

Parameter Passing



Connecting Caller and Callee

- When you call a method, how are the ***arguments*** connected to the ***formal parameters***?
- When the called method body ***returns***, how are results communicated back to the code that called the method?

Example: GCD

- Suppose we have a static method `gcd` that computes and returns the ***greatest common divisor (GCD)*** of two `ints`:

```
public static int gcd(int i, int j) {  
    ...  
}
```

- For example:

$$- \text{GCD}(24, 80) = 8$$

$$- \text{GCD}(24, 24) = 24$$

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

```
...  
int a, b;  
...  
int c = gcd(a, b);
```

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

This is the method `gcd` that is being called; `i` and `j` are its ***formal parameters***.

```
...  
int a, b;  
...  
int c = gcd(a, b);
```

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

This is a fragment of the calling program; `a` and `b` are the **arguments** to this call of `gcd`.

```
...  
int a, b;  
...  
int c = gcd(a, b);
```

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

Suppose the solid red arrow indicates where program flow-of-control has taken us so far.

```
→ ...  
int a, b;  
...  
int c = gcd(a, b);
```

How Calls Work In Java

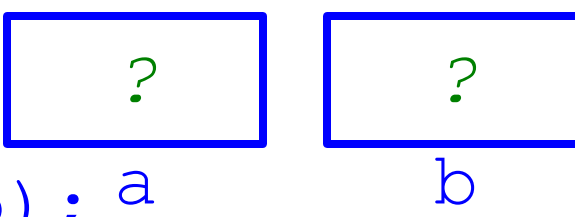
```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

```
...  
→ int a, b;  
...  
int c = gcd(a, b);
```


How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

```
...  
int a, b;  
...  
int c = gcd(a, b);
```



The diagram illustrates the state of variables during a function call. It shows two blue-outlined boxes, each containing a green question mark. The first box is positioned above the variable 'a' in the code line 'int c = gcd(a, b);', and the second box is positioned above the variable 'b'. A red arrow points from the left towards the 'int a, b;' line, indicating the point of variable declaration.

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

```
...  
int a, b;  
...  
→ int c = gcd(a, b);
```

24 80
a b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

The call to `gcd` begins ...

```
...  
int a, b;
```

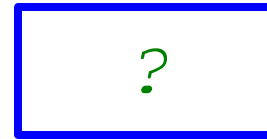
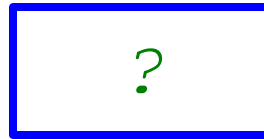
24
a

80
b

```
→ int c = gcd(a, b);
```

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```



i

j

... so the formal parameters are effectively declared ...

```
...  
int a, b;
```



24

80



```
...  
int c = gcd(a, b);
```

a

b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

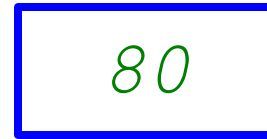


i

j

... and the argument values are **copied** to initialize them.

```
...  
int a, b;
```



24

80

```
...  
→ int c = gcd(a, b);
```

a

b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

24

i

80

j

Execution of the calling program is
“paused” at the point of the call...

```
...  
int a, b;
```

24

a

80

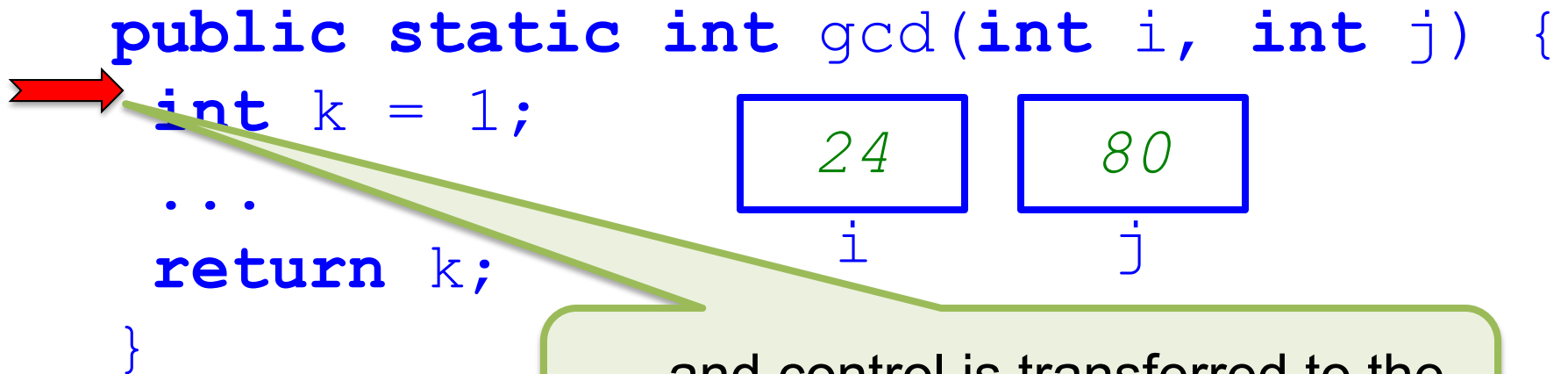
b



```
int c = gcd(a, b);
```

How Calls Work In Java

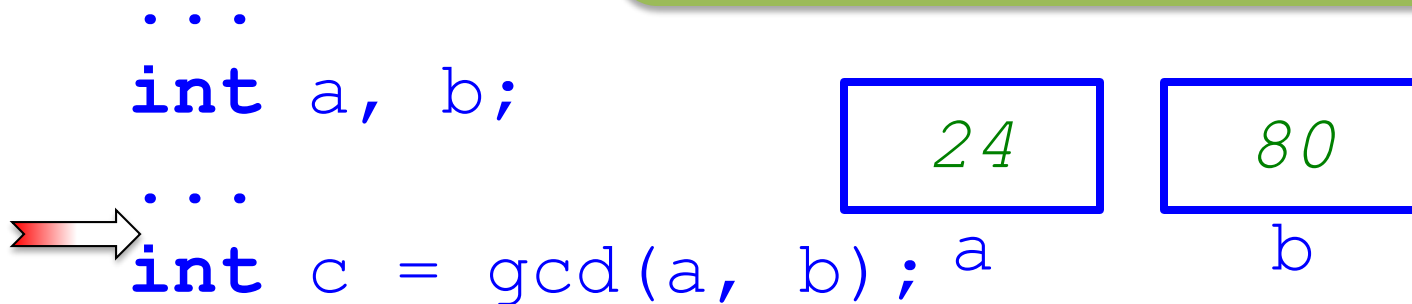
```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```



24 80
i j

... and control is transferred to the beginning of the method body.

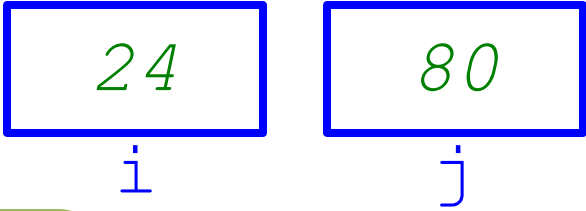
```
...  
int a, b;  
...  
int c = gcd(a, b);
```



24 80
a b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```



A diagram showing two blue-bordered boxes. The left box contains the number 24 and is labeled 'i' below it. The right box contains the number 80 and is labeled 'j' below it.

The **scope** of these variables is the calling program where they are declared.

```
int a, b;  
...  
int c = gcd(a, b);
```



A diagram showing two blue-bordered boxes. The left box contains the number 24 and is labeled 'a' below it. The right box contains the number 80 and is labeled 'b' below it.

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

The **scope** of these variables is the method body where they are declared.

24	80
i	j

```
int a, b;  
...  
int c = gcd(a, b);
```

24	80
a	b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

→

24	80	1
i	j	k

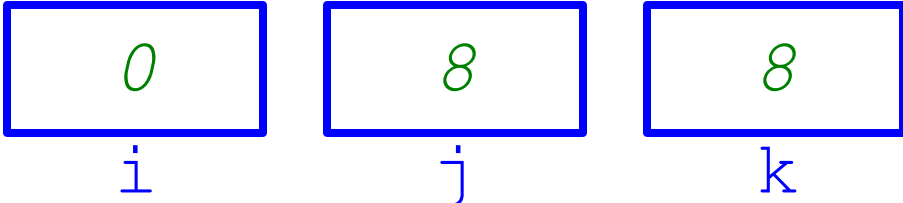
```
...  
int a, b;  
...  
int c = gcd(a, b);
```

→

24	80
a	b

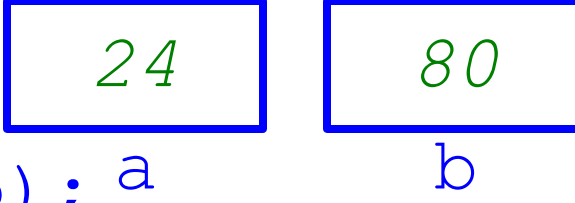
How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```



The diagram illustrates the state of variables during a function call. Three boxes represent the values of variables *i*, *j*, and *k*. The value of *i* is 0, *j* is 8, and *k* is 8. A red arrow points to the `return k;` statement in the code above.

```
...  
int a, b;  
...  
int c = gcd(a, b);
```



The diagram illustrates the state of variables during a function call. Two boxes represent the values of variables *a* and *b*. The value of *a* is 24 and *b* is 80. A red arrow points to the `gcd(a, b)` call in the code above.

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

0

i

8

j

8

k

The **return** statement immediately ends execution of method body...

```
...  
int a, b;
```

24

a

80

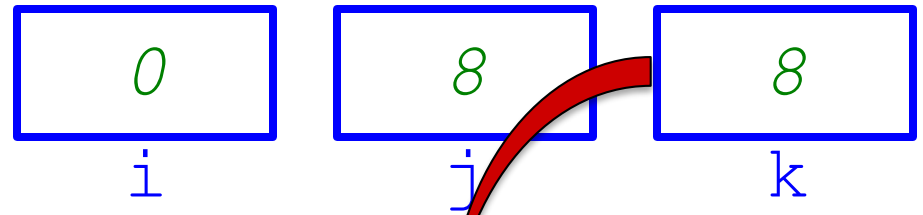
b



```
...  
int c = gcd(a, b);
```

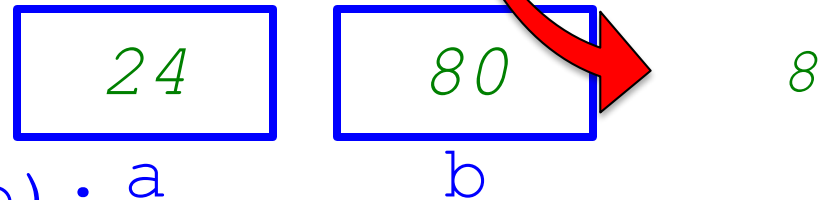
How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```



... so the returned value is **copied** back to the calling program ...

```
...  
int a, b;
```



```
→ int c = gcd(a, b);
```

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

... and the method body has finished, so its variables go away.

```
...  
int a, b;
```

24

80

8

```
→ ...  
int c = gcd(a, b);
```

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

Note that the values of the formal parameters are *not copied back* to the arguments!

```
...  
int a, b;  
...  
→ int c = gcd(a, b);
```

24 80 8
a b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

Execution of the calling program
“resumes” in mid-statement ...

```
...  
int a, b;  
...  
→ int c = gcd(a, b);
```

24 80 8
a b

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

... and the value that was returned by the call is assigned to `c`.

```
...  
int a, b;
```

```
→ int c = gcd(a, b);
```

24

80

8

a

b

c

How Calls Work In Java

```
public static int gcd(int i, int j) {  
    int k = 1;  
    ...  
    return k;  
}
```

```
...  
int a, b;
```

```
...
```

```
int c = gcd(a, b);
```

24

80

8

a

b

c



Connecting Caller and Callee

- When you call a method, how are the ***arguments*** connected to the ***formal parameters***?
 - The argument values are ***copied*** into the formal parameters to initialize them
- When the called method body ***returns***, how are results communicated back to the code that called the method?
 - *Only* the returned value is ***copied*** back to the caller; the formal parameters are simply “lost”

Names for This?

- Parameter-passing mechanism of Java:
 - May be termed ***call-by-copying*** because argument values are *copied* into formal parameters
 - May be termed ***call-by-value*** because argument *values* are copied into formal parameters
- There are other ways it might have been done (and is done in some languages)

Tracing Over a Call

Code	State
	$a = 24$ $b = 80$
<code>int c = gcd(a, b);</code>	
	$a = 24$ $b = 80$ $c = 8$