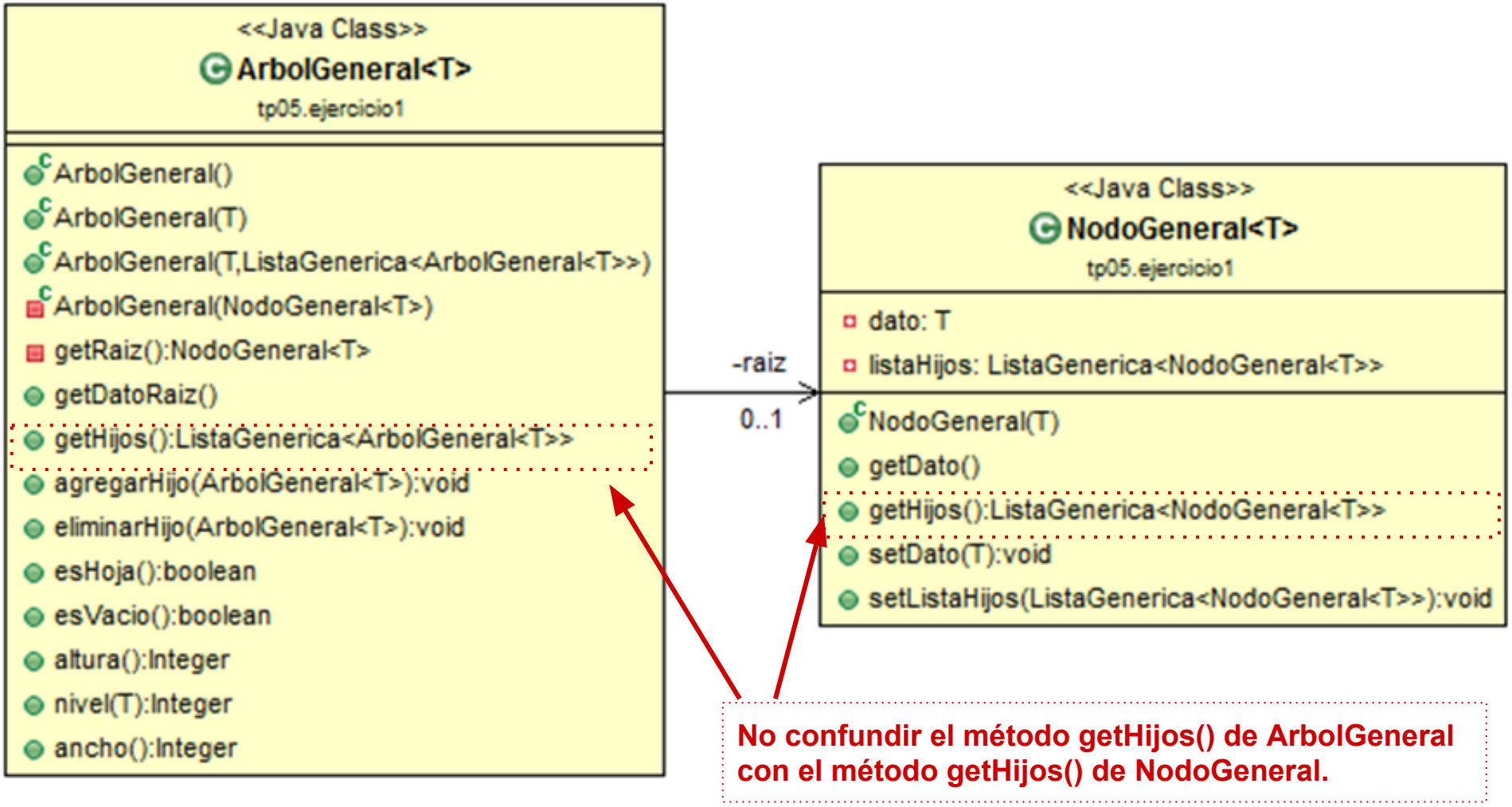
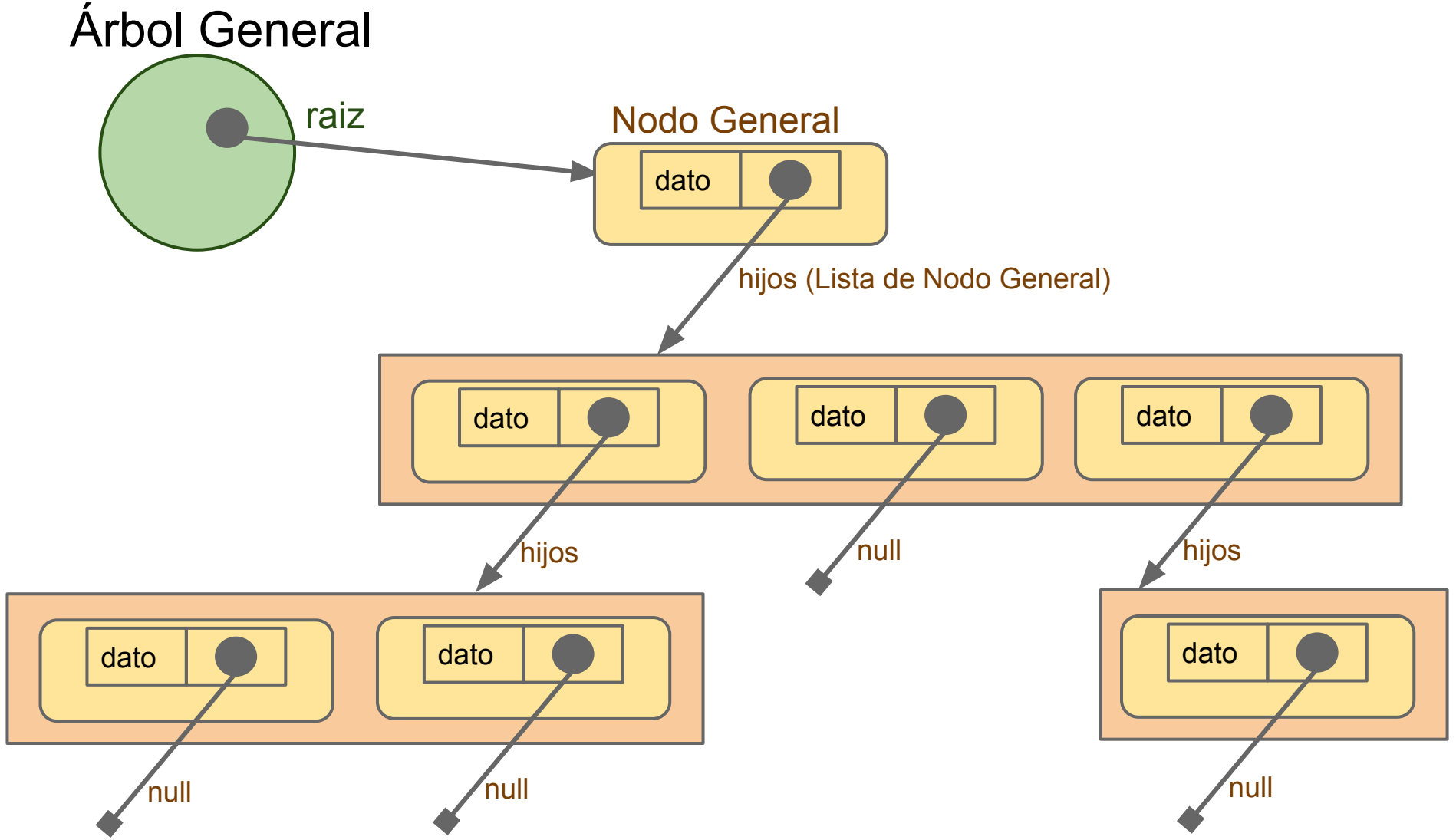
Algoritmos y Estructuras de Datos 2015

Árboles Generales

Estructura (modelo de clases)



Estructura (relación entre objetos)



Implementación de Árbol General

- **getHijos()** devuelve una **lista de árboles**. Siendo **R** la raíz del árbol original, cada árbol de la **lista de árboles** tendrá como su raíz a cada uno de los hijos (nodos generales) de **R**.

```
public ListaGenerica<ArbolGeneral<T>> getHijos() {
    ListaGenerica<ArbolGeneral<T>> lista =
                    new ListaEnlazadaGenerica<ArbolGeneral<T>>();
    ListaGenerica<NodoGeneral<T>> hijos =
                    (ListaGenerica<NodoGeneral<T>>) this.getRaiz().getHijos();
    hijos.comenzar();
    while (!hijos.fin()) {
        lista.agregarFinal(new ArbolGeneral<T>(hijos.proximo()));

    }
    return lista;
}
```

Implementación de Árbol General

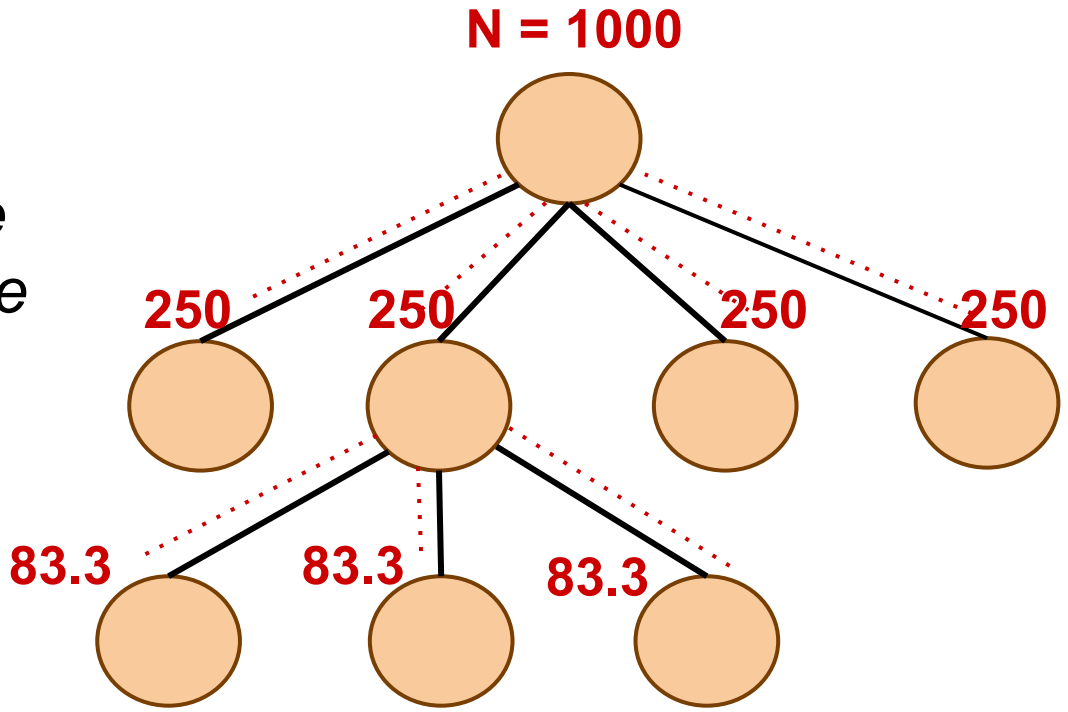
- Si el Árbol General sabe devolver sus hijos como una lista de Árboles Generales, puede enviarle mensajes a sus hijos árboles (recursión).

Pregunta: *Para agregar un hijo al árbol, ¿Puedo usar el método getHijos() que devuelve una lista y agregarle un nuevo hijo?*

NO! *Porque se agregará en la nueva lista,* ***¡usar el método agregarHijo(ArbolGeneral<T>)!****.*

Red de agua potable

Comienza en un caño maestro y el mismo se va dividiendo sucesivamente hasta llegar a cada una de las casas.



A resolver:

Considerando que ingresan **n** litros por el caño maestro, calcule cuál es el **mínimo caudal** que recibe una hoja.

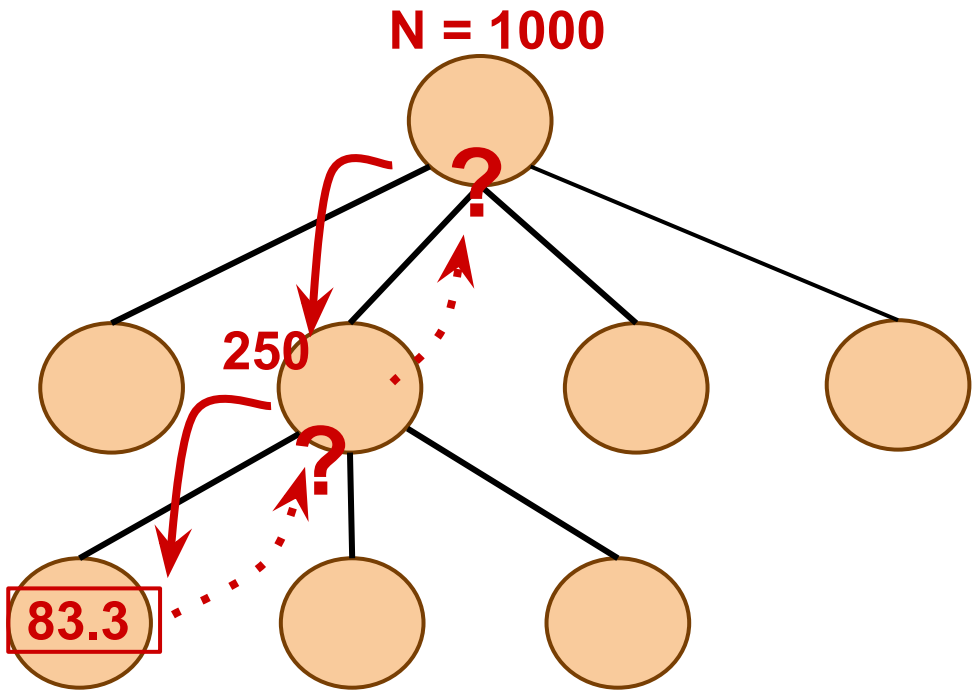
Se conoce la "estructura de la red" y "cuánto ingresa".

Clase RedDeAguaPotable

```java
package tp05.usos;

import tp03.ejercicio1.ListaGenerica;
import tp05.ejercicio1.ArbolGeneral;

public class RedDeAguaPotable {

    ArbolGeneral<Double> arbolRed;

    public RedDeAguaPotable(ArbolGeneral<Double> arbolRed) {
        super();
        this.arbolRed = arbolRed;
    }

    ..........
```

Solución 1

El padre (raiz), usa recursión para averiguar -a través de sus hijos- el caudal mínimo; los hijos devuelven información y en base a esa información el padre se queda con el mínimo.



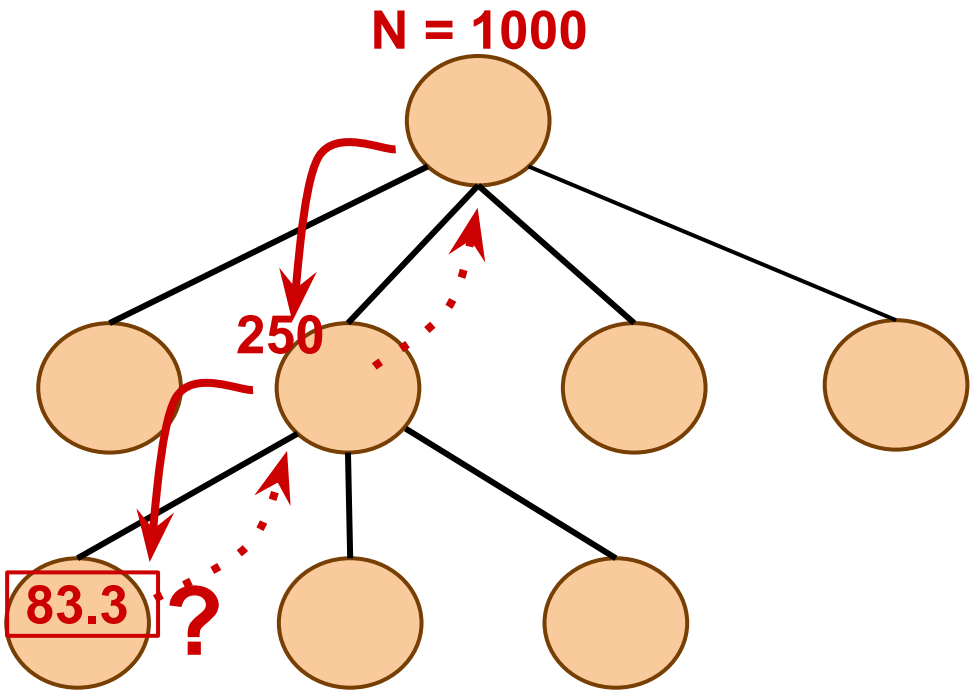
¡El valor mínimo lo voy a encontrar en una hoja!
La hoja informa a su padre para que decida...

Solución 1

```
public double minimoCaudalv1(double valorCaudal) {
    if (arbolRed.esHoja()) {
        return valorCaudal;
    } else {
        ListaGenerica<ArbolGeneral<Double>> hijos = arbolRed.getHijos();
        double min = valorCaudal;
        double calculado = 0;
        hijos.comenzar();
        while (!hijos.fin()) {
            ArbolGeneral<Double> unHijo = hijos.proximo();
            RedDeAguaPotable subRed= new RedDeAguaPotable(unHijo);
            calculado = subRed.minimoCaudalv1(valorCaudal / hijos.tamanio());
            if (calculado < min)
                min = calculado;
        }
        return min;

    }
```

Solución 2

El padre (raiz), usa recursión para averiguar -a través de sus hijos- el caudal mínimo. Los hijos que son hojas verifican el caudal mínimo.



¡El valor mínimo lo voy a encontrar en una hoja!
La hoja decide si su valor es el mínimo...

Solución 2

```
public double minimoCaudalv2(double valorCaudal){
     double[] resultado = {valorCaudal};
     minimoCaudalv2(valorCaudal, resultado);
     return resultado[0];
}
```

Solución 2

```
private void minimoCaudalv2(double valorCaudal, double[] resultMin) {
    if (arbolRed.esHoja()) {
        if (resultMin[0] > valorCaudal)
            resultMin[0] = valorCaudal;
    } else {
        ListaGenerica<ArbolGeneral<Double>> hijos = arbolRed.getHijos();
        hijos.comenzar();
        while (!hijos.fin()) {
            ArbolGeneral<Double> unHijo = hijos.proximo();
            RedDeAguaPotable subRed= new RedDeAguaPotable(unHijo);
            subRed.minimoCaudalv2(valorCaudal / hijos.tamanio(), resultMin);
        }
    }
}
```