UNLP. Facultad de Informática.
Algoritmos y Estructuras de Datos
Cursada 2018

Práctica 3
Árboles Binarios y ABB

Importante: Puede continuar trabajando en su proyecto AyED. El archivo zip descargado desde la página de la cátedra no es un proyecto eclipse, por lo tanto:

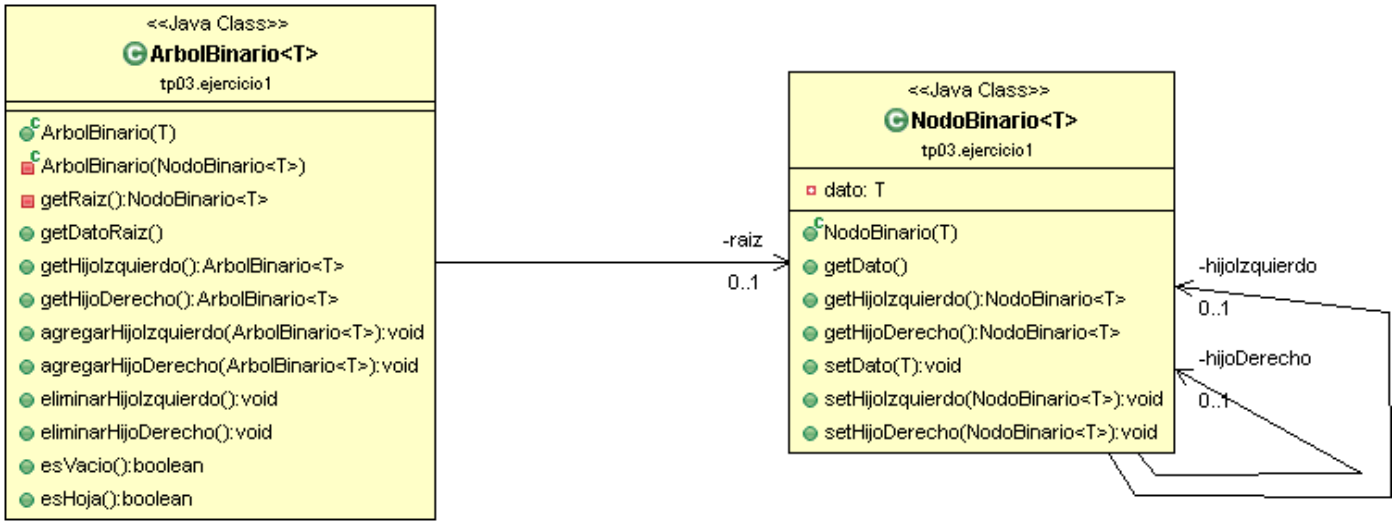
1. Descomprima el archivo zip.
2. Sobre la carpeta **src** de su proyecto AyED haga click con el botón derecho del mouse y seleccione la opción *Import* > *FileSystem*.
3. Haga click en "Browse", busque la carpeta descomprimida y seleccione la carpeta **src** (haga click para que aparezca el check seleccionado).
4. Haga click en el botón finalizar.

Objetivos

- Representar árboles binarios e implementar las operaciones de la abstracción
- Realizar distintos tipos de recorridos sobre árboles binarios
- Describir soluciones utilizando árboles binarios

Ejercicio 1

Considere la siguiente especificación de la clase Java **ArbolBinario** (con la representación hijo izquierdo e hijo derecho).



- El constructor **ArbolBinario(T dato)** inicializa un árbol que tiene como raíz un nodo binario. Este nodo binario tiene el dato pasado como parámetro y ambos hijos nulos.
- El constructor **ArbolBinario(NodoBinario<T> nodo)** inicializa un árbol donde el nodo pasado como parámetro es la raíz. (Notar que **NO** es un método público).

- El método **getRaiz():NodoBinario<T>**, retorna el nodo ubicado en la raíz del árbol. (Notar que **NO** es un método público).
- El método **getDatoRaiz():T**, retorna el dato almacenado en el NodoBinario raíz del árbol.
- Los métodos **getHijoIzquierdo():ArbolBinario<T>** y **getHijoDerecho():ArbolBinario<T>**, retornan los hijos izquierdo y derecho respectivamente de la raíz del árbol. Tenga en cuenta que los hijos izquierdo y derecho del NodoBinario raíz del árbol son nodos binarios y se deben devolver árboles binarios, por lo tanto se usa el constructor privado **ArbolBinario(NodoBinario<T> nodo)** para obtener el árbol binario correspondiente.
- El método **agregarHijoIzquierdo(ArbolBinario<T> unHijo)** y **agregarHijoDerecho(ArbolBinario<T> unHijo)** agrega un hijo como hijo izquierdo o derecho del árbol. Tenga presente que unHijo es un ArbolBinario y se debe enganchar un NodoBinario como hijo. Para ello se utiliza el método privado getRaiz.
- El método **eliminarHijoIzquierdo()** y **eliminarHijoDerecho()**, eliminan el hijo correspondiente al NodoBinario raíz del árbol receptor.
- El método **esVacio()** indica si el árbol está vacío y el método **esHoja()** indica si el árbol tiene un solo nodo.

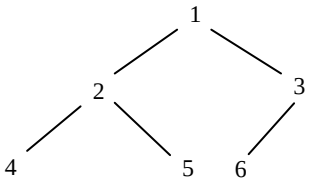
a) Analice la implementación en JAVA de las clases **ArbolBinario** y **NodoBinario** brindadas por la cátedra.

Ejercicio 2
Agregue a la clase **ArbolBinario** los siguientes métodos:

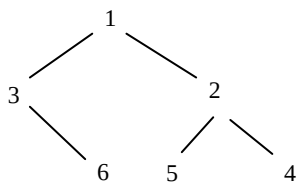
a) **contarHojas():int** Devuelve la cantidad de árbol/subárbol hojas del árbol receptor.

b) **espejo(): ArbolBinario<T>** Devuelve el árbol binario espejo del árbol receptor. Por ejemplo:

Por ejemplo, dado el árbol binario:



Retorna →



c) **entreNiveles(int n, m)** Imprime el recorrido por niveles de los elementos del árbol receptor entre los niveles n y m (ambos inclusive). (0≤n<m≤altura del árbol)

Ejercicio 3

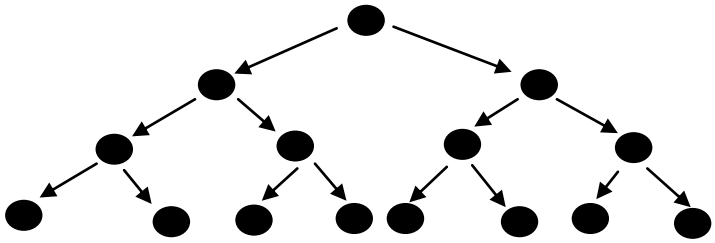
Defina una clase Java denominada **ContadorArbol** cuya función principal es proveer métodos de validación sobre árboles binarios de enteros. Para ello la clase tiene como variable de instancia un ArbolBinario<Integer>. Implemente en dicha clase un método denominado **numerosPares()** que devuelve en una estructura adecuada (sin ningún criterio de orden) todos los elementos pares del árbol (divisibles por 2). Defina la clase dentro del paquete **tp03.ejercicio3**

a) Implemente el método realizando un recorrido InOrden.

b) Implemente el método realizando un recorrido PostOrden.

Ejercicio 4

Una red binaria es una red que posee una topología de árbol binario lleno. Por ejemplo:



Los nodos que conforman una red binaria llena tiene la particularidad de que todos ellos conocen cuál es su retardo de reenvío. El retardo de reenvío se define como el período comprendido entre que un nodo recibe un mensaje y lo reenvía a sus dos hijos.
Su tarea es calcular el mayor retardo posible, en el camino que realiza un mensaje desde la raíz hasta llegar a las hojas en una red binaria llena.

a) Indique qué estrategia (recorrido en profundidad o por niveles) utilizará para resolver el problema.

b) Cree una clase Java llamada **RedBinariaLlena** (dentro del paquete **tp03.ejercicio4**) donde implementará lo solicitado en el método **retardoReenvio():int**

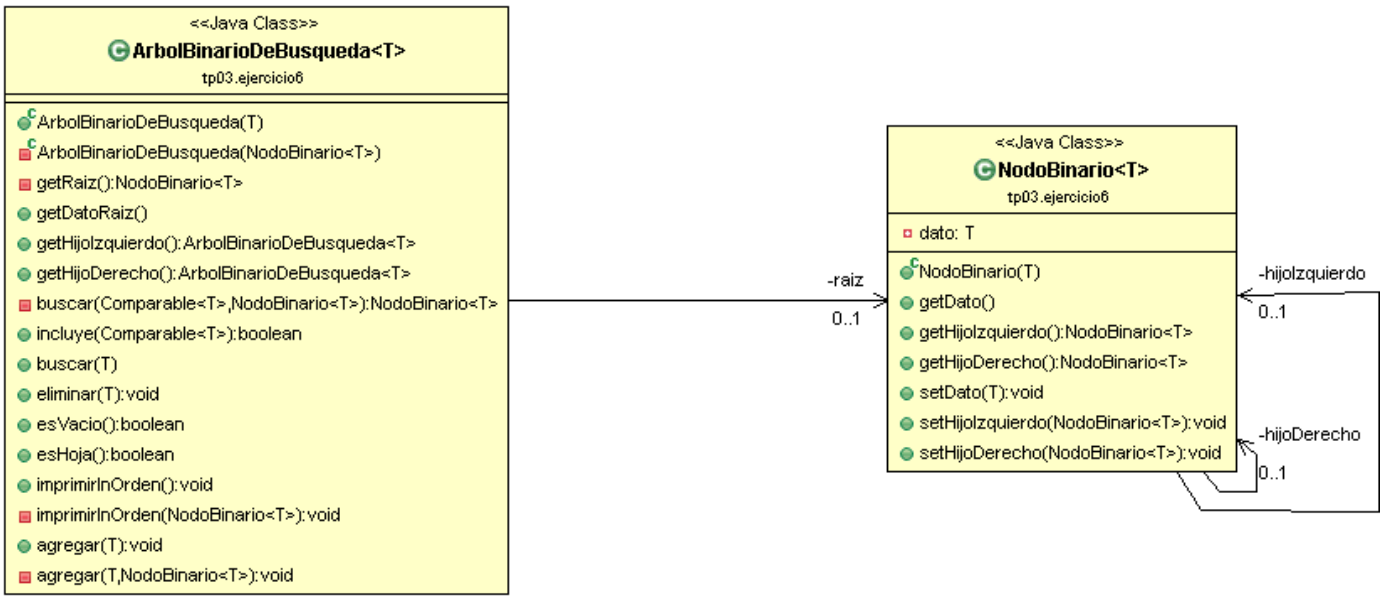
Ejercicio 5

Implemente una clase Java llamada **ProfundidadDeArbolBinario** que tiene como variable de instancia un árbol binario de números enteros y un método de instancia **sumaElementosProfundidad(int p):int** el cuál devuelve la suma de todos los nodos del árbol que se encuentren a la profundidad pasada como argumento.

Defina la clase dentro del paquete **tp03.ejercicio5**

Ejercicio 6

Considere la siguiente especificación de la clase **ArbolBinarioDeBusqueda** (con la representación hijo izquierdo e hijo derecho):



Aclaración:
En la imagen no aparece la definición de la variable de tipo en la clase ArbolBinarioDeBusqueda. La misma es **<T extends Comparable<T>>**, lo cual indica que la clase representada por la variable T implemente la interface Comparable.
Las clases que implementan la interface *java.lang.Comparable<T>* permiten que sus instancias se puedan comparar entre sí. Para lograr esto, deben implementar el método compareTo(T), el cual retorna el resultado de comparar el receptor del mensaje con el parámetro recibido. Este valor se codifica con un entero, el cual presenta la siguiente característica:

= 0: si el objeto receptor es igual al pasado en el argumento.
> 0: si el objeto receptor es mayor que el pasado como parámetro.
< 0: si el objeto receptor es menor que el pasado como parámetro.

La descripción de cada método es la siguiente:

El constructor **ArbolBinariodeBusqueda()** inicializa un árbol binario de búsqueda vacío con la raíz en null.

El constructor **ArbolBinariodeBusqueda(T dato)** inicializa un árbol que tiene como raíz un nodo binario de búsqueda. Este nodo tiene el dato pasado como parámetro y ambos hijos nulos.

El constructor **ArbolBinariodeBusqueda(NodoBinario<T> nodo)** inicializa un árbol donde el nodo pasado como parámetro es la raíz. Este método es privado y se podrá usar en la implementación de las operaciones sobre el árbol.

El método **getRaiz():NodoBinario <T>** retorna el nodo ubicado en la raíz del árbol.

El método **getDatoRaiz():T** retorna el dato almacenado en el NodoBinario raíz del árbol, sólo si el árbol no es vacío.

El método **esVacio(): boolean** indica si el árbol es vacío (no tiene dato cargado).

Los métodos **getHijoIzquierdo():ArbolBinariodeBusqueda<T>** y **getHijoDerecho():ArbolBinariodeBusqueda<T>** retornan los árboles hijos que se ubican a la izquierda y derecha del nodo raíz respectivamente. Están indefinidos para un árbol vacío.

El método **incluye (T dato)** retorna un valor booleano indicando si el dato recibido está incluido en el árbol.

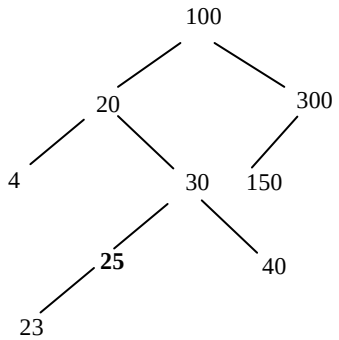
El método **buscar (T dato):T** retorna el valor almacenado en el árbol que es igual al dato recibido.

El método **agregar (T dato)** agrega el dato indicado al árbol. En caso de encontrar un elemento igual dentro del árbol, reemplaza el existente por el recibido.

a) Analice la implementación en JAVA de la clase **ArbolBinariodeBusqueda** brindada por la cátedra.

Nota: Tener presente que en ABB no se pueden almacenar cualquier objeto, ya que estos necesitan ser **comparables** para poder ordenarlos dentro de la estructura.

Ejercicio 7
Dado un **ArbolBinariodeBusqueda** de números enteros positivos y mayores a cero, se quiere obtener el camino recorrido hasta llegar a un determinado valor. En caso de ir por la rama izquierda del árbol, el valor a almacenar en el camino será el valor negativo del mismo. Es decir, en el siguiente ejemplo, si queremos encontrar el valor 25, el resultado será una lista con los valores: 100, -20, 30.



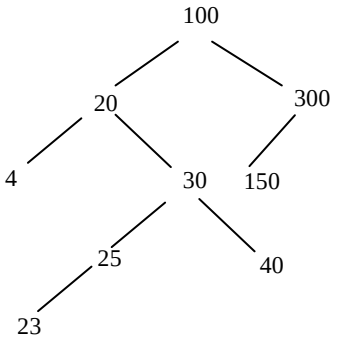
Note que el nodo con valor 20 en el resultado aparece en la lista resultante como -20 debido a que se llegó a este nodo a través del hijo izquierdo del nodo con valor 100.

La firma del método a implementar es:

ListaGenerica<Integer> caminoRecorrido (ArbolBinariodeBusqueda<Integer>)

Ejercicio 8

a. Implemente un método, que dado como parámetros un Árbol Binario de Búsqueda y un valor, devuelve el valor inmediato mayor. Es decir, dado el siguiente árbol:



- Si el valor a buscar es 30, el inmediato mayor es 40
- Si el valor a buscar es 25, el inmediato mayor es 30
- Si el valor a buscar es 300, el inmediato mayor no existe (puede devolver -1)

b**.** Plantee una solución alternativa.