Introducing Java Methods

Hui Chen

Department of Computer & Information Science

Brooklyn College

Objectives

- To define methods with formal parameters (§6.2).

- To invoke methods with actual parameters (i.e., arguments) (§6.2).

- To define methods with a return value (§6.3).

- To define methods without a return value (§6.4).

- To pass arguments by value (§6.5).

Outline

- Motivating example

- Defining and invoking value-returning methods

- Defining and invoking void methods

- Parameter passing and passing by value

- Pitfalls and errors

Motivating Problem. Compute the sum of integers

- Find the sum of integers from 1 to 10, from 20 to 30, and from 35 to 45, respectively.

Solution 1. Compute the sum of integers. Your thoughts?

```
int sum = 0;
for (int i = 1; i <= 10; i++)
   sum += i;
System.out.println("Sum from 1 to 10 is " + sum);


sum = 0;
for (int i = 20; i <= 30; i++)
   sum += i;
System.out.println("Sum from 20 to 30 is " + sum);


sum = 0;
for (int i = 35; i <= 45; i++)
   sum += i;
System.out.println("Sum from 35 to 45 is " + sum);
```

Critique on Solution 1. Compute the sum of integers.

```
int sum = 0;
for (int i = 1; i <= 10; i++)
   sum += i;
```
```
System.out.println("Sum from 1 to 10 is " + sum);
```

```
sum = 0;
for (int i = 20; i <= 30; i++)
   sum += i;
```
```
System.out.println("Sum from 20 to 30 is " + sum);
```

```
sum = 0;
for (int i = 35; i <= 45; i++)
   sum += i;
```
```
System.out.println("Sum from 35 to 45 is " + sum);
```

Solution 2. Introducing a Method

```java
public static int sum(int i1, int i2) {
  int sum = 0;
  for (int i = i1; i <= i2; i++)
    sum += i;
  return sum;
}
```

```java
public static void main(String[] args) {
  System.out.println("Sum from 1 to 10 is " + sum(1, 10));
  System.out.println("Sum from 20 to 30 is " + sum(20, 30));
  System.out.println("Sum from 35 to 45 is " + sum(35, 45));
}
```
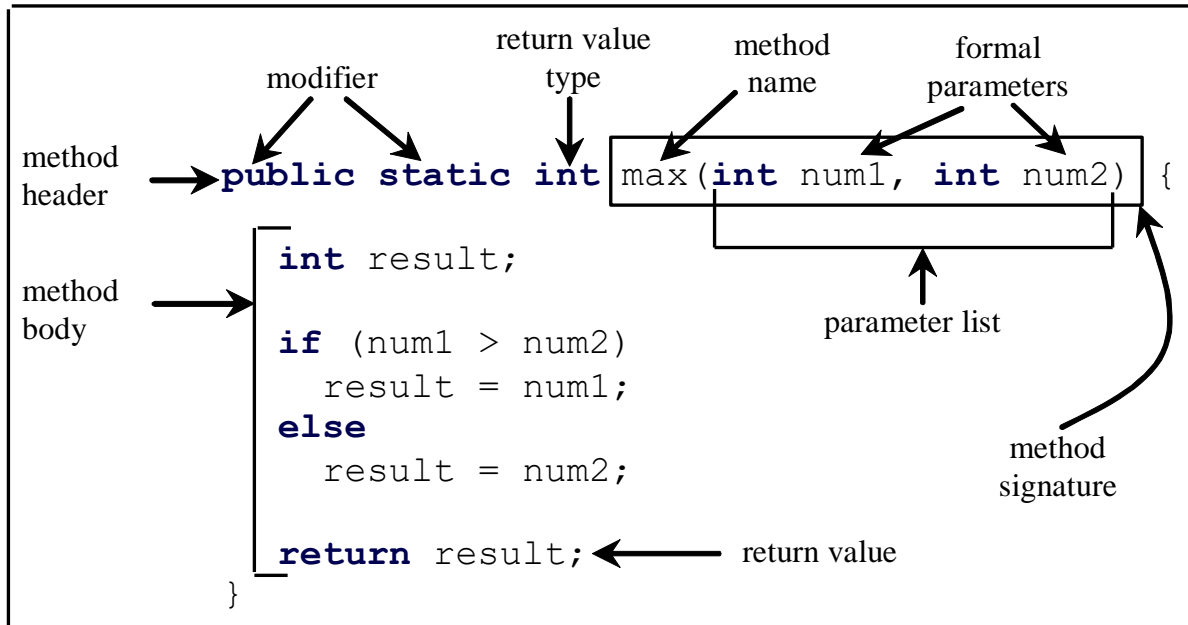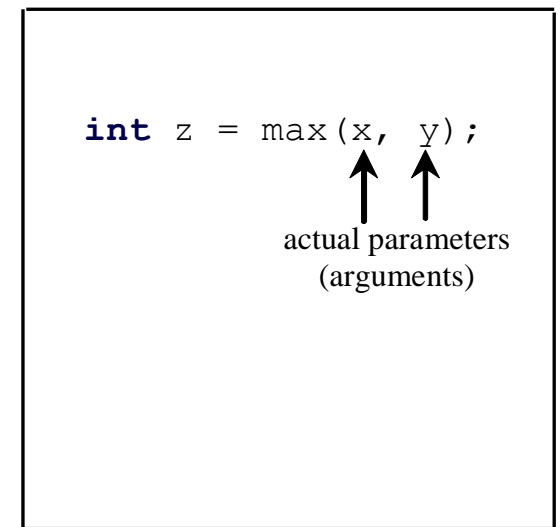
Questions?

Methods

- A method (functions, subroutines) is a collection of statements that are grouped together to perform an operation.

Defining and Invoking Methods

Define a method

```
                    return value      method        formal
      modifier          type          name        parameters

method
header    →  public static int  max(int num1, int num2)  {

method         int result;
body      →
               if (num1 > num2)
                 result = num1;
               else
                 result = num2;
                                                    method
               return result;  ←  return value    signature
             }
```

Invoke a method

```
   int z = max(x, y);

              actual parameters
                 (arguments)
```
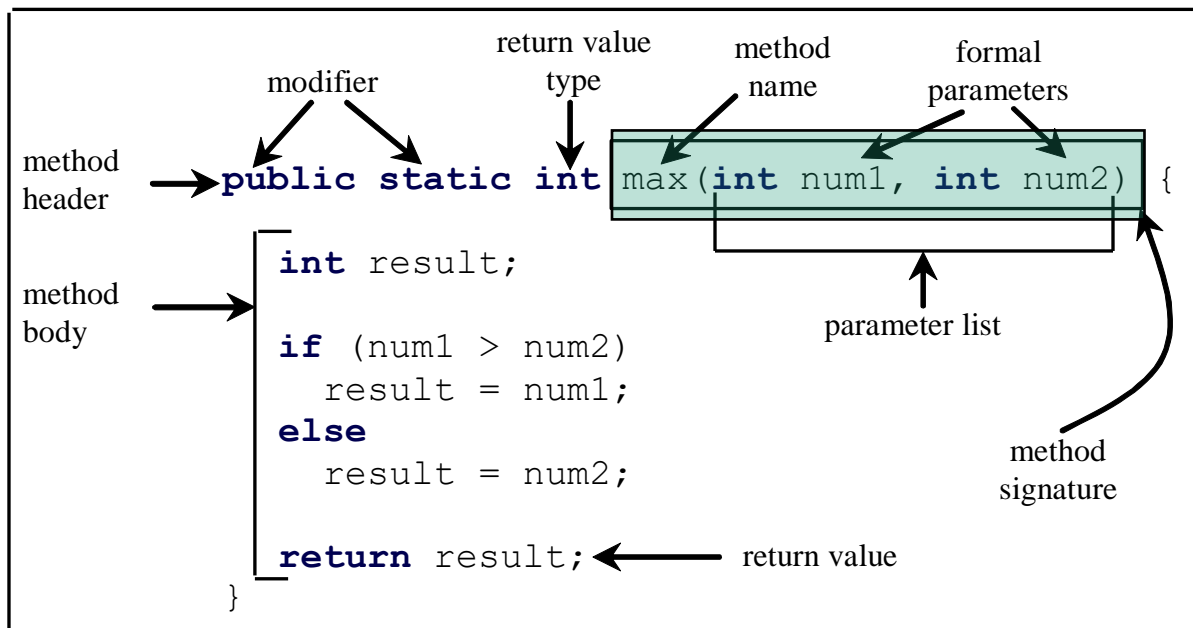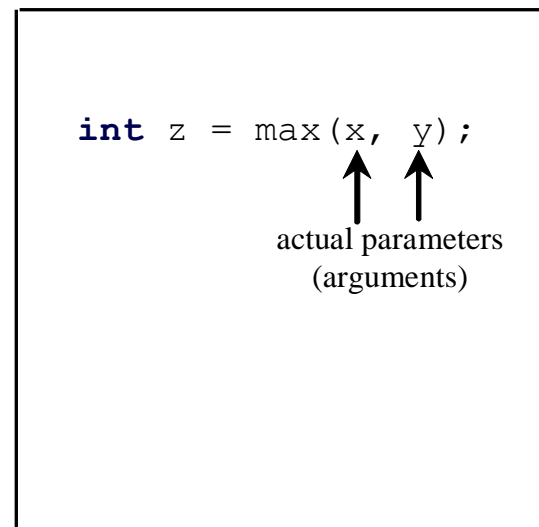
Method Signature

- *Method signature* is the combination of the method name and the parameter list.

Define a method

```
                              return value    method         formal
         modifier                  type        name        parameters

method
header   →  public static int max(int num1, int num2) {

method     int result;
body    →
           if (num1 > num2)
              result = num1;
           else
              result = num2;

           return result;  ←  return value
         }
```

parameter list

method signature

Invoke a method

```
int z = max(x, y);
```

actual parameters
(arguments)
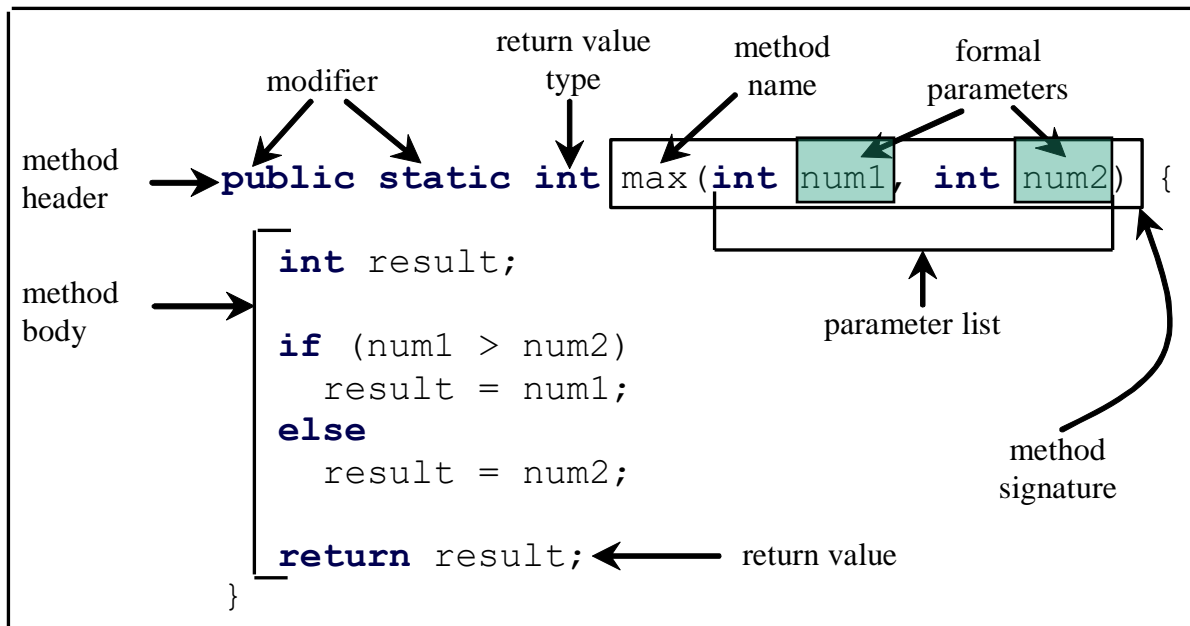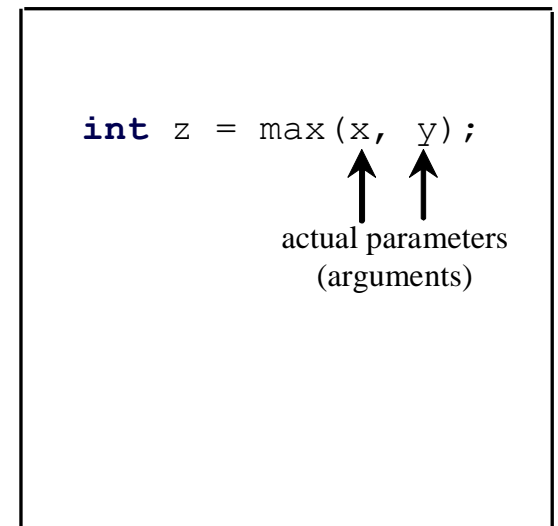
Formal Parameters

- The variables defined in the method header are known as *formal parameters*

Define a method


```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

- modifier
- return value type
- method name
- formal parameters
- method header
- method body
- parameter list
- method signature
- return value

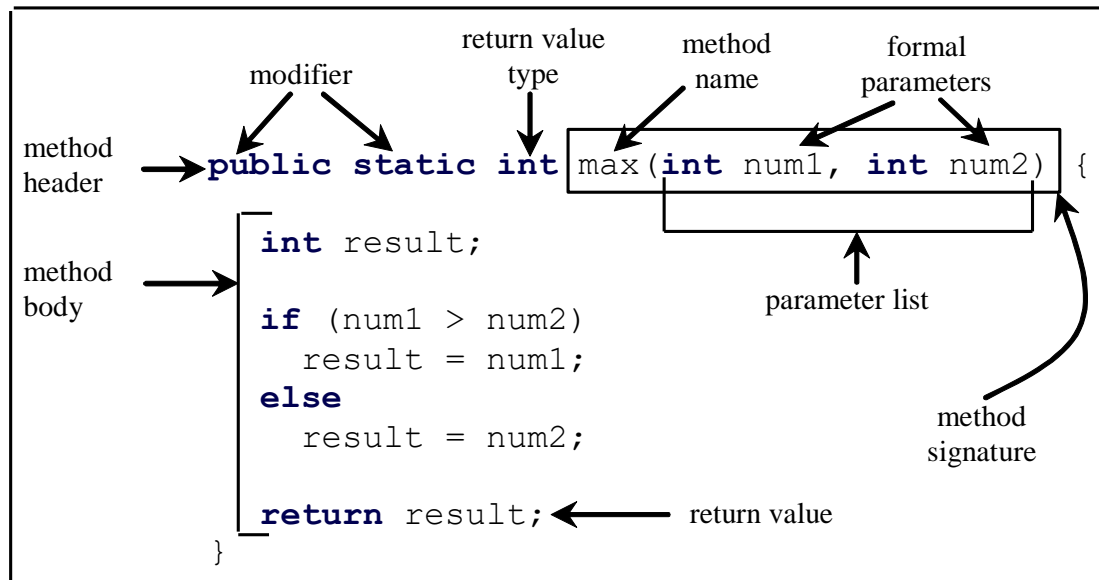Invoke a method

```
int z = max(x, y);
```
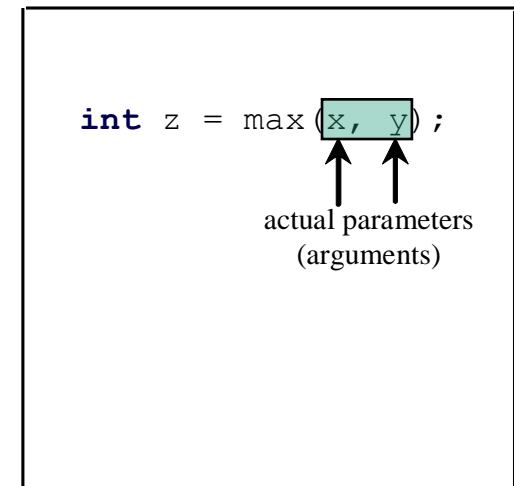
- actual parameters (arguments)

Actual Parameter (Argument)

- When a method is invoked, you pass a value to the parameter. This value is referred to as *actual parameter or argument*.

Define a method

Invoke a method

modifier   return value type   method name   formal parameters

method header → **public static int** max(**int** num1, **int** num2) {

method body →

```
int result;

if (num1 > num2)
   result = num1;
else
   result = num2;

return result;  ← return value
}
```

parameter list

method signature

`int z = max(x, y);`
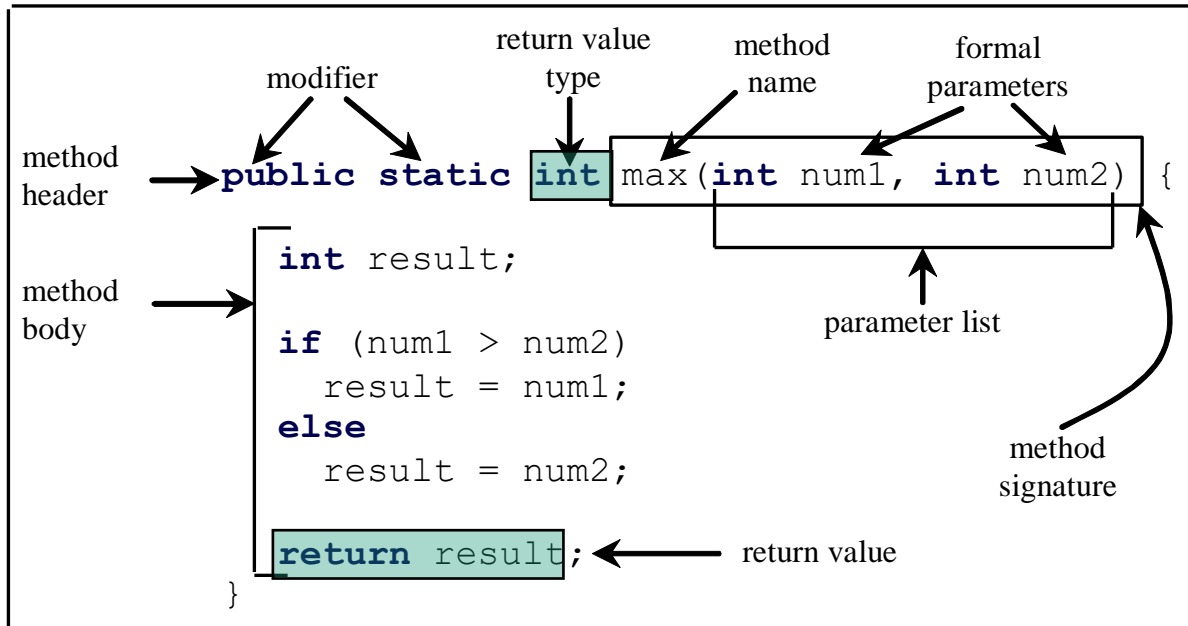
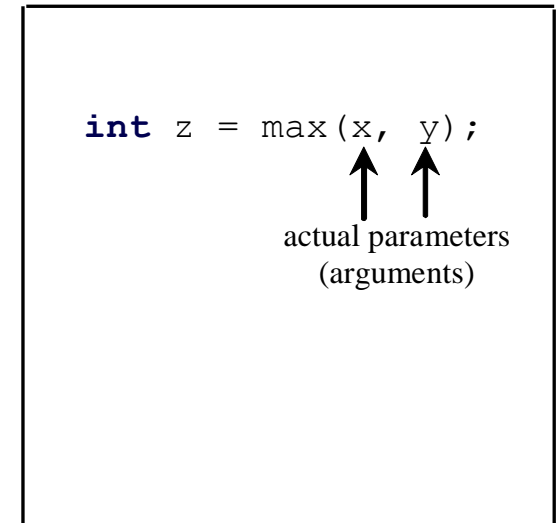actual parameters (arguments)

Return Value Type

- A method may return a value.

- The <u>returnValueType</u> is the data type of the value the method returns.

- If the method does not return a value, the <u>returnValueType</u> is the keyword <u>void</u>.

  - For example, the <u>returnValueType</u> in the <u>main</u> method is <u>void</u>.

Return Value Type

Define a method

Invoke a method

```
int z = max(x, y);
```

method header
method body

```
public static int max(int num1, int num2) {

    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

modifier → return value type → method name → formal parameters

parameter list

method signature

return value

actual parameters (arguments)

Calling/Invoking Methods

pass the value of i

pass the value of j

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);

    System.out.println(
        "The maximum between " + i +
        " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

Call Stacks

(a) The main method is invoked.

(b) The max method is invoked.

(c) The max method is being executed.

(d) The max method is finished and the return value is sent to k.

(e) The main method is finished.

Errors and Pitfalls

- A <u>return</u> statement is required for a value-returning method.

Is there an error? Where is it and Why?

```java
public static int sign(int n) {
    if (n > 0)
        return 1;
    else if (n == 0)
        return 0;
    else if (n < 0)
        return -1;
}
```

A value-returning method must return a value

- The method shown below in (a) is logically correct, but it has a compilation error because the Java compiler thinks it possible that this method does not return any value

```java
public static int sign(int n) {
    if (n > 0)
        return 1;
    else if (n == 0)
        return 0;
    else if (n < 0)
        return -1;
}
```

Correction

- To fix this problem, delete *if (n < 0)* in (a), so that the compiler will see a <u>return</u> statement to be reached regardless of how the <u>if</u> statement is evaluated

```
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else if (n < 0)
    return -1;
}
```
(a)

Should be →

```
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else
    return -1;
}
```
(b)

Questions?

void Method

- This type of method does not return a value. The method performs some actions

Problem. Given points, print letter grade

- Write a program that prompts the user to enter a point, and print the letter grade in a method

Questions?

Passing Parameters

```java
public static void nPrintln(String message, int n) {

   for (int i = 0; i < n; i++)

      System.out.println(message);

}
```

Suppose you invoke the method using
   nPrintln("Welcome to Java", 5);
What is the output?

Suppose you invoke the method using
   nPrintln("Computer Science", 15);
What is the output?

Can you invoke the method using
   nPrintln(15, "Computer Science");
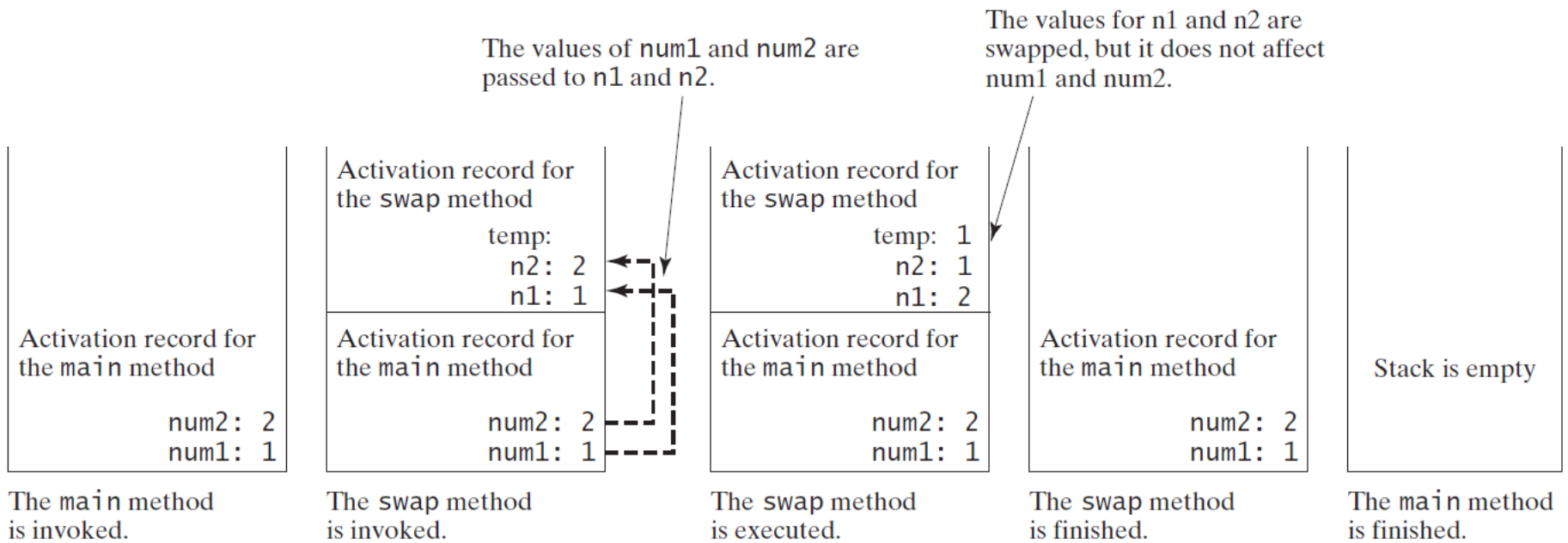
Pass By Value

- Java passes a parameter to a method by the value of the parameter.

Observe Output of Following Program

```java
public class Increment {
 public static void main(String[] args) {
   int x = 1;
   System.out.println("Before the call, x is " + x);
   increment(x);
   System.out.println("After the call, x is " + x);
 }

 public static void increment(int n) {
   n++;
   System.out.println("n inside the method is " + n);
 }
}
```

Why?

- Observe Call Stack

The values of num1 and num2 are passed to n1 and n2.

The values for n1 and n2 are swapped, but it does not affect num1 and num2.

Activation record for the swap method	Activation record for the swap method
temp: n2: 2 n1: 1	temp: 1 n2: 1 n1: 2

Activation record for the main method num2: 2 num1: 1	Activation record for the main method num2: 2 num1: 1	Activation record for the main method num2: 2 num1: 1	Activation record for the main method num2: 2 num1: 1	Stack is empty
The main method is invoked.	The swap method is invoked.	The swap method is executed.	The swap method is finished.	The main method is finished.

Questions