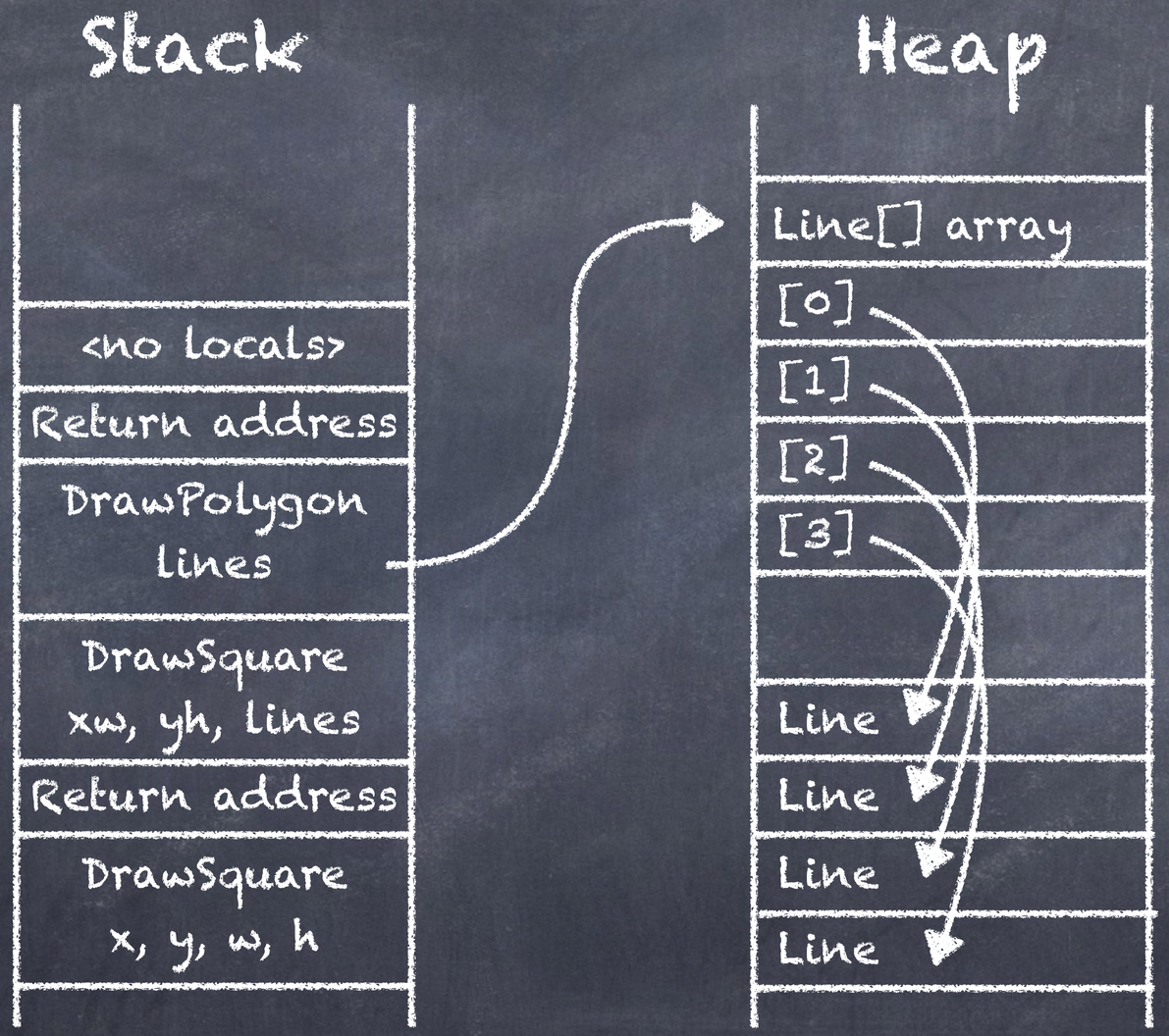
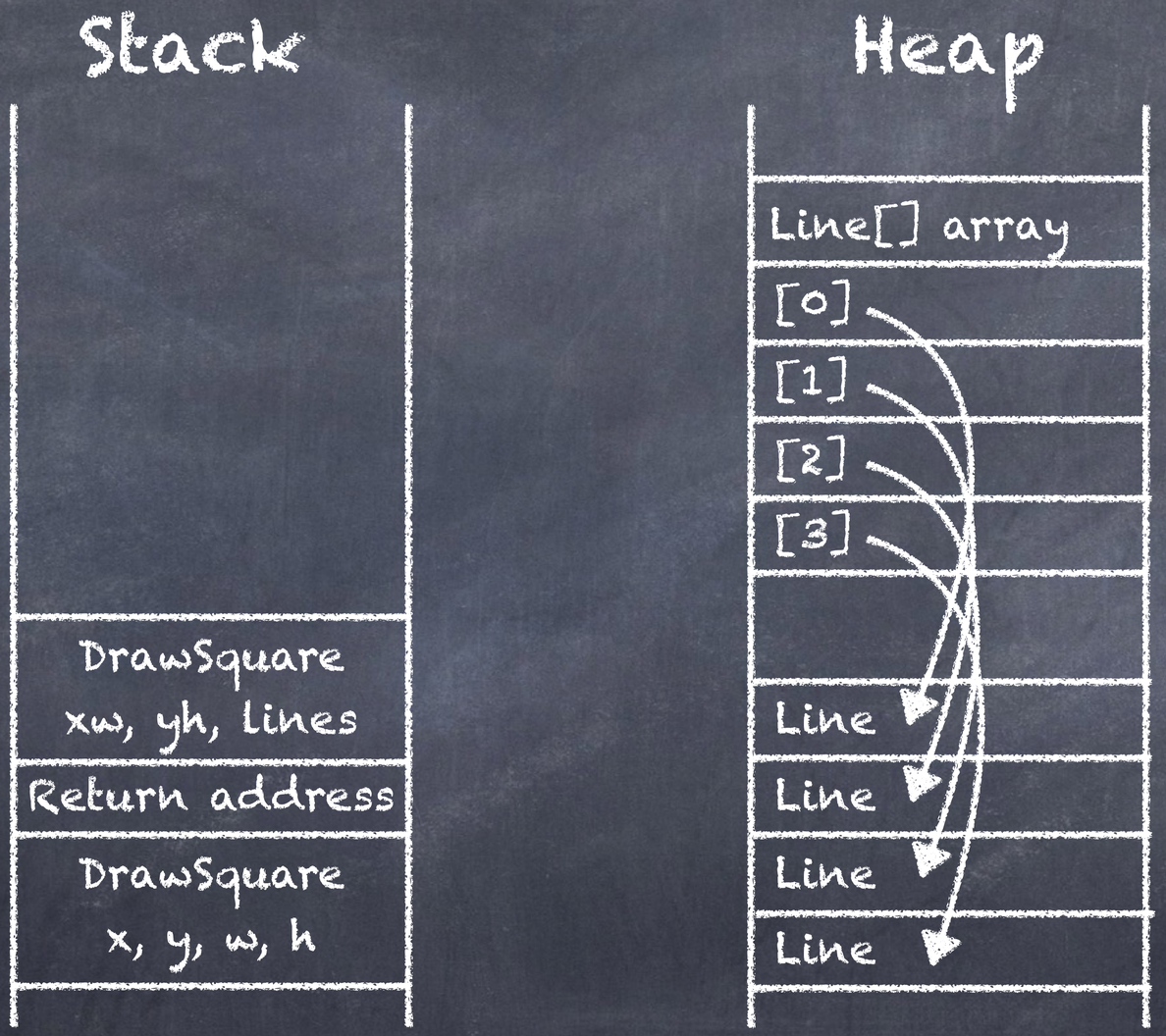
The heap

```
MyClass obj = new MyClass();
```


Stack
Heap
<no locals>
Return address
DrawPolygon
lines
DrawSquare
xw, yh, lines
Return address
DrawSquare
x, y, w, h
Line[] array
[0]
[1]
[2]
[3]
Line
Line
Line
Line


Stack
Heap
DrawSquare
xw, yh, lines
Return address
DrawSquare
x, y, w, h
Line[] array
[0]
[1]
[2]
[3]
Line
Line
Line
Line

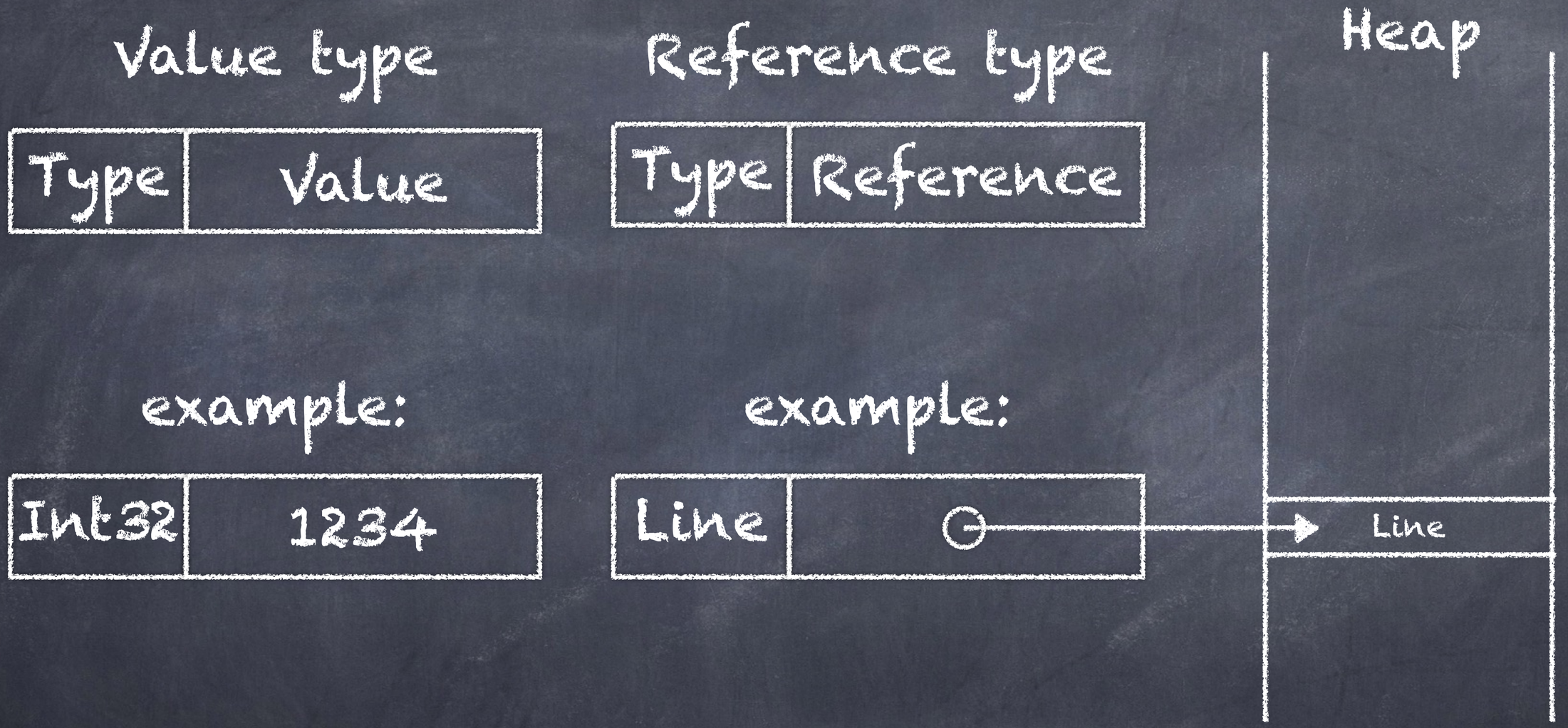
Code that uses integers on the stack is slightly faster than code that uses objects on the heap.

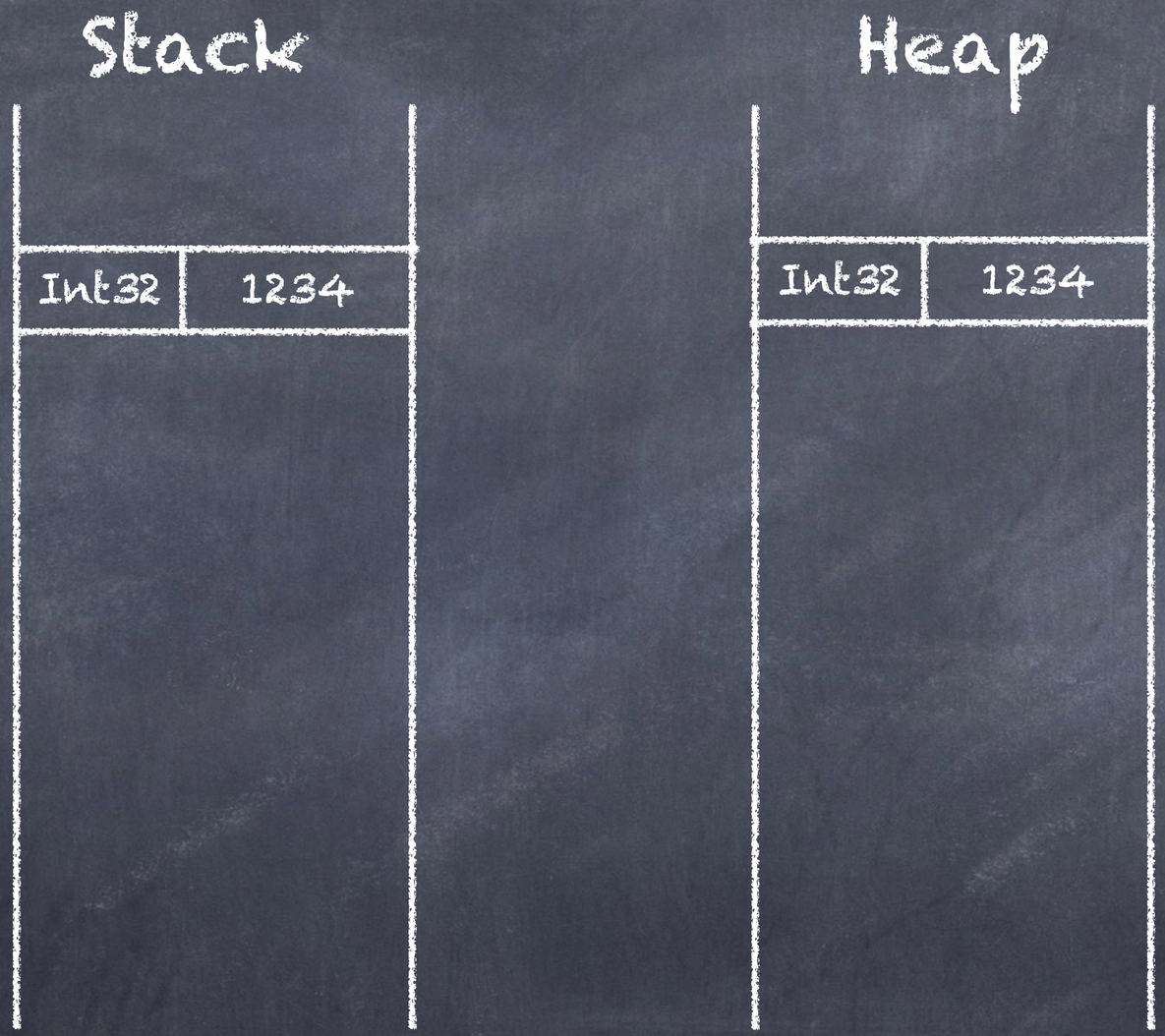
The .NET Framework will always postpone cleaning up dereferenced objects.

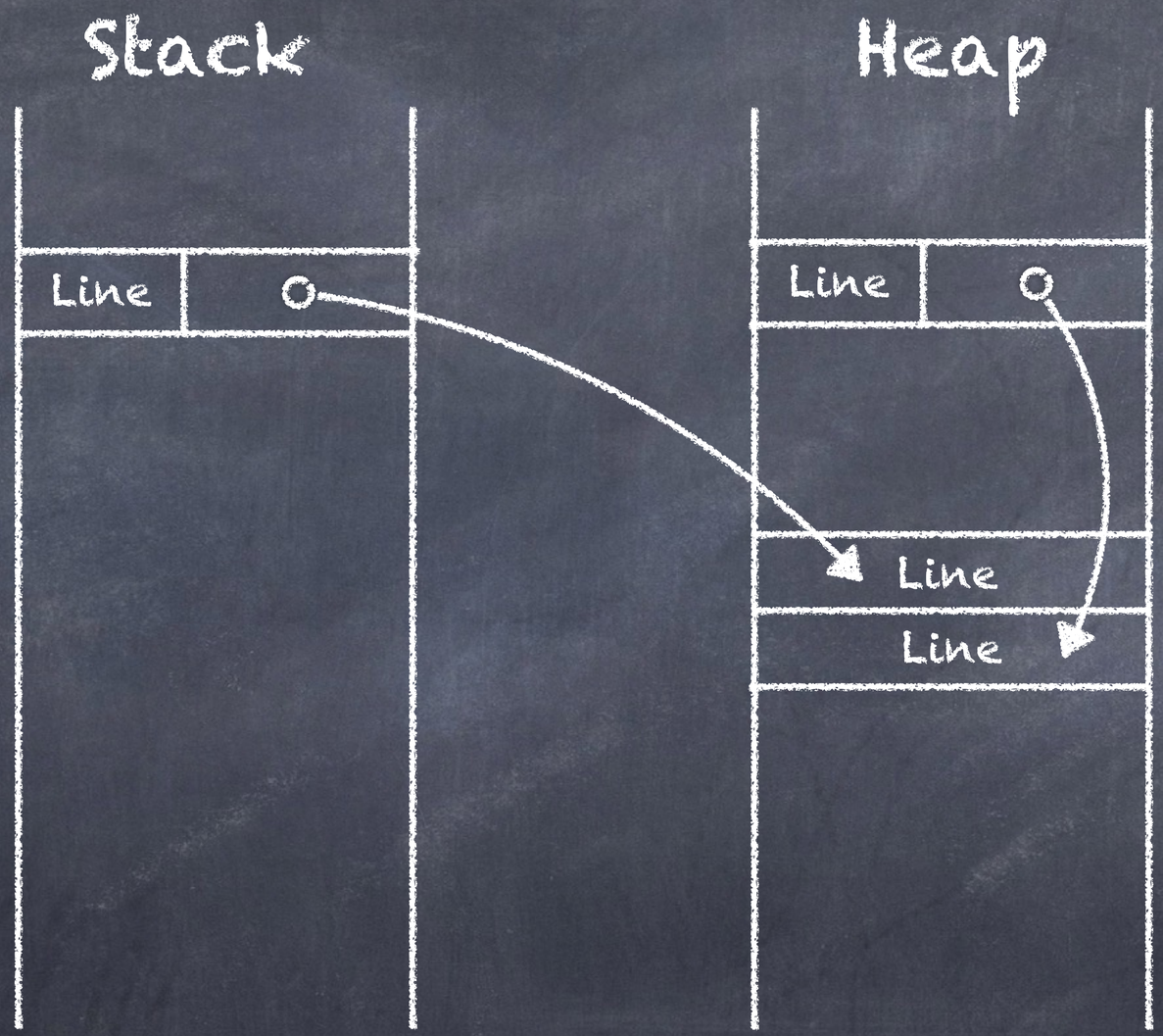
Eventually the Framework starts a process called Garbage Collection and deallocates all dereferenced objects on the heap.

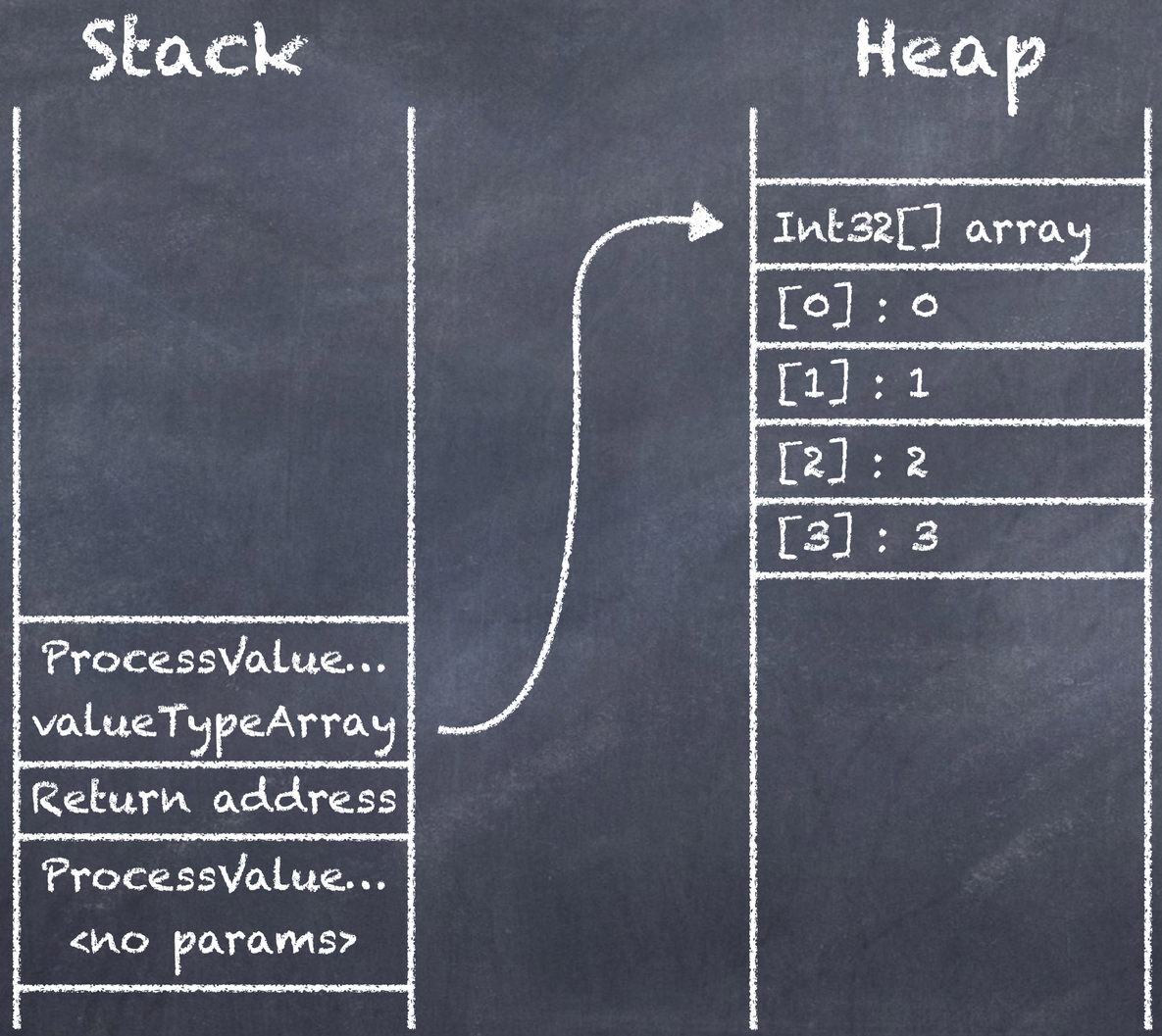
The heap

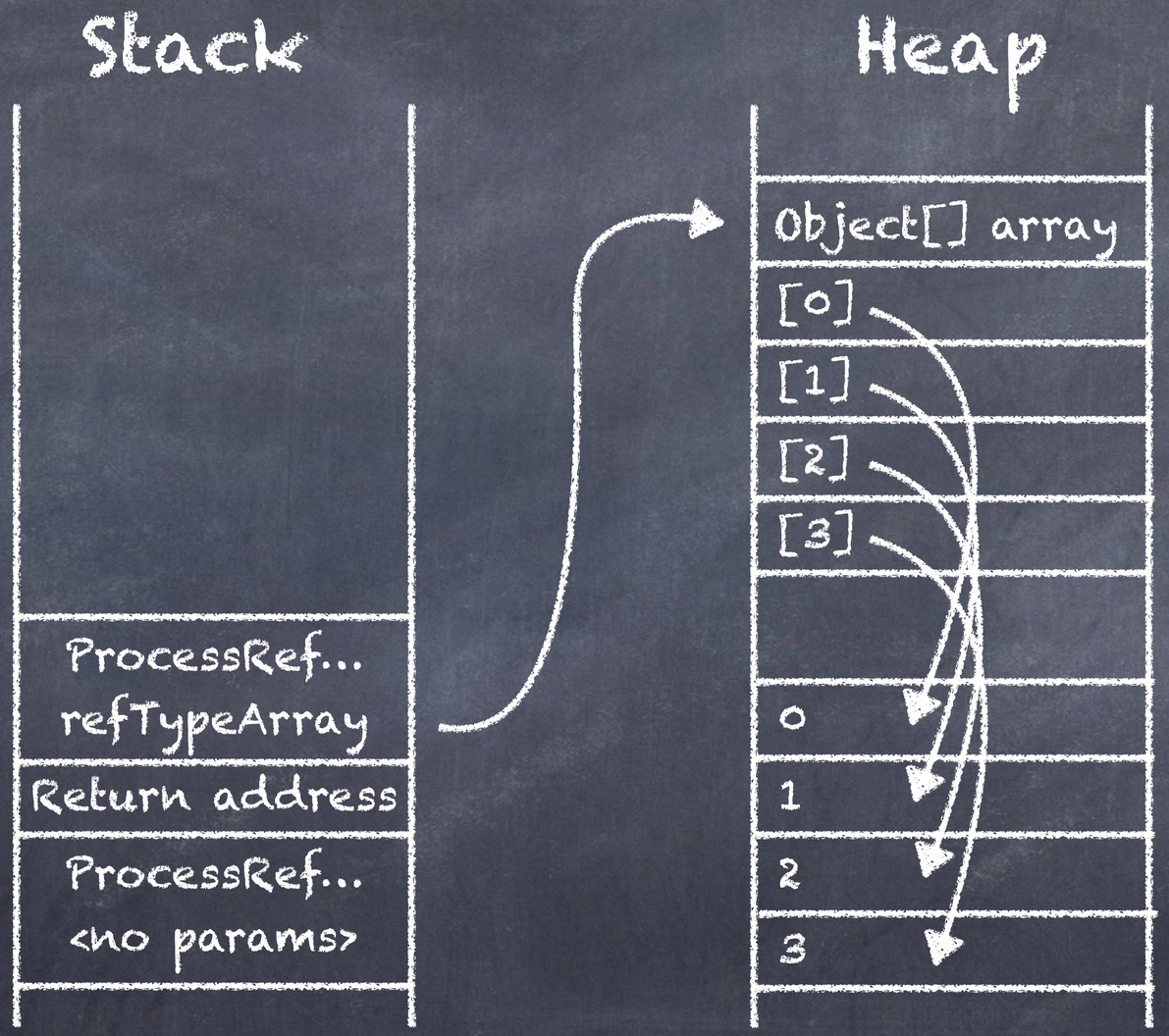
- The new keyword creates objects on the heap
- Code that uses objects on the heap is slightly slower than code using integers on the stack.
- When variables on the stack go out of scope, their corresponding objects on the heap are dereferenced, not destroyed.
- The .NET Framework eventually starts a process called Garbage Collection and deallocates all dereferenced objects on the heap.


Value type
Type
Value
Reference type
Type
Reference
Heap
example:
Int32
1234
example:
Line
Line

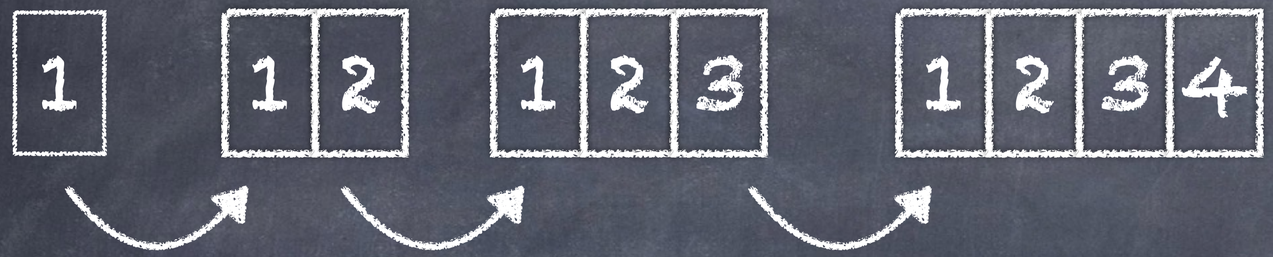

Stack
Int32
1234
Heap
Int32
1234


Stack
Heap
Line
Line
Line
Line


Stack
Heap
Int32[] array
[0] : 0
[1] : 1
[2] : 2
[3] : 3
ProcessValue...
valueTypeArray
Return address
ProcessValue...
<no params>


Stack
Heap
Object[] array
[0]
[1]
[2]
[3]
ProcessRef...
refTypeArray
Return address
ProcessRef...
<no params>
0
1
2
3

Appending to a string:



Appending to a StringBuilder: