UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional
EI1008/MT1008 Programación II
Examen final. Primera Convocatoria (23 de mayo de 2024)

PROBLEMA 1 (3 PUNTOS)

Necesitamos gestionar la información de todos los deportistas de alto nivel reconocidos en una comunidad autónoma. Para ello, ya disponemos de una clase, `Deportista`, que proporciona, entre otros:

- Un método `public String getDni()`, que devuelve el DNI del deportista.
- Un método `public String getDeporte()`, que devuelve el deporte practicado por el deportista.
- Un método `public Fecha getFechaInicio()`, que devuelve la fecha en la que el deportista fue reconocido como deportista de alto nivel.

Además, ya disponemos de la siguiente implementación parcial de la clase `DeportistasAltoNivel`:

```
public class DeportistasAltoNivel {

    // Atributos
    private String comunidad;     // Nombre de la Comunidad Autónoma
    private Deportista[] vector;  // Vector de deportistas de esa comunidad
                                  // (ordenado de menor a mayor por DNI)

    // Constructor
    public DeportistasAltoNivel(String comunidad) {
        this.comunidad = comunidad;
        this.vector = new Deportista[0];
    }
    ...
}
```

Como puedes ver, la clase `DeportistasAltoNivel` almacena en un vector la información de todos los deportistas de alto nivel de una comunidad autónoma, de modo que los datos de este vector siempre estén ordenados lexicográficamente por el DNI del deportista y su longitud coincida siempre con el número de deportistas reconocidos.

Añade a la clase `DeportistasAltoNivel` los siguientes métodos:

a) **(1,5 puntos)** `public String obtenerDeporte(String Dni)`

Cuando el vector contenga un deportista cuyo DNI coincida con el indicado, el método debe devolver el deporte practicado por tal deportista. En caso contrario, el método debe devolver `null`.

Dado que el vector de deportistas está ordenado por DNI, se exige que el coste temporal de este método sea $\mathcal{O}(\log n)$, donde `n` es la cantidad de deportistas en el vector.

b) **(1,5 puntos)** `public int eliminarObsoletos(Fecha límite)`

El método debe eliminar la información de todos los deportistas que fueron reconocidos como deportistas de alto nivel antes de la fecha `límite` indicada[1] y, además, debe devolver la cantidad de deportistas que han sido eliminados. Ten en cuenta que el tamaño del vector debe disminuir de forma acorde.

Dado que el vector de deportistas está ordenado por DNI, se exige que el coste temporal de este método sea $\mathcal{O}(n)$, donde `n` es la cantidad de deportistas en el vector.

[1] Recuerda que la clase `Fecha` implementada en prácticas proporciona el método `compareTo` para comparar dos fechas. Una llamada como `f1.compareTo(f2)` devuelve un número negativo, cero o un número positivo según f1 sea menor que, igual o mayor que f2, respectivamente.

Problema 2 (4 puntos)

Necesitamos gestionar la información de todos los estudiantes matriculados en una universidad. Para ello, ya disponemos de una clase, `Estudiante`, que proporciona, entre otros:

- Un método `public String getDni()`, que devuelve el DNI del estudiante.
- Un método `public boolean getGraduado()`, que devuelve `true` si el estudiante ha finalizado sus estudios y `false` en caso contrario.

Además, ya disponemos de la siguiente implementación parcial de la clase `EstudiantesUniversidad`:

```
public class EstudiantesUniversidad {
    private static class Nodo {
        Estudiante estudiante;
        Nodo sig;

        Nodo(Estudiante estudiante, Nodo sig) {
            this.estudiante = estudiante;
            this.sig = sig;
        }
    }

    // Atributos
    private String universidad;  // Nombre de la universidad
    private Nodo primero;        // Enlace al primer nodo de la lista de estudiantes
                                 // (nodos ordenados de menor a mayor por DNI)

    // Constructor
    public EstudiantesUniversidad(String universidad) {
        this.universidad = universidad;
        this.primero = null;
    }

    ...
}
```

Como puedes ver, la clase `EstudiantesUniversidad` utiliza una lista de nodos con enlace simple para almacenar la información de todos los estudiantes matriculados en una universidad, de modo que los nodos de esta lista siempre se mantengan ordenados lexicográficamente por el DNI del estudiante.

Añade a la clase `EstudiantesUniversidad` los siguientes métodos:

a) **(2 puntos)** `public boolean matricular(Estudiante estudiante)`

Cuando el DNI del estudiante indicado ya aparezca en la lista de estudiantes, la lista no debe modificarse y el método debe devolver `false`. En caso contrario, el estudiante debe añadirse a la lista, de modo que se mantenga el orden requerido en los nodos, y el método debe devolver `true`.

Se exige que el coste temporal de este método sea $\mathcal{O}(n)$, donde `n` es la cantidad de estudiantes en la lista.

b) **(2 puntos)** `public int eliminarGraduados()`

El método debe eliminar de la lista de estudiantes a todos aquellos que ya hayan finalizado sus estudios en la universidad y, además, debe devolver la cantidad de estudiantes que han sido eliminados.

Se exige que el coste temporal de este método sea $\mathcal{O}(n)$, donde `n` es la cantidad de estudiantes en la lista.

Problema 3 (2 puntos)

Considera las clases que han aparecido en los dos ejercicios anteriores. Añade a la clase `EstudiantesUniversidad` el siguiente método:

a) `public int contarDeportistasAN(Deportista[] deportistas)`

El método recibe un vector de objetos de la clase `Deportista`, ordenado de menor a mayor por DNI, y devuelve la cantidad de deportistas del vector que están matriculados en esa universidad.

Dado que tanto el vector de deportistas como la lista de estudiantes están ordenados por DNI, se exige que el coste temporal de este método sea $\mathcal{O}(n)$, donde `n` es la suma de la cantidad de elementos en el vector y en la lista.