Parcial 2 - PRÁCTICAS - PRG - ETSInf. Curso 2015-16
31 de mayo de 2016. 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. Así, los valores de las preguntas 3, 3 y 4 puntos equivalen en la nota final a 0.35, 0.35 y 0.5 puntos, respectivamente.)

NOMBRE:	GRUPO DE PRÁCTICAS:

1. [3 puntos] Implementar un método `toArray()` en la clase `ColaApuntes` de la práctica 5 que devuelva un array con los apuntes de la cola de apuntes.
   Nota: Recordar que los atributos definidos en la clase `ColaApunte` son `talla` (de tipo `int`), `primero` y `ultimo` (de tipo `NodoApunte`) y los de la clase `NodoApunte` son `dato` (de tipo `Apunte`) y `siguiente` (de tipo `NodoApunte`).

Solución:

```
public Apunte[] toArray() {
    Apunte[] ap = new Apunte[talla];
    NodoApunte aux = primero;
    for (int i = 0; i < talla; i++) {
        ap[i] = aux.dato;
        aux = aux.siguiente;
    }
    return ap;
}
```

2. [3 puntos] Escribir un método con perfil: `public double promedioApuntes()`, en la clase `Banco` de la práctica 5, que calcule el número promedio de apuntes de las cuentas del banco.

   Nota: Recordar que los atributos definidos en la clase `Banco` de la práctica 5 son `primero` (de tipo `NodoCuenta`) y `numCuentas` (de tipo `int`), y los de la clase `NodoCuenta` son `dato` (de tipo `CuentaAp`) y `siguiente` (de tipo `NodoCuenta`). En la clase `CuentaAp` se define el método `getNumApuntes()` que devuelve el número de apuntes de la cuenta.

Solución:

```
public double promedioApuntes() {
    double promedio = 0;
    if (numCuentas != 0){
        NodoCuenta aux = primero;
        while (aux != null) {
            promedio += aux.dato.getNumApuntes();
            aux = aux.siguiente;
        }
        promedio = promedio / numCuentas;
    }
    return promedio;
}
```

3. [4 puntos] Se desea modificar la clase `Banco` para permitir la gestión de remesas de recibos. Cada remesa viene en un fichero de texto donde cada línea tiene tres valores separados por espacios: `numCuenta` que es un entero entre 10000 y 90000, `importe` que es un valor real, y `numRecibo` que es un entero largo cuyo rango de valores está comprendido entre las constantes `MIN_NUM_RECIBO` y `MAX_NUM_RECIBO` ambas incluidas.

Se pide: asumiendo que todas las clases necesarias han sido importadas en la clase `Banco`, implementar un método con perfil: `public String gestionarRemesa(Scanner remesa)` tal que, utilizando el parámetro `Scanner` ya inicializado, lea las líneas con los datos de cada recibo del fichero, valide los datos y realice el cargo, si es posible, en la cuenta correspondiente, devolviendo un `String` con los números de recibos (`numRecibo`) procesados separados por saltos de línea. En caso de cualquier error se debe obviar la línea entera y mostrar un mensaje indicando el motivo del mismo. El `String` resultado sólo debe contener los números de recibos que sí se han podido procesar adecuadamente, esto es: que sean válidos, cuyas cuentas existan y que no hayan dado error al efectuarse el cargo. En caso de que el fichero no contenga ningún recibo válido, debe devolver un `String` vacío.

Nota: Recordar que el método `getCuenta(int)` de la clase `Banco` de la práctica 4 devuelve, si existe, la cuenta cuyo número se pasa como parámetro o `null` si no existe. El método `retirar(double)` de la clase `Cuenta` puede lanzar la excepción `IllegalArgumentException` si la cantidad a retirar es mayor que el saldo de la cuenta, y los métodos de lectura del `Scanner`, a su vez, pueden lanzar una excepción de tipo `InputMismatchException` si el tipo del token leído del fichero no coincide con el esperado por el método.

Solución:

```
public String gestionarRemesa(Scanner remesa) {
    String res = "";
    int numC = 0;  double importe = 0;  long numR = 0;
    while (remesa.hasNext()) {
        try {
            numC = remesa.nextInt();
            importe = remesa.nextDouble();
            numR = remesa.nextLong();
            Cuenta c = this.getCuenta(numC);
            if (c != null) {
                if (numR <= MAX_NUM_RECIBO && numR >= MIN_NUM_RECIBO) {
                    c.retirar(importe);
                    res += numR + "\n";
                } else { System.err.println("numRecibo erróneo"); }
            } else { System.err.println("cuenta errónea"); }
        } catch(InputMismatchException e) {
             System.err.println("línea errónea");
        } catch(IllegalArgumentException e) {
             System.err.println("importe erróneo");
        } finally { remesa.nextLine(); }
    }
    return res;
}
```