```
Object subclass: #BinaryTree
    instanceVariableNames: 'value left right'
    classVariableNames: ''
    poolDictionaries: ''
    category: nil !

!BinaryTree methodsFor: 'initialization'!

" purpose: construct a binary tree holding a single value"
" input: anInteger - the value at the (root of the) new binary tree"
initialize: anInteger
    value := anInteger.
    left := nil.
    right := nil.
!!

!BinaryTree methodsFor: 'maintaining'!

value
    ^value
!

" purpose: insert anInteger into this binary tree"
" input: anInteger - the value to insert"
" return: nothing - updates the tree"
insert: anInteger
    (anInteger < self value)
    ifTrue:
    [
        (left = nil)
        ifTrue:
        [
            left := BinaryTree new initialize: anInteger
        ]
        ifFalse:
        [
            left insert: anInteger
        ]
    ]
    ifFalse:
    [
        (right = nil)
        ifTrue:
        [
            right := BinaryTree new initialize: anInteger
        ]
        ifFalse:
        [
            right insert: anInteger
        ]
    ]

!!
```

```
!BinaryTree methodsFor: 'printing'!

" purpose: print the value"
" input: aStream - stream to output to"
" return: nothing"
printOn: aStream
    value printOn: aStream.
    ' ' printOn: aStream
!

" purpose: inorder traversal of this binary tree"
" input: nothing"
" return: nothing"
inorder
    (left ~= nil)
    ifTrue:
    [
        left inorder
    ].
    self printOn: stdout.
    (right ~= nil)
    ifTrue:
    [
        right inorder
    ]
!

" purpose: preorder traversal of this tree"
" input: nothing"
" return: nothing"
preorder
    self printOn: stdout.
    (left ~= nil)
    ifTrue:
    [
        left preorder
    ].
    (right ~= nil)
    ifTrue:
    [
        right preorder
    ]
!

" purpose: postorder traversal of this tree"
" input: nothing"
" return: nothing"
postorder
    (left ~= nil)
    ifTrue:
    [
        left postorder
    ].
    (right ~= nil)
    ifTrue:
    [
        right postorder
    ].
    self printOn: stdout

!!
```

```
"Main - for testing"
    | t |
    t := BinaryTree new initialize: 7.
    t insert: 1.
    t insert: 9.
    t insert: 0.
    t insert: 3.
    t insert: 8.
    t insert: 10.
    t insert: 2.
    t insert: 5.
    t insert: 4.
    t insert: 6.

    'inorder print:' printNl.
    t inorder.
    ' ' printNl.

    'preorder print:' printNl.
    t preorder.
    ' ' printNl.

    'postorder print:' printNl.
    t postorder.
    ' ' printNl.

!
```