UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional
EI1008/MT1008 Programación II
Examen final. Segunda Convocatoria (03 de julio de 2017)

PREGUNTA 1 (5 PUNTOS)

Queremos implementar una colección de cromos de fútbol, donde cada cromo representa un futbolista que tiene un nombre que lo identifica. Tenemos ya la clase `Cromo` que proporciona, entre otros, el siguiente método público:

```
String getNombre()  // Devuelve el nombre del futbolista
```

Necesitamos definir una nueva clase, `Colección`, que mantenga la información necesaria de todos los cromos. Para ello, vamos a utilizar un *vector ordenado por nombre de futbolista de menor a mayor*. Considera que ya tienes la siguiente definición parcial de la clase:

```
public class Colección {

    // Atributos
    private String título;   // Título de la colección

    private Cromo[] vector;  // Vector con todos los cromos de la colección
                             // (ordenado por nombre de futbolista)

    // Constructores y métodos
    ...
}
```

Ten en cuenta que en la colección puede haber cromos repetidos. En este caso, el cromo aparecerá varias veces en el vector, siempre en posiciones consecutivas (recuerda que el vector está ordenado por nombre de futbolista).

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

a) `void añadir(Cromo cromo)` **(1,75 puntos)**

Añade a la colección el cromo indicado de modo que el vector de cromos siga ordenado. En caso de que haya otros cromos con el mismo nombre, el orden entre ellos es indiferente.

b) `String másRepetido()` **(1,5 puntos)**

Devuelve el nombre del futbolista que aparece más veces en la colección. En caso de empate entre varios futbolistas, puedes devolver cualquiera de ellos.

El coste temporal de este método debe ser $\mathcal{O}(n)$, siendo n es el número de elementos en el vector de cromos.

c) `int cromosComunes(Colección otraColección)` **(1,75 puntos)**

Devuelve el número de cromos que coinciden en ambas colecciones. Cuando haya cromos repetidos, se contarán todos los que coincidan en ambas colecciones. Por ejemplo, si un futbolista aparece 3 veces en una colección y 1 vez en la otra, contará como un cromo en común; si aparece 5 veces en una colección y 3 en la otra, contará como 3 cromos en común.

El coste temporal de este método debe ser $\mathcal{O}(n)$, siendo n es el número de elementos en el vector de cromos.

Pregunta 2 (5 puntos)

Te interesa llevar un control de las notas que sacas a lo largo del curso en las distintas asignaturas en las que estás matriculado. Dado que en cada asignatura tienes una serie de notas que se ponderan para la nota final, decides crear una clase, `Nota`, que proporciona un constructor y varios métodos públicos:

- `Nota(String asignatura, double puntuación, double peso)`: constructor de la clase. Recibe el código de la asignatura a la que corresponde esa nota, su puntuación y el peso (sobre uno) que tiene en la calificación final. Por ejemplo, como este examen tiene un peso del 50 %, si sacaras un 7, deberías crear la correspondiente nota mediante

  ```
  Nota nota = new Nota("EI1008", 7, 0.50);
  ```

- `String getAsignatura()`: devuelve el código de la asignatura a la que corresponde la nota.
- `double getPuntuación()`: devuelve la puntuación asociada a la nota.
- `double getPeso()`: devuelve el peso (sobre uno) de esa nota.

Necesitas definir una nueva clase, `ListaNotas`, que mantenga la información necesaria de todas las notas obtenidas. Para ello, decides utilizar una *lista simplemente enlazada con una referencia al primer nodo y otra al último*. Considera que ya tienes la siguiente definición parcial de la clase:

```
public class ListaNotas {

    private static class Nodo {
        Nota nota;
        Nodo sig;

        Nodo(Nota nota, Nodo sig) {
            this.nota = nota;
            this.sig = sig;
        }
    }

    // Atributos
    private Nodo primero;
    private Nodo último;

    // Constructores y métodos
    ...
}
```

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

a) `double estimaNotaAsignatura(String asignatura)` **(1,25 puntos)**

Devuelve una estimación de la nota final de la asignatura dada. Si la lista de notas no contiene ninguna nota para esa asignatura, se debe lanzar la excepción `NoSuchElementException`. Para estimar la nota final de una asignatura, obtén la suma de todas sus puntuaciones ponderadas por su peso y la suma de esos pesos; después, divide la suma de las puntuaciones entre la de los pesos. Por ejemplo, si en una asignatura hay dos notas, un 3 ponderado con 0,1 y un 8 ponderado con 0,4, la estimación de la nota final se calcularía así:

$$\frac{3 \times 0,1 + 8 \times 0,4}{0,1 + 0,4} = \frac{3,5}{0,5} = 7$$

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

Devuelve un vector con los códigos de las diferentes asignaturas que hay en la lista de notas (si una asignatura tiene varias notas, su código solo debe aparecer una vez). Para implementar este método, no puedes recorrer la lista de notas más de una vez ni utilizar los métodos de la clase `Arrays`.

c) `String asignaturaMejorNota()` **(0,75 puntos)**

Devuelve el código de la asignatura con mejor nota estimada. Si se produce un empate entre varias asignaturas, puedes devolver cualquiera de ellas. Tu solución debe utilizar los métodos `asignaturas` y `estimaNotaAsignatura`, definidos anteriormente.

d) `int borrarNotasAsignatura(String asignatura)` **(1,5 puntos)**

Borra todas las notas correspondientes a la asignatura dada y devuelve la cantidad de notas que ha eliminado.