Parcial 2 - Prácticas - PRG - ETSInf - Curso 2013/14
23 de junio de 2014 - Duración: 50 minutos

1. 2.5 puntos El método `leerPalabraDesde`, que figura a continuación, lee una cadena de caracteres y un entero desde un `Scanner` y devuelve la subcadena a partir de la posición indicada por el entero leído. Al invocar al método `substring(int)` puede darse la excepción `IndexOutOfBoundsException` si el entero es negativo o más grande que la longitud de la cadena leída.

```
public static String leerPalabraDesde(Scanner t) {
    System.out.print("Introduce una palabra: ");
    String linea = t.nextLine();
    System.out.println("Introduce una posición: ");
    int pos = t.nextInt();
    t.nextLine();
    String res = linea.substring(pos);
    return res;
}

public static void main(String[] args) {
    Scanner tec = new Scanner(System.in);
    String palabra = leerPalabraDesde(tec);
    System.out.println("La palabra leída es: " + palabra);
}
```

Se pide: Modificar el método `main` para que **capture** la excepción `IndexOutOfBoundsException` y vuelva a pedir los datos, ejecutando `leerPalabraDesde` todas las veces necesarias, hasta que dicha excepción no se produzca.

Solución:

```
public static void main(String[] args) {
    Scanner tec = new Scanner(System.in);
    boolean hayError = true;
    do {
        try {
            String palabra = leerPalabraDesde(tec);
            System.out.println("La palabra leída es: " + palabra);
            hayError = false;
        } catch (IndexOutOfBoundsException e) {
              System.out.println("¡Asegúrate de introducir una posición válida!");
        }
    } while (hayError);
}
```

2. 2.5 puntos **Se pide:** Escribir un método que, dado un valor de tipo `double` que representa un saldo y un `Scanner` de un fichero de texto con la información de las cuentas de un banco, lea desde el `Scanner` y escriba en un fichero de texto *"cuentas.txt"* los números de cuenta con saldo mayor que el dado.

Cada línea del fichero del que leer 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 se tiene que tratar ninguna excepción**, ya que se propaga cualquier excepción que se pueda producir.

```
public static void eligeCuentas(double s, Scanner in) throws Exception
```

Solución:

```
public static void eligeCuentas(double s, Scanner in) throws Exception {
    PrintWriter out = new PrintWriter("cuentas.txt");
    while(in.hasNext()) {
        int numCuenta = in.nextInt();
        double saldo = in.nextDouble();
        if (saldo>s) out.println(numCuenta);
    }
    out.close();
}
```

3. 2.5 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 int numPalabrasMasN(int n)
```

que devuelva cuántas palabras aparecen más de `n` veces en el texto con el que se ha construido la `Concordancia`.

Solución:

```
// PRECONDICIÓN: n >= 1
public int numPalabrasMasN(int n) {
    int cont = 0;
    for(NodoCnc aux = prim; aux!=null; aux = aux.siguiente)
        if (aux.numLins.talla()>n) cont++;
    return cont;
}
```

4. 2.5 puntos A continuación se muestra una posible implementación incompleta del método `insNoOrd(String,int)` de la clase `Concordancia`.

```
private void insNoOrd(String pal, int numLin) {
    NodoCnc aux = prim, ant = null;
    while( /* GUARDA */ ) { ant = aux;  aux = aux.siguiente; }

    if ( /* CONDICIÓN */ )  // pal ya está en la Concordancia
        /* INSTRUCCIÓN */
    else {  // El nuevo NodoCnc con la palabra pal tiene que ser el último y
            // se ha de insertar detrás de ant
            NodoCnc nuevo = new NodoCnc(pal, numLin);
            talla++;
            if (ant==null) // el nuevo NodoCnc será el primero y el último
                /* INSTRUCCIÓN */
            else // el nuevo NodoCnc será el último
                /* INSTRUCCIÓN */
        }
}
```

Se pide: Completar el código anterior con:

a) [0.6 puntos]: la guarda del bucle.

b) [0.4 puntos]: la condición de la instrucción condicional.

c) [0.7 puntos]: la instrucción que falta si la palabra `pal` ya está en la `Concordancia`.

d) [0.4 puntos]: la instrucción que falta si la palabra `pal` tiene que ser la primera y la última de la `Concordancia`.

e) [0.4 puntos]: la instrucción que falta si la palabra `pal` tiene que ser la última de la `Concordancia`.

Solución:

```
private void insNoOrd(String pal, int numLin) {
    NodoCnc aux = prim, ant = null;
    while(aux!=null && !aux.pal.equals(pal)) {  // a)
            ant = aux;  aux = aux.siguiente;
    }

    if (aux!=null)                          // b)
        aux.numLins.encolar(numLin);        // c)
    else {
            NodoCnc nuevo = new NodoCnc(pal, numLin);
            talla++;
            if (ant==null)
                prim = nuevo;               // d)
            else ant.siguiente = nuevo;     // e)
        }
}
```