

Tree Memory Diagram

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```

Stack		Heap
Name	Value	

in/out

- Build a tree
- Visit every node with a post-order traversal

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("life", null, null)  
    root.left = new BTNode("Scala", null, null)  
    root.right = new BTNode("for", null, null)  
    root.right.left = new BTNode("coding", null, null)  
    traversal(root)  
}
```

life

in/out

- Create the root node with the value "life"

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null

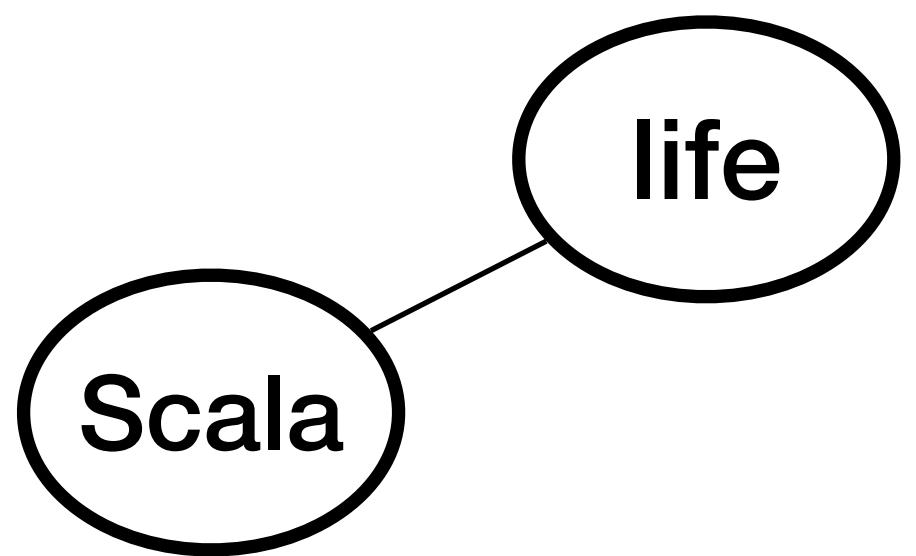
BTNode

BTNode	
value	"life"
left	null
right	0x350

Heap

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("life", null, null)  
    root.left = new BTNode("Scala", null, null)  
    root.right = new BTNode("for", null, null)  
    root.right.left = new BTNode("coding", null, null)  
    traversal(root)  
}
```



in/out

- Set the left child of the root to a node with "Scala"

Stack

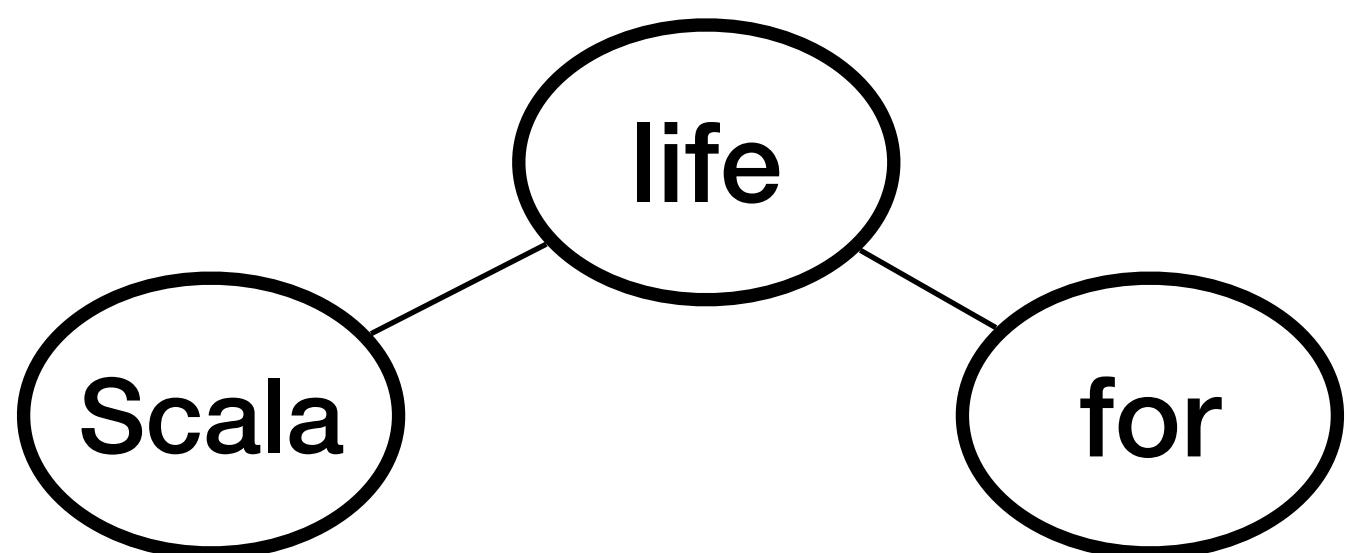
Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null

Heap

BTNode	
value	"life"
left	null 0x200
right	null
0x350	
BTNode	
value	"Scala"
left	null
right	null
0x200	

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {  
    if (node != null) {  
        traversal(node.left)  
        traversal(node.right)  
        print(node.value + " ")  
    }  
}  
  
def main(args: Array[String]): Unit = {  
    val root = new BTNode("life", null, null)  
    root.left = new BTNode("Scala", null, null)  
    root.right = new BTNode("for", null, null)  
    root.right.left = new BTNode("coding", null, null)  
    traversal(root)  
}
```



in/out

- Set the right child of the root to a node with "for"

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null

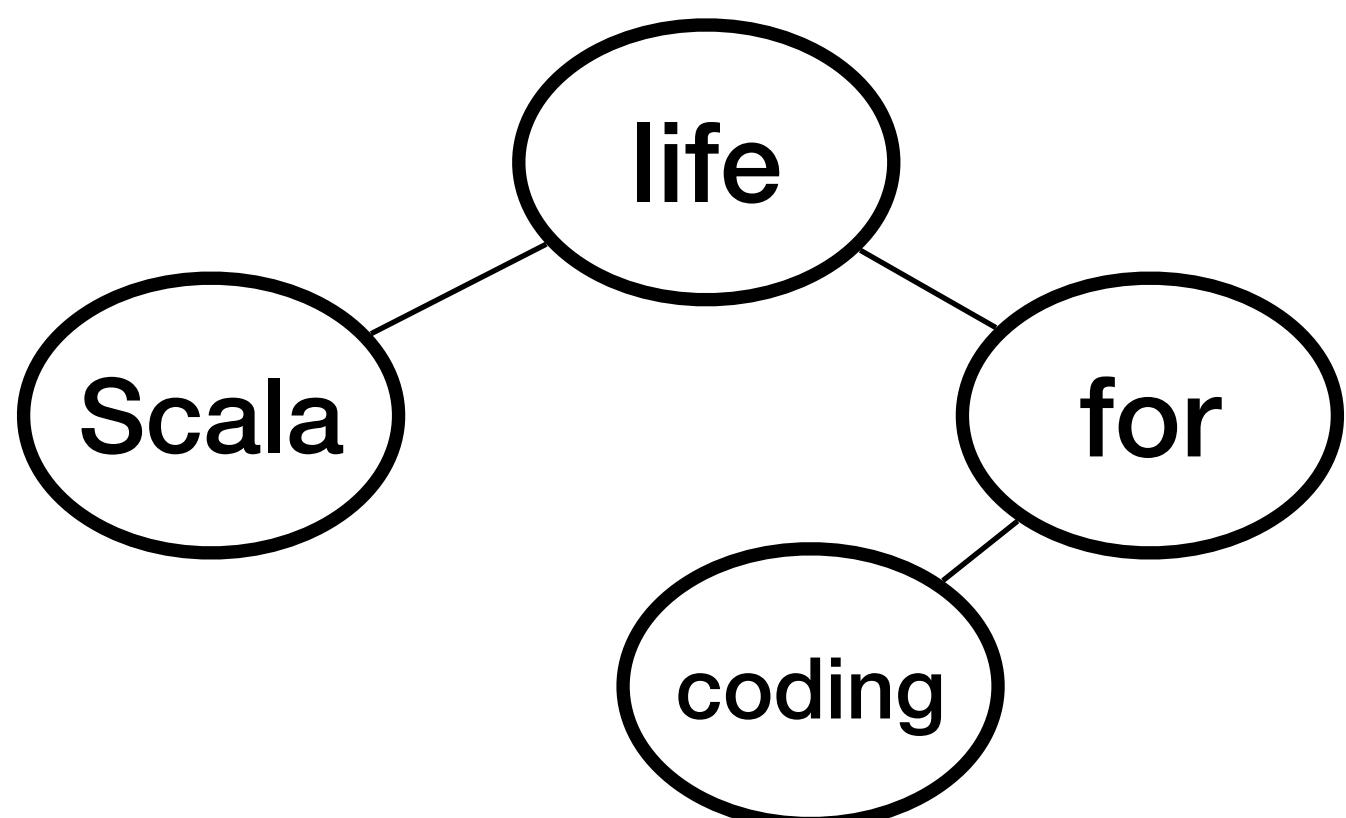
Heap

BTNode	
value	"life"
left	0x200
right	0x480
value	"Scala"
left	0x200
right	0x480
value	"for"
left	0x200
right	0x480

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out

- Set the left child of the right child of the root to a node with "coding"

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
	0x480
	0x936
	0x936

Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480

BTNode	
value	"Scala"
left	null
right	null

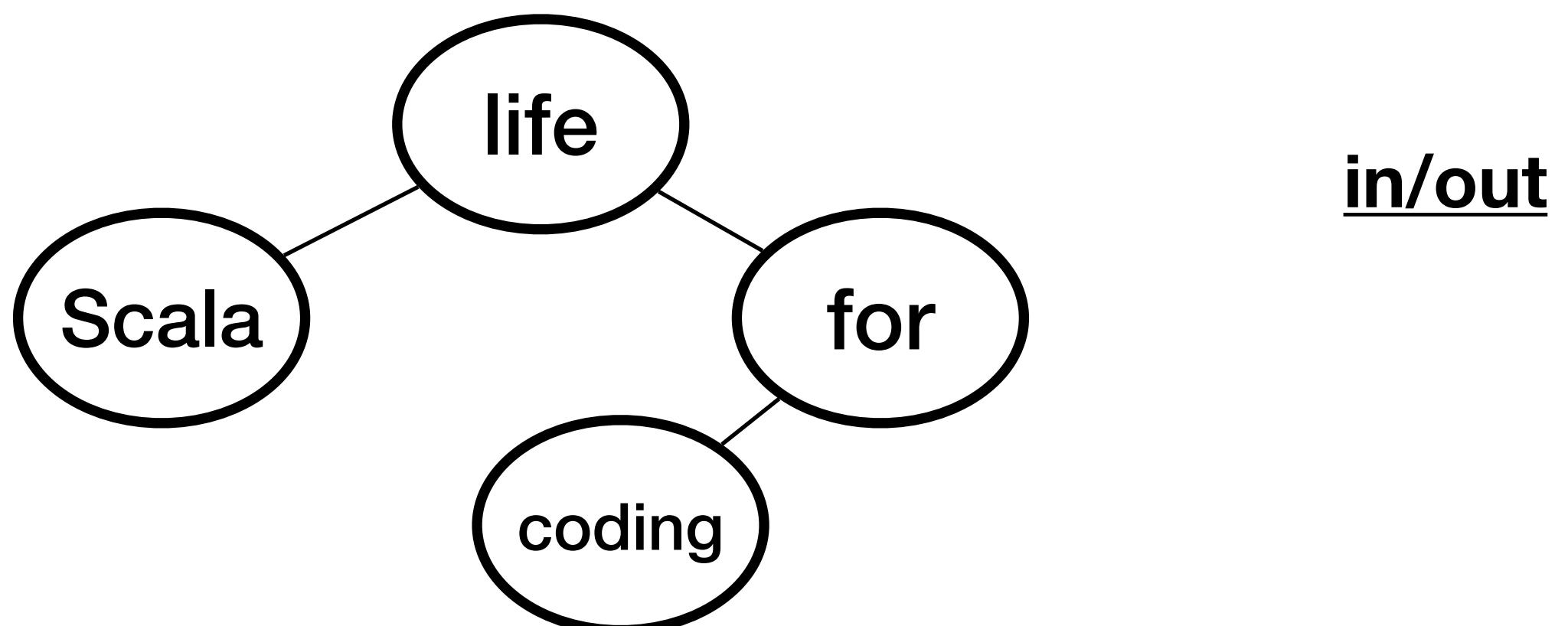
BTNode	
value	"for"
left	null 0x936
right	null

BTNode	
value	"coding"
left	null
right	null

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out

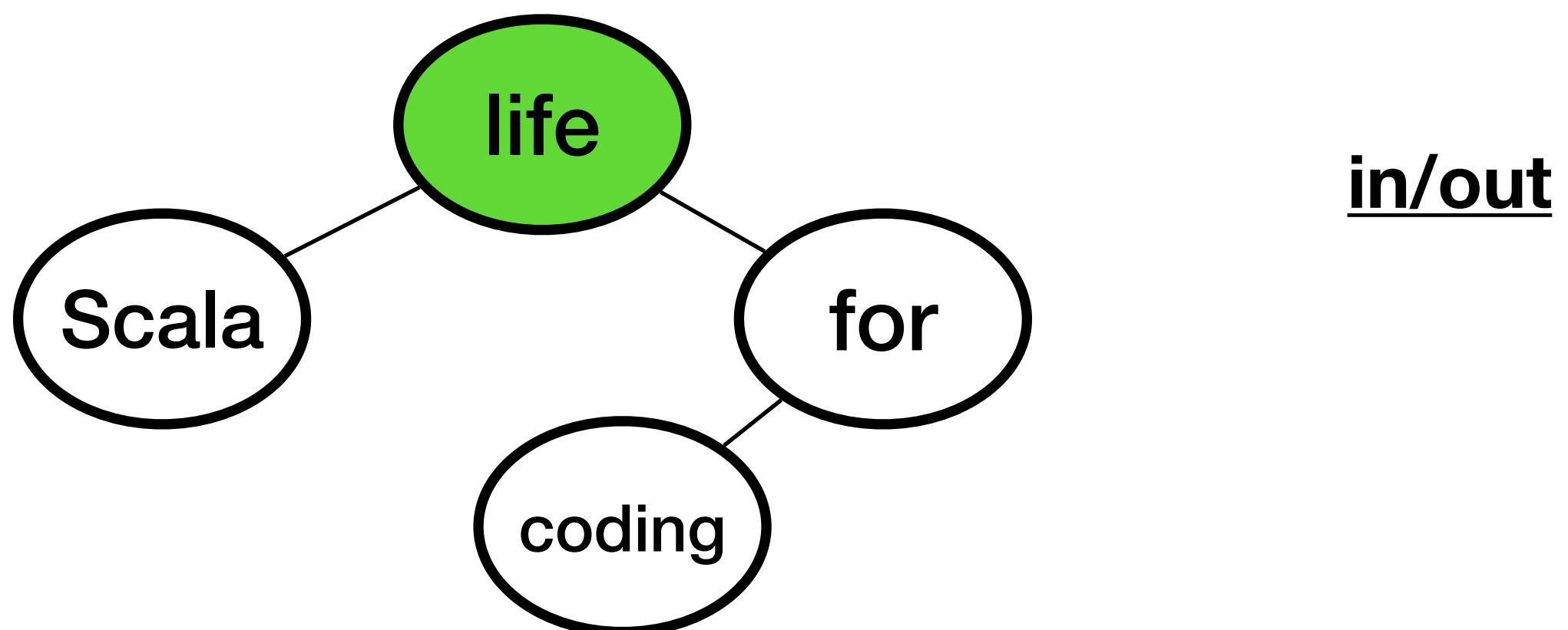
- Time to start the post-order traversal

Stack		Heap
Name	Value	
BTNode	root	0x350
	this	0x350
	value	"life"
	left	null
BTNode	right	null
	this	0x200
	value	"Scala"
	left	null
BTNode	right	null
	this	0x480
	value	"for"
	left	null
BTNode	right	null
	this	0x936
	value	"coding"
	left	null
BTNode	right	null
	value	"for"
	left	0x936
	right	null
BTNode	value	"coding"
	left	null
	right	null
	0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Initial call of traversal is on the root node (0x350)

Stack

Name	Value
------	-------

root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350

Heap

BTNode

value	"life"
left	null 0x200
right	null 0x480

BTNode

value	"Scala"
left	null
right	null

0x200

BTNode

value	"for"
left	null 0x936
right	null

0x480

BTNode

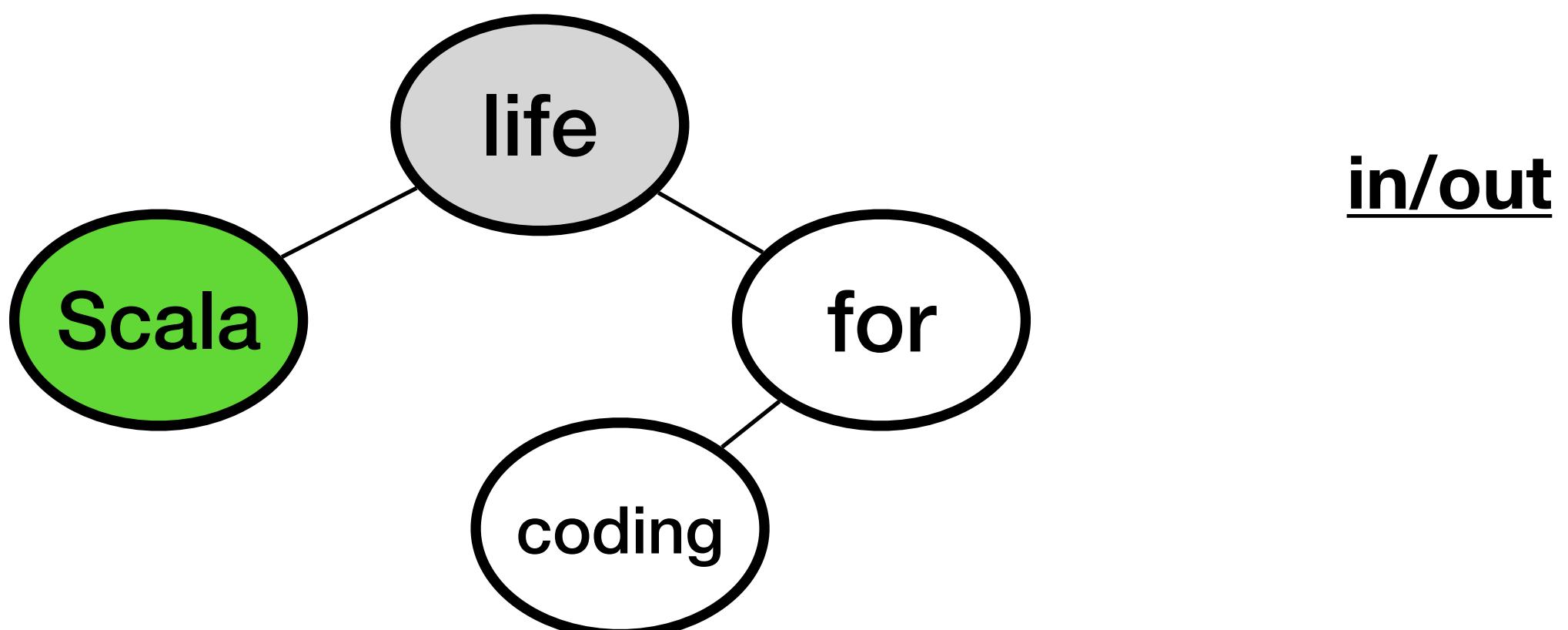
value	"coding"
left	null
right	null

0x936

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Make a recursive call on the left child first

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200

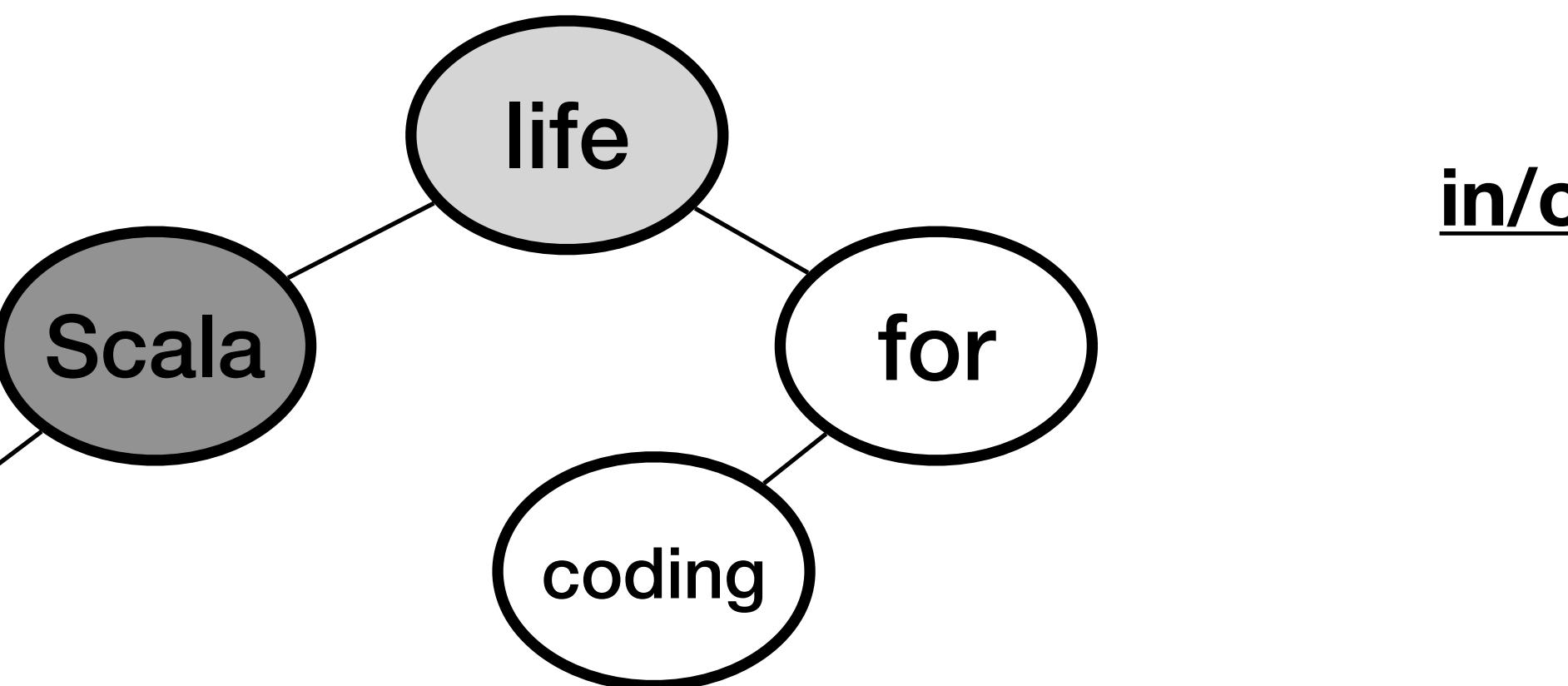
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



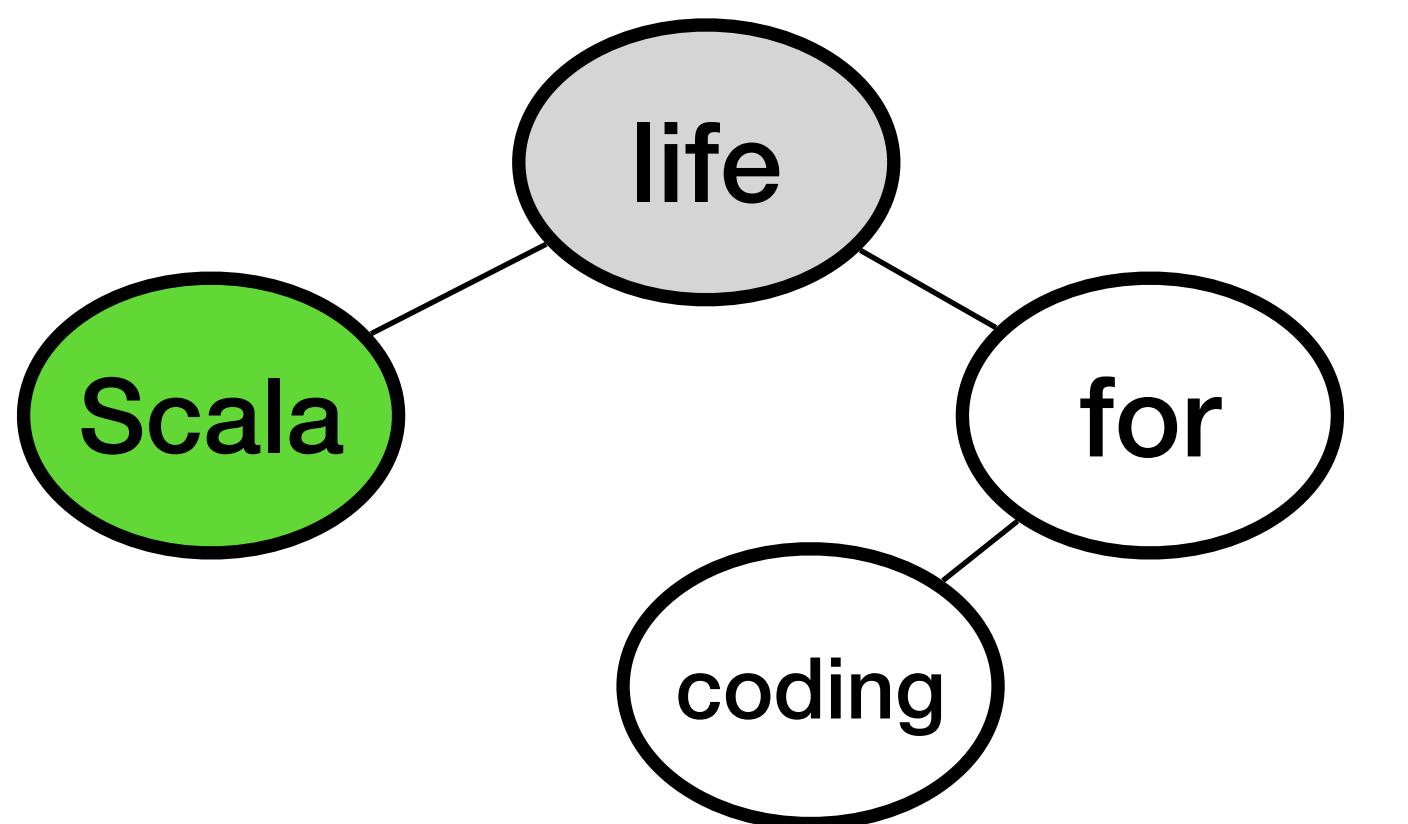
- Left node makes another recursive call on its left child

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "life"
value	"life"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"Scala"	value "Scala"
left	null	left 0x200
right	null	right 0x480
this	0x480	BTNode
value	"for"	value "for"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"coding"	value "coding"
left	null	left null
right	null	right null
node	0x350	traversal
node	0x200	traversal
node	null	traversal

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out

- A null node is our base case
- Do nothing and return to the previous frame on the stack

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null

Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480

BTNode	
value	"Scala"
left	null
right	null

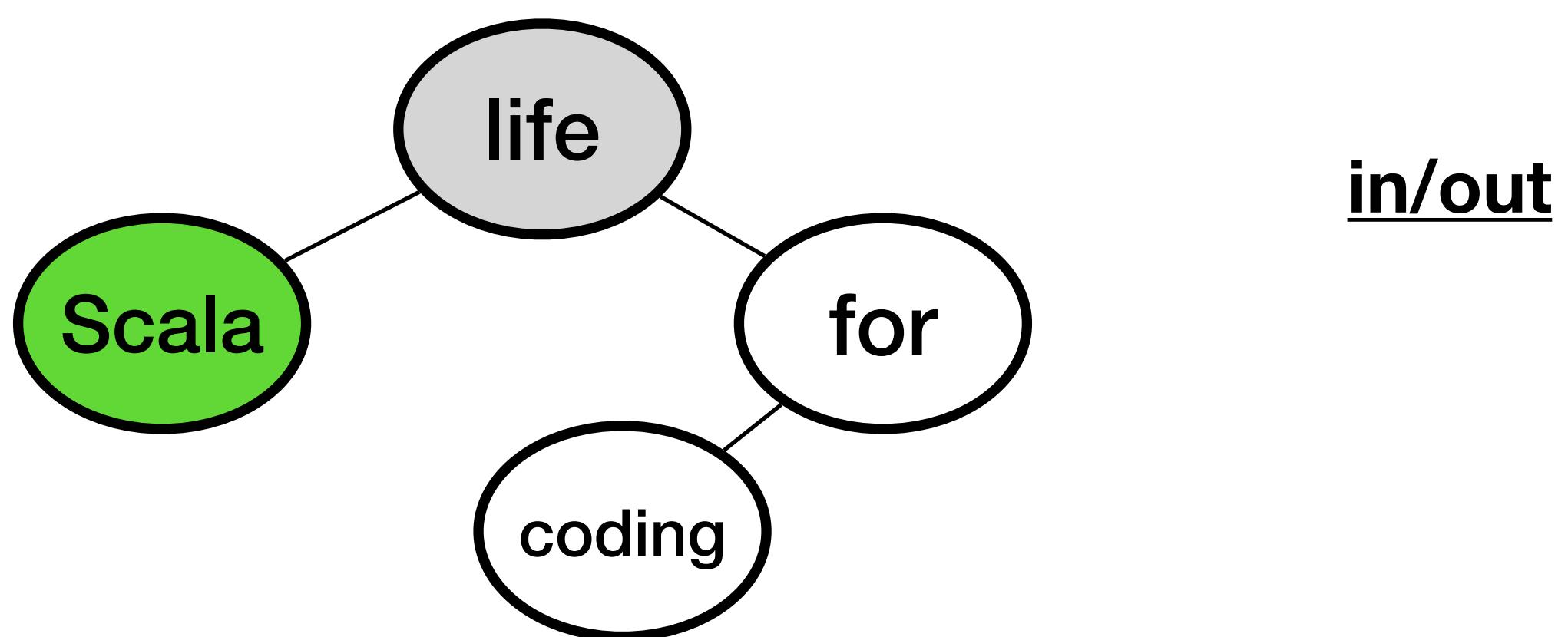
BTNode	
value	"for"
left	null 0x936
right	null

BTNode	
value	"coding"
left	null
right	null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



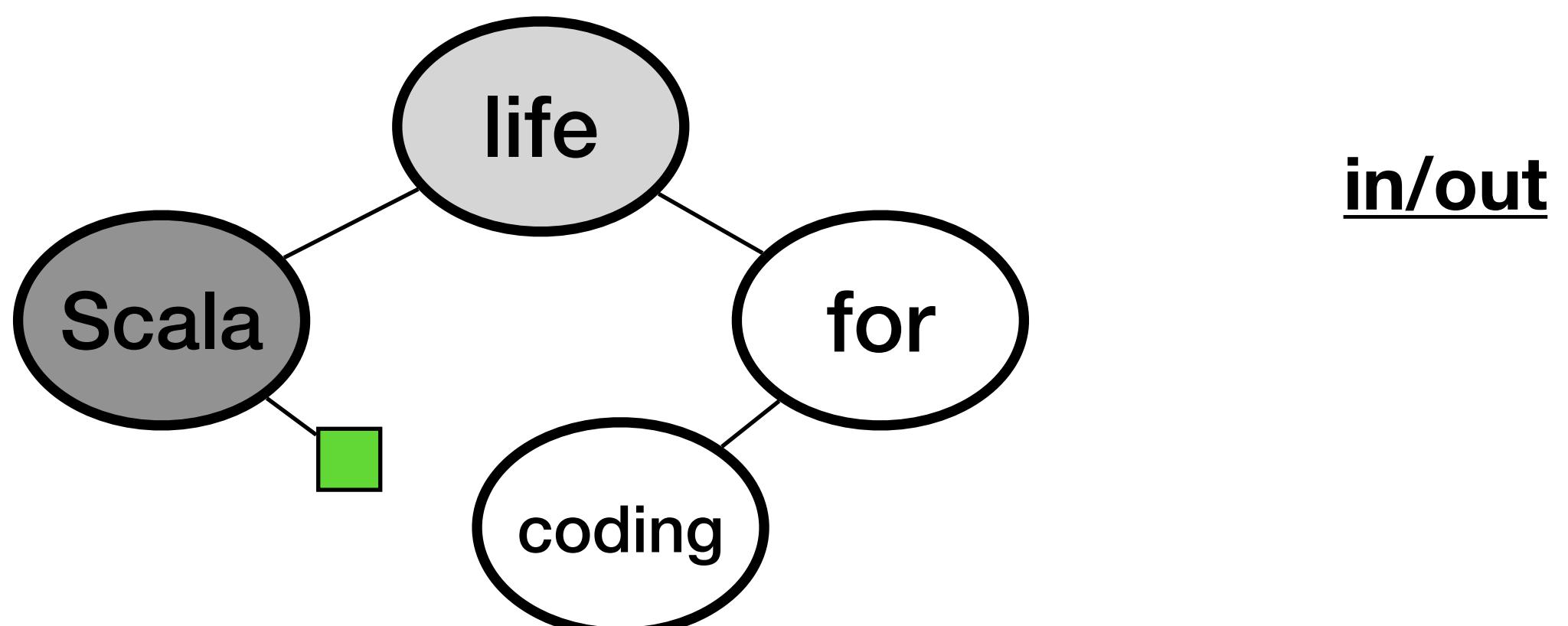
- This frame "remembers" that it made a left recursive call and needs to make the right recursive call

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value: "life"
value	"life"	left: null
left	null	right: null
right	null	
this	0x200	BTNode
value	"Scala"	value: "Scala"
left	null	left: null
right	null	right: null
this	0x480	BTNode
value	"for"	value: "for"
left	null	left: null
right	null	right: null
this	0x936	BTNode
value	"coding"	value: "coding"
left	null	left: null
right	null	right: null
node	0x350	BTNode
node	0x200	value: "Scala"
node	null	left: null
		right: null
traversal	0x480	BTNode
traversal	0x936	value: "for"
traversal	0x200	left: null
		right: null
		BTNode
		value: "coding"
		left: null
		right: null
		0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



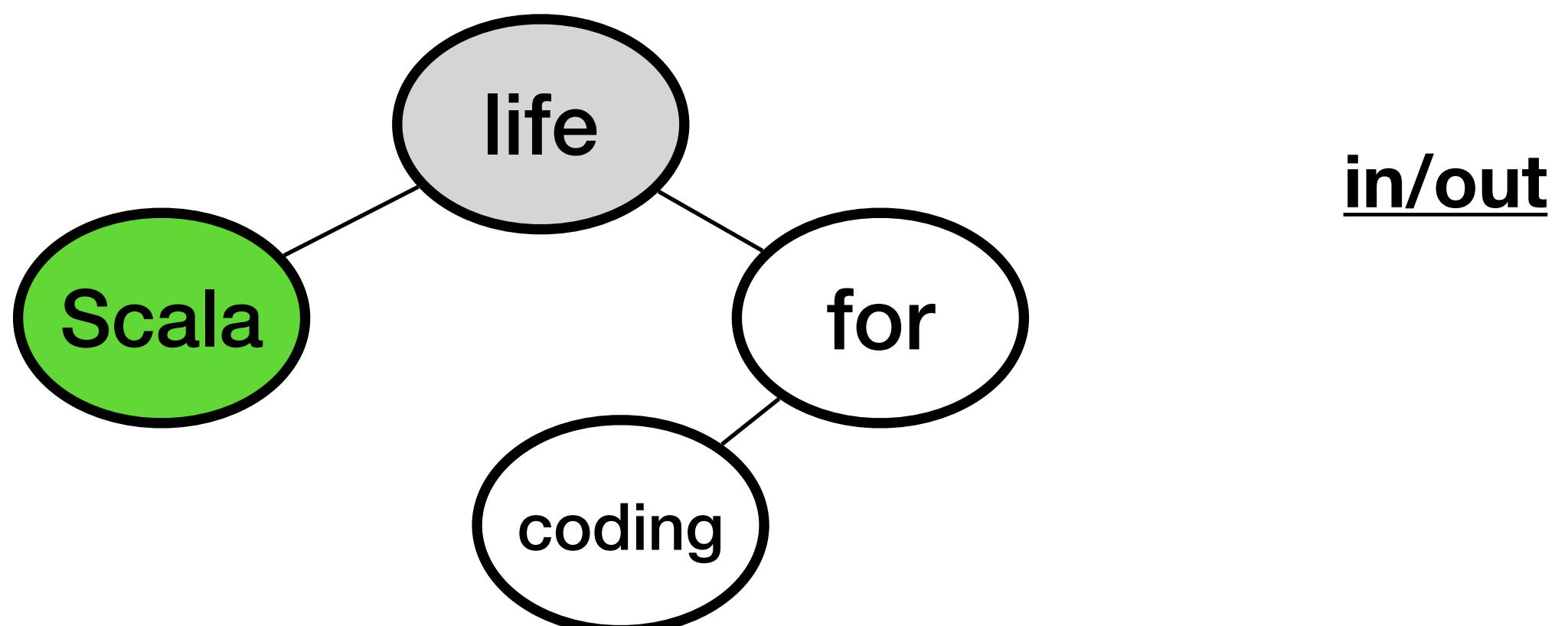
- Right recursive call is also for a null node

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "life"
value	"life"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"Scala"	value "Scala"
left	null	left 0x200
right	null	right 0x480
this	0x480	BTNode
value	"for"	value "for"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"coding"	value "coding"
left	null	left null
right	null	right null
node	0x350	traversal
node	0x200	traversal
node	null	traversal
node	null	traversal

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Return to the previous frame again
- "Remembers" that it made the right recursive call

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null

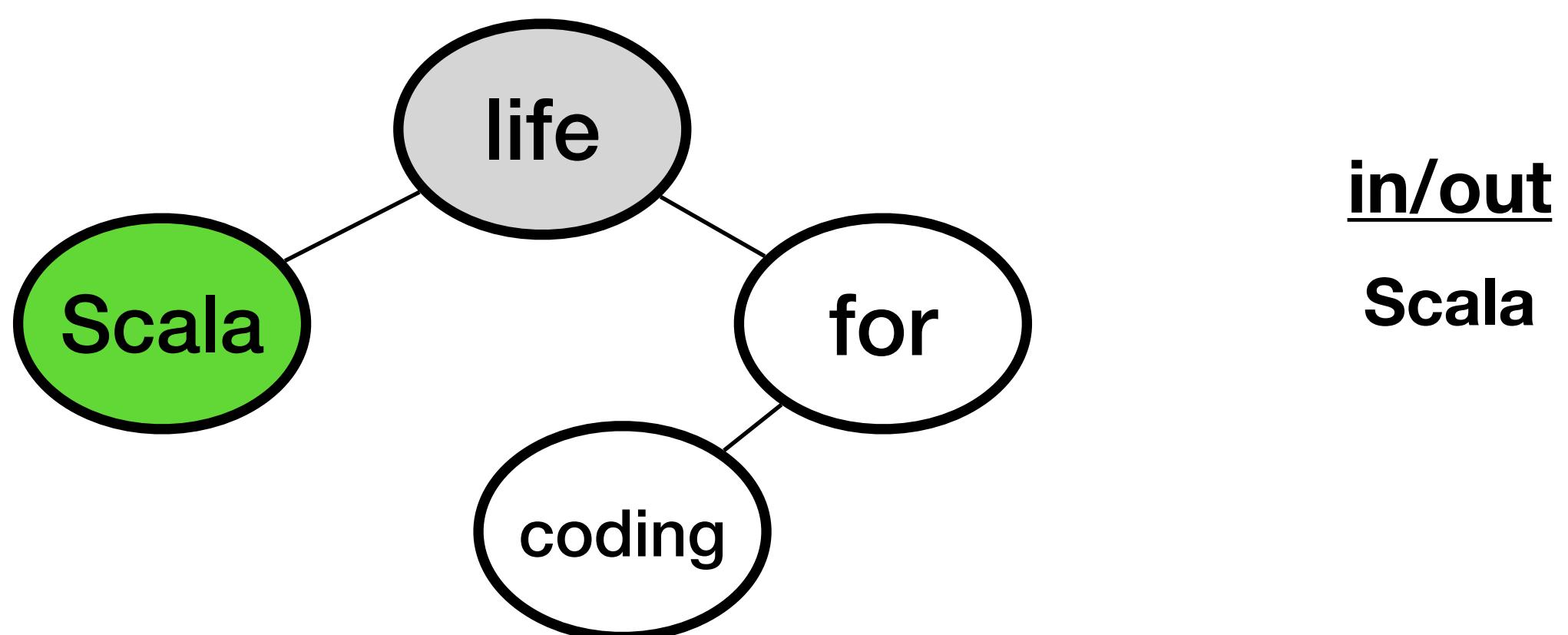
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
BTNode	0x200
value	"for"
left	null
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936
value	"for"
left	null 0x936
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936
value	"coding"
left	null
right	null

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Print Scala to the screen
- This stack frame is done and returns

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null

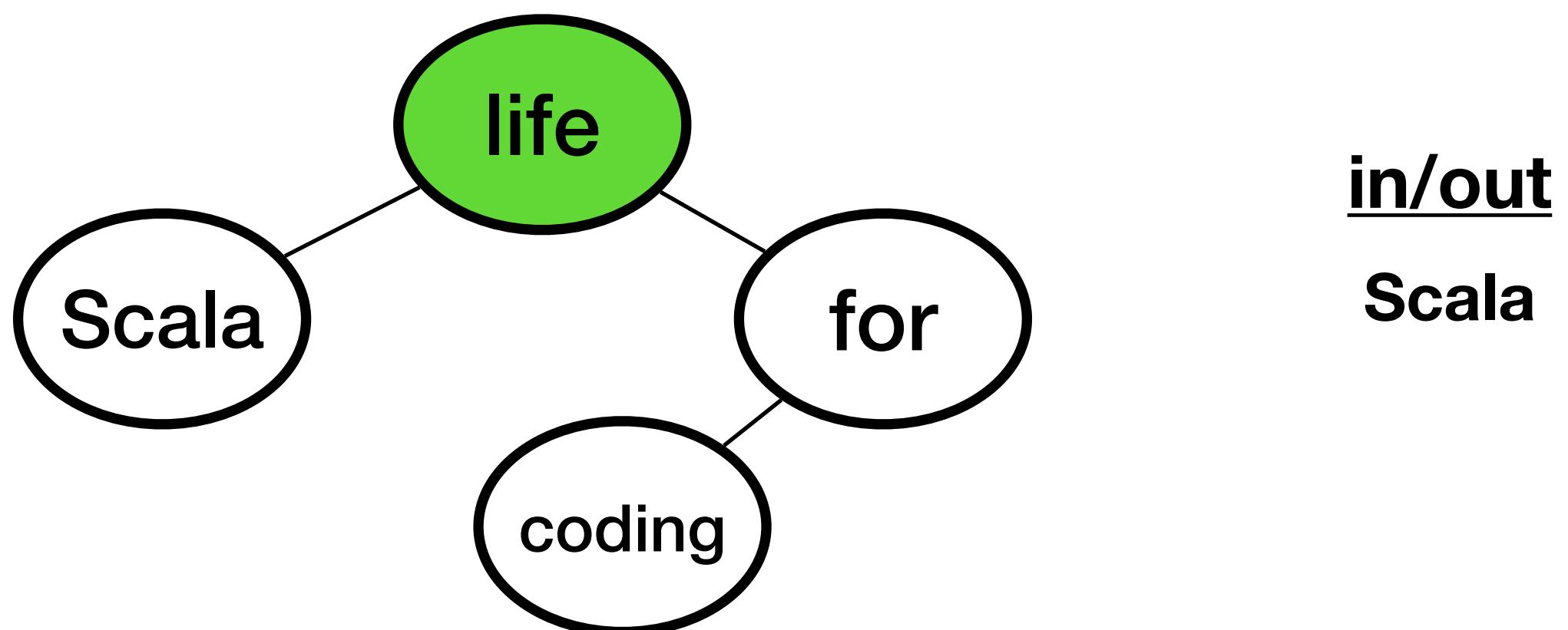
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
BTNode	0x200
value	"for"
left	null
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936
value	"for"
left	null 0x936
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936
value	"coding"
left	null
right	null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



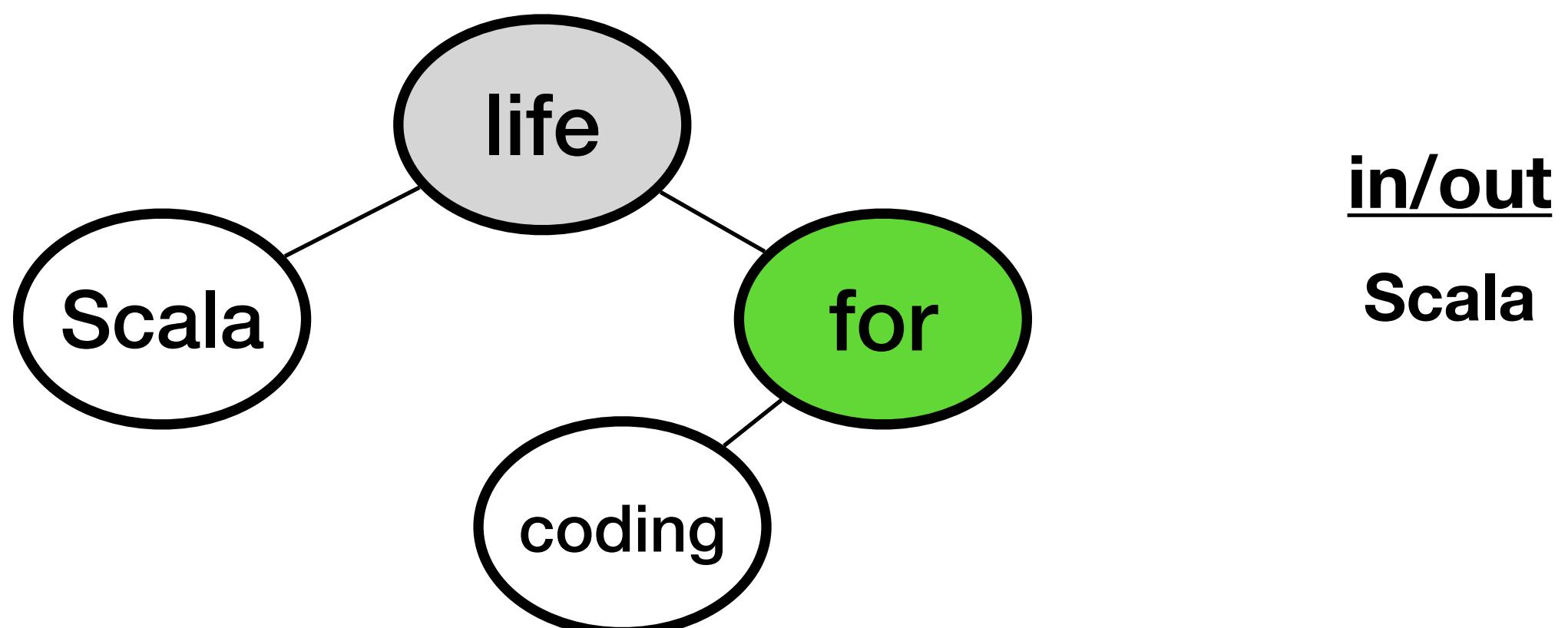
- First recursive call finally regains control (is at the top of the stack)

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "life"
value	"life"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"Scala"	value "Scala"
left	null	left 0x200
right	null	right 0x480
this	0x480	BTNode
value	"for"	value "for"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"coding"	value "coding"
left	null	left null
right	null	right null
node	0x350	BTNode
node	0x200	value "Scala"
node	null	left 0x200
node	null	right null
traversal		BTNode
traversal		value "for"
traversal		left null
traversal		right 0x480
traversal		BTNode
traversal		value "coding"
traversal		left null
traversal		right null
traversal		0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



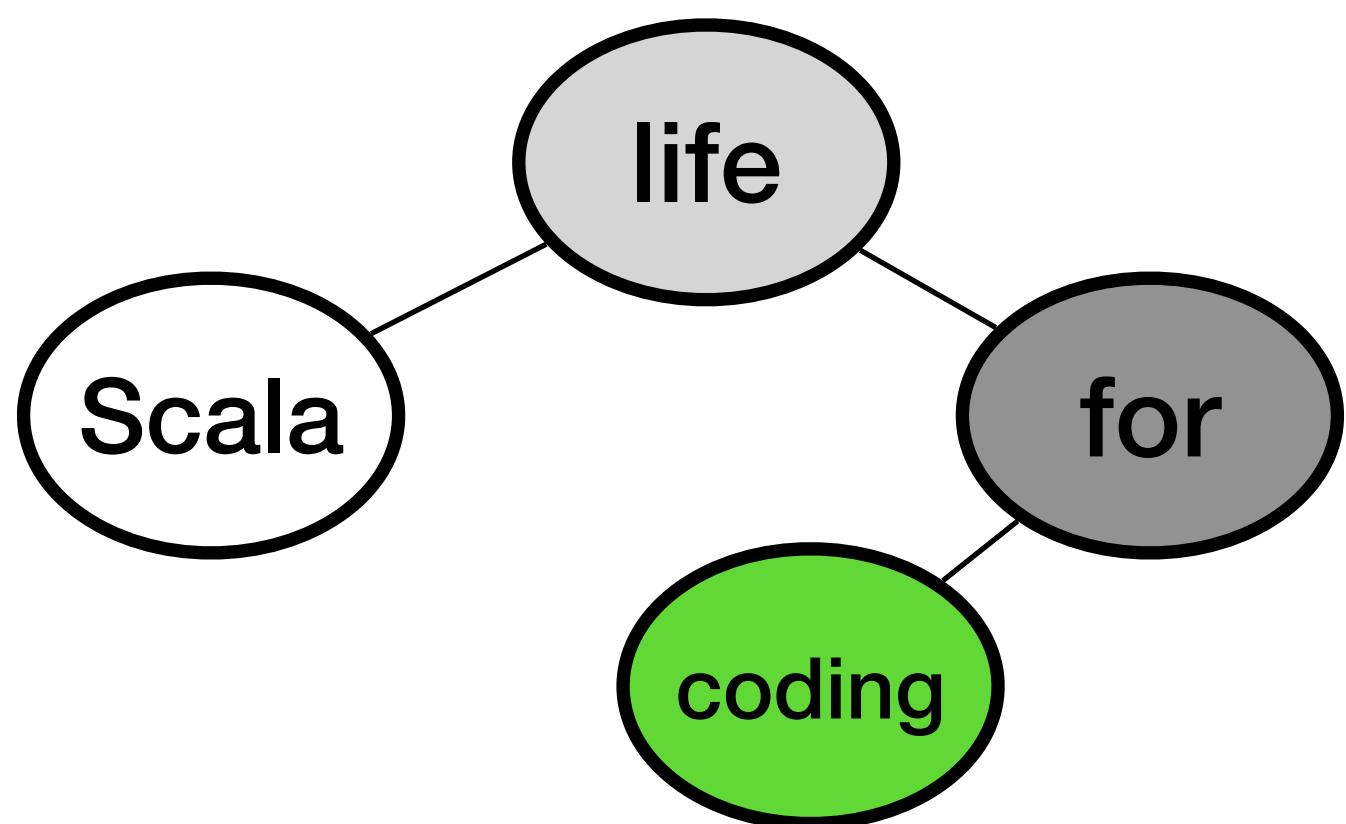
- Make a recursive call to the right of the root

Stack		Heap
Name	Value	
root	0x350	BTNode
this	0x350	value "life"
value	"life"	left null
left	null	right null
right	null	
this	0x200	BTNode
value	"Scala"	value "Scala"
left	null	left 0x200
right	null	right 0x480
this	0x480	BTNode
value	"for"	value "for"
left	null	left null
right	null	right null
this	0x936	BTNode
value	"coding"	value "coding"
left	null	left null
right	null	right null
node	0x350	traversal
node	0x200	traversal
node	null	traversal
node	null	traversal
node	0x480	traversal
node	0x936	BTNode
value	"coding"	value "coding"
left	null	left null
right	null	right null
	0x936	

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}
```

```
def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Recursive call to the left

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936

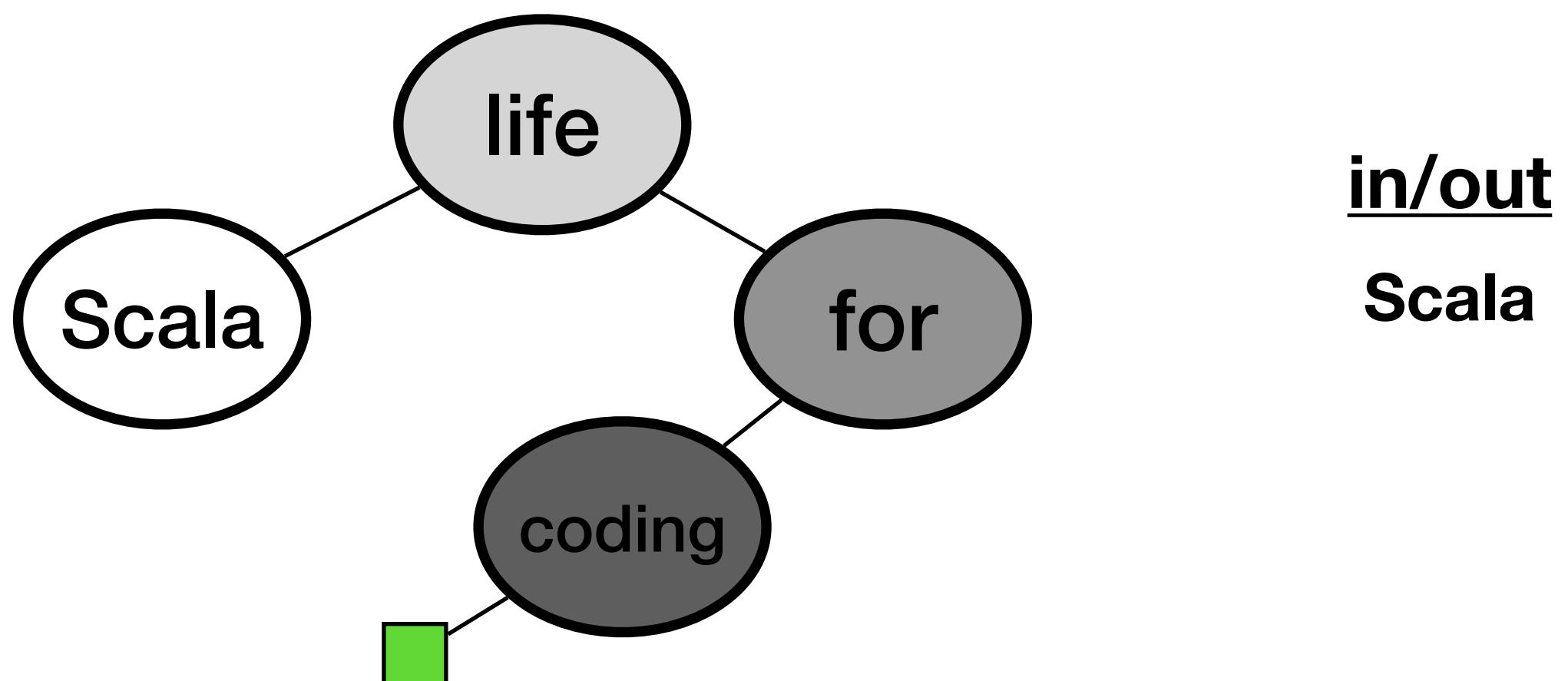
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
BTNode	0x200
value	"for"
left	null 0x936
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Recursive call of null to the left

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null

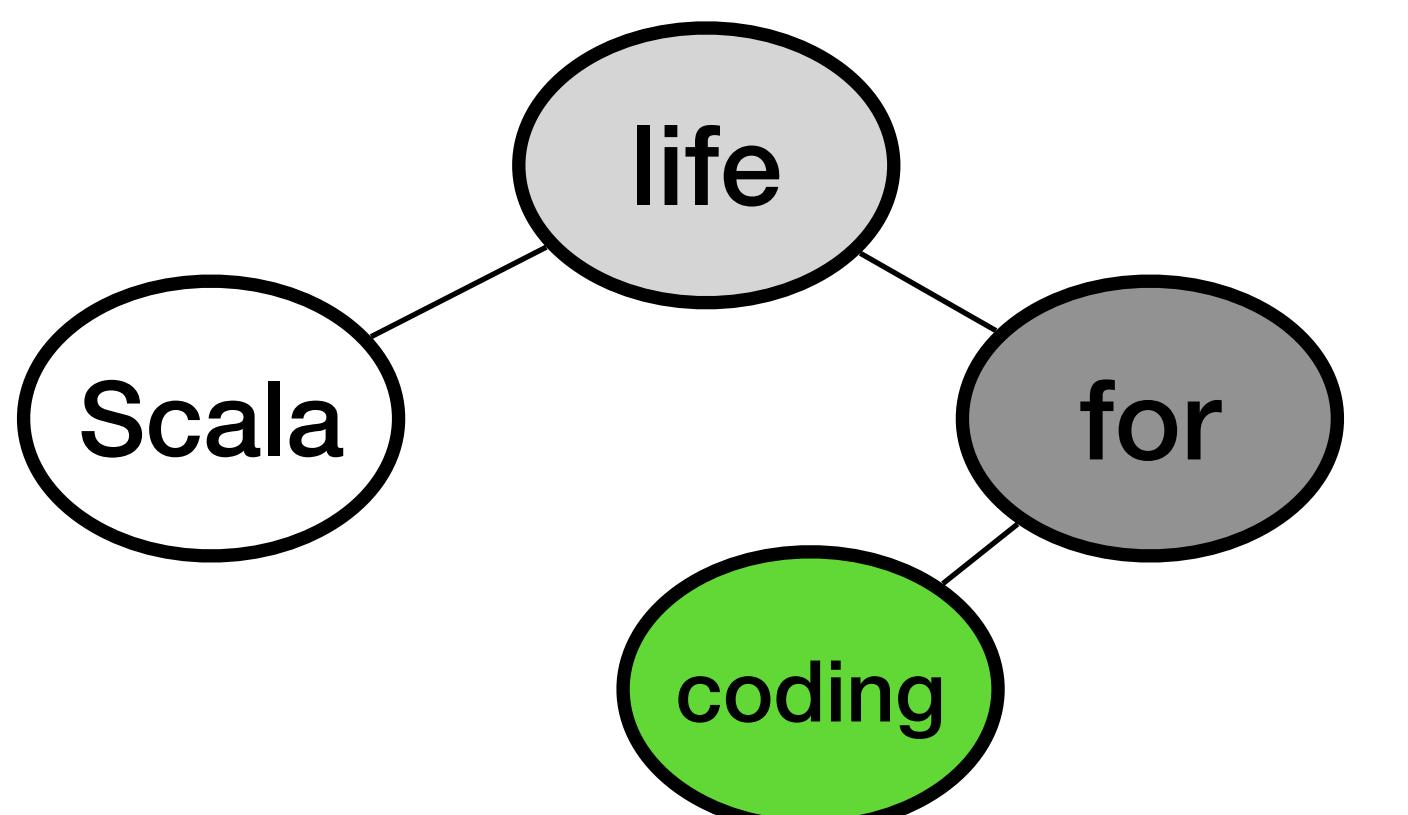
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
BTNode	0x200
value	"for"
left	null
right	null
BTNode	0x480
value	"for"
left	null 0x936
right	null
BTNode	0x936
value	"coding"
left	null
right	null

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}
```

```
def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Return to the previous frame and make the right recursive call

Stack

Name

Value

BTNode

BTNode

BTNode

BTNode

traversal

traversal

traversal

traversal

traversal

traversal

root

this

value

left

right

this

value

left

right

this

value

left

right

this

value

left

right

node

node

node

node

node

node

node

0x350

0x350

"life"

null

null

0x200

"Scala"

null

null

0x480

"for"

null

null

0x936

"coding"

null

null

0x350

0x200

null

null

0x480

0x936

null

Heap

BTNode

value

left

right

0x350

0x200

0x480

BTNode

value

left

right

0x200

BTNode

value

left

right

0x480

BTNode

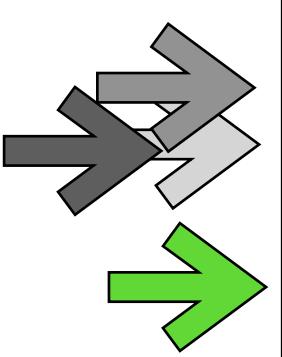
value

left

right

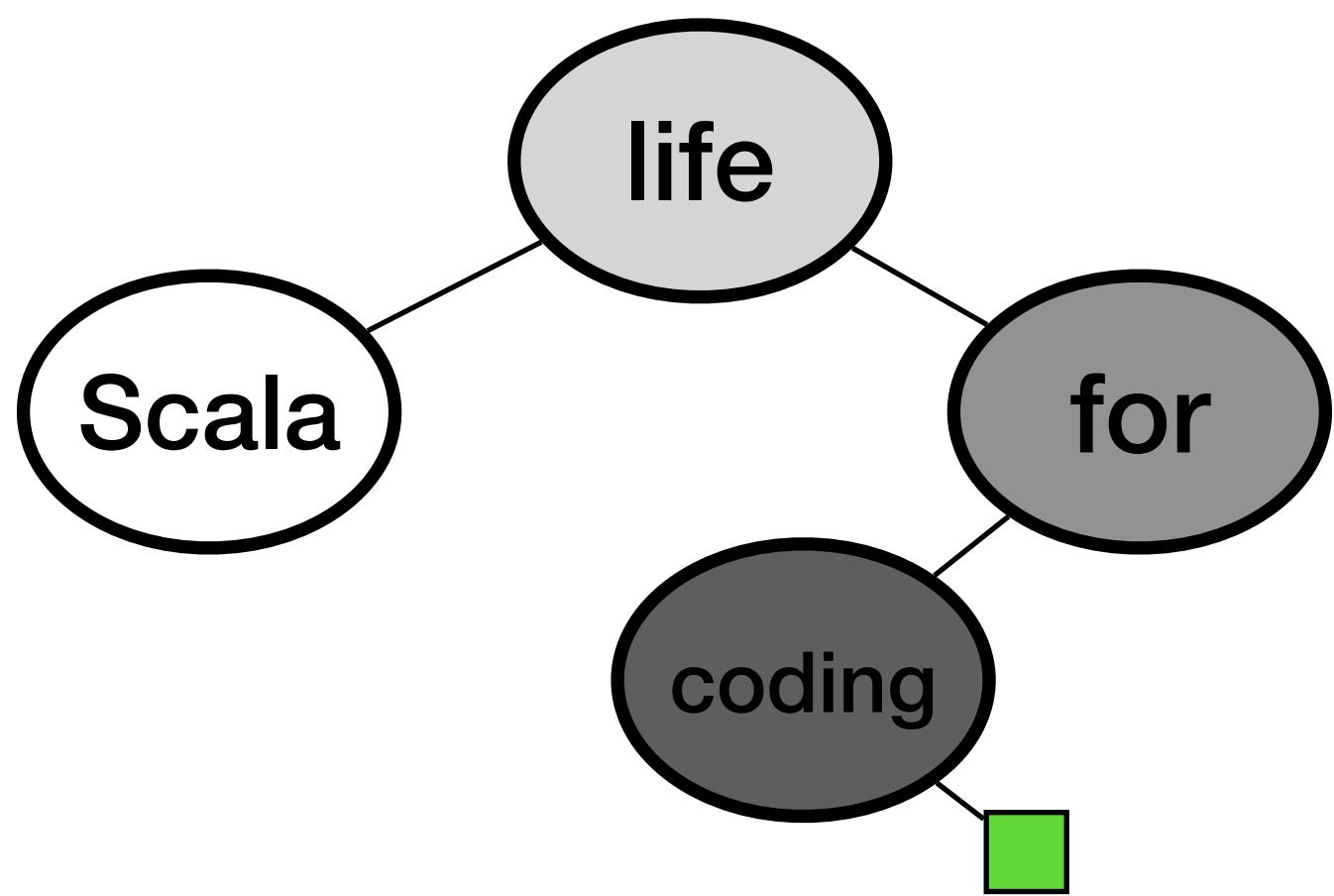
0x936

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Make the right recursive call of null

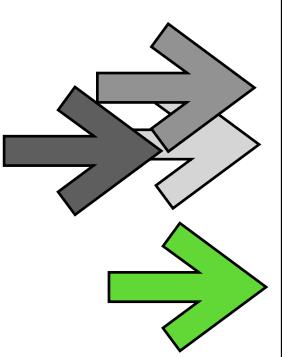
Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

Heap

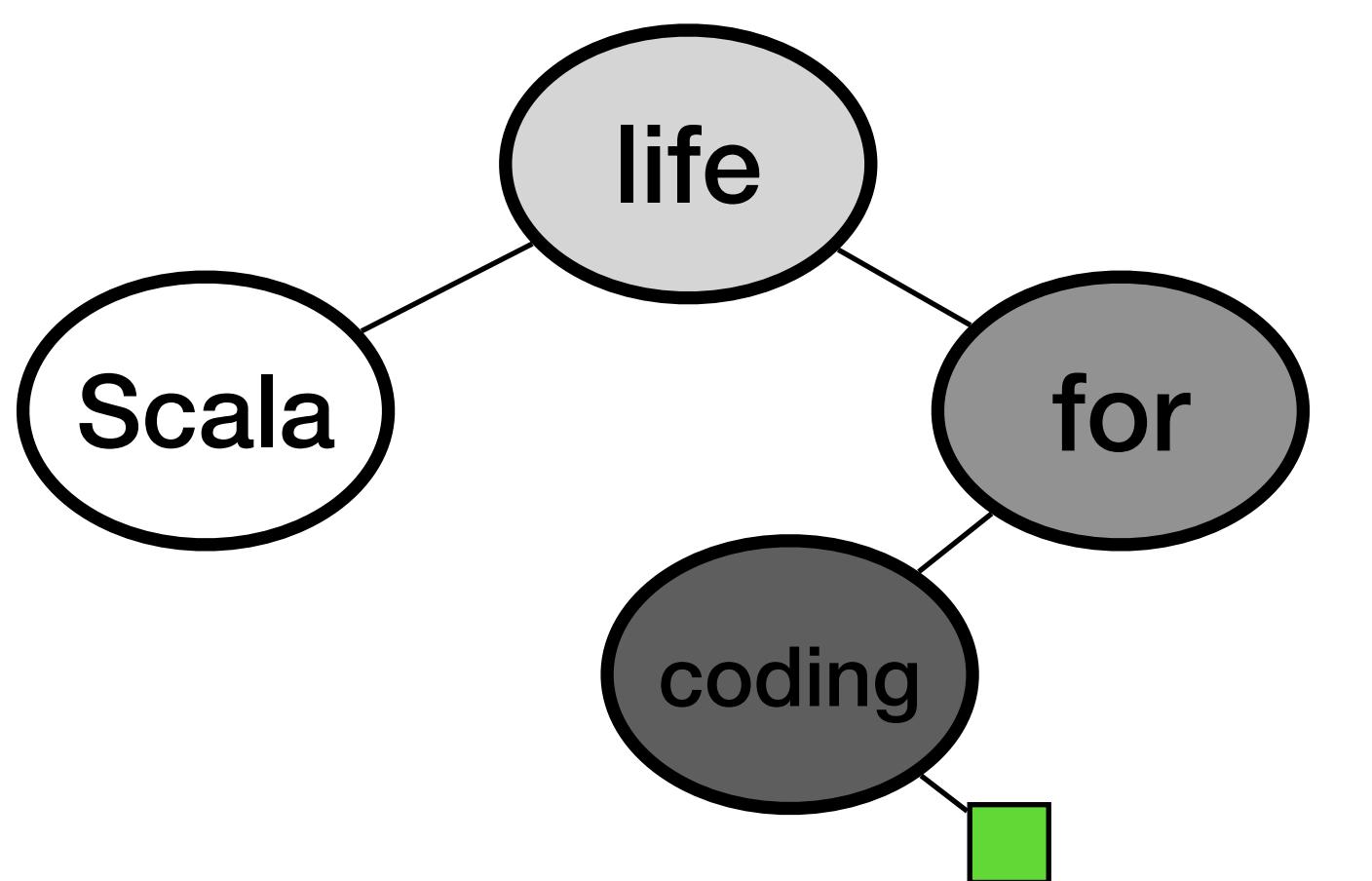
BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
0x200	
BTNode	0x936
value	"for"
left	null 0x936
right	null
0x480	
BTNode	0x936
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala

- Notice that all 4 stack frames remember what they need to do next when they regain control

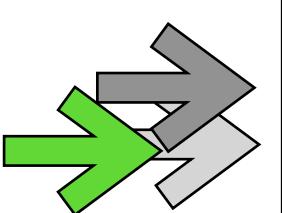
Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

Heap

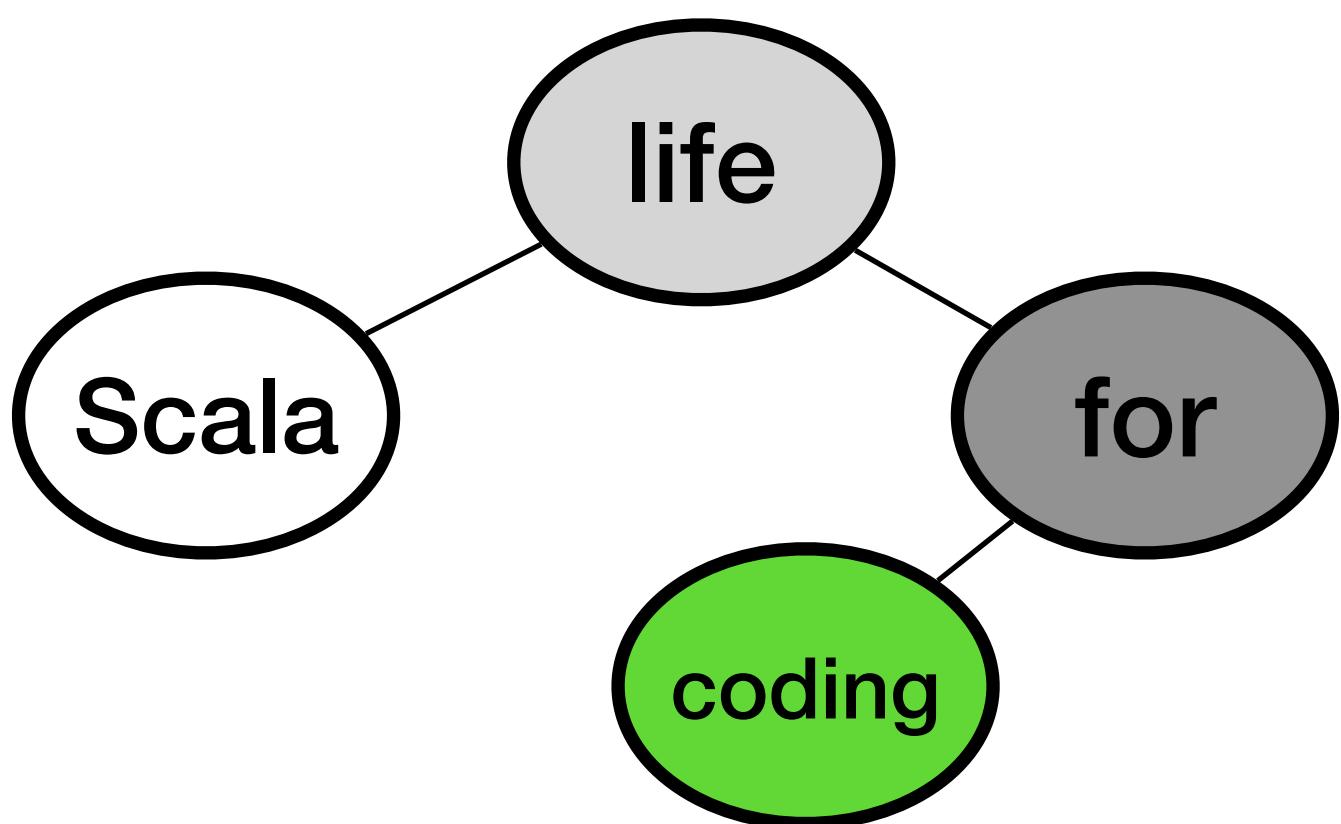
BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```



```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala

- This stack frame already made both recursive calls

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

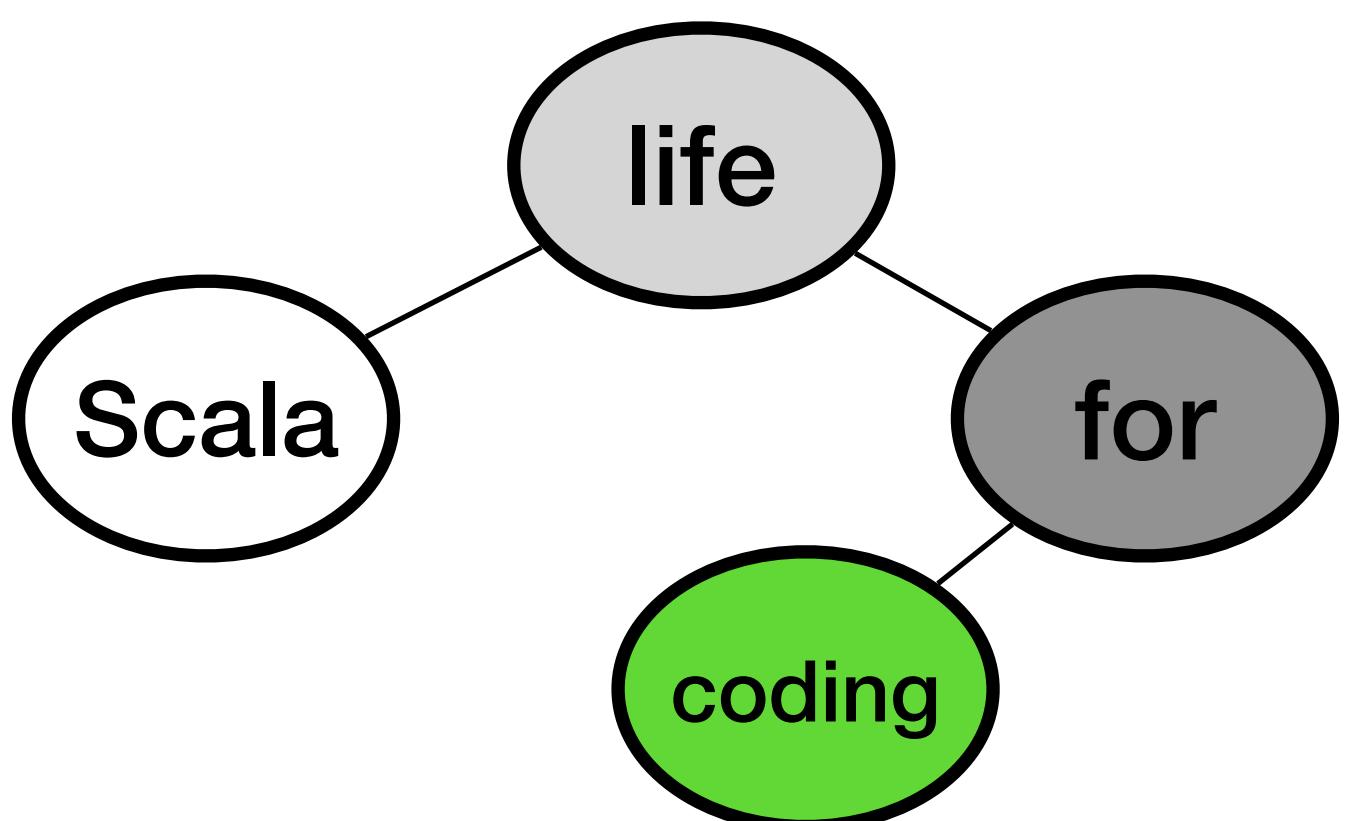
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
0x200	0x200
BTNode	0x480
value	"for"
left	null 0x936
right	null 0x480
0x936	0x936
BTNode	0x936
value	"coding"
left	null
right	null
0x936	0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Print "coding" to the screen

in/out
Scala coding

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

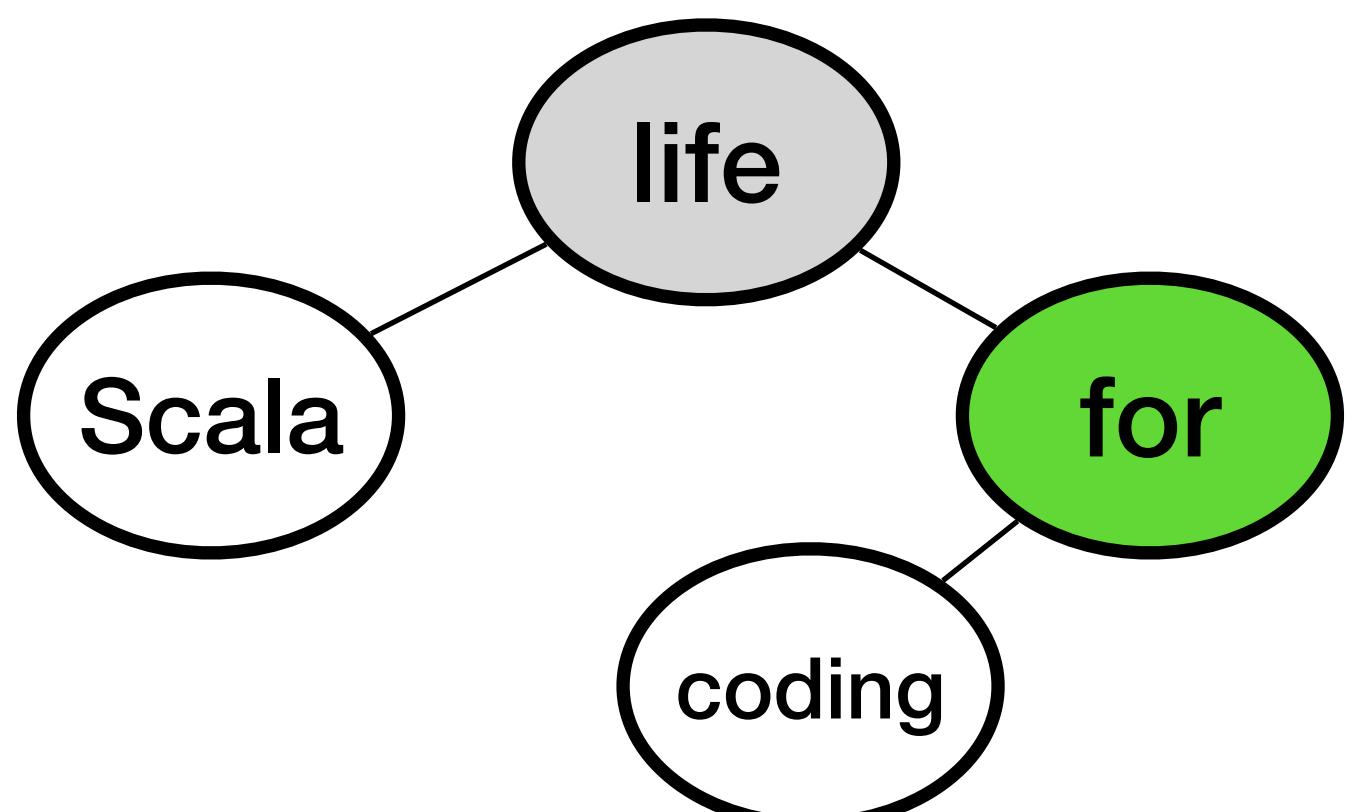
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala coding

- Return back to the previous stack frame

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

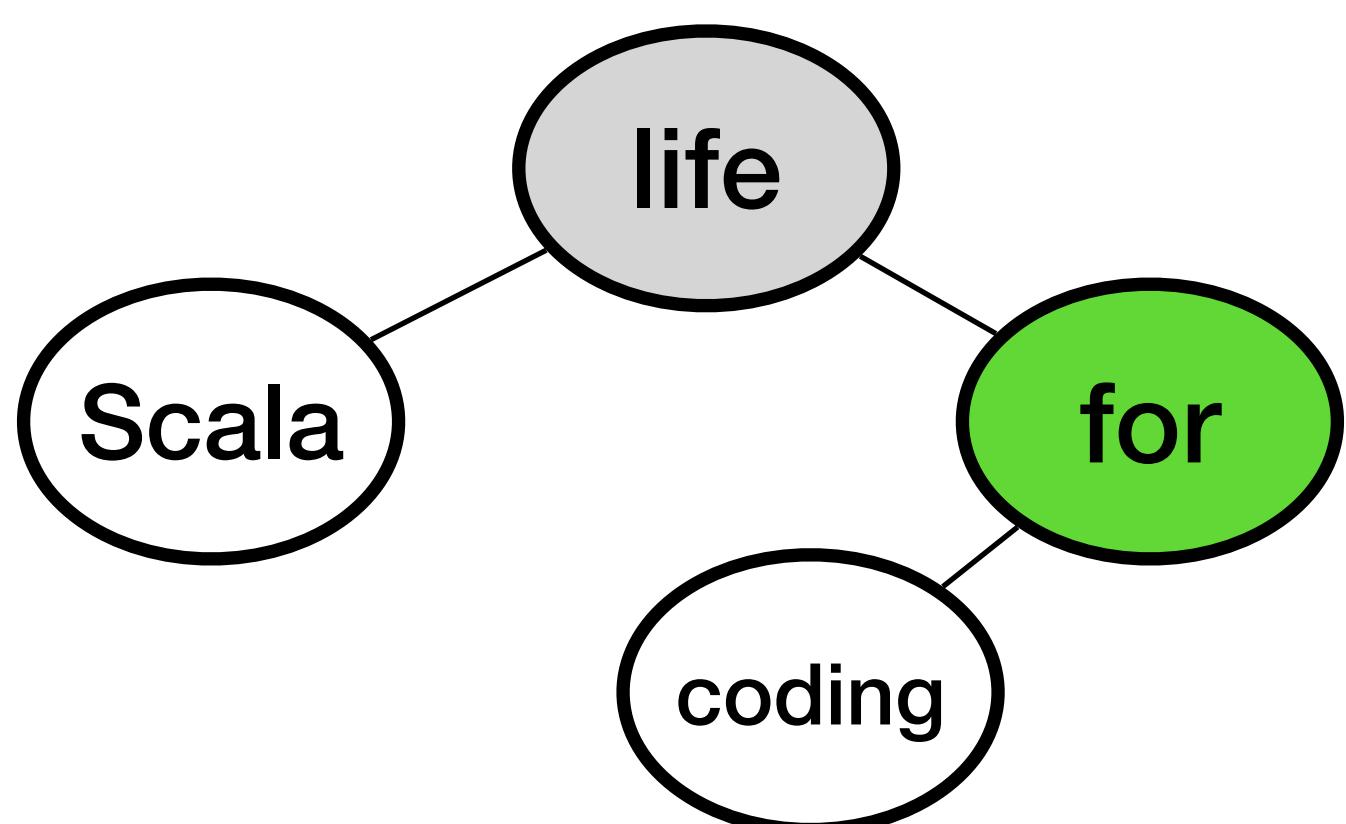
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala coding

- Make right recursive call

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null

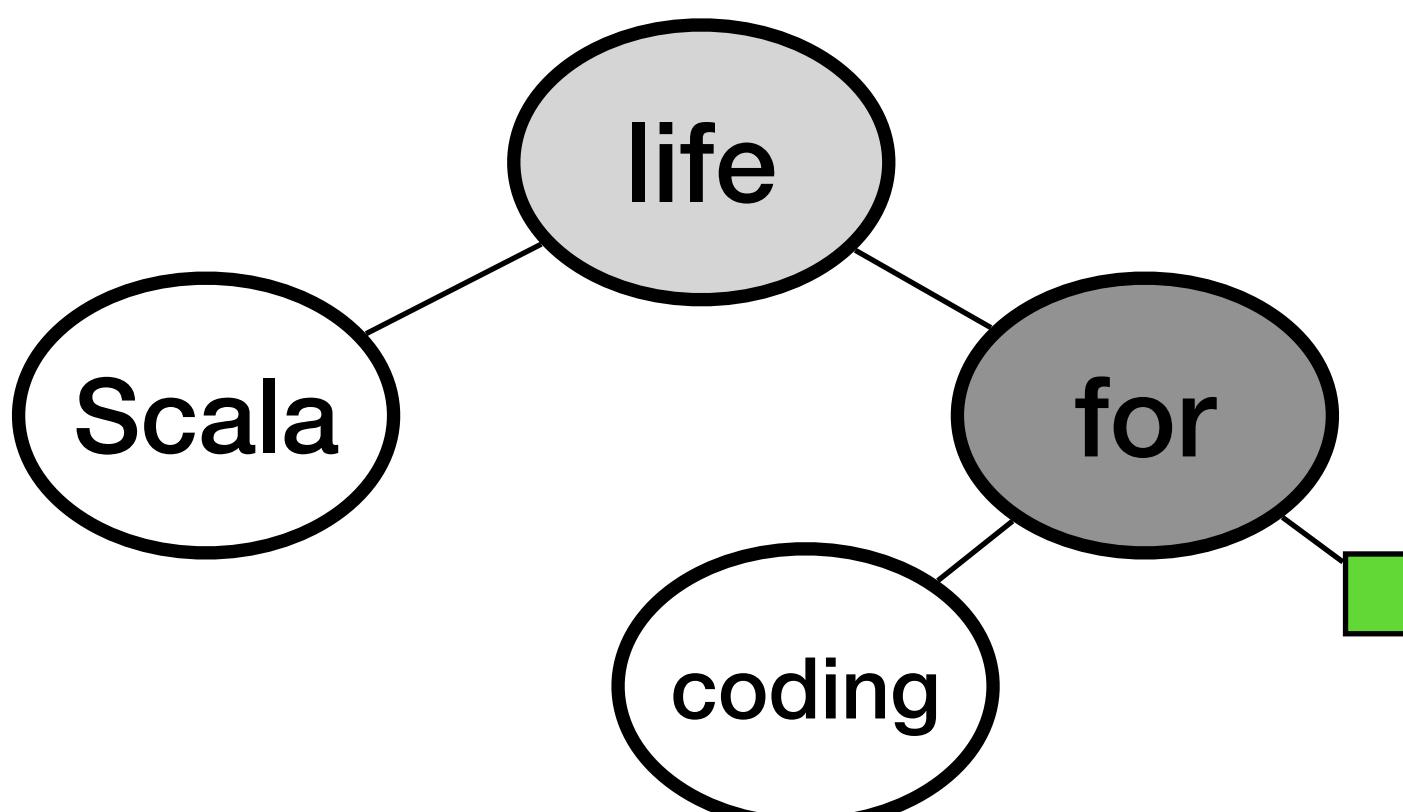
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
BTNode	0x200
value	"for"
left	null 0x936
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala coding

- Base case of null
- Just return

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

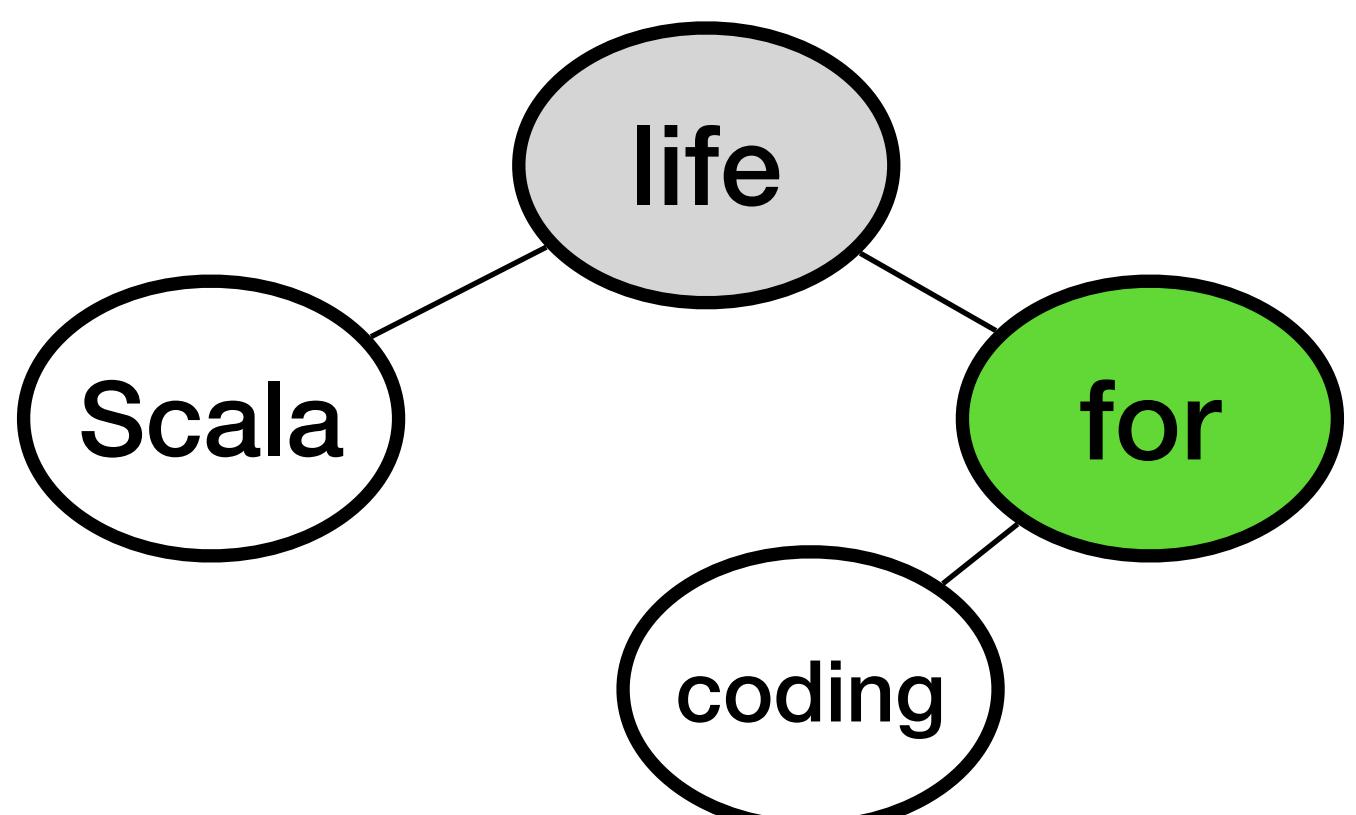
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala coding

- Stack frame done with both recursive calls

Stack

Name	Value
------	-------

root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

Heap

BTNode

value	"life"
left	null 0x200
right	null 0x480

BTNode

value	"Scala"
left	null
right	null

BTNode

value	"for"
left	null 0x936
right	null

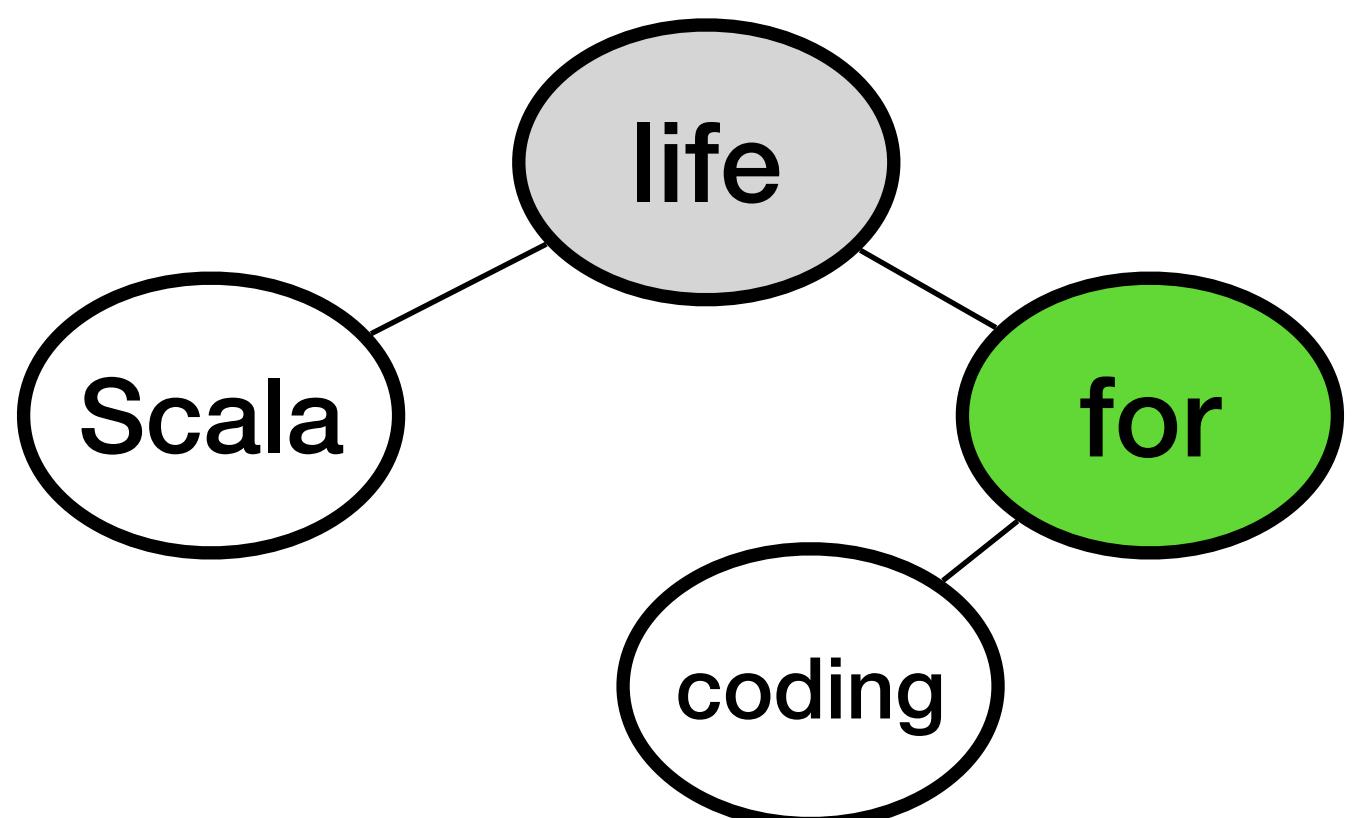
BTNode

value	"coding"
left	null
right	null

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



- Print "for" to the screen

in/out
Scala coding for

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

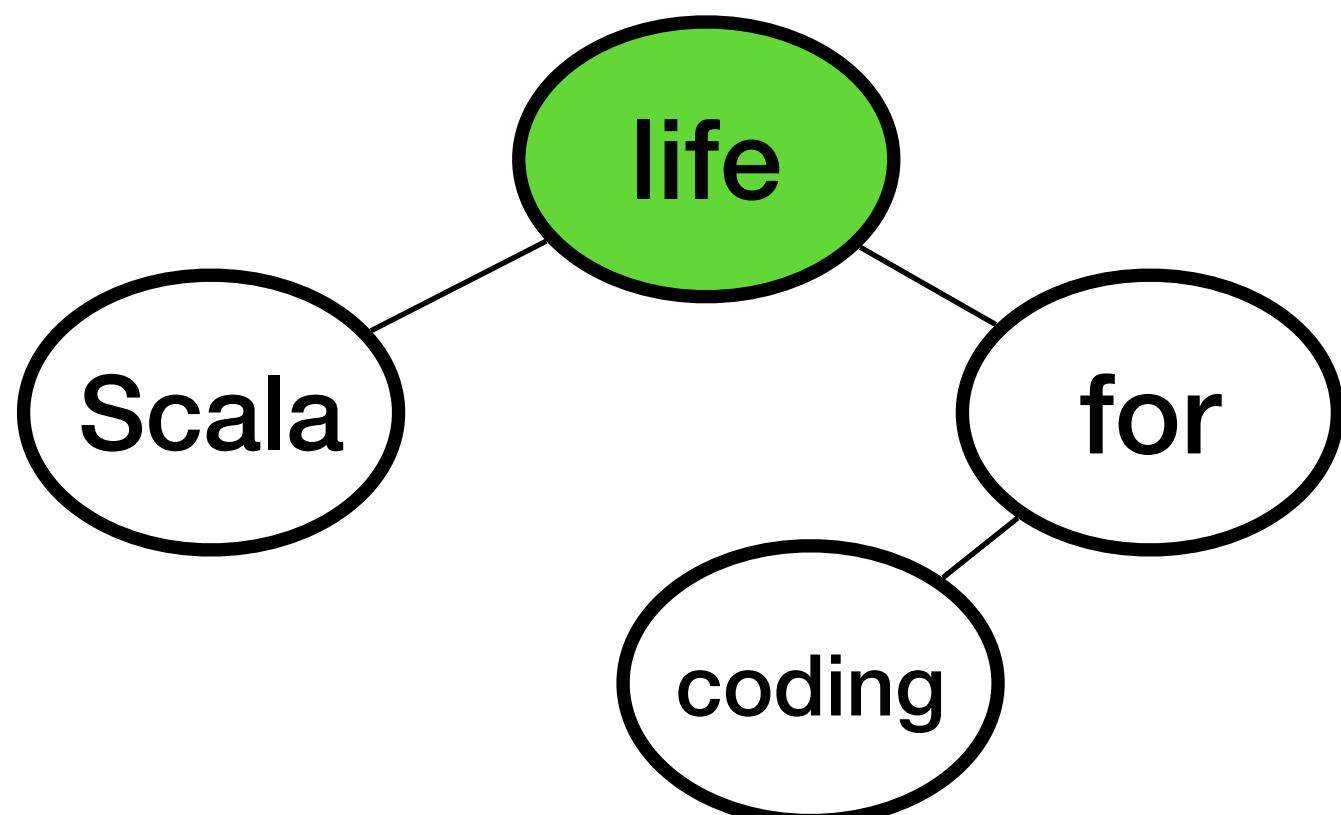
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	0x350
value	"Scala"
left	null
right	null
BTNode	0x200
value	"for"
left	null 0x936
right	null
BTNode	0x480
value	"coding"
left	null
right	null
BTNode	0x936

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
    if (node != null) {
        traversal(node.left)
        traversal(node.right)
        print(node.value + " ")
    }
}

def main(args: Array[String]): Unit = {
    val root = new BTNode("life", null, null)
    root.left = new BTNode("Scala", null, null)
    root.right = new BTNode("for", null, null)
    root.right.left = new BTNode("coding", null, null)
    traversal(root)
}
```



in/out

Scala coding for

- Root node is now done with both recursive calls

[A])

Stack		Heap	
Name	Value		
BTNode	root	0x350	
BTNode	this	0x350	
BTNode	value	"life"	
BTNode	left	null	
BTNode	right	null	
BTNode	this	0x200	
BTNode	value	"Scala"	
BTNode	left	null	
BTNode	right	null	
BTNode	this	0x480	
BTNode	value	"for"	
BTNode	left	null	
BTNode	right	null	
traversal	this	0x936	
traversal	value	"coding"	
traversal	left	null	
traversal	right	null	
traversal	node	0x350	
traversal	node	0x200	
traversal	node	null	
traversal	node	null	
traversal	node	0x480	
traversal	node	0x936	
traversal	node	null	
traversal	node	null	
traversal	node	0x936	
traversal	node	null	
traversal	node	null	
traversal	node	0x936	

The diagram illustrates the state of the stack and heap during a traversal of a binary tree. The stack contains pointers to nodes on the heap, and the heap contains node structures with their respective values and pointers to left and right children.

Stack:

- BTNode (root) - Value: 0x350
- BTNode (this) - Value: 0x350
- BTNode (value) - Value: "life"
- BTNode (left) - Value: null
- BTNode (right) - Value: null
- BTNode (this) - Value: 0x200
- BTNode (value) - Value: "Scala"
- BTNode (left) - Value: null
- BTNode (right) - Value: null
- BTNode (this) - Value: 0x480
- BTNode (value) - Value: "for"
- BTNode (left) - Value: null
- BTNode (right) - Value: null
- traversal (this) - Value: 0x936
- traversal (value) - Value: "coding"
- traversal (left) - Value: null
- traversal (right) - Value: null
- traversal (node) - Value: 0x350
- traversal (node) - Value: 0x200
- traversal (node) - Value: null
- traversal (node) - Value: null
- traversal (node) - Value: 0x480
- traversal (node) - Value: 0x936
- traversal (node) - Value: null
- traversal (node) - Value: null
- traversal (node) - Value: 0x936

Heap:

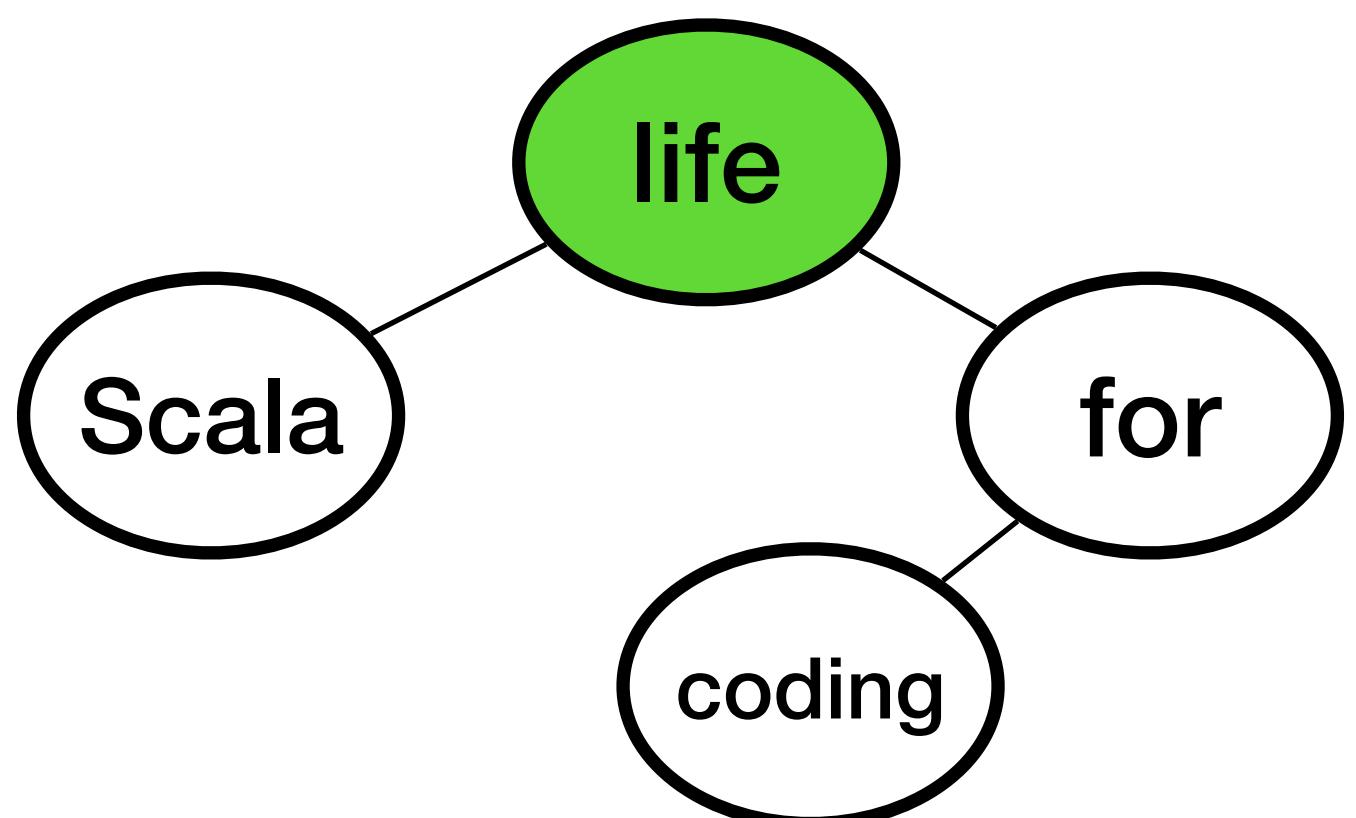
- BTNode** (value: "life", left: null, right: 0x350)
- BTNode** (value: "Scala", left: null, right: 0x200)
- BTNode** (value: "for", left: 0x200, right: null)
- BTNode** (value: "coding", left: null, right: null)

Magenta arrows indicate the current state of the stack, showing the traversal path through the heap-allocated BTNode objects.

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala coding for life

- Print "life" to the screen

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936

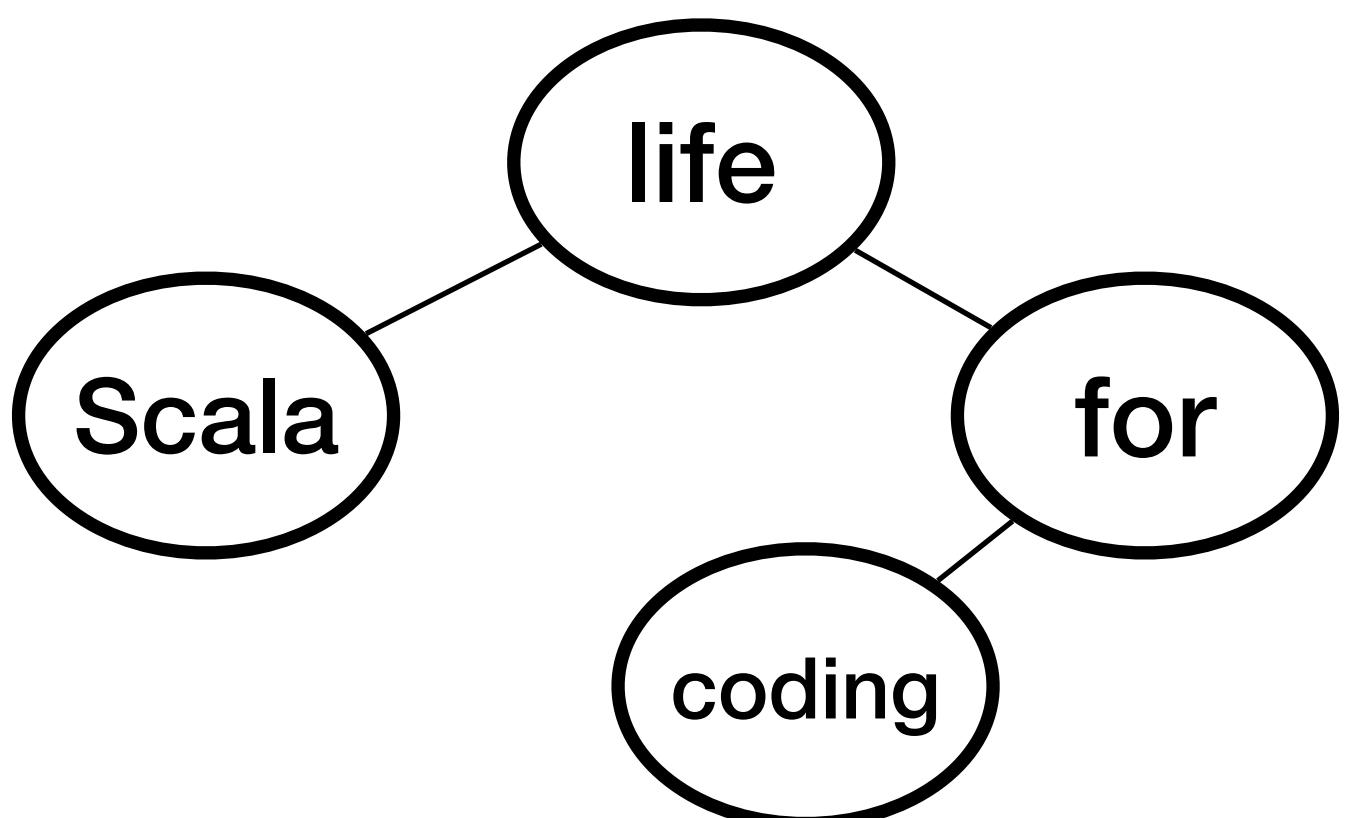
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A] (var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
  if (node != null) {
    traversal(node.left)
    traversal(node.right)
    print(node.value + " ")
  }
}

def main(args: Array[String]): Unit = {
  val root = new BTNode("life", null, null)
  root.left = new BTNode("Scala", null, null)
  root.right = new BTNode("for", null, null)
  root.right.left = new BTNode("coding", null, null)
  traversal(root)
}
```



in/out
Scala coding for life

- The traversal is over
- All stack frames have returned

Stack

Name	Value
root	0x350
this	0x350
value	"life"
left	null
right	null
this	0x200
value	"Scala"
left	null
right	null
this	0x480
value	"for"
left	null
right	null
this	0x936
value	"coding"
left	null
right	null
node	0x350
node	0x200
node	null
node	null
node	0x480
node	0x936
node	null
node	null
node	0x936
node	null
node	null
node	null

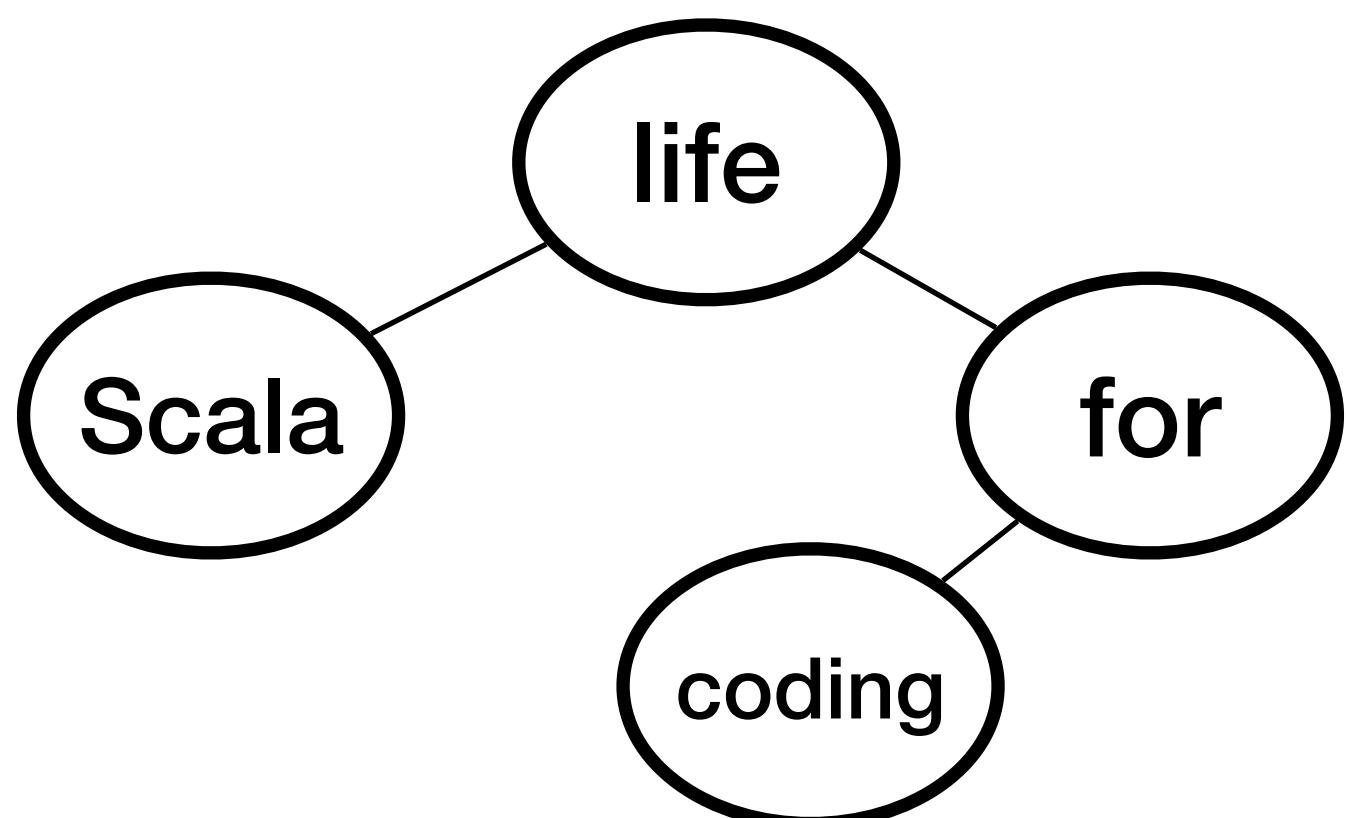
Heap

BTNode	
value	"life"
left	null 0x200
right	null 0x480
BTNode	
value	"Scala"
left	null
right	null
0x200	
BTNode	
value	"for"
left	null 0x936
right	null
0x480	
BTNode	
value	"coding"
left	null
right	null
0x936	

```
class BTNode[A](var value: A, var left: BTNode[A], var right: BTNode[A])
```

```
def traversal[A](node: BTNode[A]): Unit = {
    if (node != null) {
        traversal(node.left)
        traversal(node.right)
        print(node.value + " ")
    }
}

def main(args: Array[String]): Unit = {
    val root = new BTNode("life", null, null)
    root.left = new BTNode("Scala", null, null)
    root.right = new BTNode("for", null, null)
    root.right.left = new BTNode("coding", null, null)
    traversal(root)
}
```



in/out

scala coding for life