Parcial 2 - Prácticas - PRG - ETSInf - Curso 2013/14
10 de junio de 2014 - Duración: 1 hora y 15 minutos

1. 2.5 puntos El método `leerInt`, que figura a continuación, lee desde un `Scanner` y devuelve un valor entero en el intervalo `[lInf..lSup]`. Este método se utiliza desde el programa principal para leer un número de cuenta de 5 dígitos (entre 10000 y 99999).

```
public static int leerInt(Scanner t, String mensaje, int lInf, int lSup) {
    System.out.print(mensaje);
    int res = t.nextInt();
    return res;
}

public static void main(String[] args) {
    Scanner tec = new Scanner(System.in);
    int numC = leerInt(tec,"Introduce un número de cuenta (de 5 dígitos): ",10000,99999);
    System.out.println("El número de cuenta leído es: " + numC);
}
```

Se pide:

a) [1.5 puntos]: Modificar el método `leerInt` para que, si el valor introducido no está en el rango `[lInf..lSup]`,
- **lance** la excepción `IllegalArgumentException`, con un mensaje que indique que el valor leído no está en dicho rango,
- y la **propague explícitamente**.

Solución:

```
public static int leerInt(Scanner t, String mensaje, int lInf, int lSup)
                                                        throws IllegalArgumentException {
    System.out.print(mensaje);
    int res = t.nextInt();
    if (res>lSup || res<lInf)
        throw new IllegalArgumentException(res + " no está en [" + lInf + ".." + lSup + "]");
    return res;
}
```

b) [1 punto]: Modificar también el método `main` para que **capture** esta excepción, mostrando el mensaje de la misma mediante el método `getMessage()`.

Solución:

```
public static void main(String[] args) {
    Scanner tec = new Scanner(System.in);
    try {
        int numC = leerInt(tec,"Introduce un número de cuenta (de 5 dígitos): ",10000,99999);
        System.out.println("El número de cuenta leído es: " + numC);
    } catch(IllegalArgumentException e) {
        System.out.println(e.getMessage());
    }
}
```

2. 2.5 puntos **Se pide:** Escribir un método que, dado el nombre de un **fichero de texto** con la información de las cuentas de un banco, devuelva la suma de los saldos de todas las cuentas. Cada línea del fichero tiene dos elementos de información, un número de cuenta de tipo `int` seguido de un saldo de tipo `double`. La cabecera del método es la que figura a continuación. **No tienes que tratar ninguna excepción**, fíjate que se propaga cualquier excepción que se pueda producir.

```
public static double sumaSaldos(String nomFich) throws Exception
```

Solución:

```
public static double sumaSaldos(String nomFich) throws Exception {
    double suma = 0;
    Scanner ent = new Scanner(new File(nomFich)).useLocale(Locale.US);
    while(ent.hasNext()) {
        int numCuenta = ent.nextInt();
        double saldo = ent.nextDouble();
        suma += saldo;
    }
    ent.close();
    return suma;
}
```

3. [3 puntos] Dadas las estructuras de datos `Concordancia` y `NodoCnc`, como las vistas en prácticas, con atributos según las declaraciones siguientes:

```
Concordancia                          NodoCnc
------------                          -------
private NodoCnc prim;                 String pal;
private int talla;                    ColaIntEnla numLins;
private boolean esOrd;                NodoCnc siguiente;
private String separadores;
```

Se pide: Escribir un método dentro de la clase `Concordancia` con perfil:

```
// PRECONDICIÓN: n >= 1
public boolean masAparicionesN(int n)
```

que determine si existe en el texto con el que se ha construido la `Concordancia` una palabra que aparezca **al menos** `n` veces.

Solución:

```
// PRECONDICIÓN n >= 1
public boolean masAparicionesN(int n) {
    NodoCnc aux = prim;
    while(aux!=null && aux.numLins.talla()<n)
        aux = aux.siguiente;
    return aux!=null;
}
```

4. [1 punto] Los métodos públicos de la clase `ColaIntEnla` son el constructor `ColaIntEnla()`, los modificadores `encolar(int)` y `desencolar()`, y los consultores `primero()`, `esVacia()`, `talla()` y `toString()`.

Se pide: Enumerar cuáles de estos métodos se han utilizado en:

a) la clase `NodoCnc`

Solución: Los métodos `ColaIntEnla()` y `encolar(int)`.

b) la clase `Concordancia`

Solución: Los métodos `encolar(int)`, `talla()` y `toString()`.

5. [1 punto] **Se pide:** Justificar brevemente cuál es el coste asintótico (constante, lineal, cuadrático, logarítmico, etc.) del método:

```
public boolean esOrdenada()
```

definido en la clase `Concordancia`, que devuelve si se trata de una `Concordancia` ordenada o no.

Solución: El método `esOrdenada()` tiene un coste constante ya que es un método consultor que sólo ejecuta un `return` del atributo `esOrd`.