UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional
EI1008/MT1008 Programación II
Examen final. Segunda Convocatoria (2 de julio de 2018)

Pregunta 1 (5 puntos)

Nos han pedido desarrollar una aplicación para gestionar las calificaciones finales de los estudiantes de secundaria. Un centro educativo usa varios ficheros de texto para almacenar los resultados de sus diferentes cursos. Cada línea de uno de estos ficheros contiene el DNI de un estudiante, el nombre de una asignatura y la calificación final obtenida por el estudiante en esa asignatura. Las líneas del fichero no mantienen ningún orden concreto. A continuación se muestran algunas líneas de ejemplo de uno de estos ficheros:

```
18423999X Matemáticas 7,3
07666321J Música 5,0
18423999X Lengua_y_Literatura 4,2
33619732M Educación_Física 9,0
```

Ya tenemos implementada una clase, `Estudiante`, que proporciona, entre otros, los siguientes constructores y métodos públicos:

- `Estudiante(String dni)`: constructor de la clase. Almacena el DNI del estudiante y establece a cero los contadores internos de asignaturas suspensas.
- `String getDni()`: devuelve el DNI del estudiante.
- `void contarSuspensa(boolean esBásica)`: incrementa en uno el contador de asignaturas suspensas del estudiante; si la asignatura es básica, también incrementa el correspondiente contador.
- `int getTotalSuspensas()`: devuelve el número total de asignaturas que ha suspendido el estudiante.
- `int getBásicasSuspensas()`: devuelve el número de asignaturas básicas que ha suspendido el estudiante.

Necesitamos definir una nueva clase, `CalificacionesFinales`. Considera que la definición inicial de la clase es:

```
public class CalificacionesFinales {

    private Estudiante[] estudiantes;  // Datos de los estudiantes (sin ningún tipo de orden).

    public CalificacionesFinales() {
        this.estudiantes = new Estudiante[0];
    }

    public static boolean esBásica(String asignatura) {
       ...  // Devuelve true o false según la asignatura dada sea básica o no.
    }
}
```

Añade a la clase `CalificacionesFinales` los siguientes constructores y métodos públicos:

a) `CalificacionesFinales(String nombreFichero)` **(1,5 puntos)**

Este constructor recibe el nombre de un fichero de resultados como los descritos anteriormente e inicializa el vector de estudiantes de manera adecuada, esto es, el vector debe contener un objeto por cada estudiante que aparezca en el fichero (incluso si ha aprobado todas las asignaturas) y se deben contabilizar las asignaturas suspensas como corresponda, teniendo en cuentas cuáles son básicas y cuáles no. En el vector no puede haber entradas a `null`, por lo que su talla debe ser igual al número de estudiantes.

b) `String[] repitenCurso()` **(1,5 puntos)**

Devuelve un vector con los DNI de aquellos estudiantes que deberían repetir curso. Un estudiante repite si el total de asignaturas que suspende es mayor o igual que tres o si suspende dos asignaturas básicas. La talla del vector devuelto debe ser igual al número de estudiantes que repiten.

c) `int procedenCeip(String[] vectorDni)` **(2 puntos)**

El método tiene como parámetro un vector de DNI, *ordenado lexicográficamente de menor a mayor*, con todos los estudiantes que cursaron primaria en un determinado colegio. Como resultado, debe devolver la cantidad de estudiantes del vector `estudiantes` que aparecen en el vector de DNI.

Dado que el vector de DNI está ordenado lexicográficamente y el vector de estudiantes no, se exige que el coste temporal de este método sea $\mathcal{O}(n \log m)$, donde n es el número de elementos del vector de estudiantes y m es el número de elementos del vector de DNI.

Pregunta 2 (5 puntos)

El software de gestión de una tienda online utiliza la clase `Inventario` para guardar la información de los productos que tiene en almacén. La clase implementa una lista simplemente enlazada en la que cada nodo guarda la información de un producto: su código y sus existencias (cantidad de unidades de ese producto). De la lista enlazada, se guarda su *primer* y su *último* nodo. La definición inicial de `Inventario` es:

```
public class Inventario {
    private static class Nodo {
        String código;
        int existencias;
        Nodo sig;

        Nodo(String código, int existencias, Nodo sig) {
            this.código = código;
            this.existencias = existencias;
            this.sig = sig;
        }
    }

    private Nodo primero;
    private Nodo último;
}
```

Añade a la clase `Inventario` los siguientes métodos públicos:

a) `void darDeAlta(String código, int cantidad)` **(1,75 puntos)**

Cuando en la lista haya algún nodo cuyo código coincida con el `código` dado, el método debe actualizar las existencias de ese producto incrementándolas en la `cantidad` dada. En caso contrario, debe insertar un nuevo nodo con la información dada *al final* de la lista.

b) `Inventario aReponer(int umbral)` **(1,5 puntos)**

Crea y devuelve un nuevo inventario con todos aquellos productos cuyas existencias sean menores o iguales que el `umbral` dado. Debes mantener el criterio de añadir nodos *al final* de la nueva lista. Por cuestiones de eficiencia, *no puedes recorrer varias veces la lista original ni la nueva lista.*

c) `void darDeBajaRango(String códigoMenor, String códigoMayor)` **(1,75 puntos)**

Elimina de la lista todos aquellos nodos cuyo código sea mayor o igual que `códigoMenor` y menor o igual que `códigoMayor`. Por cuestiones de eficiencia, *no puedes recorrer varias veces la lista.*