DEPARTAMENTO Lenguajes y Ciencias de la Computación

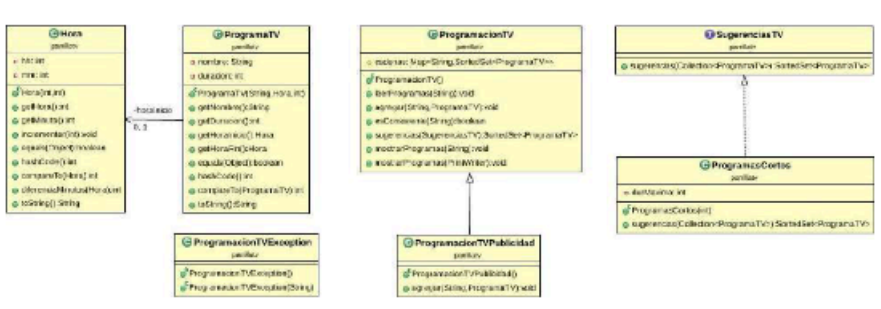
Programación Orientada a Objetos
(22 de junio de 2022)

APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA	

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- El alumnado debe haber comprobado si cumple los requisitos para acogerse a la modalidad de evaluación continua. De no ser así, además de la prueba práctica, deberá realizar un examen teórico adicional y, por lo tanto, ponerse en contacto con el profesorado.
- Al **inicio del contenido de cada fichero** realizado deberá aparecer un comentario con los **apellidos y nombre, titulación y grupo.**
- **Está permitido:**
  - Consultar los apuntes (CV), la API (Internet), la guía rápida de la API (CV).
  - Añadir métodos privados a las clases.
- **No está permitido:**
  - Intercambiar documentación con otros compañeros.
  - Recibir ayuda de otras personas. Se debe realizar personal e individualmente la solución del ejercicio propuesto.
  - Añadir métodos no privados a las clases.
  - Añadir variables o constantes a las clases.
  - Modificar la visibilidad de las variables, constantes y métodos que aparecen en el diagrama UML.
  - Modificar el código suministrado.
- Una vez terminado el ejercicio, se debe crear un archivo comprimido de la carpeta SRC del proyecto y subirlo a la tarea correspondiente del campus virtual.
- 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.
- Para la corrección del ejercicio se utilizarán **programas de detección de copias/plagios**.
- Con posterioridad a la realización del ejercicio, el profesor podrá convocar a determinado/as alumno/as para realizar **entrevistas personales** con objeto de comprobar la autoría de las soluciones entregadas.

Se desea desarrollar una aplicación que simule la distribución de las parrillas de programación de diversas cadenas de televisión en un día determinado. Para ellos se deberá crear un proyecto `prProgramasTV` *con las clases que siguen, todas ellas en el paquete* `parrillatv`*. En el campus virtual se encuentran disponibles la clase* `Hora`*, el fichero "*`programacion.txt`*" y un programa Java de pruebas (*`PruebaProgramacionTV`*). La parrilla de programación de una cadena de televisión permite conocer los programas que se emiten en las distintas horas de un día.*



1) **(0.25 ptos.)** La clase `ProgramacionTVException` que se utilizará para tratar las situaciones excepcionales que se consideren convenientes. Se tratará de excepciones **no comprobadas** (*unchecked*).

2) La clase `Hora`, disponible en el campus virtual, representa instantes de tiempo, incluyendo hora y minutos. Esta clase dispone de un constructor para crear horas a partir de dos enteros, un método para calcular la diferencia en minutos entre dos horas, y métodos habituales para acceder a sus características, para definir la igualdad, un orden natural y una representación textual. Es importante consultar la clase y la documentación que acompaña a cada método para resolver el resto del ejercicio.

3) **(1,75 ptos.)** La clase `ProgramaTV`, que mantenga información sobre el nombre de un programa de televisión (de tipo `String`), la hora de inicio del mismo (de tipo `Hora`) 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, una hora y una duración, considerando que la duración no puede ser negativa, en cuyo caso se generará la correspondiente excepción.

   3.2. Métodos para obtener el nombre del programa, la hora de inicio y la duración:

```
public String getNombre();
public Hora getHoraInicio();
public int getDuracion();
```

   3.3. Un método que devuelva la hora de finalización del programa:

```
public Hora getHoraFin()
```

   3.4. Una relación de igualdad donde dos programas son iguales cuando coinciden su nombre (independientemente de mayúsculas y minúsculas), su hora de inicio y su duración.

   3.5. Un orden natural donde un programa se considera menor que otro cuando su hora de inicio es anterior. En caso de tener un inicio coincidente, cuando su duración es menor. Y si estas también coinciden, cuando el nombre es anterior lexicográficamente, sin distinguir minúsculas y mayúsculas.

   3.6. La representación textual de un programa vendrá dada por:

```
NOMBRE EN MAYÚSCULAS@[hh:mm]-duración
```

   Por ejemplo:

```
EL INTERMEDIO@[21:30]-45
```

4) **(0,50 ptos.)** Proporciónese un orden alternativo sobre los objetos de la clase `ProgramaTV`, de forma que un programa se considere menor que otro cuando el nombre sea anterior en orden lexicográfico (independientemente de mayúsculas y minúsculas) y, en caso de igualdad, cuando la duración sea menor. En caso de igualdad también en la duración, cuando la hora de inicio sea anterior.

5) **(5,00 ptos.)** La clase `ProgramacionTV`, que debe mantener información sobre las parrillas de programación de diferentes cadenas de televisión. Así, la clase mantendrá una variable protegida, `cadenas`, que represente una estructura que sea capaz de asociar a nombres de cadenas de TV (`String`) conjuntos ordenados (según el orden natural) con los programas que se pueden ver en un día determinado en esas cadenas. La clase incluirá:

   5.1. Un constructor sin argumentos, que permita inicializar la programación a una programación vacía.

   5.2. Un método que permita añadir al nombre de una cadena de TV (proporcionada como primer argumento) un programa (proporcionado como segundo argumento):

```
public void agregar(String cadenaTV, ProgramaTV prog)
```

   5.3. Un método para leer información de la programación almacenada en un fichero, cuyo nombre se proporciona como parámetro.

```
public void leerProgramas(String fichero)
```

   Se supondrá que la información está organizada por líneas con el siguiente formato (consúltese en el fichero "programacion.txt"):

```
Nombre de la cadena de TV>Nombre del Programa@hh:mm-duración
```

   Podéis usar la expresión regular "`[>@:-]+`". Para cada línea se añadirá al nombre de la cadena el programa de TV correspondiente. En caso de que una línea no tenga el formato correcto, esta se obviará y se continuará procesando el resto de líneas.

   5.4. Métodos para mostrar la parrilla de programación sobre un fichero o sobre un objeto `PrintWriter`, respectivamente:

```
public void mostrarProgramas(String fichero)
public void mostrarProgramas(PrintWriter pw)
```

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

```
Telecinco:
    PASAPALABRA@[20:10]-55
    INFORMATIVOS TELECINCO@[21:05]-50
A3:
    TIERRA AMARGA@[17:45]-70
    ¡BOOM!@[19:00]-55
    PASAPALABRA@[20:00]-55
    NOTICIAS@[21:00]-40
    EL HORMIGUERO@[21:45]-55
```

   Después del nombre de cada cadena, aparecen dos puntos y a continuación en líneas distintas, un tabulador seguido del nombre del programa, la hora de inicio y la duración, tal como se muestra en el ejemplo.

   5.5. Un método que devuelva `true` cuando la parrilla de una determinada cadena (proporcionada como argumento) es consistente, considerando que lo es cuando ningún programa tiene una hora de finalización posterior a la hora de inicio del programa siguiente:

```
public boolean esConsistente(String cadenaTV)
```

   Si la cadena de TV no aparece en la parrilla se deberá lanzar una excepción.

(1,00 ptos.) La clase `ProgramacionTVPublicidad` que extiende el comportamiento de la clase `ProgramacionTV`, de forma que, al agregar un programa, incluya también una cuña publicitaria después del mismo. La publicidad se representará como un programa que tenga nombre "Publicidad", cuya hora inicial sea la hora final del programa agregado, y con una duración fija de 5 minutos.

(1,50 ptos.) La interfaz `SugerenciasTV` que define un método para seleccionar, según diferentes criterios (establecidos en las clases que puedan implementar la interfaz), un conjunto ordenado de programas de TV a partir de una colección de programas que se pasa como argumento. La signatura del método a definir es:

```
SortedSet<ProgramaTV> sugerencias(Collection<ProgramaTV> progs)
```

Una vez definida la interfaz, impleméntense:

7.1. Una clase `ProgramasCortos` que implemente la interfaz, de tal forma que:

i) Incluya un atributo privado para almacenar la duración de referencia (de tipo `int`).

Si has llegado hasta aquí... Te mereces una Mahou

ii) Incluya un constructor que inicialice la variable anterior (duración de referencia) al valor recibido como parámetro.

iii) Implemente el método `sugerencias`, devolviendo un conjunto ordenado (según el criterio establecido por el orden alternativo especificado en el ejercicio 4) con los programas cuya duración sea menor que la duración de referencia.

7.2. En la clase `ProgramacionTV` se ha de añadir un método:

```
public SortedSet<ProgramaTV> sugerencias(SugerenciasTV seleccion)
```

que devuelva un conjunto ordenado con las sugerencias que establezca el objeto `seleccion` pasado como argumento, de entre todos los programas de todos las cadenas almacenadas.