UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional
EI1008/MT1008 Programación II
Examen final. Segunda Convocatoria (20 de junio de 2022)

Apellidos: .....

Nombre: ..... **DNI:** .....

Problema 1 (4,5 puntos)

Una conocida librería permite la compra de libros online, animando a sus clientes a dar una valoración a los libros adquiridos. Para gestionar esta información se dispone de la clase `Libro` que proporciona, entre otros, los siguientes constructores y métodos públicos:

- `Libro(String ISBN, double puntos, int votos)`: Inicializa un objeto `Libro` con su ISBN, los puntos promedio de su valoración y el número de clientes que votaron.
- `String getISBN()`: Devuelve una cadena con el ISBN del libro.
- `double getPuntos()`: Devuelve el promedio de puntuación otorgada al libro por los compradores.
- `int getVotos()`: Devuelve el número de compradores que puntuaron el libro.

También tenemos la siguiente implementación parcial de la clase `Materia`, que almacena los libros y sus valoraciones de una determinada materia (ciencia-ficción, filosofía, cocina...):

```
public class Materia {
    private static final int TAM_INICIAL = 10;
    private Libro[] libros; // Vector de libros
                            // (ordenados de mayor a menor según su valoración)
    private int cantidad; // Cantidad real de libros almacenados en el vector

    public Materia() {
        libros = new Libro[TAM_INICIAL];
        cantidad = 0;
    }
    ...
}
```

El atributo `cantidad` indica la cantidad de libros del vector que realmente se están utilizando. Todos los elementos del vector que contengan un libro (que indica el atributo `cantidad`) están agrupados de forma contigua al principio del vector. Los libros están **ordenados por su valoración promedio de mayor a menor**, de manera que los libros con mayor puntuación aparecen primero.

Añade a la clase `Materia`:

a) **(1,5 puntos)** El método `public void eliminar(int minVotos)` que elimina del vector `libros` los libros que han sido votados por menos de `minVotos` clientes. El vector resultante ha de mantener la estructura con el número de libros a la izquierda del vector y el tamaño del vector no debe cambiar durante el proceso de borrado de libros. Recuerda mantener actualizado el atributo `cantidad`. Su coste temporal debe ser $\mathcal{O}(n)$, siendo n la talla del vector.

b) **(1 punto)** El método `public Libro getLibro(String ISBN)`, que devuelve el objeto de la clase `Libro` de entre los que están en el vector cuyo ISBN coincide con el dado. El método debe devolver `null` si no hay ningún libro en el vector con dicho ISBN.

c) **(2 puntos)** El método `public String [] mejoresLibros(Materia otraMateria, int n)` que devuelve un listado de los mejores libros (como un vector de ISBNs) según la valoración promedio de los clientes otorgada a dos secciones de la librería (`this` y `otraMateria`). Por ejemplo, si `n = 20` se devolverían los 20 libros mejor puntuados en la unión de ambas materias. Ten en cuenta que, dado que `this` y `otraMateria` son materias distintas, no puede haber libros que estén en ambas. Teniendo en cuenta que ambas secciones están ordenadas de mayor a menor valoración, se pide que el coste temporal de este método sea $\mathcal{O}(n)$.

Problema 2 (4,5 puntos)

Una conocida ONG acepta donaciones de particulares para ayudar a personas necesitadas. Queremos desarrollar un programa que permita llevar un registro del importe donado por cada persona. Tenemos ya una clase, `Donante`, que almacena internamente el DNI del donante, el importe total de las donaciones realizadas por esa persona y la fecha en la que realizó la última donación. La clase `Donante` proporciona los siguientes constructores y métodos públicos:

- `Donante(String dni, double importe)`: Inicializa un objeto de la clase `Donante` con el DNI y el importe dados. Además, establece como fecha de última donación la fecha actual.
- `String getDni()`: Devuelve el DNI del donante.
- `double getImporte()`: Devuelve el importe total donado.
- `Fecha getFechaÚltimaDonación()`: Devuelve la fecha en la que se hizo la última donación.
- `void añadirDonativo(double cantidad)`: Incrementa el importe total donado según la cantidad dada y establece la fecha de última donación a la fecha actual.

Queremos completar la implementación de la clase `RegistroDonantes`, que utiliza internamente una lista de nodos con enlace simple para almacenar la información necesaria. Cada nodo guarda la información de un donante. Los nodos están ordenados por fecha de última donación, de más recientes a más antiguas:

```
public class RegistroDonantes {

    private static class Nodo {
        Donante donante;
        Nodo sig;

        Nodo(Donante elDonante, Nodo elSiguiente) {
            this.donante = elDonante;
            this.sig = elSiguiente;
        }
    }

    // Atributo
    private Nodo primero;

    // Constructor (por defecto)
}
```

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

a) **(2 puntos)** `void añadir(String dni, double importe)`

Añade una donación a la lista. Si en la lista ya hay un donante con el DNI indicado, se debe incrementar el importe total de sus donaciones según el importe dado y actualizar la fecha de última donación. Para mantener la lista ordenada por fecha de donación, ese donante debe pasar a ocupar la primera posición de la lista. Si en la lista no aparecía el DNI dado, se debe crear un nuevo objeto `Donante` con el DNI y el importe indicados y almacenarlo en la primera posición de la lista.

El coste temporal de este método debe ser $\mathcal{O}(n)$.

b) **(1 punto)** `void eliminarDonantesSinActividad(Fecha fecha)`

Elimina de la lista toda la información de los donantes cuya fecha de última donación es anterior a la fecha dada[1].

El coste temporal de este método debe ser $\mathcal{O}(n)$.

c) **(1,5 puntos)** `void mostrarDonantesPorFecha()`

Muestra por pantalla, para cada una de las fechas que aparecen en la lista, la cantidad de donantes cuya última donación se produjo en esa fecha.

Teniendo en cuenta que la lista se encuentra ordenada por fecha, todos los objetos con una misma fecha de última donación se encontrarán almacenados de manera consecutiva en la lista. Por lo tanto, el coste temporal de este método debe ser $\mathcal{O}(n)$.

Un ejemplo de salida[2] de este método podría ser:

```
20/06/2022 2 donante(s)
19/06/2022 5 donante(s)
17/06/2022 3 donante(s)
12/06/2022 1 donante(s)
```

[1] Recuerda que la clase `Fecha` proporciona el método `compareTo`.
[2] El método `toString` de la clase `Fecha` utiliza el formato requerido.