UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional
EI1008/MT1008 Programación II
Examen final. Segunda convocatoria (04 de julio de 2016)

Pregunta 1 (5 puntos)

Queremos desarrollar una aplicación para gestionar rutas que visitan una serie de ciudades. Considera la siguiente definición (incompleta) de la clase Ruta:

```
public class Ruta {

    private static final String[] nombresDeCiudades = {...};
    private static final double distancias[][] = { {...}, {...}, ...};

    private String[] ciudades;  // Nombres de las ciudades en la ruta.

    public Ruta() {
        ciudades = new String[0]; // Inicialmente la ruta está vacía.
    }
}
```

Cada objeto de la clase `Ruta` contiene la información de una ruta concreta. Para ello, la clase dispone del atributo `ciudades` en el que se guardan los nombres de las ciudades que forman la ruta. El orden de las ciudades en el vector es importante. La ruta comienza en la primera ciudad del vector, pasa secuencialmente por cada una de las ciudades siguientes y termina en la última ciudad.

Además, la clase `Ruta` contiene dos atributos estáticos:

- `nombresDeCiudades` es un vector que contiene los nombres de todas las ciudades que conoce la clase `Ruta`. En este vector, los nombres de las ciudades se encuentran ordenados lexicográficamente de menor a mayor. Llamaremos índice de una ciudad a la posición que ocupa en este vector.
- `distancias` es una matriz que contiene la distancia entre todas las ciudades que conoce la clase `Ruta`. Si `i` y `j` son los índices de dos ciudades, las celdas `distancias[i][j]` y `distancias[j][i]` contienen la distancia entre ambas ciudades.

Añade a la clase `Ruta` los siguientes métodos:

a) **(1,25 puntos)** Un método privado, `int obtenerÍndice(String ciudad)`, que devuelva la posición de la ciudad dada en el vector `nombresDeCiudades`. Si la ciudad dada no aparece en el vector, devolverá un valor negativo.

El coste temporal de este método debe ser $\mathcal{O}(log\ N)$, siendo N el número de ciudades en el vector `nombresDeCiudades`.

b) **(0,50 puntos)** Un método privado, `double distancia(String ciudad1, String ciudad2)`, que devuelva como resultado la distancia entre las dos ciudades dadas. En este apartado debes utilizar el método definido en el apartado a). Si alguna de las dos ciudades dadas no aparece en el vector `nombresDeCiudades`, devolverá un valor negativo.

c) **(1,25 puntos)** Un método público, `double longitud()`, que devuelva como resultado la longitud total de la ruta. La longitud de una ruta se define como la suma de las distancias entre cada par de ciudades consecutivas. En este apartado debes utilizar el método definido en el apartado b).

d) **(2 puntos)** Un método público, `Ruta añadirCiudad(String ciudad)`, que devuelva una nueva ruta que contenga todas las ciudades de la ruta actual más la ciudad dada como argumento. La posición de la nueva ciudad dentro de la nueva ruta debe ser aquella para la que la longitud total de la nueva ruta sea mínima.

Por ejemplo, si tenemos la ruta [”Murcia”, ”Alicante”, ”Castellón”] y se añade la ciudad de ”Valencia” a la ruta, se deben considerar las siguientes posibilidades:

- [”Valencia”, ”Murcia”, ”Alicante”, ”Castellón”]
- [”Murcia”, ”Valencia”, ”Alicante”, ”Castellón”]
- [”Murcia”, ”Alicante”, ”Valencia”, ”Castellón”]
- [”Murcia”, ”Alicante”, ”Castellón”, ”Valencia”]

y devolver como resultado aquella cuya longitud sea menor.

Pregunta 2 (5 puntos)

Queremos desarrollar una aplicación que permita evaluar la experiencia de los clientes tras una visita a un restaurante. Para ello, se cuenta con la colaboración de voluntarios que proporcionan información sobre su experiencia tras comer o cenar en dichos restaurantes. Para almacenar la información de una valoración suministrada por un voluntario disponemos de la clase `Valoración`, que tiene, entre otros, los siguientes métodos públicos (considera que ya existe la clase `Fecha`[1], la misma que has desarrollado en las prácticas de la asignatura):

- `Fecha getFecha()`: devuelve la fecha de la valoración.
- `String getEstablecimiento()`: devuelve el nombre del establecimiento.
- `double getPuntuación()`: devuelve la valoración, en forma de puntuación, realizada por un voluntario.

Para gestionar las valoraciones vamos a utilizar una **lista simplemente enlazada en la que los nodos están ordenados por nombre del establecimiento**. Considera la siguiente definición (incompleta) de la clase `ListaValoraciones`:

```
public class ListaValoraciones {

    private static class Nodo {
        Valoración   valoración;
        Nodo         sig;

        Nodo(Valoración valoración, Nodo siguiente) {
            this.valoración = valoración;
            this.sig        = siguiente;
        }
    }

    // Atributos
    private Nodo primerNodo;

    //... Resto de la clase
}
```

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

a) **(1,75 puntos)** Un método, `void añadir(Valoración valoración)`, que reciba una valoración y la añada a la lista de valoraciones. Recuerda que la lista debe estar ordenada por nombre de establecimiento de menor a mayor. En el caso de que varias valoraciones correspondan al mismo establecimiento, el orden en el que se almacenen esos nodos es indiferente.

b) **(1,5 puntos)** Un método, `double puntuaciónMedia(String establecimiento)`, que reciba el nombre de un establecimiento y devuelva como resultado la puntuación media obtenida en las valoraciones recibidas. En caso de que el establecimiento no haya recibido ninguna valoración, devolverá `-1.0`.

c) **(1,75 puntos)** Un método, `void eliminarValoracionesAntesDe(Fecha fecha)`, que reciba una fecha y elimine de la lista todas las valoraciones anteriores a la fecha dada.

[1] La clase `Fecha` dispone del método `compareTo`. Si `f1` y `f2` son de tipo `Fecha`, `f1.compareTo(f2)` devuelve un número negativo, cero o un número positivo según `f1` sea menor que, igual o mayor que `f2`, respectivamente.