PRG - ETSInf. TEORÍA. Curso 2013-14. Recuperación Parcial 2.
23 de junio de 2014. Duración: 1h 50m.

1. 2.5 puntos Dados un `String nomFich` con el nombre de un fichero de texto y cierta palabra `pal` (una palabra es un *token* o sucesión de caracteres no separados por blancos Java), realizar un método que determine si `pal` aparece en el fichero. En el método deberá tratarse la posible excepción `FileNotFoundException`, comunicando la no existencia del fichero al usuario mediante un mensaje en la salida estándar y retornando `false`.

Solución:

```
public static boolean estaEn(String nomFich,String pal) {
   try {
         Scanner entrada = new Scanner(new File(nomFich));
         boolean encontrado = false;
         while (entrada.hasNext() && !encontrado) {
           String p = entrada.next();
           if (p.equals(pal)) encontrado = true;
         }
         entrada.close();
         return encontrado;
   } catch (FileNotFoundException e) {
         System.out.println("Fichero " + nomFich + " no encontrado");
         return false;
   }
}
```

2. 2.5 puntos Considérese la siguiente clase `NodoPersona`:

```
class NodoPersona {
   int dni;
   String nombre;
   NodoPersona siguiente;

   NodoPersona(int i, String s, NodoPersona n) {
      dni = i;
      nombre = s;
      siguiente = n;
   }
}
```

Se pide implementar los siguientes métodos estáticos en la clase `NodoPersona`:

1. (1.25 puntos) Un método llamado `contar`, tal que, dada una secuencia de `NodoPersona` y un `String`, devuelva el número de nodos en la secuencia tales que el atributo `nombre` del nodo contenga al `String`. En la clase `String` hay un método, de perfil `boolean contains(String s)`, que comprueba si el `String` que lo invoca contiene a `s`.
2. (1.25 puntos) Un método llamado `buscar`, tal que, dada una secuencia de `NodoPersona` y un `int`, busque si en la secuencia existe algún nodo cuyo atributo `dni` sea igual a dicho `int`. En el caso de encontrar el nodo, se devuelve el valor de su atributo `nombre`; en el caso de no encontrarlo, se devuelve la cadena `"Persona desconocida"`.

Solución:

```
public static int contar(NodoPersona n, String s) {
   int cont = 0;
   NodoPersona aux = n;
   while (aux!=null) {
      if (aux.nombre.contains(s)) cont++;
      aux = aux.siguiente;
   }
   return cont;
}

public static String buscar(NodoPersona n, int i) {
   NodoPersona aux = n;
   while (aux!=null) {
      if (aux.dni==i) return aux.nombre;
      aux = aux.siguiente;
   }
   return "Persona desconocida";
}
```

3. 2.5 puntos Se desea añadir a la clase `ListaPIIntEnla` un nuevo método de inserción en el punto de interés (además del método `insertar(int)` ya definido en la misma) llamado `insertarSinRepetir`, el cual realice la inserción de un elemento solo si no se encuentra en la lista; en caso contrario debe lanzar una excepción `IllegalArgumentException` (predefinida de Java y derivada de `RuntimeException`), con el mensaje `"Ya está el elemento "` seguido del entero que se ha tratado de insertar. Por ejemplo:

   - Si la lista es `9 1 4 2 9` y se quiere insertar el nuevo elemento `0`, la lista debe quedar `9 1 0 4 2 9`.
   - Si la lista es `9 1 4 2 9` y se quiere insertar el nuevo elemento `2`, la lista debe quedar `9 1 4 2 9` y se debe lanzar la excepción con el mensaje: `Ya está el elemento 2`.

   Se pide:

   1. (2.25 puntos) Escribir el método `insertarSinRepetir` según la descripción anterior. Notar que si se verifica que `e` no existe de antemano en la lista en ninguna posición, se puede usar el método `insertar` de la clase.

      Solución:

      ```
      public void insertarSinRepetir(int e) {
         NodoInt aux = primero;
         while (aux!=null && aux.dato!=e) aux = aux.siguiente;
         if (aux!=null) throw new IllegalArgumentException("Ya está el elemento " + e);
         this.insertar(e);
      }
      ```

   2. (0.25 puntos) Si este método se invocara en un método `main`, ¿sería obligatorio escribir la llamada al método dentro de una instrucción `try/catch`, o en su defecto modificar la cabecera de `main`? Razonar la respuesta.

      Solución: No, porque no lanza ninguna excepción *checked*.

4. 2.5 puntos Dada una cierta clase `Examen`, se desea escribir en ella un método para obtener el valor mayor de una `ColaIntEnla c`. Al acabar la operación, `c` debe ser igual que lo era inicialmente. Si la cola está vacía debe lanzar la excepción `NoSuchElementException` con el mensaje `"Cola vacía: máximo no definido"`. Por ejemplo:

   - Si `c` es `<- 4 -2 9 8 <-`, debe devolver `9` y dejar `c` como `<- 4 -2 9 8 <-`.
   - Si `c` es `<- -2 <-`, debe devolver `-2` y dejar `c` como `<- -2 <-`.
   - Si `c` es `<- <-`, `c` debe seguir siendo vacía, y se lanza la excepción.

Solución:

```
public static int maximo(ColaIntEnla c) {
   int n = c.talla();
   if (n==0) throw new NoSuchElementException("Cola vacía: máximo no definido");
   int e = c.desencolar();
   int maximo = e;
   c.encolar(e); n--;
   while (n>0) {
        e = c.desencolar();
        if (e>maximo) maximo = e;
        c.encolar(e); n--;
    }
   return maximo;
}
```