Ejercicio de Programación Orientada a Objetos
Curso 2016/2017
Cursos

Previo. Gestión del tiempo.

Para la realización del examen es necesario utilizar fechas. A continuación se dan algunas indicaciones:

- Para representar las fechas utilizaremos la clase **`java.util.Date`**. En Java se recomienda la **creación de fechas** mediante la clase `java.util.Calendar`. Por ejemplo:

```
Calendar calendario = Calendar.getInstance();
calendario.set(año, mes, dia);// enero = 0, febrero = 1, etc.
Date fecha = calendario.getTime();
```

- Para no equivocarte, todos los meses están definidos como constantes en la clase `Calendar`: `Calendar.JANUARY`, ...`Calendar.DECEMBER`.
- El constructor sin parámetros de la clase `Date` crea un objeto fecha con la fecha actual.
- Para comparar fechas puedes aplicar el método `compareTo` de la interfaz `Comparable`.

Por último, ten en cuenta que los objetos `Date` son *mutables*, es decir, tienen métodos que permiten cambiar su estado.

Queremos desarrollar una aplicación para la gestión de cursos organizados por una academia de formación.

La clase **Alumno** representa la persona que puede realizar un curso. Un alumno se caracteriza por tener las siguientes propiedades:

- *Nombre*: cadena de texto que identifica al alumno. Su valor no puede cambiar una vez inicializado en la construcción.
- *Dni*: es una cadena de texto que no puede cambiar una vez establecida en la construcción.
- *Crédito:* representa el dinero disponible para el pago de los cursos.
- *Cursos matriculados:* conjunto de cursos en los que el alumno se ha matriculado. El tipo de datos Curso se describe más adelante.

Para la construcción se requiere obligatoriamente el nombre y dni del alumno. El crédito disponible puede establecerse de manera opcional en el constructor. Por defecto, si no se establece, el valor inicial será de 100 euros.

La funcionalidad que ofrece la clase, además de la consulta de sus propiedades es:

- Incrementar el crédito en una cantidad establecida como parámetro.
- Decrementar el crédito en la una cantidad establecida como parámetro.
- Añadir curso. Métodos para añadir un curso al conjunto de cursos matriculados.

La clase **Curso** representa los cursos que pueden realizar los alumnos. La clase se caracteriza por las siguientes propiedades:

- *título* del curso.
- fecha de *inicio* del curso.
- fecha de *finalización* del curso.
- número de *días* de clase del curso. El número de días de clase no tiene por qué corresponderse con el número de días existentes entre la fecha de inicio y finalización.
- *precio* de matrícula. Número real que representa el dinero que hay que pagar para matricularse del curso.
- *alumnos matriculados*: conjunto de alumnos que se han matriculado del curso.
- *alumnos aptos*: conjunto de alumnos que han superado el curso con aprovechamiento.
- *número de alumnos* matriculados.

El título, las fechas de inicio y finalización, los días de clase y el precio se establecen en el constructor. Todas las propiedades se pueden consultar.

Para implementar parte de la funcionalidad de la clase Curso se tiene que hacer uso de los conceptos de **clase abstracta** y **método plantilla** dado que en la clase curso se implementan los algoritmos generales de comportamiento que dependen de funcionalidad propia de los tipos de cursos (los tipos de cursos se describen más adelante). La funcionalidad que ofrece la clase curso es la siguiente:

- Consultar si un curso *ha terminado*. Se considerará que el curso ha terminado si la fecha actual del sistema es superior a la fecha de finalización del curso.
- Consultar si un alumno ha sido calificado como apto en el curso, esto es, si se encuentra en el conjunto de alumnos aptos. La calificación de alumnos (apto / no apto) se explica a continuación.
- *Matriculación* de un alumno en un curso. Un alumno podrá matricularse en un curso si cumple los requisitos establecidos para ello. Por regla general, un alumno se podrá matricular en un curso si tiene crédito suficiente para afrontar el precio de la matrícula. En el caso de que el alumno cumpla los requisitos, dicho alumno se añadirá al conjunto de alumnos matriculados del curso. El crédito del alumno tendrá entonces que haberse actualizado y el curso quedará registrado en el conjunto de cursos en los que está matriculado el alumno. El método debe devolver un valor booleano para informar si el alumno ha podido matricularse o no.
- *Calificar.* Una vez concluido el curso, se valorará el grado de aprovechamiento de cada alumno para ver si se consideran aptos o no aptos. Por tanto, una vez finalizado el curso, para cada alumno matriculado en el mismo, si ha superado el curso con aprovechamiento, se registrará en el conjunto de alumnos aptos. El método devolverá un valor booleano que indique si se ha realizado la calificación de los alumnos o no. Las condiciones para considerar si un alumno es apto, o no apto, dependen del tipo de curso.

Existen dos tipos de cursos: online y presencial.

Un **curso online** es un tipo de curso que se caracteriza por establecer como requisito para la matriculación que el alumno haya realizado con aprovechamiento un conjunto de cursos con anterioridad y por exigir un *nivel* para ser calificado como apto. Las propiedades que caracterizan a un curso online son:

- nivel. Número entero que representa el nivel máximo que se puede alcanzar durante la realización del curso. Los alumnos tendrán que ir superando niveles de forma secuencial hasta llegar a este nivel máximo. Esta propiedad no puede cambiar.
- seguimiento de los alumnos: mapa que asocia cada alumno con un número que representa el nivel en el que se encuentran en el curso. Inicialmente todos los alumnos tienen nivel 0 cuando se matriculan.
- cursos previos. Conjunto de cursos que tendrá que haber superado el alumno para poder matricularse del curso online. Este conjunto no pueden cambiar.

El constructor recibirá como argumento, además de las propiedades propias de cualquier curso, el nivel que se puede alcanzar en el curso y un argumento de tamaño variable con los cursos previos que ha debido cursar el alumno.

La funcionalidad que ofrece la clase curso online es la siguiente:

- Consultar el nivel en el que se encuentra un alumno dado. Si el alumno no está matriculado en el curso devolverá -1. Recuerda que el método de consulta en un mapa devuelve `null` si la clave no existe en el mapa.
- Superar nivel. Este método recibe como parámetro el alumno al que se quiere registrar que ha superado el nivel y, si es un alumno matriculado en el curso, se incrementa en uno el nivel asociado en el mapa de seguimiento de los alumnos, siempre que no supere el nivel máximo. El método devolverá un valor booleano para indicar si se ha podido hacer el registro.

Además, un curso online amplía los requisitos de matriculación establecidos para los cursos, de manera que un alumno, además de tener crédito suficiente para la matrícula, tendrá que haber realizado con aprovechamiento (haber sido apto) en todos los cursos establecidos en la construcción (cursos previos).

Por último, un alumno se considera que *es apto*, esto es, que ha superado con aprovechamiento el curso online, si ha alcanzado un nivel medio, esto es, si su nivel registrado en el mapa de seguimiento es, al menos, la mitad del nivel máximo establecido para el curso.

Un **curso presencial** es un tipo de curso que, como su nombre indica, se imparte presencialmente. Las propiedades que caracterizan este tipo de cursos son:

- cupo: número máximo de alumnos que pueden matricularse.
- número mínimo de asistencias: representa el número mínimo de días que hay que asistir a clase para que se considere el curso aprovechado.
- asistencias: para cada día de clase se registran los alumnos que han asistido. Los días se representan como un entero, siendo 1 el primer día. Por tanto, debemos utilizar un mapa que asocie cada día del curso con el conjunto de alumnos que han asistido ese día.
- plazas libres: número de alumnos que aún pueden matricularse en el curso, esto es, el cupo menos el número de alumnos matriculados.

El cupo y el número mínimo de asistencias se establecen en la construcción y no pueden cambiar.

La clase añade funcionalidad para la gestión de las asistencias de los alumnos:

- *registro de asistencia*: se pasa como parámetro el número de día del curso y el alumno, y si son correctos, se añade el alumno al conjunto asociado a dicho día. Para que los valores sean correctos, el número de día debe estar dentro de los días de duración del curso y el alumno debe estar matriculado. El método informa con un booleano si la asistencia ha sido registrada o no.
- *consulta del número de asistencias* de un alumno: esto es, el número de días que ha asistido al curso.

En un curso presencial sólo podrá matricularse un alumno si, cumpliendo las condiciones generales de matriculación, quedan plazas libres, esto es, si no se ha alcanzado el cupo. Por tanto, se amplían los requisitos generales de matriculación.

Finalmente, se considera que un alumno es apto, esto es que ha superado con *aprovechamiento* el curso, si ha asistido, al menos, al número de días mínimo establecido.

3. Métodos Object

Implementa los métodos de la clase `Object` en las clases que se indican a continuación siguiendo las recomendaciones de la asignatura y según la semántica de los tipos de datos:

- Método `toString` en la clase `Alumno`. El método mostrará el nombre, dni, crédito y el número de cursos en los que está matriculado.
- Método `clone` en las clases que implementan los cursos. Ten en cuenta que:
  - Es necesario evitar los casos de aliasing que no sean correctos, en especial, los relativos a las colecciones.
  - Las colecciones de alumnos matriculados y aptos debe estar vacía. Igual que los mapas de seguimiento y control de asistencia de los cursos online y presencial.
  - Siempre interesa redefinir el método clone en los descendientes, aunque solo sea para aplicar la regla covariante.

4. Orden de los alumnos matriculados.

Declara un método en los cursos que retorne un listado ordenado de alumnos matriculados. El método debe estar parametrizado con el criterio de ordenación (*Comparator*). Implementa un criterio de ordenación que ordene los alumnos por dni.

5. Programa.

Implementa el siguiente programa para probar la funcionalidad:

- Declara una variable que referencie a un Alumno con DNI “34678904” y nombre “Pepe”.
- Declara una variable que referencie a un Alumno con DNI “17679456” y nombre “Andrea” con crédito inicial de 125€.
- Declara una variable que referencie a un curso presencial de título “Diseño de Bases de Datos” con fecha de inicio y fin 5/05/2014. El precio del curso es de 50€, la duración es de un día de clase, un cupo de 20 y el número mínimo de asistencias 1.
- Declara una variable que referencie a un curso online de título “Administración de Bases de Datos” con fecha de inicio 12/05/2014 y fin 16/05/2014. El precio del curso es de 25€, la duración es de 5 días, tiene un nivel 4 y como requisito que se haya realizado previamente el curso presencial de “Diseño de Bases de Datos”.
- Matricula a los dos alumnos en el curso presencial.
- Registra la asistencia del alumno “Pepe” en el día 1 del curso presencial.
- Califica a los alumnos del curso presencial y muestra en la consola los alumnos aptos. Sólo debe aparecer “Pepe”.
- Matricula a los dos alumnos en el curso online.
- Muestra la lista de los alumnos matriculados en el curso online. Sólo debe aparecer “Pepe”.
- Registra en el curso online que “Pepe” ha superado el nivel.
- Califica a los alumnos del curso online y muestra en la consola los alumnos aptos. La colección debe estar vacía porque “pepe” está en el nivel 1 y el mínimo es 2.
- Muestra los alumnos matriculados en el curso presencial ordenados utilizando el criterio de ordenación implementado.