PRÁCTICA

Enunciado general

Con el objeto de aportar nuestro granito de arena en la lucha contra la pandemia, la Agencia Europea del Medicamento (EMA) nos ha encargado el desrrollo de un software para la gestión de la vacunas contra el COVID-19.

Para ello, crearemos tres clases y una interface en un único paquete llamado gal.teis.vacunas:

- Clase abstracta **Vacunas**.
- Tres clases de las vacunas ya autorizadas enlace
- Una clase, por lo menos, de alguna vacuna no autorizada enlace
- Clase **OpVacunas**
- Clase **Aplicación** (método main())
- Interface **IAutorizable**

Interfaz IAutorizable

La interfaz **IAutorizable** se usará para autorizar o no los objetos de la clase **Vacuna**.

Tendrá dos métodos:

- **autorizar()** que hace que una vacuna esté autorizada para su uso.
- **rechazar()** que hace que una vacuna no esté autorizada para su uso.

Clase abstracta Vacuna

La clase **Vacuna** deberá almacenar: el **código**, el **nombre**, el **principio activo**, **farmacéutica**, el **precio recomendado**.

El comportamiento de la clase vendrá determinado por:

- Un constructor que inicializará los datos a los valores que se indiquen.
- Dos métodos públicos, uno para acceder al atributo del precio y otro para darle valor. Hacer lo mismo con el resto de los atributos, si procede.
- El **código** tendrá el siguiente formato:
  - Comenzará por la letra V seguida de una vocal en mayúsculas.
  - A continuación, tres o cuatro letras minúsculas.
  - Finaliza, o con dos números del 4 al 7, o bien con el número 8.
- Método **toString()** que mostrará los datos de una vacuan de la siguiente manera (siempre que esté autorizada por la EMA).

```
Datos de la vacuna:
   Código          VAedf45
   Nombre          COVID-19 vacuna AstraZeneca
   P. activo       Adenovirus de chimpacé
   Farmaceutica    AstraZeneca
   Precio          2.9 €
```

- Método **equals()** e **hashCode()** que determinarán que dos vacunas son iguales si sus códigos los son también.

Clase concreta de cada tipo de vacuna

Estas clases que pueden tener nombres como **VacunaAstraZeneca**, **VacunaPfizer**, etc. que heredará de la clase abstracta **Vacuna** e implementará la interfaz **IAutorizable**.

Cada una de estas clases deberá tener los siguientes atributos **boolean** indicando si ha superado o no cada una de las fases necesarias para su aprobación.

- **fase1Superada**
- **fase2Superada**
- **fase3Superada**

Para sabes si la información de las distintas fases ya es la definitiva se deberá utilizar el atributo *byte* siguiente, que tendrá un valor correspondiente a las fases de las que se ha almacenado su resultado. Cuando el resultado de las tres fases hayan sido introducidas, este atributo tendrá el valor **3**.

- **fasesCompletadas**

Los métodos para modificar el valor de **fase1Superada**, **fase2Superada** y **fase3Superada** son los que modificarán el valor del atributo **fasesCompletadas**. De esta forma se interpretará correctamente el valor *false* por defecto que tienen los atributos *boolean*, que no significa que la fase no ha sido superada, sino que aun no se ha introducido el resultado de la fase.

La implementación de los métodos de la interfaz **IAutorizable** deberán analizar los atributos **fase…** y devolver *true* o *false* en función del método a implementar (autorizada(), rechazada()). Para que una vacuna sea autorizada deberá verificar que todas las fases de investigación se han completado con éxito, analizando el valor de **fasesCompletadas**, **fase1Superada**, **fase2Superada** y **fase3Superada**, en cualquier otro caso, las vacunas no serán autorizadas.

Clase OpVacunas

La clase **OperacionesVacunas** deberá almacenar en un *ArrayList* las vacunas identificado por el nombre **almacen** y deberá tener un atributo para saber el número de vacunas identificado por el nombre **total**.

Las vacunas que se guardan en **almacen** serán aquellas que estén pendientes de aprobar por la EMA, que las aprobará si tienen todas las fases de investigación finalizadas con éxito.

El comportamiento de la clase será el siguiente:

- Método que permitirá agregar una vacuna pendiente de análisis al **almacen** e incrementar el valor de **total**. Este método devolverá el resultado de la operación (si fue posible la operación o no). Deberá comprobar que la vacuna no está ya almacenada en base a su **código** (utilizando el método **equals()**)
- Método que permita eliminar una vacuna del listado en base a su **código** utilizando el método **equals()**. Este método devolverá el resultado de la operación (si fue posible la operación o no).
- Método que permitirá introducir el resultado de las tres fases de prueba necesarias para autorizar una vacuna.

- Método que muestra el listado completo de vacunas con la información de los atributos comunes a todas las vacunas.
- Método que muestre el listado de todas las vacunas con su nombre, código y si están aprobadas o no.
- Método que muestre el listado de las vacunas autorizadas (nombre y código).
- Método que muestre el listado de las vacunas no autorizadas (nombre y código).
- Método que muestre el listado de las vacunas pendientes de autorización (nombre y código).
- Método que busque una vacuna en función de su código y nos informe de si está autorizada, rechazada o pendiente de autorización.

Clase Aplicación

Realizará las operaciones que se corresponden con el menú siguiente:

```
1.Buscar vacuna.
2.Agregar vacuna.
3.Eliminar vacuna.
4.Introducir resultado de las fases de una vacuna.
5.Ver vacunas autorizadas.
6.Ver vacunas no autorizadas.
7.Ver vacunas pendientes de autorizar/rechazar.
```