Como devolver más de un resultado en un método Java

Motivación Algunas veces es necesario que un método devuelva más de un resultado pero nos encontramos con la restricción que Java nos permite un único resultado para cada método.

Una forma de hacer esto es definir una clase con uno o más atributos cuyo tipo de dato depende de lo que el método necesita informar, con los métodos constructor, get y set correspondientes. Una instancia de dicha clase se pasa como parámetro al método en cuestión. Este objeto puede ser modificado durante la ejecución del método. Al finalizar el método el objeto modificado es visible desde el método llamador. Así, este mecanismo funciona a modo de pasaje de parámetros de entrada salida.

Ejemplo Como ejemplo se presenta el método *eliminar* de árbol binario de búsqueda, que se ha implementado utilizando un método recursivo privado *eliminarAux* donde el hijo le devuelve al padre el nodo que queda como raíz de dicho subarbol después de la eliminación del elemento solicitado. Dicho resultado se utiliza para actualizar el subárbol correspondiente (ver el uso en las llamadas recursivas en el método *eliminarAux*).

Creación de la Clase Resultado (Ver Algoritmo 0.1) Dado que el método *eliminarAux* recorre parte del árbol hasta llegar al elemento que se desea eliminar o encuentra una rama null, se desea devolver un valor lógico indicando si la eliminación fue exitosa o no. Para ello se definió una clase *Resultado* para mantener y pasar el valor *boolean* necesario. Dicha clase puede crearse dentro del paquete "utiles" (donde puede ser usada por varias clases). Si se desea sólo para un caso particular, puede crearse de manera interna a la clase ArbolBinBusq.

Como se usa (Ver Algoritmo 0.2) El método *eliminarAux* debe recibir una instancia ya creada e inicializada de la clase Resultado (que en el ejemplo se ha llamado *exito*). Dicha instancia ha de ser creada e inicializada en el método *eliminar* que es el que necesita recibir el valor resultado. Luego, si el método *eliminarAux* encuentra un error (cuando no se encuentra el elemento, es decir, el Nodo n es nulo en alguna llamada recursiva), modifica el valor lógico de la instancia *exito* en *false*. En caso que el elemento se encuentre, el estado de la instancia nunca cambia. Al retornar el control al método *eliminar* utiliza el valor de la instancia *exito* y lo retorna al programa llamador.

Algoritmo 0.1 Clase Resultado para mantener y devolver un valor boolean

```
class Resultado {

    private boolean valor;

    public Resultado(boolean ini) {

        valor = ini;

    }

    public boolean get() {

        return valor;

    }

    public void set(boolean nuevo) {

        valor = nuevo;

    }

}
```

Algoritmo 0.2 Como se usa la clase Resultado para devolver un mensaje de error en *eliminarAux*

```
public boolean eliminar(String element) {
    Resultado exito = new Resultado(true);
    this.raiz = eliminarAux(this.raiz, element, exito);
    return exito.get();
}

private NodoArb eliminarAux(NodoArb n, String e, Resultado exito) {

    NodoArb salida = n;

    if (n == null) {
        exito.set(false);
    } else {
        if (n.getElem().equals(e)) {
            // n tiene el elemento buscado
            // caso 1: n es hoja
            if (n.getIzquierdo() == null && n.getDerecho() == null) {
                salida = null;
            } else {
                // ... caso 2: n tiene un solo hijo
                // ... caso 3: n tiene dos hijos
            }
        } else {
            // no es el elemento buscado
            if (n.getElem().compareTo(e) > 0) {
                // si el elem buscado es menor que n, baja al HI de n
                n.setIzquierdo(eliminarAux(n.getIzquierdo(), e, exito));

            } else {
                // el elem buscado es mayor que n, baja al HD de n
                n.setDerecho(eliminarAux(n.getDerecho(), e, exito));
            }
        }
    }

    return salida;
}
```