Parcial 2 - PRÁCTICAS - PRG - ETSInf. Curso 2016-17
5 de junio de 2017. Duración: 1 hora

Nota: El examen se evalúa sobre 10 puntos, pero su peso específico en la nota final de la asignatura es de 1,2 puntos.

NOMBRE:	GRUPO DE PRÁCTICAS:

1. 5 puntos Se desea modificar la clase `Banco` de la práctica 4, para permitir la gestión de remesas de transferencias. Una remesa viene en un fichero de texto en el que cada línea contiene los datos de una transferencia con tres valores separados por espacios:

   ```
   numCuenta1 numCuenta2 importe
   ```

   en donde `numCuenta1`, `numCuenta2` son enteros entre 10000 y 99999 que corresponden a números de cuenta existentes en el banco, e `importe` es un valor real positivo.

   Es decir, los datos de una línea indican que se quiere hacer una transferencia de `importe` euros desde la cuenta número `numCuenta1` a la cuenta número `numCuenta2`.

   Se pide: suponiendo que todas las clases necesarias han sido importadas en la clase `Banco`, implementar un método con perfil

   ```
   public void hacerTransf(Scanner remesa)
   ```

   tal que, utilizando el parámetro `Scanner` ya inicializado, lea cada línea del fichero y retire el `importe` de la cuenta con número de cuenta `numCuenta1` para ingresarlo a continuación en la cuenta con número de cuenta `numCuenta2`.

   En caso de cualquier error en una línea, se debe obviar toda la línea mostrando un mensaje que indique el motivo del mismo, y seguir tratando el resto de datos.

   Nota: Recordar que el método `retirar(double)` de la clase `Cuenta` puede lanzar una excepción de tipo `IllegalArgumentException` si la cantidad a retirar es mayor que el saldo de la cuenta, y que los métodos de lectura del `Scanner` pueden lanzar una excepción `InputMismatchException` si el tipo del token leído no coincide con el esperado por el método.

   Recordar también que el método `getCuenta(int)` de la clase `Banco` devuelve, si existe, la cuenta cuyo número se pasa como parámetro, o `null` si no existe –aunque en este ejercicio nunca sucederá, ya que los números de cuenta del fichero son de cuentas existentes en el banco–.

Solución:

```
public void hacerTransf(Scanner remesa) {
    int numC1 = 0, numC2 = 0;  double importe = 0;
    while (remesa.hasNext()) {
        try {
            numC1 = remesa.nextInt();
            numC2 = remesa.nextInt();
            importe = remesa.nextDouble();
            Cuenta c1 = this.getCuenta(numC1);
            Cuenta c2 = this.getCuenta(numC2);
            c1.retirar(importe);
            c2.ingresar(importe);
        } catch (InputMismatchException e) {
            System.err.println("línea errónea");
        } catch (IllegalArgumentException e) {
            System.err.println("importe erróneo");
        } finally { remesa.nextLine(); }
    }
}
```

2. [5 puntos] Sea la clase `ConjuntoString` de la práctica 5:

```
public class ConjuntoString {

    private NodoString primero;
    private int talla;

    /** Crea un conjunto vacío */
    public ConjuntoString() {
        this.primero = null;
        this.talla = 0;
    }

    ....


}
```

en donde la secuencia enlazada `primero` contiene los elementos que forman parte del conjunto; todos los métodos se implementan de manera que, para cualquier conjunto de la clase, dicha secuencia está ordenada ascendentemente por el orden de `String` y sin elementos repetidos.

Se pide implementar un nuevo método dentro de la clase, con perfil

```
public ConjuntoString subconjuntoPalabras(char c)
```

que devuelva un `ConjuntoString` con todos los elementos del conjunto que empiecen por el carácter `c`. Se deberá procurar que el método sea $O(n)$, siendo n la talla de `this`.

Recordar que se puede consultar el carácter que aparece en la posición i-ésima de un `String` con el método `charAt(int i)`.

Solución:

```
public ConjuntoString subconjuntoPalabras(char c) {
    ConjuntoString cs = new ConjuntoString();
    NodoString aux = primero;
    NodoString ultCs = null;
    boolean fin = false;
    while (aux != null && !fin) {
        String p = aux.dato;
        if (p.charAt(0) > c) { fin = true; }
        else if (p.charAt(0) == c) {
            // Insertar al final de cs:
            NodoString nuevo = new NodoString(p);
            if (cs.primero == null) { cs.primero = nuevo; }
            else { ultCs.siguiente = nuevo;  }
            ultCs = nuevo;
            cs.talla++;
        }
        aux = aux.siguiente;
    }
    return cs;
}
```