control1SaludAbril21.pdf

Cerebrandex

Programación Orientada a Objetos

1º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

DEPARTAMENTO LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN	**PROGRAMACIÓN ORIENTADA A OBJETOS** (Prueba realizada el 12 de abril de 2021)	
APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA	

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- Al inicio del contenido de cada fichero deberá indicarse **el nombre del alumno, titulación, grupo y código del equipo** que está utilizando**.**
- 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.
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, el alumno *no debe pararse en él indefinidamente*. Puede abordar otros.
- **Está permitido:**
  - Consultar la API y los apuntes.
- **No está permitido**:
  - Utilizar otra documentación electrónica o impresa.
  - Intercambiar información con otros compañeros.
  - Utilizar soportes de almacenamiento o teléfonos móviles.
- Una vez terminado el ejercicio subir, a la tarea creada en el campus virtual para ello, un solo archivo con los ficheros **fuente** (.java) que hayáis realizado.

Se desea desarrollar una aplicación para gestionar las personas vacunadas contra una determinada enfermedad altamente contagiosa. Para ello, se creará un proyecto `prVacunados` con las clases `Persona`, `PersonaDosis`, `SistemaVacunacion` y `SistemaVacunacionDosis` en el paquete `vacunas`, y la clase distinguida `PruebaVacunacion`, que se proporciona con el enunciado, en el paquete por defecto.

1) **(1.25 ptos.)** La clase `Persona` mantendrá información sobre las personas que han de vacunarse. En concreto, su nombre (`String`), su edad (`int`) y una variable llamada `inmunizado` que indica si ya está vacunado o no (`boolean`). También dispondrá de:
   a. *(0.5 ptos.)* Un constructor dados valores para el nombre y la edad. Si la edad es un número negativo se lanzará una excepción de tipo `RuntimeException`. Inicialmente, una persona se considera no vacunada.
   b. *(0.25 ptos.)* Métodos `String getNombre()`, `int getEdad()` y `boolean inmunizado()`, que permiten obtener los valores almacenados en los atributos nombre, edad e inmunizado, respectivamente.
   c. *(0.25 ptos.)* Un método `void vacunar()`, que permite indicar que a la persona ya se le ha administrado la vacuna y está inmunizada.
   d. *(0.25 ptos.)* La representación textual de una persona (`String toString()`) está formada por el nombre, la edad y la cadena INMUNIZADO en caso de estar inmunizado, separados por coma. Si no estuviera inmunizado se debe incluir la cadena NO INMUNIZADO. Toda la información está encerrada entre paréntesis. Por ejemplo:

```
(Benito Pérez, 70, NO INMUNIZADO)
```

2) **(3.75 ptos.)** La clase `SistemaVacunacion` almacenará la información de la colección de personas que están en espera o ya han sido vacunadas en el array denominado `personas`. La variable de instancia `numPersonas` (`int`) sirve para conocer el número de personas que hay almacenadas en el array y para gestionar el array, ya que los elementos se mantienen al principio del mismo (`0 ... numPersonas-1`). Además,

existe una variable de instancia `vacunas (int)` que indica cuántas vacunas hay disponibles. La clase dispondrá de:

a. *(0.5 ptos.)* Un constructor sin argumentos que utiliza el valor almacenado en la **constante** de clase `CAP_INICIAL` (`int`) como tamaño inicial del array. El valor de la constante `CAP_INICIAL` será 5. Inicialmente el array se encuentra vacío y no se dispone de vacunas.

b. *(0.5 ptos.)* Un método para buscar una persona en la colección dado su nombre y su edad (`int buscarPersona(String, int)`). Debe devolver la posición en la que se encuentra el objeto `Persona` correspondiente o -1 en caso de que no se encuentre en el array. Para localizar a una persona deben coincidir tanto la edad como el nombre, ignorando mayúsculas y minúsculas. Este es un método **privado**.

c. *(0.75 ptos.)* Un método para añadir un objeto de tipo `Persona` en el array de personas (`void anyadirPersona(Persona)`) . Si la persona que se desea añadir ya está en la colección, no se hará nada. En caso de tener que añadir la persona y el array esté lleno, se duplicará su capacidad automáticamente. Puedes usar el método `buscarPersona` anterior para la implementación de este método. Este es un método **protegido**.

d. *(0.25 ptos.)* Un método para añadir una persona sin vacunar a la lista de personas, dados su nombre y su edad (`void addPersona(String, int`). Este método habrá de crear un nuevo objeto de la clase `Persona` con dichos argumentos e invocará al método `anyadirPersona(Persona)` para añadirlo a la colección.

e. *(0.25 ptos.)* Un método para comprar una cantidad de vacunas dada (`void comprarVacunas(int)`). Se debe incrementar el número de vacunas en esa cantidad. En caso de que el argumento sea negativo o cero no se hace nada.

f. *(1 pto.)* Un método para vacunar a las personas comprendidas en un rango de edad (`void vacunar(int edadMinima, int edadMaxima))`. Toda persona de la colección que no esté inmunizada y cuya edad sea mayor o igual a `edadMinima` y menor o igual que `edadMaxima` será vacunada. Es posible que haya menos vacunas que las necesarias, por lo que el proceso de vacunación parará cuando se hayan agotado las vacunas.

g. *(0.5 ptos.)* Un método para proporcionar la representación de los objetos de la clase (`String toString()`). Está compuesta por la subcadena "`Vacunas : `" y el número de vacunas que quedan. Después, se añade un espacio ("` `") y la cadena devuelta por la llamada `Arrays.toString(personas)`. Ejemplo:

```
Vacunas: 16 [(Alberto Gómez, 64, INMUNIZADO), (María Abad, 50, NO INMUNIZADO),
(Sara Campos, 80, INMUNIZADO), null, null]
```

3) **(2.5 ptos.)** La clase `PersonaDosis` se comporta como la clase `Persona`, con la diferencia de que se guarda una **constante** de instancia con las dosis de vacuna que han de administrarse a una persona para llegar a su inmunización (`int dosisNecesarias`), así como una variable con el número de dosis que se han administrado a la persona (`int dosisAdministradas`). Se debe implementar:

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

1/6
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

Si ya tuviste sufi con tanto estudio...
Te dejamos este espacio
para desahogarte.

Pinta, arranca,
llora... tú decides ;)

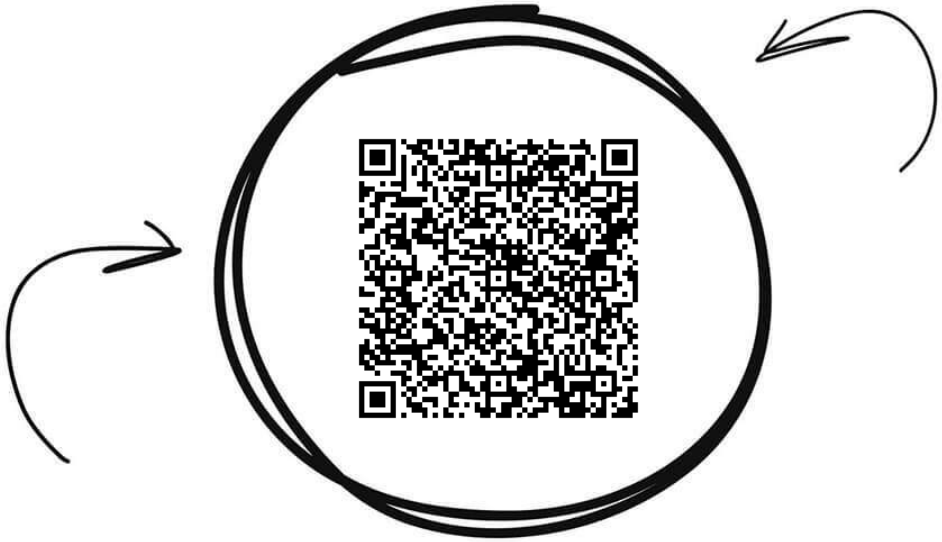
¿Te sientes más liberado?
Sigue siéndolo con la **Cuenta NoCuenta:**
libre de comisiones*, y de lloraditas.

¡Quiero una de esas!

*TIN 0 % y TAE 0 %.

do your thing

Programación Orientada a Obj...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

a. *(1.5 pto.)* Un constructor dado su nombre, edad y el número de dosis necesarias. En caso de que el número de dosis necesarias sea menor o igual a 0 se lanzará una excepción de tipo `RuntimeException`. Inicialmente, la persona no habrá recibido ninguna dosis de la vacuna.
b. *(1 pto.)* Una redefinición del método `void vacunar()` para que incremente en una unidad el número de dosis administradas. Si la persona ya ha recibido todas las dosis necesarias, estará inmunizada y habrá que actualizar el atributo heredado `inmunizado` para que refleje tal situación.

4) **(2.5 pto.)** La clase `SistemaVacunacionDosis` se comporta como la clase `SistemaVacunacion`, con la diferencia de que las dosis que hay que administrar de una vacuna para inmunizarse depende de la enfermedad. La variable `dosisVacunas` (`int`) indica el número de dosis que hay que administrar para que una persona pueda considerarse inmunizada. Son necesarios:
   a. *(1.5 pto.)* Un constructor que recibe las dosis necesarias para que una persona se inmunice. En caso de que el número de dosis sea negativo o cero se lanzará una excepción de tipo `RuntimeException`.
   b. *(1 pto.)* Una redefinición del método `void addPersona(String, int)`, que añade la persona con nombre y edad dadas a la lista de personas. En esta ocasión se creará un objeto de clase `PersonaDosis` con el nombre y edad dados y donde el número de dosis necesarias coincide con el fijado en `dosisVacuna`.

Como parte del enunciado del examen se puede encontrar la clase `PruebaVacunacion.java` que se puede utilizar en el paquete por defecto del proyecto para comprobar el correcto funcionamiento de las clases requeridas. La salida esperada como salida de dicho programa de prueba está incluida como un comentario al final del fichero `PruebaVacunacion.java`.