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:

- Interface **IAutorizable**
- Clase abstracta **VacunaAutorizacion**.
- Clase **Vacuna**
- Clase **VacAlmacen**
- Clase **Aplicación** (método main())

Interfaz IAutorizable

La interfaz **IAutorizable** se usará para autorizar o no los objetos creados a partir de **Vacuna** y tendrá dos métodos:

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

El valor devuelto de los dos métodos indica si la operación se ha realizado o no.

Clase abstracta VacunaAutorizacion

Esta clase debe implementar la interface **IAutorizable**.

Tendrá los siguientes atributos de tipo **boolean** que indican si ha superado o no cada una de las fases necesarias para su aprobación.

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

Para saber si la información de las distintas fases ya es la definitiva se deberá utilizar el atributo *byte* **fasesCompletadas**, que tendrá el 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**.

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.

La decisión de rechazar una vacuna es una decisión de la EMA, más allá de que una vacuna haya superado todas las fases (que podría ser).

La clase también tendrá los siguientes atriburos:

- *LocalDate* **fechaResultado** que recogerá la fecha en que la vacuna ha sido rechazada o autorizada.
- *boolean* **autorizada** y *boolean* **rechazada** que contendrá el resultado de la operación **autorizar()** y **rechazar()** respectivamente. Es importante que existan los dos atributos ya que si los dos valores son *false*, significa que aún no se an completado las fases de investigación.

Además de los métodos de la interfaz y los de modificación de **faseiSuperada** debemos tener métodos que:

- Devuelva el valor de **fechaResultado.**
- Devuelva el resultado de la investigación de la **útima** fase que se ha implementado con la vacuna.

Cuestiones a tener en cuenta:

- Deben existir métodos para ver si una vacuna ha sido autorizada o no, pero la autorización solo se debe producir con los métodos implementados de **IAutorizable** .
- El valor de **fechaResultado** solo se podrá modificar desde el método **autorizar()** y **rechazar()** y se corresponderá con la fecha actual del sistema en el momento de llevar a cabo la operación.
- El valor de **autorizada** solo podrá ser modificado desde **autorizar().**
- El valor de **rechazada** solo podrá ser modificado desde **rechazar ().**

Si una vacuna ha sido rechazada no puede ser posteriormente autorizada, de la misma forma, si ha sido rechazada no podrá ser autorizada.

Clase 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 vacuna 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 €
```

  En el caso de que la vacuna no esté autorizada por la EMA mostrará

```
Datos de la vacuna:
  Código          ZSgtf23
  Nombre          COVID-19 vacuna Sputnik V
  P. activo       Adenovirus Ad26 y Ad5
```

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

Clase VacAlmacen

La clase **VacAlmacen** deberá gestionar una colección de vacunas y tener los métodos necesarios para ejecutar las distintas operaciones del menú que se mustra en la clase **Aplicación**.

Clase Aplicación

Realizará las operaciones que se corresponden con el menú siguiente:

```
1.Listar todas las vacunas y mostrar todos sus datos
2.Buscar vacuna.
3.Agregar vacuna.
4.Eliminar vacuna.
5.Introducir resultado de las fases de la vacuna.
6.Autorizar/Rechazar vacuna.
7.Ver vacunas autorizadas.
8.Ver vacunas rechazadas.
9.Ver vacunas pendientes de autorizar/rechazar.
10.Ver la última fase investigada de cada vacuna almacenada.
```

El punto 5 debe dar la posibilidad de introducir las distintas fases de investigación. Esta operación se puede gestionar de distintas formas.

EXAMEN

1. **Añadir una nueva *interfaz*** llamada **`IGestionVacunasPais`** que tendrá la siguiente implementación:

```
public interface IGestionVacunasPais {
      public void asignar(long numAsignadas);
}
```

   El método **`void asignar(long numAsignadas)`** debe permitir la asignación de vacunas.

2. **Añadir una nueva *clase*** al proyecto llamada **`AsignaVacuna`** que implemente la interfaz **`IGestionVacunasPais`**. Esta clase se utilizará para almacenar información de **país/vacuna**.

Atributos

`Vacuna laVacuna`	Objeto Vacuna
`String pais`	Nombre del país al que se le asigna la vacuna
`long asignadas`	Número de vacunas de un tipo asignadas

Estos atributos solo tendrán métodos **`get`** para mostar su contenido no se implementarán métodos **`set`**

3. En la clase **`AsignaVacuna`** se debe implementar solo un constructor con la siguiente implementación:

```
public AsignaVacuna(Vacuna laVacuna, String pais) {
        this.laVacuna = laVacuna;
        this.pais = pais;
}
```

4. En la clase **`AsignaVacuna`** debemos implementar el método **`equals()`** para determinar que **dos objetos son iguales si el código de la vacuna y el nombre del país coinciden**.
5. En la clase **`AsignaVacuna`** debemos implementar el método **`hashCode()`** que calcule el código *hash* del objeto en función **del código de la vacuna y el nombre del país**.
6. En la clase **`AsignaVacuna`** debemos implementar los métodos de la interfaz siguiendo las directrices siguientes:

Método	Características de la implementación
`void asignar(long numAsignadas)`	• Da valor al atributo **`asignadas`**. • **@param `numAsignadas`** número de vacunas que se quieren asignar a un país.

7. **Crear 2 nuevas entradas en el menú** que tendrán como fuente de datos la lista donde están las vacunas. A esa lista la llamo en este enunciado **`listaVacunas`**, aunque en cada programa se puede usar el nombre que ya tiene. Tendremos que comprobar que la lista no está vacía antes de hacer cualquier operación sobre ella. Esta lista puede tener todas las vacunas o solo las vacunas autorizadas, las dos opciones son válidas. Se debe agregar un comentario explicando la opción elegida.
   Solo se podrán asignar vacunas autorizadas a los distintos países.

   Los objetos **`AsignaVacuna`** que se creen al asignar **vacuna/país** deben almacenarse en una lista que se llamará **`listaVacunasAsignadas`**. La lista no puede ser de tipo *ArrayList* ni *Map*, pero puede ser de cualquier otro tipo de los estudiados.

Entrada de menú	**Características de la implementación**

`11. Asignar vacunas a país`	• Pedir el **código** de la vacuna, el **nombre** del país y el **número** de vacunas a asignar.
	• Si es necesario, confirmar que la vacuna está autorizada.
	• Se tiene que buscar en la lista **`listaVacunasAsignadas`** si ya se ha asignado la vacuna al país.
	  ○ En el caso de que ya exista esa asignación (*sí existe el objeto en* ***`listaVacunasAsignadas`***), se debe mostrar el valor que tiene asignado actualmente y pedir confirmación para cambiar el valor. Al final, mostar un mensaje resumen de la operación realizada.
	  ○ Si no se ha asignado ese tipo de vacuna al país *(no existe el objeto en* ***`listaVacunasAsignadas`***), se ha de:
	  a) Crear un objeto del tipo **`AsignaVacuna`**.
	  b) Agregar el objeto a **`listaVacunasAsignadas`**.
	  c) Asignar el número de vacunas con el método ***`asignar()`*** del objeto recién creado.
	  d) Mostrar un mensaje resumen de la operación realizada.
	• Controlar que no se asigne el mismo tipo de vacuna al mismo país. Si esto se produce, sería un error muy grave.

`12. Ver las vacunas asignadas a un país`	• Debe **preguntar** por el país y **mostrar** el número y el tipo de vacunas asignadas.