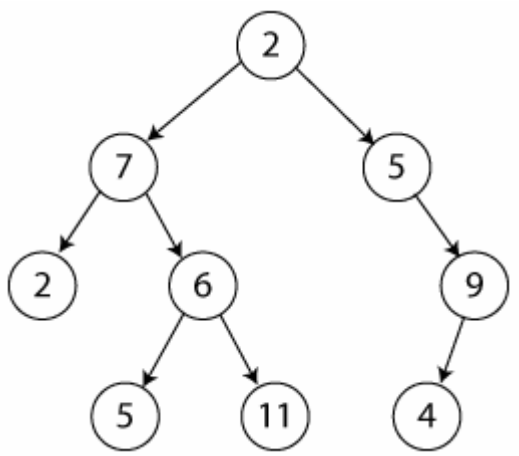
Actividad recursividad

Actividad evaluable sobre recursividad

Existe una estructura en programación llamada arbol binario. Un árbol binario tiene una estructura similar a la siguiente:



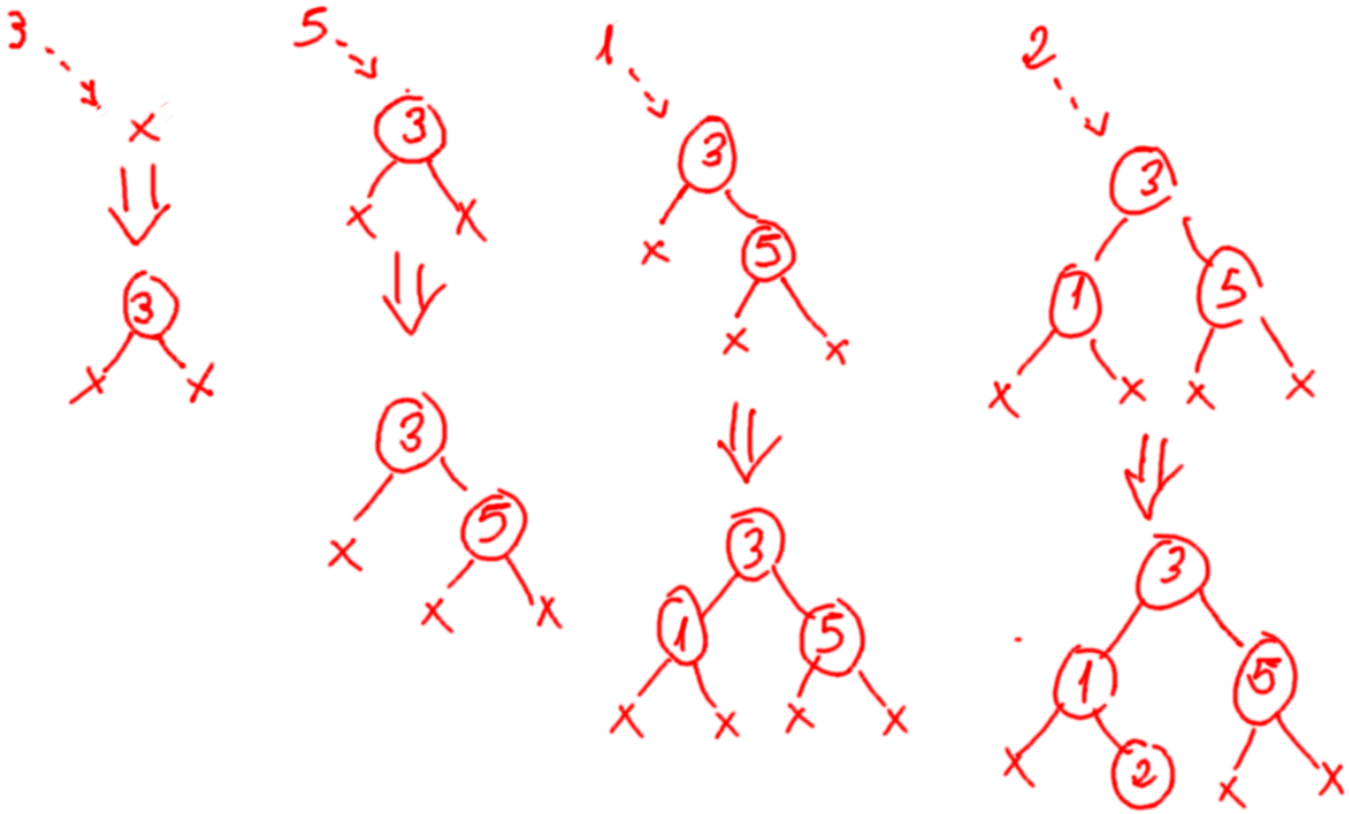
Cada uno de los elementos del árbol se denomina *nodo*. Cada nodo, contiene referencias a otros dos elementos, llamados *nodo hijo izquierdo* y *nodo hijo derecho*. El nodo principal se llama *nodo raíz*

```
class Nodo{
    private int valor;
    private Nodo hijoIzquierdo;
    private Nodo hijoDerecho;

    // Métodos
}
```

Los **árboles binarios de búsqueda**, son un tipo de árbol binario que cumplen un conjunto de reglas:

- Al insertar un nuevo valor *v*, se compara con el valor del nodo *raíz*:
    - Si el nodo raíz es nulo, se crea un nuevo nod raíz con el valor
    - Si *v* es menor que el valor del nodo raíz, se intenta insertar en el subárbol izquierdo.
    - En otro caso, se intenta insertar en el subárbol derecho.
- Para cada nodo, se vuelve a comparar del mismo modo. Esta operación se repite hasta encontrar un nodo que tiene un hijo vacío dónde insertar el nuevo nodo.



Una clase *Nodo* tendrá una estructura similar a la siguiente:

```
class Nodo{
    int valor;
    Nodo hijoIzquierdo;
    Nodo hijoDerecho;

    public Nodo(int valor){
        this.valor = valor;
        hijoIzquierdo = null;
        hijoDerecho = null;
    }

    public void setValor(int valor){
        this.valor = valor;
    }

    public void setHijoIzquierdo(Nodo nodo){
        this.hijoHizquierdo = nodo;
    }

    public void setHijoDerecho(Nodo nodo){
        this.hijoDerecho = nodo;
    }
```

```
    public void addValor(int valor){
        // Método recursivo:
        // Si valor < this.valor
        //       añadir en el subárbol izquierdo*
        // En caso contrario
        //       añadir en el subárbol derecho
        //
        // * Un subárbol es un nodo que apunta a otros.
        // ** El caso base es un subárbol nulo.
    }

    public boolean inArbol(int valor){
        // Método recursivo:
        //
        // Un valor está en el árbol si está
        // en el nodo actual o bien en alguno de sus hijos.
        // El caso base es:
        //      - Valor encontrado -> true
        //  - Nodo nulo -> false
    }

    public String recorridoEnOrden(){
        // La cadena devuelta es la concatenación en orden
        // de las siguientes cadenas:
        //
        //      - La cadena correspondiente al hijo izquierdo
        //      - El valor del nodo
        //      - La cadena correspondiente al hijo derecho.
        //
        //      El caso base es:
        //       - Un nodo hijo vacío -> ""
    }
}
```

Para crear un árbolo binario, necesitamos un primer nodo, que enlace a los demás. A este primer nodo raíz, podemos llamarle Árbol:

```
    Nodo árbol = new Nodo(valor);
```

1. Escribe el método *addValor*, para que el nuevo nodo se inserte correctamente en el árbol.
2. Escribe el método *inArbol* que comprueba si un cierto valor está presente en el árbol.
3. Escribe el método "recorridoEnOrden" que realiza el recorrido en orden de un árbol binario.