链滴

动手写一颗二叉树

作者：JackHoo
原文链接：https://ld246.com/article/1510301481372
来源网站：链滴

二叉树的遍历，查找，删除等等

说到二叉树一般都跟递归有比较大的关系

数据结构如下：

```
package data_structrue;

/**
 * Created by hushangjie on 2017/11/10.
*/
 public class BinaryTreeNode {
    private int data;
    private BinaryTreeNode leftChild;
    private BinaryTreeNode rightChild;

    public void setData(int data) {
        this.data = data;
    }

    public void setLeftChild(BinaryTreeNode leftChild) {
        this.leftChild = leftChild;
    }

    public void setRightChild(BinaryTreeNode rightChild) {
        this.rightChild = rightChild;
    }

    public int getData() {

        return data;
    }

    public BinaryTreeNode getLeftChild() {
        return leftChild;
    }

    public BinaryTreeNode getRightChild() {
        return rightChild;

    }
}

package data_structrue;

/**
 * Created by hushangjie on 2017/11/10.
 */
public class BinaryTree {
    private BinaryTreeNode root;

    public BinaryTree() {
    }
```

```
  public BinaryTree(BinaryTreeNode root) {
      this.root = root;
  }

  public void setRoot(BinaryTreeNode root) {
      this.root = root;
  }

  public BinaryTreeNode getRoot() {

      return root;
  }

  //clear
public void clear(BinaryTreeNode node) {
      if (node != null) {
          clear(node.getLeftChild());
          clear(node.getRightChild());
          node = null;
      }
  }

  //clear root
public void clear() {
      clear(root);
  }

  //
public boolean isEmpty() {
      return root == null;
  }

  //求二叉树的高度
public int height() {
      return height(root);
  }

  public int height(BinaryTreeNode node) {
      if (node == null) {
          return 0;
      } else {
          int l = height(node.getLeftChild());
          int r = height(node.getRightChild());
          return l < r ? r + 1 : l + 1;
      }
  }

  //获取二叉树的总节点数
public int size() {
      return size(root);
  }

  public int size(BinaryTreeNode node) {
      if (node == null) {
```

```
            return 0;
        } else {
            return 1 + size(node.getLeftChild()) + size(node.getRightChild());
        }
    }

    //先根遍历
    public void preOrder(BinaryTreeNode node){
        if (node != null) {
            System.out.println(node.getData());
            preOrder(node.getLeftChild());
            preOrder(node.getRightChild());
        }
    }

    //中跟遍历
    public void inOrder(BinaryTreeNode node) {
        if (node != null) {
            inOrder(node.getLeftChild());
            System.out.println(node.getData());
            inOrder(node.getRightChild());
        }
    }
    //后根遍历
    public void postOrder(BinaryTreeNode node){
        if (node != null) {
            preOrder(node.getLeftChild());
            preOrder(node.getRightChild());
            System.out.println(node.getData());
        }
    }

}
```