Algoritmos y Estructuras de Datos 2015

Árboles Binarios

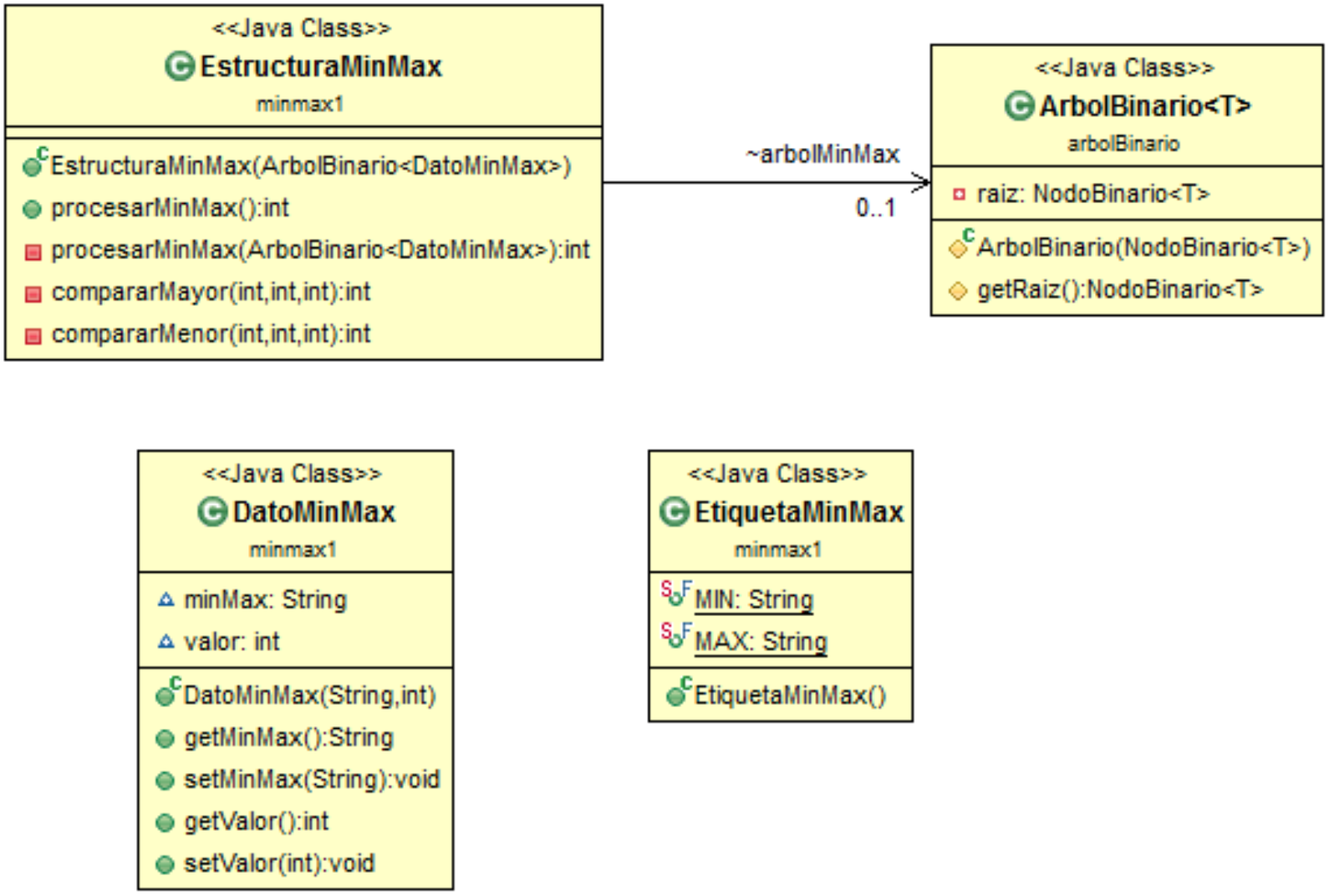
Ejercicio Árbol MinMax

Modelizar e implementar en Java la siguiente situación:
Considere un árbol binario no vacío con dos tipos de nodos: ***nodos MIN*** *y* ***nodos MAX****. Cada nodo tiene un* ***valor entero*** *asociado.*

Se puede definir el valor de un árbol de estas características de la siguiente manera:

- *Si la raíz es un nodo MIN, entonces el valor del árbol es igual al mínimo valor entre: (i) El entero almacenado en la raíz. (ii) El valor correspondiente al subárbol izquierdo, si el mismo no es vacío. (iii) El valor correspondiente al subárbol derecho, si el mismo no es vacío.*
- *Si la raíz es un nodo MAX, entonces el valor del árbol es igual al máximo valor entre los valores nombrados anteriormente.*

Solución 1


<<Java Class>>
EstructuraMinMax
minmax1
EstructuraMinMax(ArbolBinario<DatoMinMax>)
procesarMinMax():int
procesarMinMax(ArbolBinario<DatoMinMax>):int
compararMayor(int,int,int):int
compararMenor(int,int,int):int
~arbolMinMax
0..1
<<Java Class>>
ArbolBinario<T>
arbolBinario
raiz: NodoBinario<T>
ArbolBinario(NodoBinario<T>)
getRaiz():NodoBinario<T>
<<Java Class>>
DatoMinMax
minmax1
minMax: String
valor: int
DatoMinMax(String,int)
getMinMax():String
setMinMax(String):void
getValor():int
setValor(int):void
<<Java Class>>
EtiquetaMinMax
minmax1
MIN: String
MAX: String
EtiquetaMinMax()

Solución 1

```java
package minmax1;

import arbolBinario.ArbolBinario;

public class EstructuraMinMax {
    ArbolBinario<DatoMinMax> arbolMinMax;

    public EstructuraMinMax(ArbolBinario<DatoMinMax> arbolMinMax) {
        super();
        this.arbolMinMax = arbolMinMax;
    }

    public int procesarMinMax() {
        return procesarMinMax(this.arbolMinMax);
    }
```

Solución 1

```
private int procesarMinMax(ArbolBinario<DatoMinMax> arbolMinMaxParam) {
    if (arbolMinMaxParam.esHoja())
        return arbolMinMaxParam.getDatoRaiz().getValor();
    else {
        int datoHI = 0;
        int datoHD = 0;
        int resultado = 0;
        if (arbolMinMaxParam.getDatoRaiz().getMinMax()
                .equals(EtiquetaMinMax.MIN)) {
            if (!arbolMinMaxParam.getHijoIzquierdo().esVacio())
                datoHI = procesarMinMax(arbolMinMaxParam.getHijoIzquierdo());
            else
                datoHI = Integer.MAX_VALUE;
            if (!arbolMinMaxParam.getHijoDerecho().esVacio())
                datoHD = procesarMinMax(arbolMinMaxParam.getHijoDerecho());
            else
                datoHD = Integer.MAX_VALUE;
            resultado = compararMenor(arbolMinMaxParam.getDatoRaiz()
                    .getValor(), datoHI, datoHD);
```

Solución 1

```
        } else if (arbolMinMaxParam.getDatoRaiz().getMinMax()
                .equals(EtiquetaMinMax.MAX)) {
            if (!arbolMinMaxParam.getHijoIzquierdo().esVacio())
                datoHI = procesarMinMax(arbolMinMaxParam.getHijoIzquierdo());
            else
                datoHI = Integer.MIN_VALUE;
            if (!arbolMinMaxParam.getHijoDerecho().esVacio())
                datoHD = procesarMinMax(arbolMinMaxParam.getHijoDerecho());
            else
                datoHD = Integer.MIN_VALUE;
            resultado = compararMayor(arbolMinMaxParam.getDatoRaiz()
                    .getValor(), datoHI, datoHD);
        }
        return resultado;
    }
}
```

Solución 1

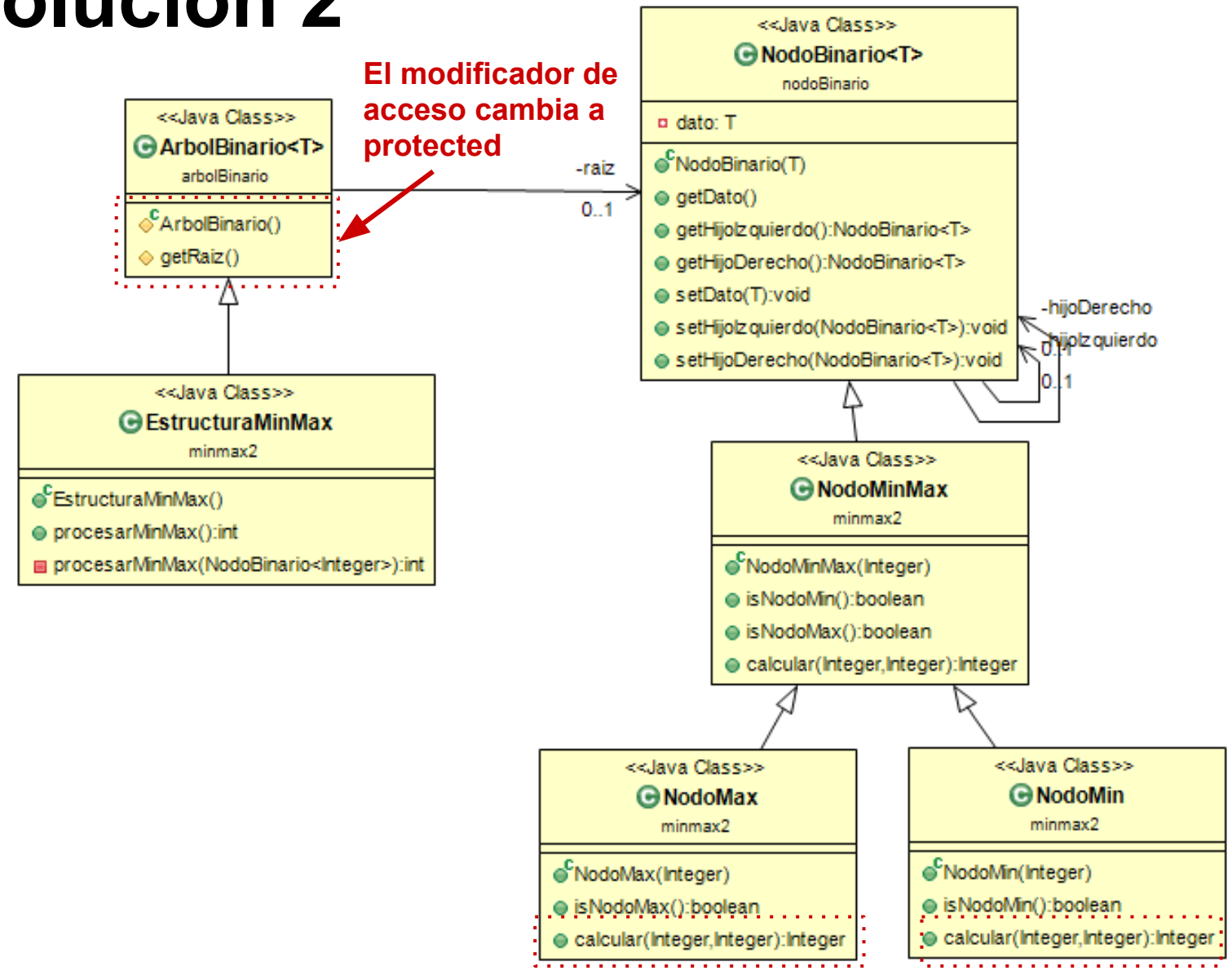
```
private int compararMenor(int valor, int datoHI, int datoHD) {
    if (datoHI < valor && datoHI < datoHD)
        return datoHI;
    if (datoHD < valor && datoHD < datoHI)
        return datoHD;

    return valor;
}

private int compararMayor(int valor, int datoHI, int datoHD) {
    if (datoHI > valor && datoHI > datoHD)
        return datoHI;
    if (datoHD > valor && datoHD > datoHI)
        return datoHD;

    return valor;
}
```

Solución 2


El modificador de acceso cambia a protected
<<Java Class>>
ArbolBinario<T>
arbolBinario
ArbolBinario()
getRaiz()
-raiz
0..1
<<Java Class>>
NodoBinario<T>
nodoBinario
dato: T
NodoBinario(T)
getDato()
getHijoIzquierdo():NodoBinario<T>
getHijoDerecho():NodoBinario<T>
setDato(T):void
setHijoIzquierdo(NodoBinario<T>):void
setHijoDerecho(NodoBinario<T>):void
-hijoDerecho
-hijoIzquierdo
0..1
0..1
<<Java Class>>
EstructuraMinMax
minmax2
EstructuraMinMax()
procesarMinMax():int
procesarMinMax(NodoBinario<Integer>):int
<<Java Class>>
NodoMinMax
minmax2
NodoMinMax(Integer)
isNodoMin():boolean
isNodoMax():boolean
calcular(Integer,Integer):Integer
<<Java Class>>
NodoMax
minmax2
NodoMax(Integer)
isNodoMax():boolean
calcular(Integer,Integer):Integer
<<Java Class>>
NodoMin
minmax2
NodoMin(Integer)
isNodoMin():boolean
calcular(Integer,Integer):Integer

Solución 2

```java
package minmax2;

import nodoBinario.NodoBinario;

public class NodoMinMax extends NodoBinario<Integer> {

    public NodoMinMax(Integer dato) {
        super(dato);
    }

    public boolean isNodoMin(){
        return false;
    }

    public boolean isNodoMax(){
        return false;
    }

    //este método sólo lo necesito en esta clase para aplicar polimorfismo
    //en realidad en nuestra solución no será invocado
    public Integer calcular(Integer dato1, Integer dato2) {
        return this.getDato();
    }
}
```

Solución 2

```
package minmax2;

public class NodoMin extends NodoMinMax {

    public NodoMin(Integer dato) {
        super(dato);
    }

    public boolean isNodoMin(){
        return true;
    }

    public Integer calcular(Integer dato1, Integer dato2) {
        if (dato1 < dato2 && dato1 < this.getDato())
            return dato1;
        if (dato2 < dato1 && dato2 < this.getDato())
            return dato2;
        return this.getDato();
    }
}
```

Solución 2

```
package minmax2;

public class NodoMax extends NodoMinMax {

    public NodoMax(Integer dato) {
        super(dato);
    }

    public boolean isNodoMax() {
        return true;
    }

    public Integer calcular(Integer dato1, Integer dato2) {
        if (dato1 > dato2 && dato1 > this.getDato())
            return dato1;
        if (dato2 > dato1 && dato2 > this.getDato())
            return dato2;
        return this.getDato();
    }
}
```

Solución 2

```java
package minmax2;

import nodoBinario.NodoBinario;
import arbolBinario.ArbolBinario;

public class EstructuraMinMax extends ArbolBinario<Integer> {

    public int procesarMinMax() {
        return procesarMinMax(this.getRaiz());
    }

    private int procesarMinMax(NodoBinario<Integer> raiz) {
        if (raiz.getHijoIzquierdo() == null && raiz.getHijoDerecho() == null)
            return this.getDatoRaiz();
        else {
            int datoHI = procesarMinMax(raiz.getHijoIzquierdo());
            int datoHD = procesarMinMax(raiz.getHijoDerecho());
            int resultado = ((NodoMinMax)raiz).calcular(datoHI, datoHD);
            return resultado;
        }
    }

}
```