UNIVERSITAT JAUME·I

1º Ingeniería Informática
1º Matemática Computacional
EI1008/MT1008 Programación II
Examen final. Segunda Convocatoria (1 de julio de 2019)

PREGUNTA 1 (5 PUNTOS)

Disponemos de la clase `Cuenta` para representar la situación de una cuenta bancaria. Cada cuenta se identifica mediante su número de cuenta (nunca habrá dos cuentas con un mismo número). Esta clase proporciona los siguientes constructores y métodos públicos:

- `Cuenta(String númeroCuenta, String titular)`: inicializa una nueva cuenta con los datos que se indican y un saldo inicial de cero euros.
- `String getNúmeroCuenta()`: devuelve el número de cuenta.
- `String getTitular()`: devuelve el nombre del titular.
- `double getSaldo()`: devuelve el valor actual del saldo.
- `void ingresar(double importe)`: incrementa el saldo de la cuenta según el importe dado.
- `void reintegrar(double importe)`: decrementa el saldo de la cuenta según el importe dado.

Para la gestión de las cuentas de una sucursal bancaria necesitamos definir una nueva clase, `Sucursal`. Considera que ya tenemos la siguiente definición parcial:

```
public class Sucursal {

   // Atributos
   private String código;     // Código que identifica a la sucursal bancaria.

   private Cuenta[] cuentas;  // Vector de cuentas ordenado de menor a mayor por número de cuenta.
                              // Su tamaño coincidirá siempre con la cantidad de cuentas en la
                              // sucursal (es decir, nunca habrá componentes a null).
   // Constructor
   public Sucursal(String código) {
      this.código = código;
      this.cuentas = new Cuenta[0];  // Vector con 0 cuentas bancarias.
   }
}
```

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

a) `void añadirCuenta(String númeroCuenta, String titular)` **(1,5 puntos)**.

Crea un nuevo objeto de la clase `Cuenta` con los datos indicados (supón que no habrá otra cuenta con ese número) y lo añade a la sucursal de modo que el vector de cuentas siga ordenado por número de cuenta.

El coste temporal de este método debe ser $\mathcal{O}(n)$, siendo n el número de cuentas en la sucursal.

b) `void asentarOperación(char tipo, String c1, String c2, double importe)` **(2 puntos)**.

El tipo de operación puede ser `'I'` (ingreso), `'R'` (reintegro) o `'T'` (transferencia). Para los ingresos y reintegros, el saldo de la cuenta cuyo número es `c1` se debe actualizar según el importe dado (en estas operaciones, `c2` no se utiliza). Para las transferencias, se debe realizar un reintegro del importe dado en la cuenta `c1` y un ingreso del mismo importe en la cuenta `c2`.

Dado que el vector de cuentas bancarias está ordenado por número de cuenta, se exige que el coste temporal de este método sea $\mathcal{O}(log\ n)$, siendo n el número de cuentas en la sucursal.

c) `int borrar(String titular)` **(1,5 puntos)**.

Borra de la sucursal todas las cuentas que pertenezcan al titular dado y devuelve la cantidad de cuentas que ha borrado.

El coste temporal de este método debe ser $\mathcal{O}(n)$, siendo n el número de cuentas en la sucursal.

Recuerda que el tamaño del vector debe coincidir con el número de cuentas en la sucursal.

Pregunta 2 (5 puntos)

Necesitamos desarrollar una aplicación de control parental para dispositivos móviles. Ya hemos definido una clase, `TiempoUso`, que proporciona los siguientes constructores y métodos públicos:

- `TiempoUso(String usuario, String aplicación, int minutos)`: constructor de la clase. Almacena la cantidad de minutos que un usuario ha estado utilizando una aplicación.
- `String getUsuario()`: devuelve el nombre del usuario.
- `String getAplicación()`: devuelve el nombre de la aplicación.
- `int getMinutos()`: devuelve la cantidad de minutos que el usuario ha usado la aplicación.
- `void aumentar(int minutos)`: incrementa en la cantidad de minutos indicada el tiempo de uso de la aplicación por parte del usuario.

Necesitamos definir una nueva clase, `ControlParental`, que controle el tiempo de uso de todos los usuarios registrados. Para ello, utilizaremos una *lista simplemente enlazada con una referencia al primer nodo*. Considera que ya tenemos la siguiente definición parcial de la clase:

```
public class ControlParental {
    private static class Nodo {
        TiempoUso uso;
        Nodo sig;

        Nodo(TiempoUso uso, Nodo sig) {
            this.uso = uso;
            this.sig = sig;
        }
    }

    private Nodo primero;
}
```

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

a) `void añadir(String usuario, String aplicación, int minutos)` **(2 puntos)**.

Cuando en la lista ya exista un objeto para ese usuario y esa aplicación, el método debe incrementar la cantidad de minutos almacenada según la cantidad dada. En caso contrario, debe crear un nuevo objeto con la información necesaria y añadirlo a la lista.

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

b) `boolean seDebeBloquear(String usuario, int límite))` **(1 punto)**.

Debe devolver `true` cuando el tiempo total dedicado por el usuario a todas las aplicaciones supere el límite de minutos indicado y `false` en otro caso.

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

c) `void borrar(String usuario)` **(2 puntos)**.

Borra de la lista todos los nodos correspondientes a ese usuario. Por cuestiones de eficiencia, *no puedes recorrer varias veces la lista.*