Programación Orientada a Objetos

Curso 2023/2024

Convocatoria Junio

Alumno/a: _____GRUPO:_____

Preparación del examen

- Crea la carpeta examen" en el Escritorio.
- Arranca Eclipse y utiliza la carpeta "examen" como espacio de trabajo.
- Crea un proyecto Java que incluya tu grupo, apellidos y nombre, siguiendo la plantilla: **GXX*Apellidos-Nombre-Junio***. Por ejemplo "G11-MartinezLopez-Juan-Junio".
- Al acabar el examen, comprime la carpeta con el proyecto (no el espacio de trabajo). El nombre del fichero comprimido debe coincidir con el nombre del proyecto con extensión .rar, .zip o la correspondiente al compresor utilizado.
- Sube el fichero a la tarea del **Aula Virtual** "Entrega examen convocatoria junio".

Previo. Gestión de fechas.

- Para representar las fechas utilizaremos la clase **`java.time.LocalDate`**, que implementa una fecha sin información de la zona horaria de acuerdo con el sistema de calendario ISO-8601, en el formato (por defecto) de `año-mes-día`, por ejemplo `2024-06-16`.
- La clase `LocalDate` no dispone de constructores. Para crear objetos se utiliza el método de clase *`of`*, que recibe como parámetro el año, mes (como un entero o un valor del enumerado `Month`) y día. Por ejemplo:

```
LocalDate fecha = LocalDate.of(2024, 06, 16);
LocalDate fecha = LocalDate.of(2024, Month.JUNE, 16);
```

- Las fechas disponen de métodos para poder comparar: `isAfter`, `isBefore` e `isEqual`.
- La clase `LocalDate` ofrece métodos para acceder a sus propiedades como, por ejemplo, `getYear()` para retornar el año.

El objetivo del ejercicio es el desarrollo de un sistema de *catas de vinos*.

Bloque 0

Este apartado contiene ejercicios comunes a las dos partes del examen. Será valorado en el bloque 2.

a) Implementa la clase **Vino** de acuerdo con la siguiente especificación:

Un **vino** se valorará en la cata y se caracteriza por las siguientes propiedades:

- Fecha de producción (`LocalDate`).
- Nombre: Nombre comercial del vino.
- Color: el color del vino podrá ser tinto, blanco o rosado.
- Maduración: el vino se clasificará según su tiempo de maduración en joven (sin maduración), crianza (6 meses de maduración), reserva (12 meses de maduración) y gran reserva (36 meses de maduración).
- Coste: valor del vino por litro.

Se ofrece un constructor que recibe como argumentos un valor para todas las propiedades. También se ofrece un segundo constructor en el que se omite el coste, el color y el tipo de maduración y, en ese caso, se asume un coste es de 2€ y se clasifica como blanco y joven. Esta clase implementa *objetos inmutables*.

b) Crea una clase **Programa** con la siguiente funcionalidad:

- Crea un vino con fecha de producción 01/06/2020, nombre “Marqués”, de color tinto, maduración reserva, y con un coste de 15€.
- Crea un vino con fecha de producción 01/10/2015, nombre “Constantino”, de color tinto, maduración gran reserva, y con un coste de 30€.
- Crea un vino con fecha de producción 01/09/2022, nombre “El Purísimo”, de color rosado, maduración crianza, y con un coste de 5€.
- Crea un vino con el constructor por defecto del nombre “Don Pinpón” con fecha de producción 01/10/2023.
- Crea un vino con el constructor por defecto del nombre “Gil y Gil” con fecha de producción 01/11/2020.
- Crea una colección de vinos y añade los vinos anteriores.

Bloque 1 (4,5 ptos): Bloque compensable por parciales

NOTA: Este apartado no es obligatorio para el alumnado que haya aprobado los parciales. Sin embargo, e mantendrá la nota sacada por parciales si se implementa y se obtiene menos nota en este apartado.

1. (**0,35 ptos**) Extensión de la clase **Vino:**

a) Implementa los métodos `equals/hashCode` de modo que dos vinos son iguales si tienen la misma fecha de producción, nombre, color y maduración.

b) Implementa el orden natural de los vinos primero según el tipo de maduración y segundo según el orden natural de la fecha de producción.

2. (**0,25 ptos**) Implementa la clase **Valoración** de acuerdo con la siguiente especificación:

Los catadores valorarán los vinos según unas categorías concretas. Así, una valoración se caracteriza por las siguientes propiedades:

- Catador: cadena de texto que representa a la persona que realiza la valoración.
- Vino: vino del que se hace la valoración.
- Calificación olfativa: valor entero de la categoría olfativa.
- Calificación visual: valor entero de la categoría visual.
- Calificación gustativa: valor entero de la categoría gustativa.
- Nota: valor medio de las categorías que se valoran.

En la construcción de una valoración se establece el catador, el vino y el valor de las calificaciones olfativa, visual y gustativa. Esta clase implementa *objetos inmutables*.

3. (**1,5 ptos**) Implementa la clase **Cata** de acuerdo con la siguiente especificación:

Una **cata** permite gestionar las valoraciones de los catadores sobre una colección de vinos a evaluar. Por tanto, una cata se caracteriza por las siguientes *propiedades*:

- Nombre. Cadena de texto que identifica a la cata. Esta propiedad puede cambiar.
- Vinos. Colección de vinos a valorar.
- Valoraciones. Colección con las valoraciones que se han emitido de los vinos registrados.

En la construcción de una cata se establece solo el nombre. Las colecciones estarán inicialmente vacías.

La clase ofrece la siguiente *funcionalidad*:

- Registrar un vino. Recibe como parámetro un vino y si el vino no está registrado ya, lo añade a la colección de vinos de la cata. El método devuelve un valor booleano para indicar si el vino se ha podido registrar.
- Consultar la colección de catadores que han realizado alguna valoración.
- Consultar los vinos que ha valorado un catador. Devuelve la colección con los vinos que ha valorado el catador que se pasa como parámetro. Si no ha valorado ninguno devolverá una colección vacía.
- Consultar si un vino ha sido valorado por un catador. Recibe como parámetro el nombre del catador y el vino y devuelve un valor booleano verdadero si el vino ha sido valorado por el catador, o un valor falso si el vino no ha sido valorado o no es un vino registrado en la cata.
- Valorar un vino por un catador. El método recibe como parámetro el nombre del catador, el vino que se va a valorar y las calificaciones olfativa, visual y gustativa correspondientes. Si el catador es apto para la cata, el vino está registrado en ella, y el vino aún no lo ha valorado el catador, se crea el objeto valoración, se añade a la colección de valoraciones y se devuelve el valor verdadero para indicar que la valoración se ha registrado con éxito. En caso contrario, se devuelve falso. El criterio para decidir si un catador *es apto* depende del tipo de cata. Es un requisito para la implementación de este método que se utilice el concepto de **método plantilla**.
- Consultar las valoraciones organizadas por vinos. El método devuelve un mapa en el que las claves son los vinos y cada uno tiene asociada la colección de las valoraciones emitidas por los catadores.
- Método *sobrecargado* de consulta de valoraciones. El método devuelve una colección con las valoraciones asociadas al vino indicado como argumento. Si el vino está registrado, pero no ha sido valorado aún devolverá una colección vacía. En otro caso, devolverá una referencia nula.
- Consultar la puntuación de un vino pasado como argumento. La puntuación se calcula como la media de las notas establecidas por los catadores que lo han valorado. Si el vino no está registrado en la cata, el método devolverá el valor -1.
- Consultar el mejor o mejores vinos de la cata (en caso de empate), es decir, los vinos que tienen la mejor puntuación. El método devolverá una colección con los vinos con la mejor puntuación o una colección vacía si ningún vino tiene valoraciones aún en la cata.

4. (**0,75 ptos**) Implementa la clase **CataProfesional** de acuerdo con la siguiente especificación:

Una cata profesional es un tipo de cata que se caracteriza porque sólo se valoran vinos "reserva" o "gran reserva" y los catadores son sumilleres profesionales. Por tanto, este tipo de catas se caracteriza por mantener una colección con los nombres de los sumilleres que se inicializa en la construcción y no puede cambiar. El constructor recibe como parámetro el nombre de la cata y la colección de sumilleres como un argumento variable.

Así, un catador *es apto* para una cata profesional, si pertenece a la colección de sumilleres registrados. Además, solo se pueden registrar vinos de maduración “reserva" y “gran reserva".

Por último, este tipo de cata tiene una propiedad que identifica al *responsable*. Se considera responsable al primero de los sumilleres que se establecen en la construcción.

5. (**0,7 ptos**) Implementa un nuevo tipo de cata denominada **CataAmateur** que se caracteriza porque cualquier catador es apto y sólo se van a catar vinos jóvenes. A diferencia de las catas profesionales, en una cata amateur los catadores pueden rectificar sus valoraciones. Esto es, pueden reemplazar una valoración que hayan registrado por una nueva. En la rectificación se tiene que volver a establecer la puntuación de todas las categorías que se valoran. Si la nueva valoración no es válida se mantendrá la que ya estaba registrada.

6. (**0,65 ptos**) Redefinición de métodos de `Object`:

- Implementa `toString` en todas las clases siguiendo las recomendaciones de la asignatura.
- Implementa `clone` en la jerarquía de catas siguiendo las recomendaciones de la asignatura. Las copias de las catas mantendrán los vinos, pero no sus valoraciones. En las catas profesionales se mantendrán los sumilleres.

7. (**0,3 ptos**) Extiende el programa del bloque 0:

- Crea una cata profesional de nombre "Bodega Talavera" con los sumilleres "Andoni" y "Gemma".
- Crea una cata amateur de nombre "Taller iniciación".
- Crea una lista de catas y añade las dos catas anteriores.
- Recorre la colección vinos del bloque 0 y registra todos los vinos en todas las catas. Muestra los vinos registrados en cada cata. Sólo se deben haber registrado dos vinos en cada una de ellas.
- Recorre la colección de catas:
  - Muestra el nombre de la cata
  - Si es una cata profesional, muestra el nombre del responsable
  - Para cada vino de la cata
    - "Andoni" valora el vino con las calificaciones 8, 7 y 9 (olfativa, visual y gustativa)
    - "Custodio" valor el vino con las calificaciones 5, 8, 9
  - Si la cata es Amateur, "Custodio" cambia la valoración del vino "Gil y Gil" a 9, 9, 9.
  - Muestra la colección de los mejores vinos.

Bloque 2 (2 ptos)

8. (**0,4 ptos**) Bloque 0, ejercicio a)

9. (**0,1 ptos**) Bloque 0, ejercicio b)

10. (**0,25 ptos**) Implementa el control de precondiciones en la construcción de los vinos.

11. (**0,45 ptos**) En el programa del bloque 0, utilizando la lista con los vinos, implementa las siguientes consultas utilizando el procesamiento basado en *streams*:

- a) Consulta si hay algún vino rosado y muestra el resultado en la consola.
- b) Consulta cuántos vinos fueron producidos antes del 2021 y muestra el resultado en la consola.

Para los que han hecho el bloque 1:

- c) Construye una lista con los vinos que tengan un coste mayor de 5€, ordenados atendiendo al orden natural (ejercicio 1.b). Muestra la lista obtenida por la consola.

Para los que han hecho <u>sólo</u> el bloque 2:

- c) Construye una lista con los nombres de los vinos que tengan un coste mayor de 5€, ordenados alfabéticamente. Muestra la lista obtenida por la consola.

12. (**0,3 ptos**) Implementa un método de clase en una clase de utilidad (`Utils`) para registrar vinos por teclado. El método debe devolver una colección con los vinos que se han registrado o lanzar una *excepción comprobada* si se produce un error en el registro de un vino por teclado (los parámetros introducidos por el usuario no son correctos). El usuario podrá registrar tantos vinos como considere. Durante la ejecución:

- Se muestra un mensaje al usuario solicitando el nombre del vino.
- Se muestra un mensaje al usuario solicitando la fecha de producción del vino.
- Se crea el vino utilizando el constructor con los valores por defecto para el color, la maduración y el coste y se registra en la colección.
- Se muestra un mensaje al usuario indicando que el vino se ha registrado con éxito.
- Por último, se muestra un mensaje al usuario preguntando si se quiere registrar un nuevo vino.

Notas:

- La clase `java.util.Scanner` se puede utilizar para pedir datos por teclado creando un objeto de la siguiente forma: `new Scanner(System.in);`
- La clase `Scanner` dispone del método de instancia `nextLine()` que devuelve una cadena de texto con el contenido de la línea introducida por teclado.
- `LocalDate.parse(String)` convierte la cadena de texto que se pasa como parámetro (en el formato "2024-05-27") en un objeto de tipo `LocalDate`. Lanza la excepción *runtime* `DateTimeParseException` si la cadena no representa una fecha válida.

13. (**0,5 ptos**) Este ejercicio tiene dos pasos:

a) Implementa una clase `Utils` un método de clase **genérico** denominado *buscarReemplazar* que reciba como parámetro una colección, un objeto del mismo tipo que la colección y una función (`Function`) y se encargue de reemplazar las ocurrencias del valor pasado como parámetro en la colección, por el resultado de aplicar la función sobre dicho valor. Por ejemplo, dada la colección `numeros` = {4, 5, 7, 9, 4, 4} si aplicamos el método solicitando que se reemplace el número 4 por su cuadrado, la colección quedaría {16, 5, 7, 9, 16, 16}

b) Prueba el método en el programa principal utilizando el ejemplo anterior sobre la colección `numeros`.