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 `int`s:

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

• For example:
  – `GCD(24, 80) = 8`
  – `GCD(24, 24) = 24`
How Calls Work In Java

```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

```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.

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

```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

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

```java
int a, b;
...
int c = gcd(a, b);
```
How Calls Work In Java

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

```java
... int a, b;
... int c = gcd(a, b); a ? ? b
```
How Calls Work In Java

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

... int a, b; ...

```java
int c = gcd(a, b); // Example input values: a = 24, b = 80
```
How Calls Work In Java

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

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

The call to `gcd` begins ...

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

24

80
public static int gcd(int i, int j) {
    int k = 1;
    ...
    return k;
}

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

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

... and the argument values are copied to initialize them.
How Calls Work In Java

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

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

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

int a, b;
...
int c = gcd(a, b);
How Calls Work In Java

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

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

```java
int a, b;
...
int c = gcd(a, b);
```
How Calls Work In Java

```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.

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

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

7 January 2019 OSU CSE 18
How Calls Work In Java

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

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

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

... and the method body has finished, so its variables go away.
How Calls Work In Java

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

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

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

Execution of the calling program “resumes” in mid-statement ...
How Calls Work In Java

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

... and the value that was returned by the call is assigned to \( c \).
public static int gcd(int i, int j) {
    int k = 1;
    ... 
    return k;
}

... 
int a, b;
...
int c = gcd(a, b); a 24 b 80 c 8
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

<table>
<thead>
<tr>
<th>Code</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$a = 24$</td>
</tr>
<tr>
<td></td>
<td>$b = 80$</td>
</tr>
<tr>
<td>int $c = \text{gcd}(a, b);$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$a = 24$</td>
</tr>
<tr>
<td></td>
<td>$b = 80$</td>
</tr>
<tr>
<td></td>
<td>$c = 8$</td>
</tr>
</tbody>
</table>