1. En el contexto del diseño e implementación de una aplicación de gestión de bibliotecas, se han identificado los tipos Persona, Libro y MedioAudiovisual. A partir de la definición de cada una debe escribir su **interfaz** correspondiente en Java. Para ello identifique sus **propiedades**, en función de la descripción del problema. Para cada propiedad decida su tipo y si debe ser consultable y/o modificable. Defina también los **tipos enumerados** que necesite para representar las propiedades de los tipos anteriores.

   a) Las **personas** que son socias de las bibliotecas tienen datos que necesitamos conocer, como el *DNI* (por ejemplo, "3224326H"), el *nombre* ("Antonio"), los *apellidos* ("González Hernández"), y la *fecha de nacimiento* (por ejemplo, el 2 de agosto de 1991), la edad (por ejemplo, 25) y el email (por ejemplo, "agonzalez91@gmail.com"). Todos estos datos pueden cambiarse una vez creado un usuario en el sistema, para subsanar posibles errores.
   b) Cada **libro** posee un código llamado *ISBN*, formado por una serie de dígitos y otros caracteres (por ejemplo, "978-0385536516"), que lo identifica de manera inequívoca. Otros datos del libro que necesitamos manejar son el *título* (por ejemplo, "Dexter: El oscuro pasajero"), el nombre del *autor* ("Jeff Lindsay"), el *número de páginas* (por ejemplo, 288), la *fecha de adquisición* (12 de febrero de 2005) y el *precio* que costó en euros (19,90). Además, el sistema almacena una estimación del *número de copias vendidas* del libro (por ejemplo, 2.500.000). La aplicación necesita conocer para cada libro si se trata o no de un *best-seller* (un libro se considera un *best-seller* si ha vendido más de 500.000 copias; este valor podría cambiar en futuras versiones de nuestra aplicación). Cada libro tiene asignado un *tipo de préstamo*, que puede ser diario, semanal o mensual. Una vez creado un libro en el sistema, sólo el número de copias vendidas y el tipo de préstamo pueden ser cambiados. Además de los métodos de consulta y modificación de las propiedades, se desea disponer de un método que devuelva el número de días que puede ser prestado el libro, en función del tipo de préstamo (diario: 1 día; semanal: 7 días; mensual: 30 días).
   c) Los **prestamos** están conformados por un *usuario*, que es la persona que realiza el préstamo, un *libro*, una *fecha de préstamo* y una *fecha de devolución*, que se calcula a partir de la fecha de préstamo y el tipo de préstamo del libro. La fecha de préstamo es el único dato que puede ser cambiado, para permitir que un préstamo sea renovado.
   d) Un **medio audiovisual** posee un *código* único alfanumérico (por ejemplo, "d0223"), un *título* (por ejemplo, "Breaking Bad – Temporada 1"), la *duración* del vídeo o el audio (1 hora y 5 minutos), la *fecha de adquisición* (22 de marzo de 2009) y el *precio* que costó en euros (39,95). Existen audiovisuales de tres *tipos*: vídeo, música y audiolibro. Además, el sistema necesita conocer el *número de discos* que conforman el audiovisual. Cada audiovisual tiene asignado un *tipo de préstamo*, que indica el periodo por el que el audiovisual puede ser retirado por los usuarios de la biblioteca. Los tipos de préstamo son diario, semanal y mensual. Una vez dado de alta un audiovisual en el sistema, sólo se puede cambiar el tipo de préstamo, el resto de las propiedades permanecerán inalterables.

2. Implemente las clases correspondientes a los tipos del ejercicio anterior: PersonaImpl, LibroImpl y MedioAudiovisualImpl. A continuación, se proporciona información acerca de los constructores y la representación como cadena de estas clases.

   a) Las **personas** se representan textualmente mediante el DNI, seguido de un guión, los apellidos, una coma, el nombre, otro guión y la fecha de nacimiento (por ejemplo, "28864657W – García Vaquero, Pascual – 15/09/1998"). Se requiere que la clase que implemente al tipo contenga dos constructores, uno que permita crear objetos para representar personas a partir de todas sus propiedades básicas, y otro con todas las propiedades básicas excepto el email, que se dejará con el valor null.
   a) Los **libros** son representados textualmente en la aplicación mediante su título, seguido entre paréntesis del texto "ISBN:" y el ISBN del libro (por ejemplo, "Dexter: El oscuro pasajero (ISBN: 978-

0385536516)"). La clase que implemente el tipo debe contener un único constructor que reciba como parámetros todos los datos necesarios para representar a un libro en el sistema.

b) Los **prestamos** son representados textualmente en la aplicación mediante el título del libro, seguido del dni del usuario, la fecha del préstamo y la fecha de devolución, con estos tres datos entre paréntesis, bien identificados y separados por punto y coma. Por ejemplo, "Dexter: El oscuro pasajero (usuario: 28864657W, fecha préstamo: 12/12/2016, fecha devolución: 12/01/2017)". La clase debe tener un único constructor que reciba como parámetros todos los datos necesarios para representar a un libro en el sistema.

c) Los **medios audiovisuales** son representados textualmente en la aplicación mediante su título, seguido entre paréntesis del texto "DVD" y el código, si se trata de un vídeo, o del texto "CD" y el código, si se trata de música o un audiolibro (por ejemplo, "Breaking Bad – Temporada 1 (DVD d0223)"). La clase que implementa el tipo debe contener un único constructor que reciba como parámetros todos los datos necesarios para dar de alta un nuevo audiovisual en el sistema.

3. En el marco de la implementación de una aplicación para gestionar universidades, se definen los tipos Asignatura, Beca, Persona, Espacio, Nota y Tutoría. A partir de la definición de cada una debe escribir su **interfaz** correspondiente en Java. Para ello identifique sus **propiedades**, en función de la descripción del problema. Para cada propiedad decida su tipo y si debe ser consultable y/o modificable. Defina también los **tipos enumerados** que necesite para representar las propiedades de los tipos anteriores.

   a) Una **asignatura** se identifica mediante un *nombre* (por ejemplo "Fundamentos de Programación") y un *código* numérico de 7 dígitos (por ejemplo, "0000230"). Cada asignatura tiene un número determinado de *créditos* (se admiten asignaturas con número no entero de créditos). Las asignaturas pueden ser de tres *tipos*, según el periodo del curso en el que se imparten: anual, de primer cuatrimestre y de segundo cuatrimestre. Cada asignatura es impartida en un *curso*, y su docencia es asumida por un *departamento* concreto. Ninguna de las propiedades que definen a una asignatura, salvo el departamento, varía una vez creada la misma.

   b) Las **becas** se identifican mediante un *código* alfanumérico único. Cada beca tiene asignada una *cuantía total* en euros (puede contener decimales), y una *duración* en meses (las becas siempre duran un número determinado de meses completos). Existen becas de tres *tipos*: ordinaria, de movilidad y de empresa. Para cada beca es necesario conocer su *cuantía mensual*, que se calcula dividiendo la cuantía total entre el número de meses que dura la beca. Una vez concedida una beca su código y su tipo nunca cambian, no así la cuantía y la duración, que pueden variar debido a alegaciones por parte de los alumnos o por cambios legislativos.

   c) El tipo **persona**[1] se utiliza para modelar a todas las personas (ya sean profesores o alumnos) que participan en la aplicación. Cada persona tiene un *DNI* (formado siempre por ocho dígitos y una letra), un *nombre*, unos *apellidos*, una *fecha de nacimiento*, una dirección de *email* y una *edad*. Todas las propiedades de la persona pueden variar una vez creada.

   d) Un **espacio** representa un lugar físico ubicado en un centro y en el cual se realizan tareas docentes. Un espacio puede ser de varios *tipos*: un aula de teoría, un laboratorio, un seminario, un aula de examen o de otro tipo. Cada espacio tiene un *nombre* (por ejemplo, "A3.10") y una *capacidad* dada por el número máximo de personas que admite. Además, un espacio está ubicado en una determinada *planta*. Todas las propiedades de un espacio pueden variar una vez creado, con excepción de la planta.

   e) Una **nota** representa la calificación obtenida por un alumno en una *asignatura* de un *curso académico* concreto, dado por el año de comienzo del mismo (por ejemplo, 2015 para el curso 2015/16). La nota corresponde a una *convocatoria*, que puede ser la primera, la segunda, o la tercera, y tiene un *valor* numérico comprendido entre 0 y 10. Si el valor numérico es mayor o igual a 9, la nota puede ser distinguida con una *mención de honor*. La nota también tiene una *calificación*, que se calcula a partir del valor numérico y de la mención de honor, y que puede ser suspenso (si el valor numérico es menor

[1] Este tipo tiene la misma definición que el tipo Persona del ejercicio 1.

que 5), aprobado (si el valor numérico es mayor o igual que 5 y menor que 7), notable (si el valor numérico es mayor o igual que 7 y menor que 9), sobresaliente (si el valor numérico es mayor o igual que 9 y la nota no tiene mención de honor) o matrícula de honor (si el valor numérico es mayor o igual que 9 y la nota tiene mención de honor). Ninguna de las propiedades que definen a una nota varía una vez creada la misma.

f) La **tutoría** representa un intervalo de tiempo que todo profesor tiene reservado para atender a sus alumnos. Una tutoría tiene lugar un *día de la semana* (de lunes a viernes[2]) y tiene una *hora de comienzo*, una *hora de fin* y una *duración*, que es la diferencia entre la hora de fin y la hora de comienzo. Ninguna de las propiedades que definen a una tutoría varía una vez creada la misma.

4. Implemente las clases correspondientes a los tipos del ejercicio anterior: AsignaturaImpl, BecaImpl, EspacioImpl, NotaImpl, PersonaImpl y TutoriaImpl. A continuación, se proporciona información acerca de los constructores y la representación como cadena de estas clases.

   b) Las **asignaturas** serán representadas textualmente en la aplicación mediante el código de la misma entre paréntesis, seguido del nombre, por ejemplo, "(0000230) Fundamentos de Programación". Se requiere que la clase que implemente al tipo contenga un constructor que permita crear objetos para representar asignaturas a partir de todas las propiedades básicas que las definen.

   c) Con respecto a las **becas**, la aplicación las representará textualmente mediante el código y el tipo, separados por comas y colocados entre corchetes (por ejemplo, "[ABB2024, MOVILIDAD]"). Se requiere que la clase que implemente al tipo contenga dos constructores: uno que permita crear becas una vez conocidas todas sus propiedades básicas, y otro que permita crear becas de las que no se conoce aún su cuantía ni su duración. Se supondrán en este caso la cuantía y duración mínimas, que son 1500,0 € y 1 mes, respectivamente (tenga en cuenta que la cuantía mínima podría variar en futuras versiones de la aplicación).

   d) Una **persona** se representará mediante el DNI, seguido de un guión, los apellidos, una coma, el nombre, otro guión y la fecha de nacimiento. Por ejemplo: "28864657W – García Vaquero, Pascual – 15/09/1998". Se requiere que la clase que implemente al tipo contenga dos constructores, uno que permita crear objetos para representar personas a partir de todas sus propiedades básicas, y otro con todas las propiedades básicas excepto el email, que se inicializará con una cadena vacía.

   e) Un **espacio** se representará mediante su nombre seguido de la planta entre paréntesis. Por ejemplo, "A3.10 (planta 3)". La clase que implemente el tipo tendrá un único constructor con todas sus propiedades básicas.

   f) Una **nota** se representará mediante la asignatura, el curso académico (formado por el año de inicio del curso seguido de un guión y de los dos últimos dígitos del año de final del curso), la convocatoria, el valor numérico y la calificación, separados por comas. Por ejemplo, "(0000230) Fundamentos de Programación, 2015-16, PRIMERA, 7.5, NOTABLE". La clase que implemente el tipo tendrá dos constructores, uno que permita crear objetos para representar notas a partir de todas sus propiedades básicas, y otro con todas las propiedades básicas excepto la mención de honor, que se inicializará con un valor falso.

   g) Una **tutoría** se representará por un carácter que indica el día (L, M, X, J o V), seguido de la hora de comienzo, un guión y la hora de fin. Por ejemplo, "X 10:30-12:30". La clase que implemente el tipo tendrá dos constructores, uno con el día de la semana, la hora de comienzo y la hora de fin, y otro con el día, la hora de comienzo y la duración.

[2] Utilice el tipo enumerado `DayOfWeek`, http://docs.oracle.com/javase/8/docs/api/java/time/DayOfWeek.html.

5. Se desea implementar los tipos Artista y Cancion, que se usarán en el desarrollo de una aplicación de *streaming* de música basada en Spotify (http://www.spotify.com). En una primera fase del desarrollo, se usará la siguiente definición de sus propiedades y características:

   Artista:
   - Propiedades:
     - Id, de tipo String, consultable. Se trata de una cadena formada por números y letras que identifica de manera inequívoca al artista en nuestro sistema. Esta propiedad es necesaria puesto que pueden existir varios artistas distintos con el mismo nombre.
     - Nombre, de tipo String, consultable.
     - Género, de tipo String, consultable. Indica a qué género musical se dedica mayoritariamente el artista. Dado que existen multitud de géneros distintos, se ha decidido representar esta propiedad mediante una cadena de texto, en lugar de mediante un tipo enumerado.
     - Popularidad, de tipo Integer, consultable y modificable. Se trata de un valor que ira de 0 (artista muy poco popular) a 100 (artista muy popular).
     - URL Imagen, de tipo String, consultable y modificable. Almacena una dirección URL a una imagen del artista.
   - Constructor: recibe un parámetro por cada una de las propiedades anteriores.
   - Representación como cadena: el nombre del artista, seguido de su identificador entre corchetes.

   Cancion:
   - Propiedades:
     - Id, de tipo String, consultable. Se trata de una cadena formada por números y letras que identifica de manera inequívoca a la canción en nuestro sistema. Esta propiedad es necesaria puesto que pueden existir varias canciones distintas con el mismo nombre.
     - Nombre, de tipo String, consultable.
     - Artista, de tipo Artista, consultable. Representa al intérprete de la canción.
     - Duración, de tipo Duration, consultable.
     - Número de pista, de tipo Integer, consultable. Indica el número que ocupa esta canción en el álbum al que pertenezca.
     - Popularidad, de tipo Integer, consultable y modificable. Se trata de un valor que ira de 0 (canción muy poco popular) a 100 (canción muy popular).
   - Constructor: recibe un parámetro por cada propiedad básica del tipo.
   - Representación como cadena: el nombre de la canción, seguido del artista entre paréntesis; por ejemplo, “Whole Lotta Love (Led Zeppelin [36QJpDe2go2KgaRleHCDTp])”.

   Se pide:

   a) Implemente las interfaces Artista y Cancion y las clases ArtistaImpl y CancionImpl.
   b) Implemente una clase TestCancion que incluya un método main en el que se cree una canción y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado, y cada una de las propiedades del artista que la interpreta.

6. Se desea implementar una aplicación para manejar información de películas y series de televisión, inspirado en The Movie Database (http://www.tmdb.com). Para ello se han definido los siguientes tipos:

   MiembroStaff, representa una persona que trabaja en una película o serie de televisión:
   - Propiedades:
     - Id, de tipo Integer, consultable. Representa un número que sirve para identificar, de forma única, al miembro del staff de una película.
     - Nombre, de tipo String, consultable y modificable. Representa el nombre de la persona que participa en la película.

  - Fecha de nacimiento, de tipo LocalDate, consultable y modificable.
  - Fecha de defunción, de tipo LocalDate, consultable y modificable. Representa la fecha de fallecimiento de la persona. Si el miembro del staff aún vive, su valor es nulo.
  - Edad, de tipo Integer, consultable. Se calcula a partir de la fecha de nacimiento. Si la persona ya ha fallecido, esta propiedad representa la edad que tenía en el momento de su muerte.
  - Lugar de nacimiento, de tipo String, consultable y modificable. Ciudad, estado o país en el que nació la persona.
  - alias, de tipo String, consultable y modificable. Nombre alternativo con el que se conoce a la persona. Por ejemplo, el actor Dwayne Johnson tiene como alias "The Rock". Si no se le conoce ningún alias, esta propiedad tendrá el valor null.
- Constructores:
  - C1: recibe un parámetro por cada propiedad básica del tipo.
  - C2: recibe un parámetro por cada propiedad básica del tipo, excepto la fecha de defunción, que se queda con el valor null.
  - C3: recibe el id y el nombre (el resto de propiedades básicas se quedan con el valor null).
- Representación como cadena: el nombre, seguido del alias, entre paréntesis, si tiene alias, seguido de un guión y el id. Por ejemplo: "Juan Antonio Bayona (Jota Bayona) - 1235" o "Steven Spielberg – 1236".

Pelicula, representa una pelicula**:**
- Propiedades:
  - Id, de tipo Integer, consultable. Representa un número que sirve para identificar, de forma única, a la película.
  - Título, de tipo String, consultable. Representa el título de la película en su versión española (puede coincidir con el título original).
  - Título original, de tipo String, consultable y modificable. Representa el título original de la película.
  - Idioma original, de tipo String, consultable y modificable. Idioma principal en el que está rodada la película.
  - Fecha de estreno, de tipo LocalDate, consultable y modificable. Fecha en la que se estrenó la película.
  - Duración, de tipo Duration, consultable y modificable. Representa la duración (en minutos) de la película.
  - Tipo de metraje, de tipo TipoMetraje, consultable. Se calcula a partir de la duración. Los tipos de metraje posibles son CORTOMETRAJE (menos de 30 minutos), MEDIOMETRAJE (de 30 a 60 minutos) y LARGOMETRAJE (más de 60 minutos).
  - Género, de tipo String, consultable y modificable. Género principal de la película.
  - Productora, de tipo String, consultable y modificable. Nombre de la empresa que produce la película.
  - País, de tipo String, consultable y modificable. País de origen de la película.
  - Director, de tipo MiembroStaff, consultable y modificable. Representa el director de la película.
- Constructores:
  - C1: recibe un parámetro por cada propiedad básica del tipo.
  - C2: recibe sólo el id y el título (el resto de propiedades básicas se quedan con el valor null).
- Representación como cadena: el id seguido de un guión, el título, seguido del año de la fecha de estreno entre paréntesis, siempre que esté disponible la fecha de estreno (por ejemplo, "1222 - Tiempos modernos (1936)". Si no está disponible la fecha de estreno, entonces, se mostrará solo el id y el título (por ejemplo, "1224 - Un monstruo viene a verme").

SerieTV:
- Propiedades:

- Id, de tipo Integer, consultable. Representa un número que sirve para identificar, de forma única, a la serie de televisión.
- Nombre, de tipo String, consultable. Representa el nombre de la serie.
- Fecha de primera emisión, de tipo LocalDate, consultable y modificable. Fecha en la que se emitió por primera vez la serie.
- Fecha de última emisión, de tipo LocalDate, consultable y modificable. Fecha en la que se emitió el último capítulo de la serie.
- Cadena de TV, de tipo String, consultable y modificable. Representa el nombre de la cadena principal de televisión que ha producido la serie.
- Género, de tipo String, consultable y modificable. Género principal de la serie.
- Nombre original, de tipo String, consultable y modificable. Nombre en el idioma original de la serie.
- Idioma original, de tipo String, consultable y modificable. Idioma original de la serie.
- Popularidad, de tipo Double, consultable. Número que representa la popularidad de la serie en un sitio web.
- Estado, de tipo EstadoSerie, consultable y modificable. Una serie puede estar en uno de los siguientes estados: FINALIZADA, si la serie ha terminado y ya no se producen nuevos episodios; EN_CURSO, si la serie no está en emisión, pero aún se emitirán nuevas temporadas; y EN_PRODUCCION, si la serie se está rodando, y aún no se ha emitido ningún episodio de la misma.
- Número de temporadas, de tipo Integer. Consultable y modificable. Indica el número de temporadas grabadas de la serie.

■ Constructores:
- C1: recibe un parámetro por cada propiedad básica del tipo.
- C2: recibe el id, el nombre, el estado y el número de temporada.
- C3: recibe el id y el nombre y que crea una serie con estado EN_CURSO, y una temporada.

■ Representación como cadena: El id seguido de un guión, el nombre de la serie y su estado entre paréntesis, por ejemplo: “1111 - Juego de tronos (EN_CURSO)”.

Se pide:

a) Implementar las interfaces MiembroStaff, Pelicula y SerieTV y las clases MiembroStaffImpl, PeliculaImpl y SerieTVImpl.
b) Implementar tres clases TestMiembroStaff, TestPelicula y TestSerieTV que incluyan un método main en el que se creen varios miembros del Staff, varias películas y varias SeriesTV, y se muestre por la consola la representación como cadena y las propiedades de los objetos creados. Use todos los constructores e invoque a todos los métodos.

7. Se desea implementar el tipo Sesion, que representa cada una de las sesiones de proyección de películas en un cine, y que se usará en el desarrollo de una aplicación de gestión de este tipo de negocios. En una primera fase del desarrollo, se usará la siguiente definición de sus propiedades y características:

Sesion:

■ Propiedades:
- Número de sala, de tipo Integer, consultable.
- Película, de tipo Pelicula, consultable[3].
- Fecha, de tipo LocalDate, consultable. Indica la fecha en que se proyectará la película.
- Hora de inicio, de tipo LocalTime, consultable.
- Hora de fin, de tipo LocalTime, consultable. La hora de fin de la película se calcula a partir de la hora de inicio y la duración de la película, teniendo en cuenta además que se añaden 10

[3] El tipo de esta propiedad se corresponde con el tipo Pelicula del ejercicio anterior.

minutos al inicio de la sesión para proyectar los anuncios, y 10 minutos en mitad de la película si esta sesión tiene descanso.
    - Duración, de tipo Duration, consultable. Se calcula de igual manera que la hora de fin.
    - Precio, de tipo Double, consultable y modificable. Expresado en euros.
    - VOS, de tipo Boolean, consultable. Indica si la película se proyecta o no en Versión Original Subtitulada.
    - Descanso, de tipo Boolean, consultable y modificable. Indica si la sesión incluye un descanso a mitad de la película.
    - Calificación por edades, de tipo CalificacionPorEdades, consultable. Implemente el tipo enumerado CalificacionPorEdades, que permite los valores TODOS_LOS_PUBLICOS, MAYORES_7, MAYORES_12, MAYORES_16 y MAYORES_18.
    - Sesión golfa, de tipo Boolean, consultable. Indica si la sesión se lleva a cabo de madrugada (comienza a partir de las 0:00 y antes de las 8:00).
    - Sesión matinal, de tipo Boolean, consultable. Indica si la sesión se lleva a cabo por la mañana (comienza a partir de las 8:00 y antes de las 14:00).
- Otras operaciones:
    - void retrasa(Integer minutos): retrasa la hora de inicio de la sesión los minutos indicados mediante el parámetro del método.
- Constructores:
    - C1: recibe un parámetro por cada una de las propiedades básicas del tipo.
    - C2: recibe los mismos parámetros que C1 salvo VOS y descanso, propiedades para las que se supondrá el valor `false`.
- Representación como cadena: el número de la sala, la fecha y hora de inicio, y el título de la película. Si la sesión se proyecta en Versión Original Subtitulada, se usará el título original de la película. Por ejemplo: "Sala 2 (10/12/2016 17:30) Inteligencia Artificial".

Se pide:

a) Implemente la interfaz Sesion y la clase SesionImpl.
b) Implemente una clase TestSesion que incluya un método main en el que se cree una sesión y se muestre por la consola tanto la representación como cadena de la misma, como cada una de las propiedades por separado.

8. Se desea implementar una aplicación que sirva para registrar los datos de las audiencias de series de televisión. En una primera iteración del desarrollo de la aplicación se definen los tipos:

Emision: representa la emisión de un episodio concreto en una cadena de televisión y fecha determinadas.
- Propiedades:
    - Serie, de tipo SerieTV[4], consultable. Nombre de la serie para la que se hace la medición de audiencia.
    - Cadena de TV, de tipo String, consultable. Nombre de la cadena de televisión en la que se lleva a cabo la emisión del episodio.
    - Fecha de emisión, de tipo LocalDateTime, consultable y modificable. Indica el día y hora de emisión de la serie.
    - Duración, de tipo Duration, consultable y modificable. Indica el tiempo que está en emisión el programa.
    - Fecha fin de emision, de tipo LocalDateTime, consultable. Representa el día y hora en la que se acaba la emisión. Se obtiene sumando la fecha de inicio la duración de la serie.
- Constructores:

[4] Se corresponde con el tipo del mismo nombre definido en el ejercicio 6.

- C1: recibe la serie, la cadena de televisión, la fecha de emisión y la duración.
- C2: recibe la serie, la cadena de televisión, la fecha de emisión y la fecha de finalización de la emisión.

- Representación como cadena: Nombre de la serie, seguido de la cadena en la que se realiza la emisión, y la fecha de emisión con el formato que se muestra en el siguiente ejemplo: "Juego de tronos; Antena 3;17/07/12-10:36 (martes)".

Medicion Audiencia:

- Propiedades:
  - Emision, de tipo Emision, consultable. Emision sobre la que se realiza la medición de audiencia.
  - Espectadores, de tipo Long, consultable. Número de espectadores que ven la emisión.
  - Share, de tipo Float, consultable. El share es el porcentaje de hogares que están viendo la emisión en relación al número total de espectadores que tienen la TV encendida en ese momento. Por ejemplo, suponga que se miden las audiencias con 10000 televisores, de los cuales solamente hay encendidos 1000. En un momento concreto, 500 de ellos están sintonizando TVE 1. Decimos que TVE 1 en ese momento tiene un share del 50% (500 espectadores de 1000 televisores encendidos).
  - Rating, de tipo Float, consultable. El rating es el porcentaje de hogares o telespectadores que están viendo un programa en relación al número total de hogares o telespectadores considerados para el cálculo de audiencias. En el ejemplo anterior, decimos que TVE 1 tiene un 5% de rating (500 espectadores de 10000).
- Constructores:
  - C1: recibe un parámetro por cada propiedad básica del tipo.
  - C2: recibe el id de una serie de televisión, el nombre de la serie, la cadena de televisión en la que se emitió, la fecha y hora de emisión, la duración, el número de espectadores, el share y el rating.
- Representación como cadena: la representación como cadena de la emisión, seguida del número de espectadores (precedido por la cadena "E:"), y el share (precedido por "S:" y antecedido por el símbolo %). Por ejemplo, "Juego de tronos; Antena 3;17/07/12-10:48 (martes) E: 2506000 S: 20.7%".

Se pide:

a) Implementar las interfaces Emision y MedicionAudiencia y las clases EmisionImpl y MedicionAudienciaImpl.
b) Implementar las clases TestEmision y TestMedicionAudiencia en la que cree objetos con cada uno de los constructores definidos en cada clase y muestre las propiedades de los objetos creados por consola.

9. Se desea implementar los tipos Participante y ContratoArrendamiento, que se usarán en el desarrollo de una aplicación de gestión de contratos de alquiler de inmuebles. Se usará la siguiente definición de sus propiedades y características:

Participante:

- Propiedades:
  - Nombre, de tipo String, consultable.
  - Apellidos, de tipo String, consultable.
  - Estado civil, de tipo EstadoCivil, consultable y modificable. Implemente el tipo enumerado EstadoCivil, cuyos posibles valores son SOLTERO, CASADO, DIVORCIADO y VIUDO.
  - Dni, de tipo String, consultable.
  - Dirección, de tipo String, consultable y modificable.
  - Fecha de nacimiento, de tipo LocalDate, consultable.
  - Edad, de tipo Integer, consultable.

- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: los apellidos, seguidos por una coma y el nombre, más el DNI entre paréntesis. Por ejemplo: "Vázquez Montalbán, Manuel (12345678Z)".

ContratoArrendamiento:

- Propiedades:
  - Arrendador, de tipo Participante, consultable.
  - Arrendatario, de tipo Participante, consultable.
  - Dirección (del inmueble arrendado), de tipo String, consultable.
  - Fecha, de tipo LocalDate, consultable.
  - Duración, en días, de tipo Integer, consultable y modificable.
  - Renta anual, de tipo Double, consultable y modificable.
  - Renta mensual, de tipo Double, consultable.
  - Fianza, de tipo Double, consultable.
  - Prorrogable, de tipo Boolean, consultable.
- Constructores:
  - C1: recibe un parámetro por cada propiedad básica del tipo.
  - C2: recibe un parámetro por cada propiedad básica del tipo salvo la fecha, entendiéndose en ese caso que el contrato que se está creando tendrá como fecha la actual.
- Representación como cadena: la dirección del inmueble, y los nombres y apellidos del arrendador y del arrendatario, con el formato mostrado en el siguiente ejemplo: "Calle Pajaritos 12, 1º D, 41001, Sevilla: Vázquez Montalbán, Manuel (arrendador) - Cámara, Sixto (arrendatario)"

Se pide:

a) Implemente las interfaces Participante y ContratoArrendamiento y las clases ParticipanteImpl y ContratoArrendamientoImpl.
b) Implemente una clase TestContrato que incluya un método main en el que se cree un contrato y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado, y cada una de las propiedades de ambos participantes en el contrato.

10. Se desea implementar una aplicación para gestionar reservas hoteleras, inspirada en Booking (http://www.booking.com). En una primera iteración se quiere implementar el tipo Hotel, con la siguiente descripción:

Hotel:

- Propiedades:
  - Nombre, de tipo String, consultable y modificable. Nombre del hotel.
  - Dirección, de tipo String, consultable y modificable. Dirección del hotel.
  - Ciudad, de tipo String, consultable y modificable. Ciudad en la que está situada el hotel.
  - Telefono, de tipo String, consultable y modificable. Número de teléfono del hotel.
  - Cadena hotelera, de tipo String, consultable y modificable. Nombre de la cadena hotelera a la que pertenece el hotel.
  - Descripción, de tipo String, consultable y modificable. Contiene una pequeña descripción del hotel.
  - Categoría del hotel, de tipo CategoriaHotelera, consultable y modificable. Puede tomar los valores UNA, DOS, TRES, CUATRO O CINCO, que representan el número de estrellas del hotel. También se incluirá el valor OTROS, para aquellos tipos de alojamiento en los que no proceda hablar de la categoría.
  - Tipo de alojamiento, de tipo TipoAlojamiento, consultable y modificable. Puede tomar los valores APARTAMENTO, HOTEL, PENSION, BED_AND_BREAKFAST.

  - Categoría de precio, de tipo CategoriaPrecio, consultable y modificable. Puede tomar los valores BAJA, MEDIA, ALTA y LUJO, que categoriza los hoteles según el precio de las habitaciones.
  - Puntuación, de tipo Float, consultable y modificable. Media de las puntuaciones que le otorgan los usuarios del sitio web.
  - Número de Comentarios, de tipo Integer, consultable y modificable. Número de usuarios que han realizado un comentario sobre el hotel y le han dado una puntuación.
  - Admite mascotas, de tipo Boolean, consultable y modificable. Indica si el hotel admite mascotas.
  - Está adaptado, de tipo Boolean, consultable y modificable. Indica si el hotel está adaptado para minusválidos.
- Constructores:
  - C1: recibe un parámetro por cada propiedad básica del tipo.
  - C2: recibe el nombre, la dirección, la ciudad, el teléfono, la cadena hotelera, y la categoría y que permite crear un hotel de precio medio, sin ningún comentario de los usuarios, que no admite mascotas ni está adaptado.
  - C3: recibe el nombre, la cadena hotelera y la categoría del hotel, y permite crear un hotel de precio medio, sin comentarios, que no admite mascotas ni está adaptado (el resto de propiedades básicas quedarán con el valor null).
- Representación como cadena: El nombre del hotel, seguido entre paréntesis de tantos asteriscos como estrellas tenga (por ejemplo, “Hotel Alfonso XIII - A Luxury Collection Hotel (*****)”), o N/A si la categoría hotelera es OTRO (ej., “Hotel Dawson Place, Juliette's Bed and Breakfast (N/A)”).

Se pide:

a) Implementar la interfaz Hotel y la clase HotelImpl.
b) Implementar una clase TestHotel en la que cree tres hoteles, uno con cada constructor. Use los métodos modificadores para cambiar los valores por defecto de algunas de las propiedades no inicializadas en los constructores C2 y C3.

11. Se desea realizar una aplicación para gestionar apuestas sobre partidos de fútbol. En una primera iteración se deciden implementar los siguientes tipos:

PartidoFutbol

- Propiedades:
  - Fecha, de tipo LocalDateTime, consultable. Fecha de celebración y hora de comienzo del partido.
  - Equipo local, de tipo String, consultable. Nombre del equipo local.
  - Equipo visitante, de tipo String, consultable. Nombre del equipo visitante.
  - Goles local, de tipo Integer, consultable. Número de goles del equipo local.
  - Goles visitante, de tipo Integer, consultable. Número de goles del equipo visitante.
  - Resultado, de tipo ResultadoQuiniela, consultable. Puede tomar los valores UNO, EQUIS, DOS. El resultado será UNO, si el equipo local tiene más goles que el equipo visitante; será EQUIS, si los dos equipos han marcado el mismo número de goles, y DOS, si el equipo visitante ha marcado más goles que el local.
- Constructores: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: La fecha del partido, seguido del equipo local, el equipo visitante, los goles del equipo local, los goles del equipo visitante, y entre paréntesis el resultado de la quiniela, tal y como se muestra en el siguiente ejemplo: “24-09-16-> Sporting Gijón vs Barcelona: 0-5 (2)”. Tenga en cuenta que el resultado de la quiniela se debe representar con los caracteres 1, X ó 2.

Apuesta

- Propiedades:

- Id de usuario, de tipo String, consultable. Identificador que del usuario que realiza la apuesta.
- Fecha de la apuesta, de tipo LocalDateTime, consultable. Registra el momento en el que se realiza la apuesta.
- Partido, de tipo PartidoFutbol, consultable. Partido para el que se realiza la apuesta.
- Cantidad apostada, de tipo Float, consultable. Indica la cantidad que el usuario ha apostado en el partido.
- Goles local, de tipo Integer, consultable. El número de goles que el apostante cree que marcará el equipo local.
- Goles visitante, de tipo Integer, consultable. El número de goles que el apostante cree que marcará el equipo visitante.
- Ganadora, de tipo Boolean. Derivada. Es cierta si la apuesta es ganadora. Una apuesta es ganadora si el número de goles del equipo local y del equipo visitante de la apuesta, coinciden con el número de goles del equipo local y visitante del partido.

- Constructores:
  - C1: recibe un parámetro por cada propiedad básica del tipo.
  - C2: recibe el id del usuario que realiza la apuesta, la cantidad que apuesta, los goles que cree que va a marcar el equipo local, los goles que cree que va a marcar el equipo visitante, la fecha en la que se juega el partido, el nombre del equipo local, el nombre del equipo visitante, los goles que ha marcado el equipo local y los goles que ha marcado el equipo visitante. Crea una apuesta cuya fecha es la fecha en la que se crea el objeto en el programa.
- Representación como cadena: La fecha de la apuesta, seguida del id del usuario que realiza la apuesta, el equipo local, el visitante y la cantidad apostada. "21-09-16 04:15:00:00, demoUser, Sporting Gijón-Barcelona, 10.0".

Se pide:

a) Implementar las interfaces PartidoFutbol y Apuesta y las clases PartidoFutbolImpl y ApuestaImpl.
b) Implementar las clases TestPartidoFutbol y TestApuesta en la que cree objetos con cada uno de los constructores definidos en cada clase y muestre las propiedades de los objetos creados por consola.

12. Se desea implementar el tipo FincaAgricola, según la siguiente definición de sus propiedades y características:

FincaAgricola:

- Propiedades:
  - Nombre, de tipo String, consultable.
  - Número Registral, de tipo String, consultable.
  - Extensión (en hectáreas), de tipo Double, consultable y modificable.
  - Tipo de Finca, de tipo TipoFinca, consultable. Defina TipoFinca cuyos posibles valores son CEREAL, HUERTA, DEHESA y VINICOLA.
  - Acogida a Política Agraria Común (Acogida PAC), de tipo Boolean, consultable.
  - Precio por hectárea, de tipo Double, consultable y modificable.
  - Precio de la finca, de tipo Double, consultable.
- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: nombre, seguido por una coma y el número registral entre paréntesis. Por ejemplo: "Las Norias, (1245)".

PropietarioAgricola :

- Propiedades:
  - Nombre y apellidos, de tipo String, consultable.
  - Dni, de tipo String, consultable.
  - Finca en propiedad, de tipo FincaAgricola, consultable y modificable. Se supone que sólo se puede poseer una finca.

  - Cotización IRPF, tipo Double, consultable. La cotización se calcula como el 2% del precio de la finca en propiedad. Este porcentaje de cotización puede cambiar en el futuro.
- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: el nombre del propietario, datos de la finca y lo que cotiza, por ejemplo: "Propietario: Miguel de Cervantes Saavedra, Las Norias, (1245), 70.5".

Se pide:

a) Implemente la interfaz FincaAgricola y la clase FincaAgricolaImpl.
b) Implemente una clase TestFincaAgricola que incluya un método main en el que se cree una finca y se muestre por la consola tanto la representación como cadena de la misma, como cada una de las propiedades por separado.

13. Se desea implementar el tipo JugadorFutbol**,** según la siguiente definición de sus propiedades y características:

JugadorFutbol:
- Propiedades:
  - Nombre, de tipo String, consultable.
  - Apellidos, de tipo String, consultable.
  - Apodo, de tipo String, consultable
  - Posición preferente, de tipo TipoPosicion, consultable y modificable. Implemente el tipo enumerado TipoPosicion, cuyos posibles valores son PORTERO, DEFENSA, MEDIO y DELANTERO.
  - Número de dorsal, de tipo Integer, consultable.
  - Tarjetas amarillas, de tipo Integer, consultable y modificable.
  - Tajetas rojas, de tipo Integer, consultable y modificable.
  - Número de partidos de sanción, de tipo Integer, consultable. Se calcula a partir de las tarjetas amarillas y rojas: un partido de sanción por cada cinco tarjetas amarillas acumuladas y 2 por cada tarjeta roja.
- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: el apodo, un espacio en blanco y el número de dorsal entre paréntesis. Si no tuviese apodo (propiedad igual a null), se sustituiría por apellidos-nombre. Por ejemplo: "Mudo Vázquez (22)" o "Vázquez-Franco (22)".

Se pide:

a) Implemente la interfaz JugadorFutbol y la clase JugadorFutbolImpl.
b) Implemente una clase TestJugadorFutbol que incluya un método main en el que se cree un jugador y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado.

14. Se desea implementar el tipo Triangulo**,** según la siguiente definición de sus propiedades y características:

Triangulo:
- Propiedades:
  - LadoA, de tipo Double, consultable y modificable.
  - LadoB, de tipo Double, consultable y modificable.
  - LadoC, de tipo Double, consultable y modificable.
  - Clasificación según lados, de tipo ClasificacionLados, consultable. Se calcula a partir de la longitud de los lados. Implemente el tipo enumerado ClasificacionLados, cuyos posibles valores son EQUILATERO, ISOSCELES y ESCALENO.

  - Clasificación según ángulos, de tipo ClasificacionAngulos, consultable. Se calcula a partir de la longitud de los ladosno[5]. Implemente el tipo enumerado ClasificacionAngulos, cuyos posibles valores son RECTANGULO, ACUTANGULO y OBTUSANGULO.
  - Perimetro, de tipo Double, consultable.
- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena el apodo: la palabra Triangulo, seguida de dos puntos (:) y un espacio en blanco y entre paréntesis las longitudes de los tres lados separados por punto y coma (;). Ejemplo: "Triangulo: (2.5;10.7;5.0)".

Se pide:

a) Implemente la interfaz Triangulo y la clase TrianguloImpl.
b) Implemente una clase TestTriangulo que incluya un método main en el que se creen distintos triángulos y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado. Cree triángulos de todos los tipos (clasificación según lados y clasificación según ángulos).

15. Se desea implementar los tipos Fruto y Arbol según la siguiente definición de sus propiedades y características:

Fruto:
- Propiedades:
  - Nombre, de tipo String, consultable.
  - Semilla, de tipo Boolean, consultable. Indica si el fruto tiene o no semilla(s).
  - Tiempo de maduración, de tipo Duration, consultable.
- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: nombre y entre corchetes([ ]) el tiempo de maduración medido en días. Ejemplo: "Melocotón [105]"

Arbol:
- Propiedades:
  - Nombre, de tipo String, consultable.
  - Forma de las hojas, de tipo FormaHojas, consultable y modificable. Defina un tipo enumerado FormaHojas cuyos valores pueden ser ACICULAR, ACORAZONADA, LANCEOLADA, LINEAL, OVALADA y SAGITADA.
  - Fruto, de tipo Fruto, consultable.
  - Fecha de floración, de tipo LocalDate, consultable y modificable.
  - Fecha de comienzo de la recolección, de tipo LocalDate, consultable. Se calcula sumándole a la fecha de floración el tiempo de maduración del fruto.
- Constructor: recibe un parámetro por cada propiedad básica del tipo.
- Representación como cadena: nombre de árbol seguido entre paréntesis del nombre del fruto. Ejemplo: Melocotonero (Melocotón)

Se pide:

a) Implemente las interfaces Fruto y Arbol y las clases FrutoImpl y ArbolImpl.
b) Implemente una clase TestArbol que incluya un método main en el que se cree un árbol y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado. Muestre también cada una de las propiedades del fruto del árbol.

---

[5] Tras averiguar cuál es el lado mayor del triángulo, que puede asimilarlo a la hipotenusa, aplique el teorema de Pitágoras para saber si el triángulo tiene un ángulo recto, es decir: la suma de los cuadrados de los catetos es igual al cuadrado de la hipotenusa. En un triángulo obtuso, se cumple que la suma de los cuadrados de los lados de menor longitud es mayor que el cuadrado del lado mayor. Si no se cumplen las condiciones anteriores el triángulo tiene todos los ángulos agudos.

16. Se desea implementar el tipo Regalo para una determinada tienda *online*, según la siguiente definición de sus propiedades y características:

    Regalo:
    - Propiedades:
        - Nombre, de tipo String, consultable y modificable.
        - Código, de tipo String, consultable.
        - Tipo, de tipo TipoRegalo, consultable. Defina el tipo enumerado TipoRegalo, que puede tomar los valores MUSICA, LIBRO, ROPA y COMPLEMENTO.
        - Precio, de tipo Double, consultable y modificable.
        - Descuento, de tipo Integer, consultable y modificable. Representa el porcentaje que se descuenta como bonificación por compra electrónica.
        - Precio con descuento, de tipo Double, consultable. Se calcula a partir del precio y el descuento.
    - Constructor: recibe un parámetro por cada propiedad básica del tipo.
    - Representación como cadena: el nombre, código entre paréntesis y precio con descuento. Por ejemplo: "La España vacía (1298), 70.5".

    Se pide:

    a) Implemente la interfaz Regalo y la clase RegaloImpl.
    b) Implemente una clase TestRegalo que incluya un método main en el que se cree un regalo y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado.

17. Se desea implementar el tipo Vuelo, que se usará en el desarrollo de una aplicación de gestión de aeropuertos. Se usará la siguiente definición de sus propiedades y características:

    Vuelo:
    - Propiedades:
        - Código, de tipo String, consultable.
        - Origen, de tipo String, consultable.
        - Destino, de tipo String, consultable.
        - Fecha de salida, de tipo LocalDateTime, consultable y modificable.
        - Fecha de llegada, de tipo LocalDateTIme, consultable y modificable.
        - Duración, de tipo Duration, consultable. Se calcula a partir de la fecha de salida y de llegada.
        - Número de plazas, de tipo Integer, consultable.
        - Número de pasajeros, de tipo Integer, consultable y modificable.
        - Completo, de tipo Boolean, consultable. Indica si el vuelo está completo, es decir, no quedan plazas libres.
    - Constructores: uno que recibe un parámetro por cada propiedad básica del tipo; otro que recibe los mismos parámetros salvo el número de pasajeros, que se incializa a 0.
    - Representación como cadena: el código de vuelo entre paréntesis, seguido del origen y el destino separados por un guión y la fecha y hora de salida. Por ejemplo: "(IB2089) Sevilla – Madrid 25/12/2016 17:25"

    Se pide:

    a) Implemente la interfaz Vuelo y la clase VueloImpl.
    b) Implemente una clase TestVuelo que incluya un método main en el que se cree un vuelo y se muestre por la consola tanto la representación como cadena del mismo, como cada una de las propiedades por separado.