solucion-prLibreria.pdf

miau_33

Programación Orientada a Objetos

1º Grado en Ingeniería Informática

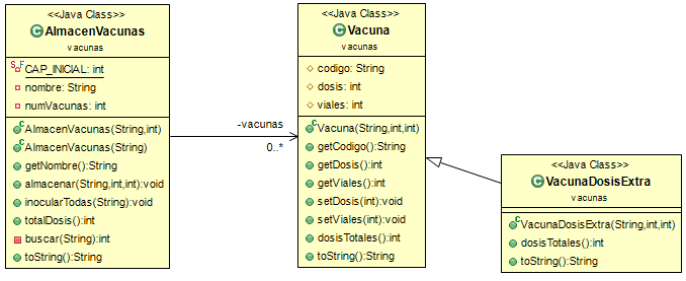
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- Al **inicio del contenido de cada fichero** realizado deberá aparecer un comentario con tus **apellidos y nombre, titulación y grupo.**
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, *no debes pararte en él indefinidamente*. Puedes abordar otros.
- **Está permitido:**
  - Consultar los apuntes (CV), la API (Internet), la guía rápida de la API (CV).
  - Añadir métodos privados a las clases.
- **No está permitido**:
  - Intercambiar documentación con otros compañeros.
  - Recibir ayuda de otras personas. Se debe realizar personal e individualmente la solución del ejercicio propuesto.
  - Añadir métodos no privados a las clases.
  - Añadir variables o constantes a las clases.
  - Modificar la visibilidad de las variables, constantes y métodos que aparecen en el diagrama UML.
  - Modificar el código suministrado.
- Una vez terminado el ejercicio, debéis subir (a la tarea creada en el campus virtual para ello) un fichero comprimido de la carpeta **src** que hayáis realizado y usáis vuestros apellidos y nombre para su denominación **(Apellido1Apellido2Nombre.rar** o **.zip)**.
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización.
- Para la corrección del ejercicio se utilizarán **programas de detección de copias/plagios**.
- Con posterioridad a la realización del ejercicio, el profesor podrá convocar a determinado/as alumno/as para realizar **entrevistas personales síncronas** con objeto de comprobar la autoría de las soluciones entregadas.

Proyecto prVacunas

Se desea crear una aplicación para gestionar el almacenamiento de determinados tipos de vacunas. Se han de crear, dentro de un paquete denominado `vacunas`, las clases que aparecen en el diagrama UML: `Vacuna`, `AlmacenVacunas`, `VacunaDosisExtra`.



También se debe crear, en el paquete "por defecto" del proyecto, una clase distinguida de prueba: `PruebaVacuna`. Por último, se suministran en el campus virtual la clase distinguida `PruebasAdicionales` para realizar más pruebas, que debe incluirse también en el paquete "por defecto" del proyecto. A continuación, se describen cada una de las clases a definir:

A. (***1.5 ptos.***) La clase `Vacuna` almacena el código de una vacuna (`String`), el número de viales de los que se dispone (`int`) y el número de dosis por vial (`int`). La clase deberá proporcionar:
   1. Un constructor en el que se proporciona el código, número de viales y dosis por vial. Si los viales o las dosis por vial son negativos o cero, el constructor lanzará una excepción del tipo `RuntimeException` con el mensaje correspondiente.
   2. Tres métodos necesarios para devolver el código, número de viales y dosis por vial.
   3. Dos métodos para modificar el número de viales y las dosis por vial, dados nuevos valores como argumentos. En caso de que los nuevos valores no sean válidos (negativos o cero) se lanzará una excepción del tipo `RuntimeException` con el mensaje correspondiente.
   4. El método `int dosisTotales()` devuelve el total de dosis de la vacuna, teniendo en cuenta el número de dosis por vial y el número de viales.
   5. La representación como cadena de caracteres de un objeto de la clase `Vacuna` debe contener el código de la vacuna junto con el número de viales y las dosis por vial, con el formato:
      `(<CÓDIGO>: <viales> x <dosis> dosis)`
      Por ejemplo, `(PFIZER-001: 200 x 5 dosis)`
      Obsérvese que el código va en mayúsculas.

B. (***0.5 ptos.***) Crea una aplicación (clase distinguida `PruebaVacuna`) para probar la clase anterior. En esta aplicación se crean tres objetos de tipo `Vacuna`. El primero con código "Moderna-001", con 100 viales y 5 dosis por vial. El segundo con código "AstraZeneca-001", con 200 viales y 8 dosis por vial. Y un tercer objeto con código "astrazeneca-001" (todo en minúscula), con 160 viales y un número de dosis por vial de 10. A continuación, **para cada una** de las tres vacunas se ha de mostrar por pantalla la siguiente información:
   El número total de dosis de la vacuna <VACUNA> es <TOTAL DOSIS>
donde <VACUNA> es la representación textual del apartado A.5 y <TOTAL DOSIS> es el resultado de aplicar a cada vacuna el método `dosisTotales()`. Por último, debe comprobase si los dos últimos objetos creados son iguales o no, entendiendo que dos vacunas son iguales si tienen el mismo código, independientemente de mayúsculas o minúsculas, y el número de dosis totales es la misma para ambas. Es decir, el mensaje que ha de mostrarse es:
   Las vacunas (ASTRAZENECA-001: 200 x 8 dosis) y (ASTRAZENECA-001: 160 x 10 dosis) son iguales
Mediante una expresión similar a la utilizada para obtener el mensaje anterior, debería obtenerse el mensaje siguiente, comprobando que las dos primeras vacunas creadas no son iguales:
   Las vacunas (MODERNA-001: 100 x 5 dosis) y (ASTRAZENECA-001: 200 x 8 dosis) no son iguales

C. (***6 ptos.***) La clase `AlmacenVacunas` almacena en un array una colección de vacunas. Se utilizará una variable de instancia para indicar el número de vacunas almacenadas en el array en un instante dado. La clase también almacena el `nombre` del almacén. La clase deberá proporcionar:
   1. Dos constructores: al primero de ellos se le proporciona el nombre del almacén y un entero que indica el tamaño del array de vacunas, mientras que al segundo sólo se le proporciona el nombre del comprador y crea el array con un tamaño 3 (constante `CAP_INICIAL`). El primer constructor lanzará una excepción del tipo `RuntimeException` si el tamaño especificado fuera menor o igual que cero.

2. El método `void almacenar(String codigo, int viales, int dosis)`, que recibe como parámetros el código, número de viales y las dosis por vial para almacenar una determinada vacuna. Con estos datos se añadirá una vacuna al almacén. Este método ha de tener en cuenta que dicha vacuna puede ser que se incluya por primera vez en el array o que ya esté incluida. Si la vacuna ya estuviera (el código coincide, independientemente de mayúsculas o minúsculas, con alguna almacenada en la lista), se ignorará la dosis por vial y únicamente se habrá de incrementar el número de viales de la vacuna existente con el que se pasa como parámetro (no se ha de introducir de nuevo la vacuna en la lista).
   Si el array se llena, se duplicará su tamaño.
3. El método `String getNombre()` devuelve el nombre del comprador.
4. El método `double totalDosis()` devuelve el número de dosis total que suponen todas las vacunas almacenadas en el array.
5. El método `void inocularTodas(String codigo)` simula la inoculación de todas las dosis de una vacuna, con el código indicado en el argumento. Esto supondrá eliminar de array de vacunas aquella cuyo código se recibe como parámetro. Si dicha vacuna no está, no se hace nada.
6. La representación como cadena de caracteres de un objeto de la clase `AlmacenVacunas` tendrá el formato `nombreVacuna = [vac1,vac2,…,vacn]` donde cada vacuna (`vaci`) se representará con el formato especificado en A.5.

D. (***2 ptos.***) La clase `VacunaDosisExtra` tiene la misma funcionalidad que la clase `Vacuna`, pero a la hora de calcular el total de dosis se tiene en cuenta que es posible obtener una dosis extra de cada vial. La clase deberá proporcionar:
   1. Un constructor con tres parámetros: el código de la vacuna, un entero que indica el número de viales de la vacuna y otro entero que indica las dosis por vial (sin la extra). Se lanzará una excepción `RuntimeException` si el número de viales o dosis por vial fuera menor o igual que cero.
   2. El método `int totalDosis()` redefine el método equivalente de `Vacuna` para devolver el número de dosis, contemplando la posibilidad de extraer una dosis adicional de cada vial.
   3. El método `String toString()` debe devolver la misma cadena de caracteres que en un objeto de la clase Vacuna, añadiendo al final:
      `… + <nro. dosis extra> extra`
      Por ejemplo, `(PFIZER-001: 200 x 5 dosis) + 200 extra`

En el campus virtual se facilita la clase `PruebasAdicionales` para probar las clases desarrolladas. La salida por consola de la ejecución de dicha clase sería la siguiente:

```
(PFIZER-001: 150 x 5 dosis)
(PFIZER-001: 150 x 5 dosis)+150 extras
Los tres almacenes de vacunas son:
Carlos Haya = [(MODERNA-001: 300 x 5 dosis), (ASTRAZENECA-001: 360 x 8 dosis), (J&J-001: 200 x 10 dosis), (PFIZER-001: 150 x 6 dosis)]
Hospital Materno = [(J&J-001: 400 x 10 dosis), (MODERNA-001: 100 x 10 dosis)]
Hospital Clínico = [(PFIZER-001: 200 x 6 dosis)]
El número total de dosis en el Hospital Carlos Haya es: 7280
ERROR. Se ha intentado almacenar una vacuna con un número de viales o dosis negativo
```