la siguiente línea, con un tabulador como sangrado, todas las sesiones consecutivas.

6. (*1,00 puntos*) Definir una clase `CarteleraPorPeliculas` que presente un comportamiento similar al de la clase `Cartelera`, pero que mantenga ordenada la cartelera por los títulos de las películas. Se valorará que la reutilización sea la mayor posible.