Static Methods
Static Method

- A *static method* (*class method*) is a block of code with a name, using which it can be *called (invoked)* to perform its computation.
- The method “takes over” execution when it is called, until it *returns* to the calling program at the point it was called.
- Also known as a routine, subroutine, operation, function, or procedure.
Anatomy of a Static Method

```java
private static int distance(
    int a, int b) {
    int d = b - a;
    return d;
}
```
Anatomy of a Static Method

private static int distance(int a, int b) {
    int d = b - a;
    return d;
}
Anatomy of a Static Method

```java
private static int distance(
    int a, int b) {
    int d = b - a;
    return d;
}
```

The *method body* consists of the block of code that is executed when the method is called.
Anatomy of a Static Method

private static int distance(
    int a, int b) {
    int d = b - a;
    return d;
}

private limits the places from which this static method may be called: only from within this class; public here would allow it to be called from other classes, too.
Anatomy of a Static Method

```java
private static int distance(int a, int b) {
    int d = b - a;
    return d;
}
```

*Int* means the method provides a value of this type to the caller when it returns; *void* here would mean the method provides no value at all.
Anatomy of a Static Method

```java
private static int distance(int a, int b) {
    int d = b - a;
    return d;
}
```

`distance` is the name of this static method, which is used when calling it.
Anatomy of a Static Method

```java
private static int distance(
    int a, int b) {
    int d = b - a;
    return d;
}
```

**int a** is the first *formal parameter* of the method, whose initial value is supplied at the point of the call; **int b** is the second formal parameter.
Anatomy of a Static Method

```java
private static int distance(int a, int b) {
    int d = b - a;
    return d;
}
```

`int d` is a **local variable** of the method.
private static int distance(
    int a, int b) {
    int d = b - a;
    return d;
}
Return Statements

- Every *path of execution* through a method that returns a value *must* end in a return statement with an expression of the return type of the method.

- A method that does not return a value *may* have return statement(s) without any such expression; but by default, it returns to the caller anyway when the method body completes execution.
Resources

• *Big Java Late Objects*, Chapter 5