

Java Arrays

Computer Science OOD
Boston University

Christine Papadakis

Arrays

An Array is a *fixed* data structure.

It is a *container* that holds
a fixed number of elements of the same data type.

The length of an array is established when
the array is declared and
the physical size of the array
cannot be altered during run time.

Arrays

Specifically:

An array variable references
a *contiguous* memory allocation of
some fixed number of elements
of the same data type.

Why is this important?

This is why arrays are more efficient and require less overhead than Python lists!

Contiguous allocation means that all the elements of the array are stored in consecutive memory cells.

The fact that all the elements are of the *exact data type* means that each element is allocated the same number of bytes.

Therefore if we know the address location of the first element, we can create an offset from that starting address and ***directly*** access every element.

Array Variables

- We use a variable to represent the array as a whole.
- Example of declaring an array *variable*:

```
int[] temps;
```

- the `[]` indicates that variable temps represents an array
- the `int` indicates that the elements will be of type `int`
- note that the array itself has not been allocated, only the *variable that will reference* the array has been declared.

Array Variables

- We use a variable to represent the array as a whole.
- Example of declaring an array *variable*:

```
int[] temps = {1, 2, 3, 4, 5};
```

- the `[]` indicates that variable `temps` represents an array
- the `int` indicates that the elements will be of type `int`
- creates an array in memory, *initialized with the specified elements*, and assigns the memory location (of the first element) to variable `temps`.

Array Variables

- We use a variable to represent the array as a whole.
- Example of declaring an array *variable*:

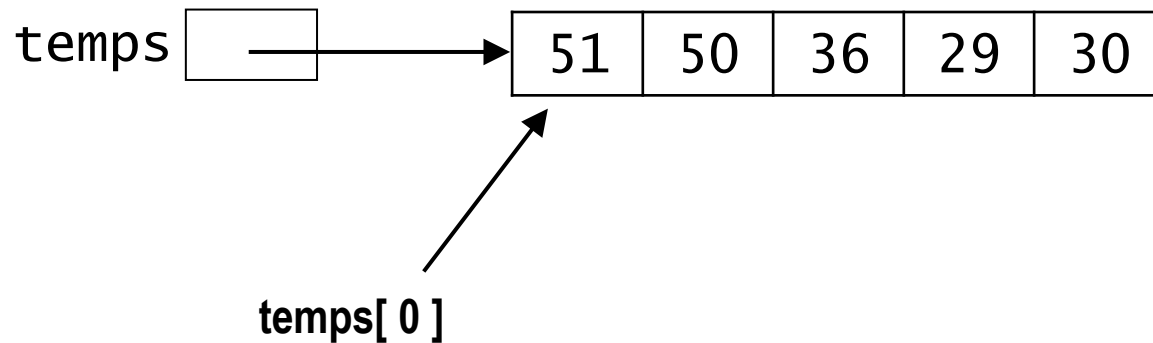
```
int[] temps = new int[5];
```

- the `[]` indicates that variable `temps` represents an array
- the `int` indicates that the elements will be of type `int`
- creates an array of five elements (initialized to default value) in memory and assigns the memory location (of the first element) to variable `temps`.

Arrays and References

- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

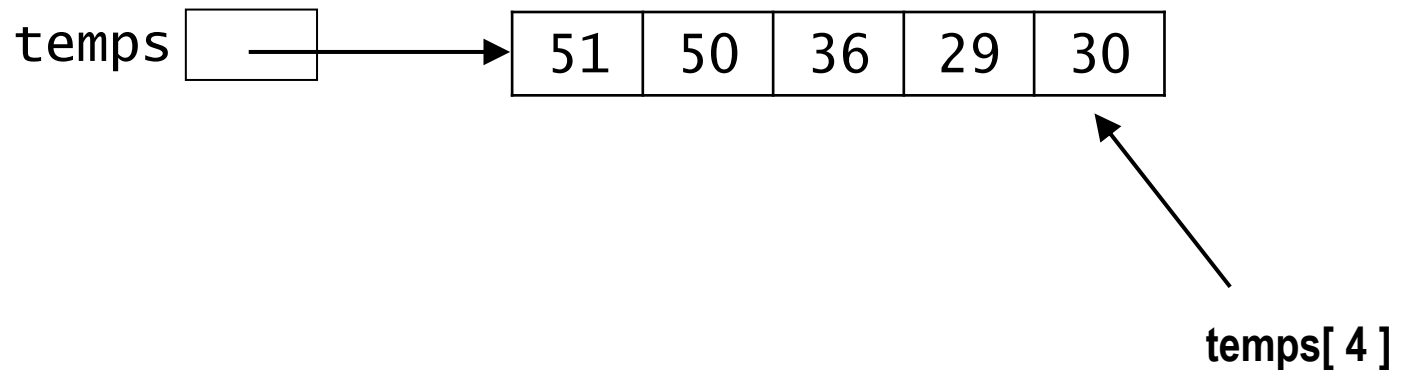
```
int[] temps = {51, 50, 36, 29, 30};
```



Arrays and References

- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

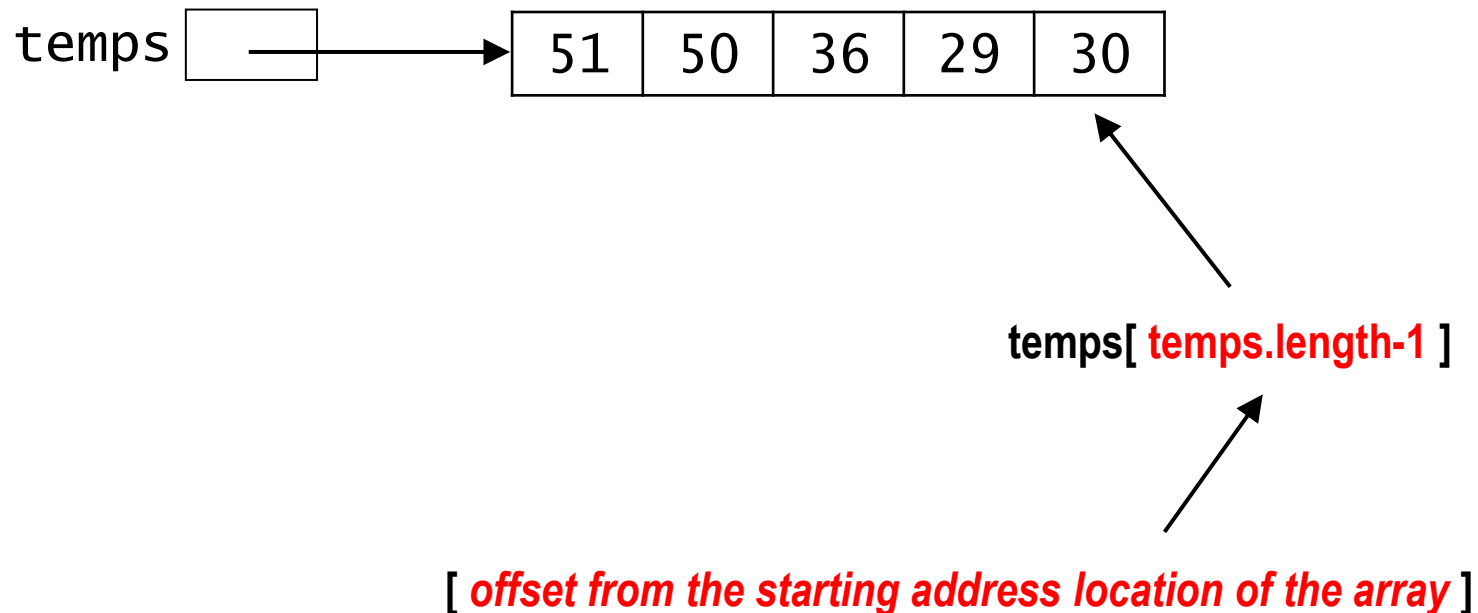
```
int[] temps = {51, 50, 36, 29, 30};
```



Arrays and References

- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

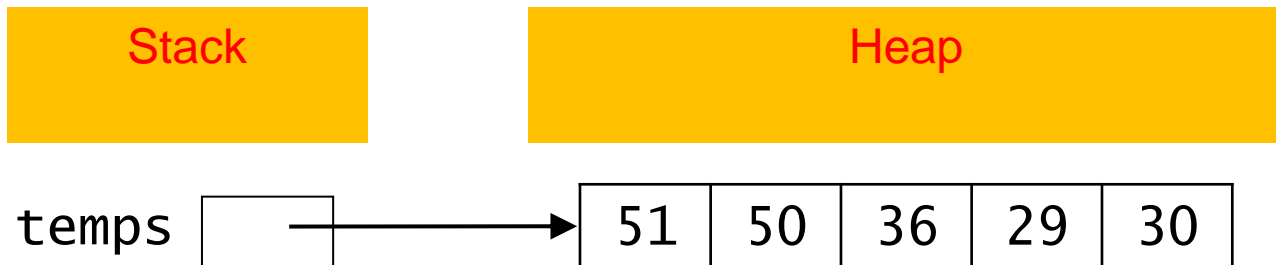
```
int[] temps = {51, 50, 36, 29, 30};
```



Arrays and References

- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

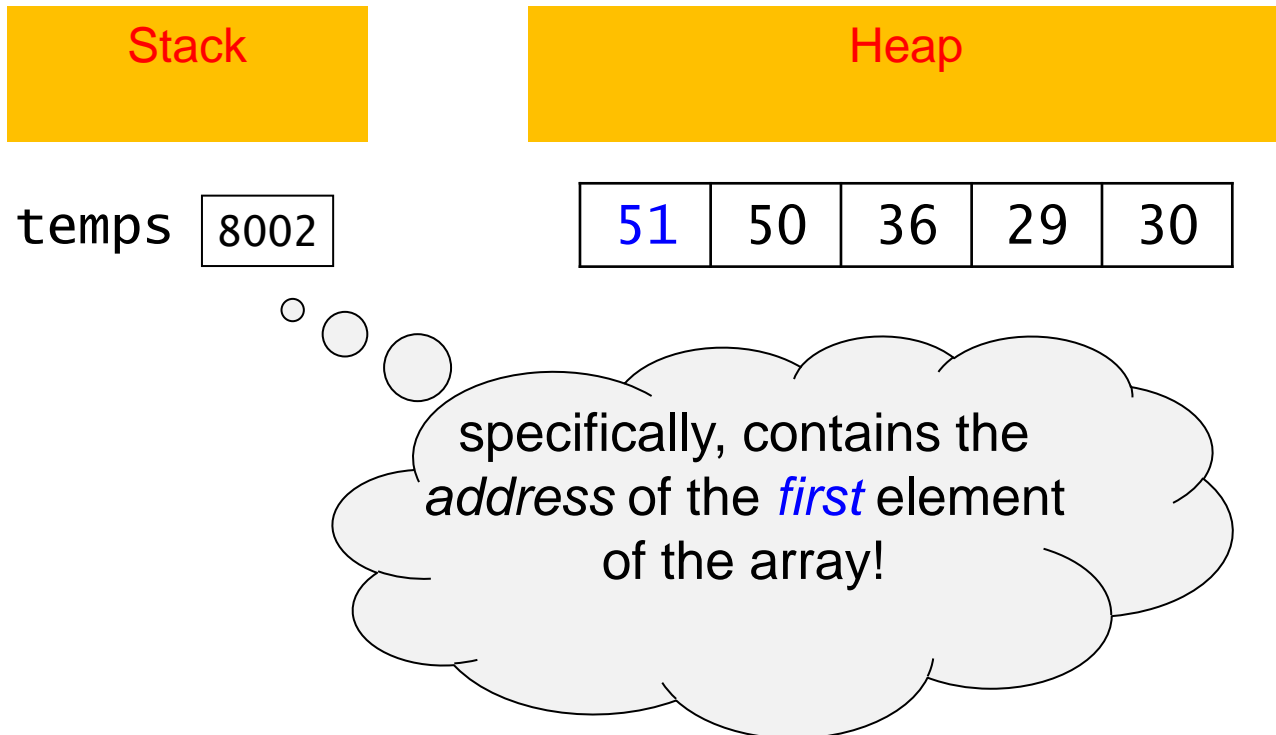
```
int[] temps = {51, 50, 36, 29, 30};
```



Arrays and References

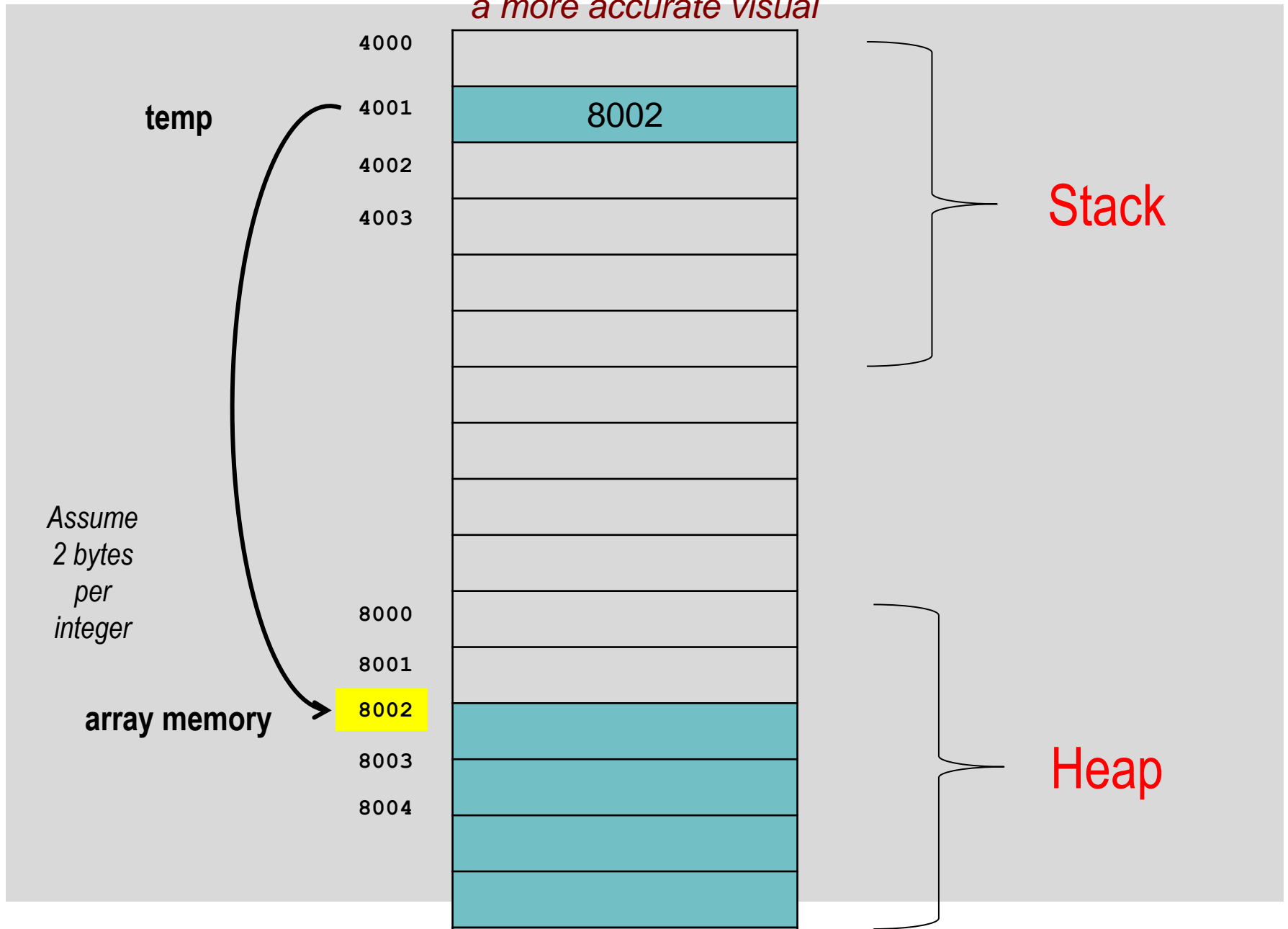
- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

```
int[] temps = {51, 50, 36, 29, 30};
```



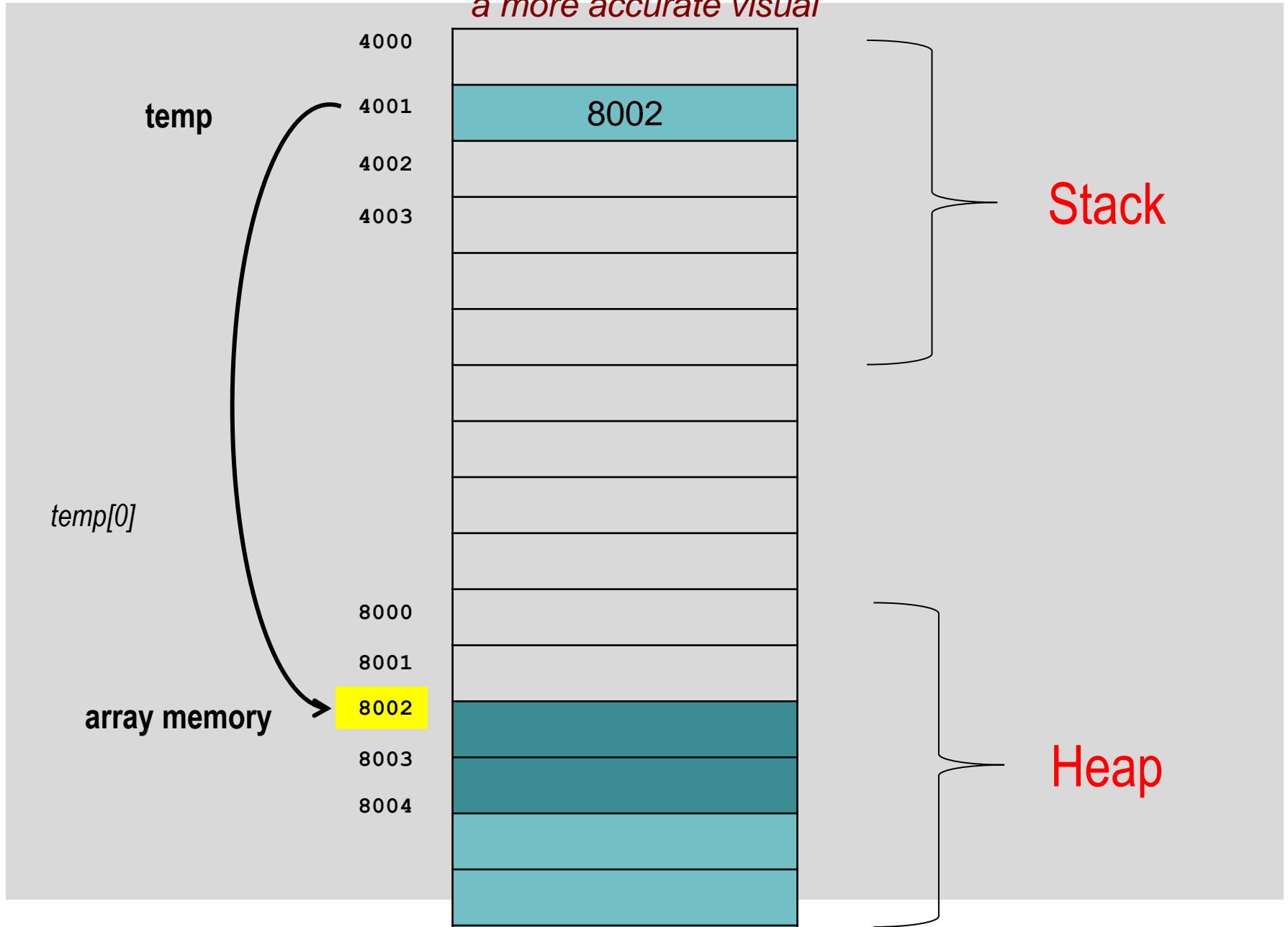
Arrays and References:

a more accurate visual



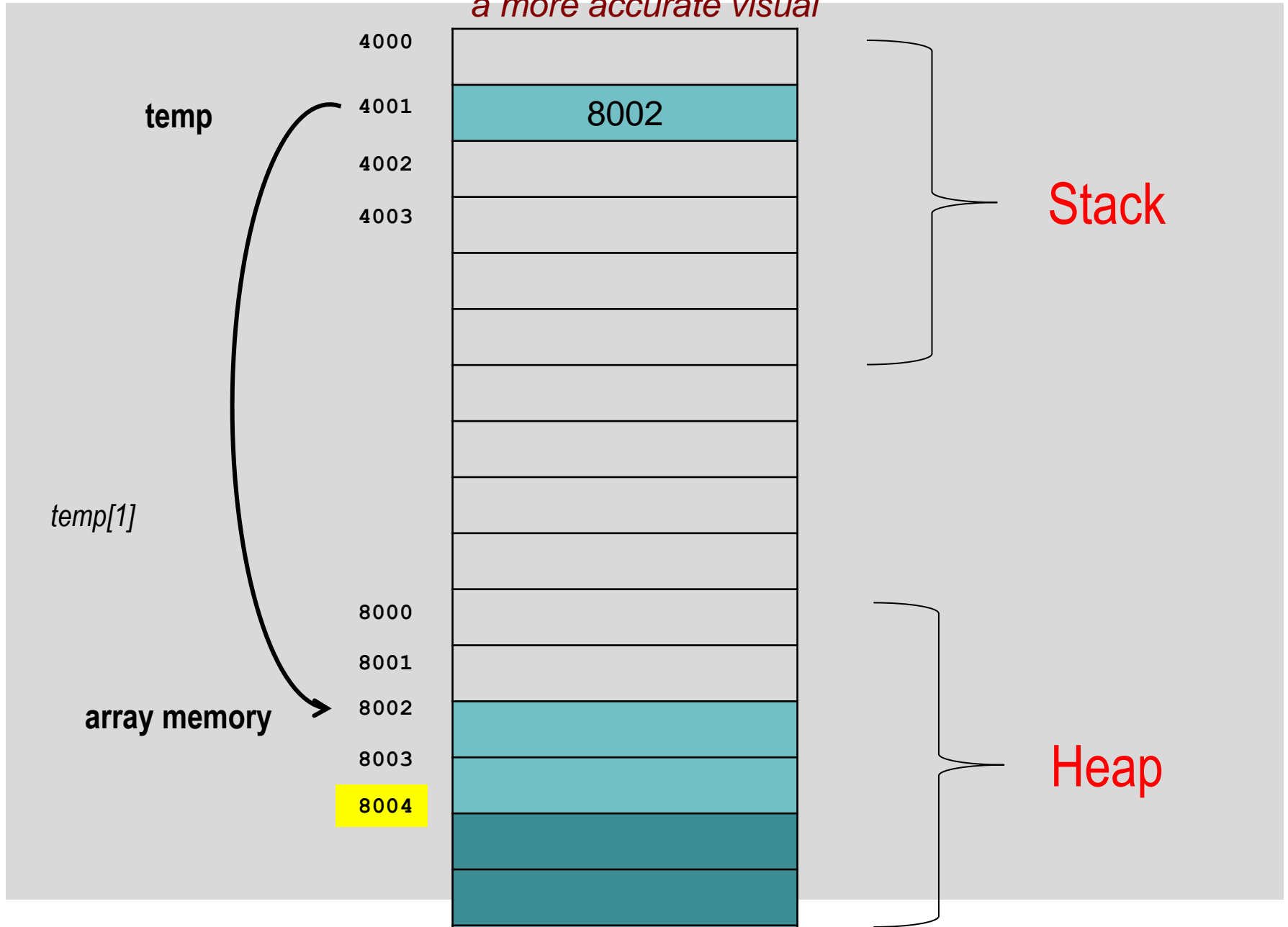
Arrays and References:

a more accurate visual



Arrays and References:

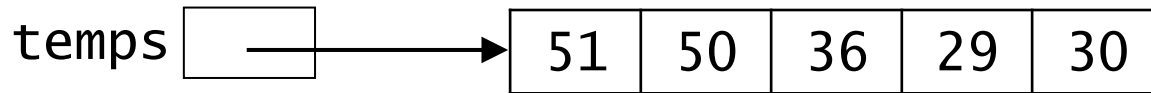
a more accurate visual



Arrays and References

- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

```
int[] temps = {51, 50, 36, 29, 30};
```



- If we print an array variable, we print the contents or value of the variable, which is the memory address of the array!

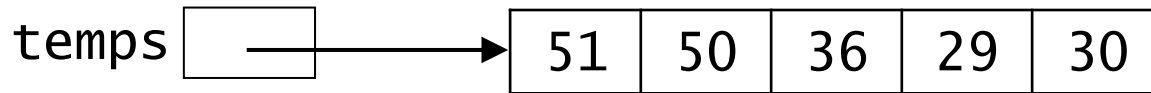
```
System.out.println(temps);
```

```
output:  
[I@1e1fd124
```


Arrays and References

- An array variable does *not* store the array itself.
- It stores a *reference* to the array.
 - the memory address of the array

```
int[] temps = {51, 50, 36, 29, 30};
```



- To print the contents of the array, we must first invoke a *static* method of the Array class that returns a string representation of the specified array:

```
System.println( Arrays.toString(temps) );
```

Processing a Sequence Using a Loop

Index-based:

```
for (int i = 0; i < array.length; i++) {  
    do something with array[i]  
}
```

where *array* is the array variable

Element-based:

```
for (int val: array) {  
    do something with val  
}
```

where *array* is the array variable

- Index-based is more flexible:
 - you can use it to *change* the element with index *i*
 - you can keep track of where you saw a given value

Processing a Sequence Using a Loop

Index-based:

```
for (int i = 0; i < array.length; i++) {  
    do something with array[i]  
}
```

. The loop iterates over the elements of the array, so variable type depends on the array data type.

array variable

Element-based:

```
for (int val: array) {  
    do something with val  
}
```

where *array* is the array variable

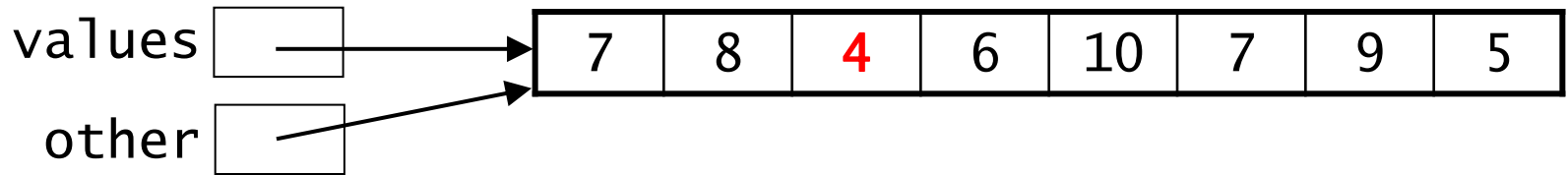
- Index-based is more flexible:
 - you can use it to *change* the element with index *i*
 - you can keep track of where you saw a given value

Array Assignment

Copying a Reference Variable

- What does this do?

```
int[] values = {7, 8, 9, 6, 10, 7, 9, 5};  
int[] other = values;
```



- Given the lines of code above, what will the lines below print?

```
other[2] = 4;  
System.out.println(values[2] + " " + other[2]);
```

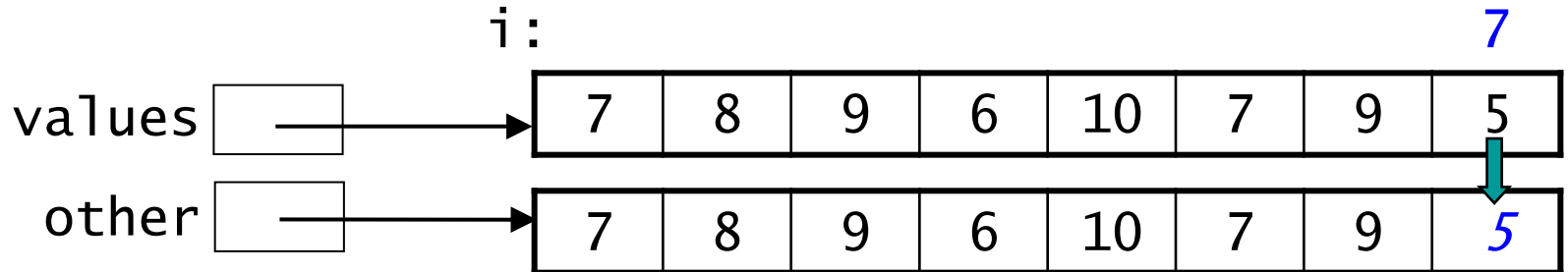
Shallow Copy

Copying an Array

- To actually create a **deep** copy of an array, we can:
 - create a new array of the same length as the first
 - traverse the arrays and copy the individual elements

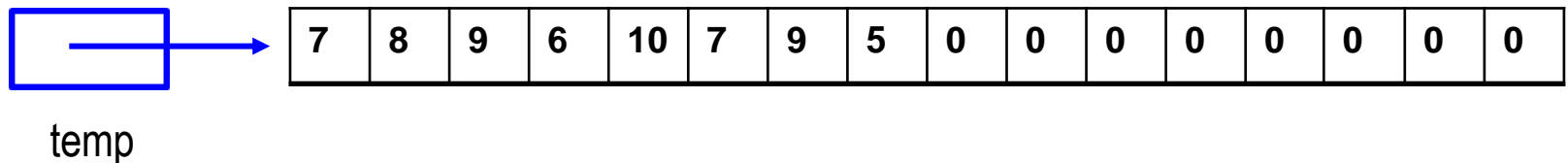
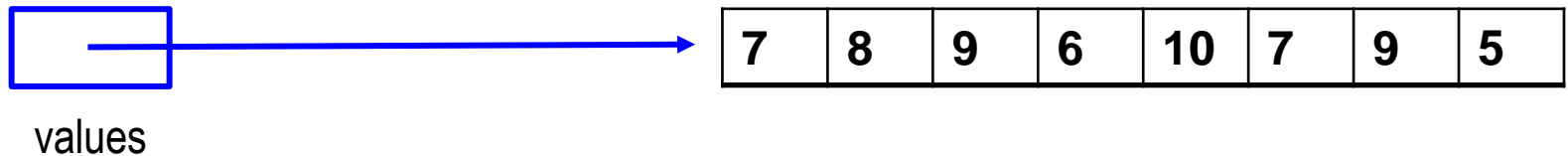
- Example:

```
int[] values = {7, 8, 9, 6, 10, 7, 9, 5};  
int[] other = new int[values.length];  
for (int i = 0; i < values.length; i++) {  
    other[i] = values[i];  
}
```



Example: Memory layout

```
int[] values = {7, 8, 9, 6, 10, 7, 9, 5};
...
int[] temp = new int[16];
for (int i = 0; i < values.length; i++) {
    temp[i] = values[i];
}
```



Example: Memory Layout Growing an Array

```
int[] values = {7, 8, 9, 6, 10}
...
int[] temp = new int[16];
for (int i = 0; i < values.length; i++)
    temp[i] = values[i];
}
```

*This
memory remains
“in limbo”
waiting for the
garbage collector
to free it!*



values



temp

7	8	9	6	10	7	9	5	0	0	0	0	0	0	0	0
---	---	---	---	----	---	---	---	---	---	---	---	---	---	---	---

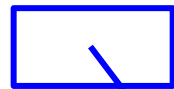
7	8	9	6	10	7	9	5
---	---	---	---	----	---	---	---

```
// re-assign the temporary array
values = temp
```

Example: Memory Layout

Growing an Array

```
int[] values = {7, 8, 9, 6, 10, 7, 9, 5};  
...  
int[] temp = new int[16];  
for (int i = 0; i < values.length; i++) {  
    temp[i] = values[i];  
}
```



values

7	8	9	6	10	7	9	5
---	---	---	---	----	---	---	---



temp

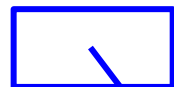
7	8	9	6	10	7	9	5	0	0	0	0	0	0	0	0
---	---	---	---	----	---	---	---	---	---	---	---	---	---	---	---

```
// re-assign the temporary array  
values = temp
```

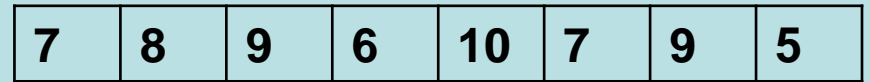

Example: Memory Layout

Growing an Array

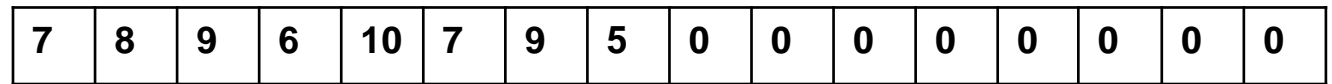
```
int[] values = {7, 8, 9, 6, 10, 7, 9, 5};  
...  
int[] temp = new int[values.length*2];  
for (int i = 0; i < values.length; i++) {  
    temp[i] = values[i];  
}
```



values



temp



```
// re-assign the temporary array  
values = temp
```

A 2-D Array

- A 2-D array, *an array of arrays*
- To declare a 2-D array we must specify both the row and the column dimension as:

```
int[][] twoDim = new int [5][4];
```

row dimension



The diagram illustrates the two dimensions of a 2D array. Two blue arrows originate from the labels 'row dimension' and 'column dimension' at the bottom. The 'row dimension' arrow points to the number '5' in the array declaration 'int [5][4];'. The 'column dimension' arrow points to the number '4' in the same declaration. Both labels and arrows are in blue.

column
dimension

A 2-D Array

- A 2-D array, *an array of arrays*
- To declare a 2-D array we must specify both the row and the column dimension as:

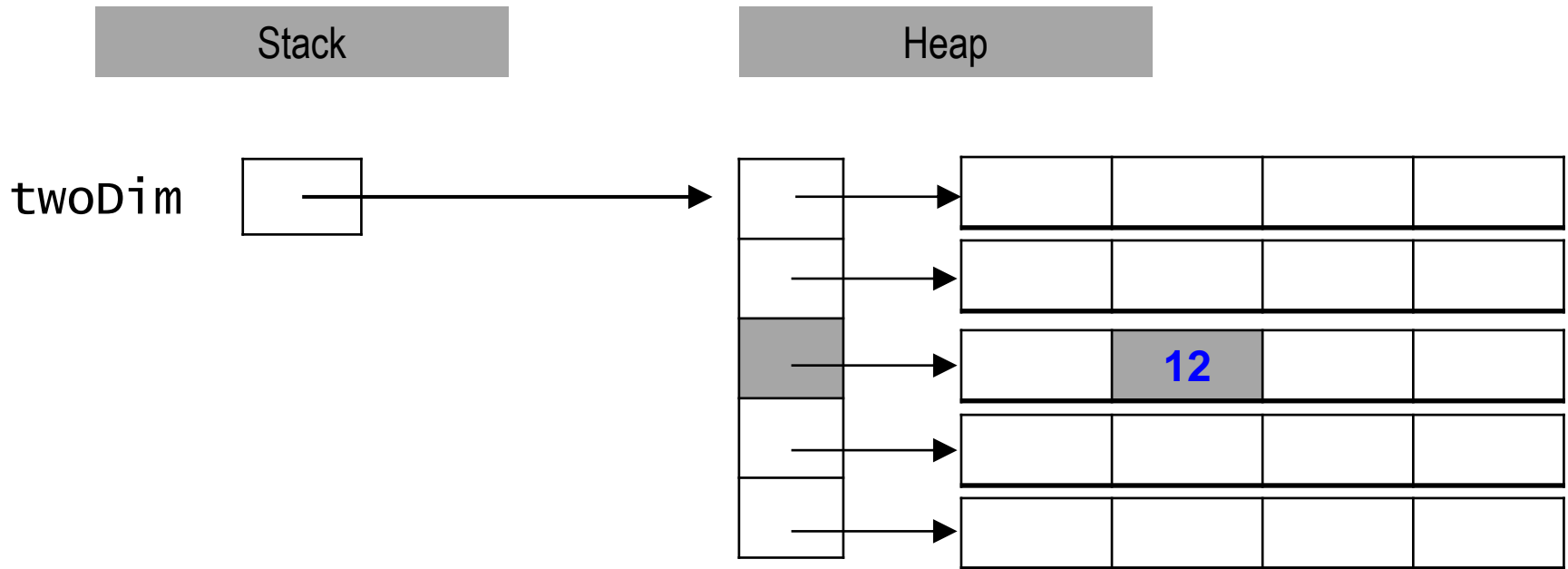
```
int[][] twoDim = new int [5][4];
```

- This statement is equivalent to:

```
int[][] twoDim = new int[5][];  
twoDim[0] = new int[4];  
twoDim[1] = new int[4];  
twoDim[2] = new int[4];
```

```
int[][] twoDim;  
twoDim = new int[5][4];
```

A 2-D Array



```
twoDim[2][1] = 12;
```

Maintaining a Game Board

- For a *simple* Tic-Tac-Toe board, we *could* use a 2-D array to keep track of the state of the board:

```
char[][] board = new char[3][3];
```

- Alternatively, we could create *and* initialize it as follows:

```
char[][] board = { {' ' , ' ' , ' ' , ' ' },  
                   { ' ' , ' ' , ' ' , ' ' },  
                   { ' ' , ' ' , ' ' , ' ' } };
```

- If a player puts an X in the middle square, we could record this fact by making the following assignment:

```
board[1][1] = 'X';
```

Processing All of the Elements in a 2-D Array

- To perform some operation on all of the elements in a 2-D array, we typically use a nested loop.
 - example: finding the maximum value in a 2-D array.

```
public static int maxValue(int[][] arr) {  
    int max = arr[0][0];  
    for (int r = 0; r < arr.length; r++) {  
        for (int c = 0; c < arr[r].length; c++) {  
            if (arr[r][c] > max) {  
                max = arr[r][c];  
            }  
        }  
    }  
    return max;  
}
```

Sources

Thanks to contributing slides from:

- David Sullivan, PhD

Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

Stack

Heap

s_var	10
-------	----

Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

c	
b	
a	
i	
args	

Heap

Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

c	
b	
a	
i	
args	

Heap

Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	11
-------	----

Stack

c	
b	
a	
i	10
args	

Heap

Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

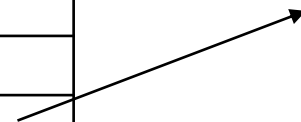
s_var	11
-------	----

Stack

c	
b	
a	
i	10
args	

Heap

2
4
6
8



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

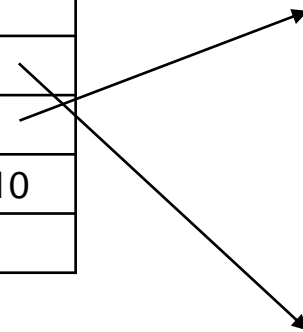
s_var	11
-------	----

Stack

c	
b	
a	
i	10
args	

Heap

2
4
6
8
0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

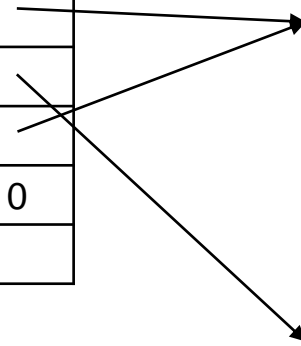
s_var	11
-------	----

Stack

c	
b	
a	
i	10
args	

Heap

2
4
6
8
0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

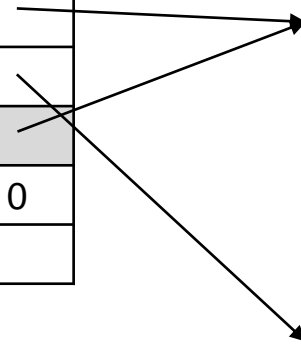
s_var	11
-------	----

Stack

c	
b	
a	
i	10
args	

Heap

2
4
6
8
0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	11
-------	----

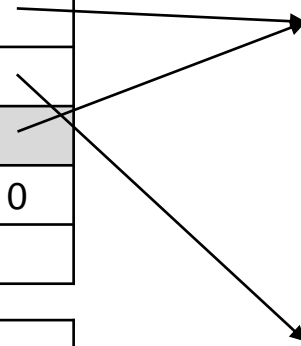
Stack

c	
b	
a	
i	10
args	

i	
b	
a	
ret	

Heap

2
4
6
8
0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	11
-------	----

Stack

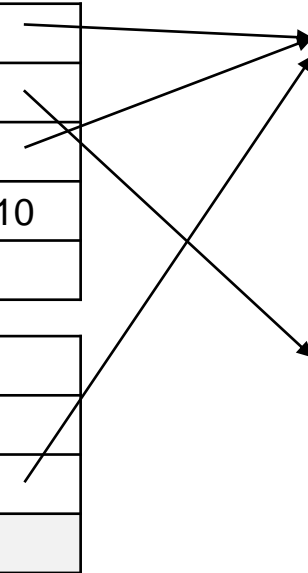
c	
b	
a	
i	10
args	

i	
b	
a	
ret	

Heap

2
4
6
8

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

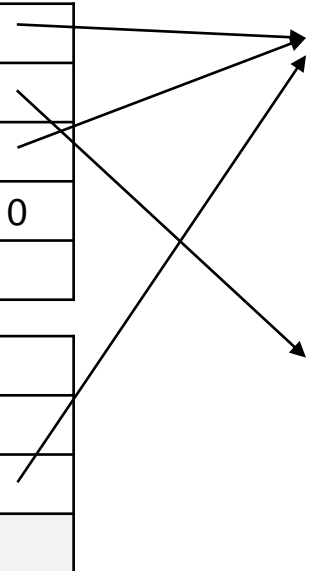
c	
b	
a	
i	10
args	

i	
b	
a	
ret	

Heap

2
4
6
8

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

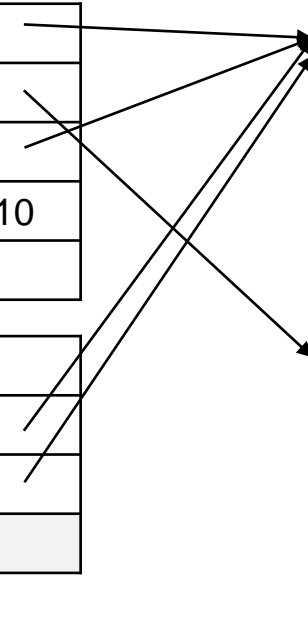
c	
b	
a	
i	10
args	

i	
b	
a	
ret	

Heap

2
4
6
8

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

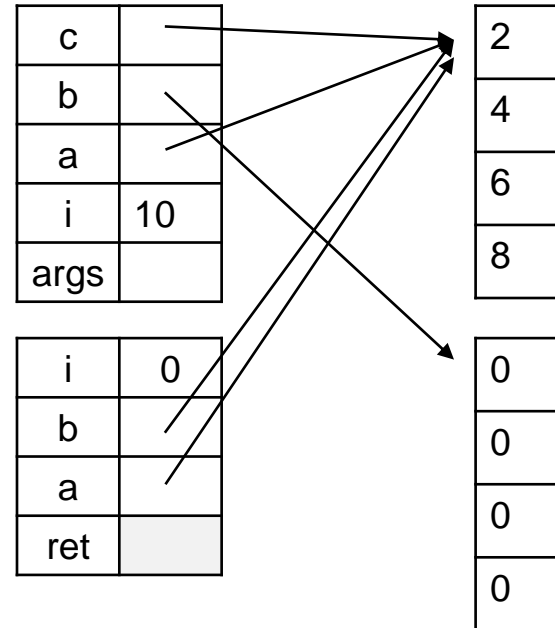
c	
b	
a	
i	10
args	

i	0
b	
a	
ret	

Heap

2
4
6
8

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {
```

```
    static int s_var = 10;
```

```
    public static void foo(int[] a) {
```

```
        s_var--;
```

```
        int[] b = a;
```

```
        for (int i = 0; i < b.length; i++) {
```

```
            b[i] = ++a[i];
```

```
        }
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        int i = s_var++;
```

```
        int[] a = {2, 4, 6, 8};
```

```
        int[] b = new int[a.length];
```

```
        int[] c = a;
```

```
        foo(a);
```

```
    }
```

```
}
```

Static

s_var	10
-------	----

Stack

c	
b	
a	
i	10
args	

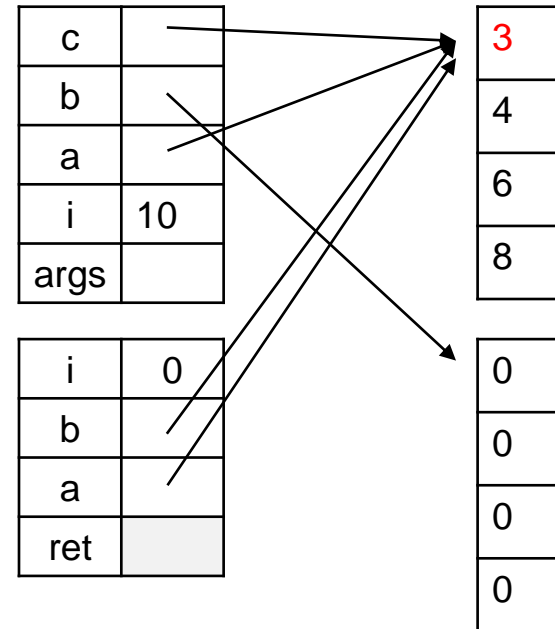
i	0
b	
a	
ret	

Heap

3
4
6
8

0
0
0
0

++a[i]
b[i] = a[i]



Understanding Java Arrays

```
public class ArrayTest {
```

```
    static int s_var = 10;
```

```
    public static void foo(int[] a) {
```

```
        s_var--;
```

```
        int[] b = a;
```

```
        for (int i = 0; i < b.length; i++) {
```

```
            b[i] = ++a[i];
```

```
        }
```

```
    }
```

```
    public static void main(String[] args) {
```

```
        int i = s_var--;
```

```
        int[] a = {2, 4, 6, 8};
```

```
        int[] b = new int[a.length];
```

```
        int[] c = a;
```

```
        foo(a);
```

```
    }
```

```
}
```

Static

s_var	10
-------	----

Stack

c	
b	
a	
i	10
args	

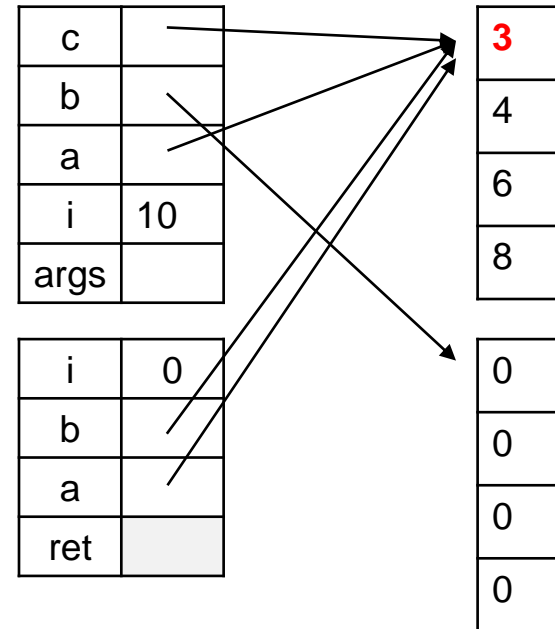
i	0
b	
a	
ret	

Heap

3
4
6
8

0
0
0
0

++a[i]
b[i] = a[i]



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

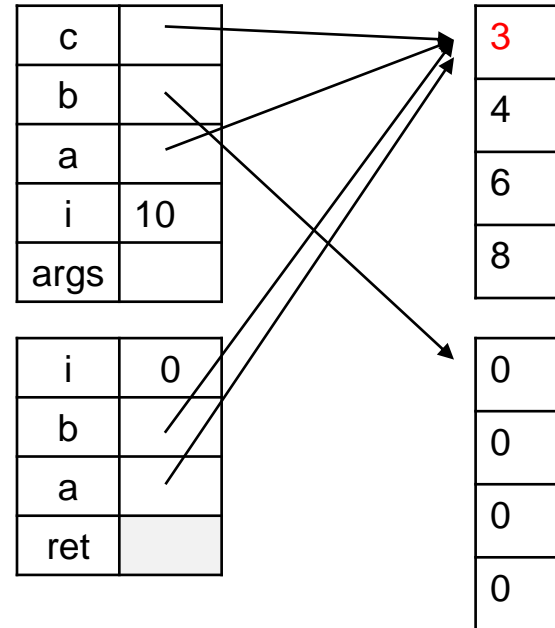
c	
b	
a	
i	10
args	

i	0
b	
a	
ret	

Heap

3
4
6
8

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

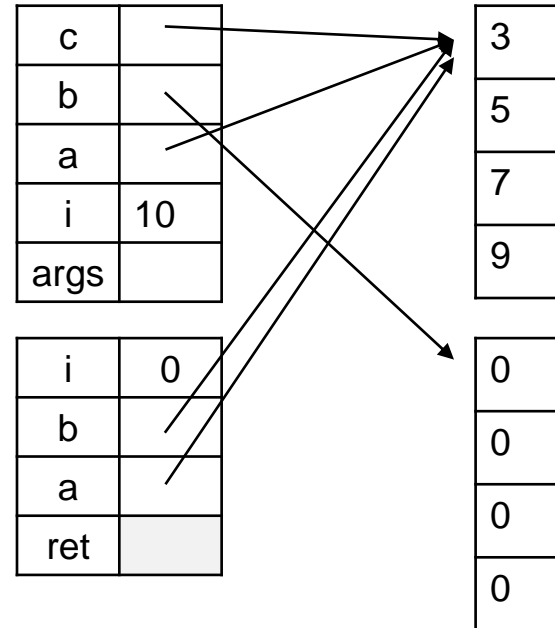
c	
b	
a	
i	10
args	

i	0
b	
a	
ret	

Heap

3
5
7
9

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

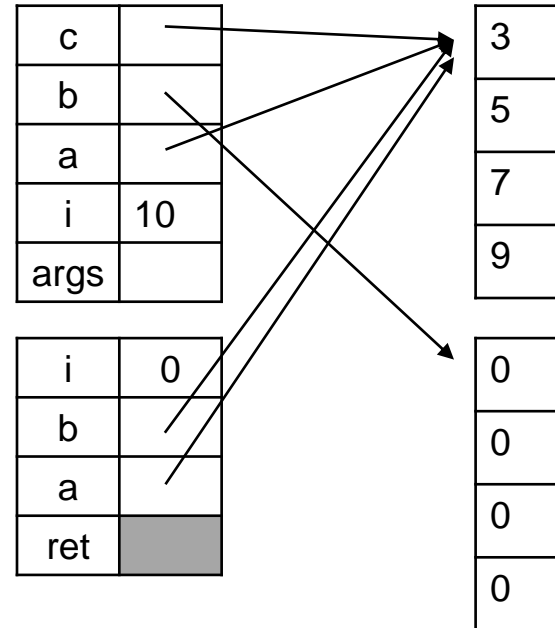
c	
b	
a	
i	10
args	

i	0
b	
a	
ret	

Heap

3
5
7
9

0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

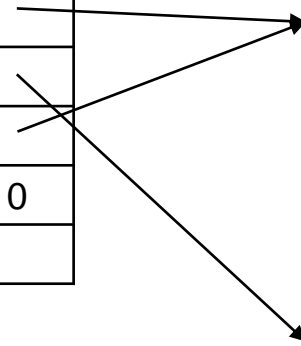
s_var	10
-------	----

Stack

c	
b	
a	
i	10
args	

Heap

3
5
7
9
0
0
0
0



Understanding Java Arrays

```
public class ArrayTest {  
  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

s_var	10
-------	----

Stack

Heap

3

5

7

9

0

0

0

0

Understanding Java Arrays

```
public class ArrayTest {  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

Stack

Heap

3

5

7

9

0

0

0

0

Understanding Java Arrays

```
public class ArrayTest {  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

Stack

Heap



3
5
7
9
0
0
0
0

Understanding Java Arrays

```
public class ArrayTest {  
    static int s_var = 10;  
  
    public static void foo(int[] a) {  
        s_var--;  
        int[] b = a;  
  
        for (int i = 0; i < b.length; i++) {  
            b[i] = ++a[i];  
        }  
    }  
  
    public static void main(String[] args) {  
        int i = s_var++;  
        int[] a = {2, 4, 6, 8};  
        int[] b = new int[a.length];  
        int[] c = a;  
  
        foo(a);  
    }  
}
```

Static

Stack

Heap

