INTRODUCTION TO PROGRAMMING WITH JAVA - CEJV416

Lecture #7

Methods

The for loop

```
for(initializationExpression; booleanExpression;
incrementExpression)
{
    statements
}
```
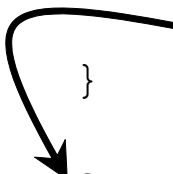
Caution

Adding a semicolon at the end of the <u>for</u> clause before the loop body is a common mistake, as shown below:

Logic
Error

```
for (int i=0; i<10; i++);
{

  System.out.println("i is " + i);

}
```

break

```java
public class TestBreak {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      sum += number;
      if (sum >= 100)
        break;
    }

    System.out.println("The number is " + number);
    System.out.println("The sum is " + sum);
  }
}
```

The syntax of the break statement

```
break;
```

A break statement that exits the inner loop

```java
for (int i = 1; i < 4; i++)
{
    System.out.println("Outer " + i);
    while (true)
    {
        int number = (int) (Math.random() * 10);
        System.out.println("   Inner " + number);
        if (number > 7)
            break;
    }
}
```

How to code break statements

- To jump to the end of the current loop, you can use the break statement.

A labeled break statement that exits the outer loop
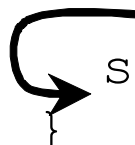
```
outerLoop:
for (int i = 1; i < 4; i++)
{
    System.out.println("Outer " + i);
    while (true)
    {
        int number = (int) (Math.random() * 10);
        System.out.println("   Inner " + number);
        if (number > 7)
            break outerLoop;
    }
}
```

How to code labeled break statements

- To jump to the end of an outer loop from an inner loop, you can *label* the outer loop and use the labeled break statement.

continue

```java
public class TestContinue {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      if (number == 10 || number == 11)
        continue;
      sum += number;
    }

    System.out.println("The sum is " + sum);
  }
}
```

The syntax of the continue statement

```
continue;
```

A continue statement that jumps to the beginning of a loop

```java
for (int j = 1; j < 10; j++)
{
    int number = (int) (Math.random() * 10);
    System.out.println(number);
    if (number <= 7)
        continue;
    System.out.println("This number is greater than 7");
}
```
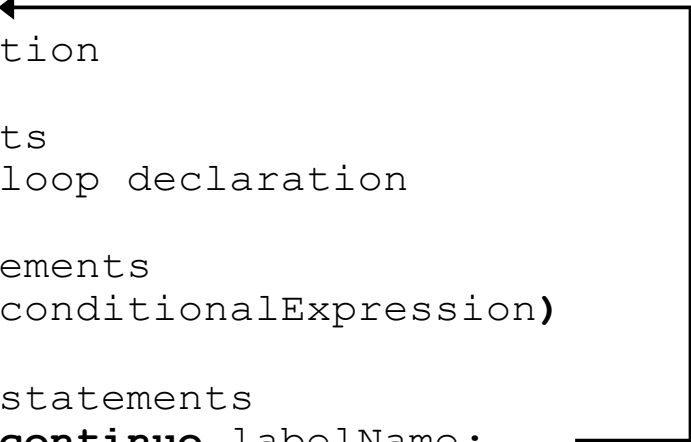
How to code continue statements

- To skip the rest of the statements in the current loop and jump to the top of the current loop, you can use the continue statement.

The syntax of the labeled continue statement

```
continue labelName;
```

The structure of the labeled continue statement

```
labelName:  ◄─────────────────────────────────┐
loop declaration                               │
{                                              │
    statements                                 │
    another loop declaration                   │
    {                                          │
        statements                             │
        if (conditionalExpression)             │
        {                                      │
            statements                         │
            continue labelName;  ──────────────┘

        }

    }

}
```

A labeled continue statement that jumps to the beginning of the outer loop

```
outerLoop:
for(int i = 1; i < 20; i++)
{
    for(int j = 2; j < i-1; j++)
    {
        int remainder = i%j;
        if (remainder == 0)
            continue outerLoop;
    }
    System.out.println(i);
}
```

Opening Problem

Find the sum of integers from <u>1</u> to <u>10</u>, from <u>20</u> to <u>30</u>, and from <u>35</u> to <u>45</u>, respectively.

Problem

```
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);
```

Problem

```java
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);
```
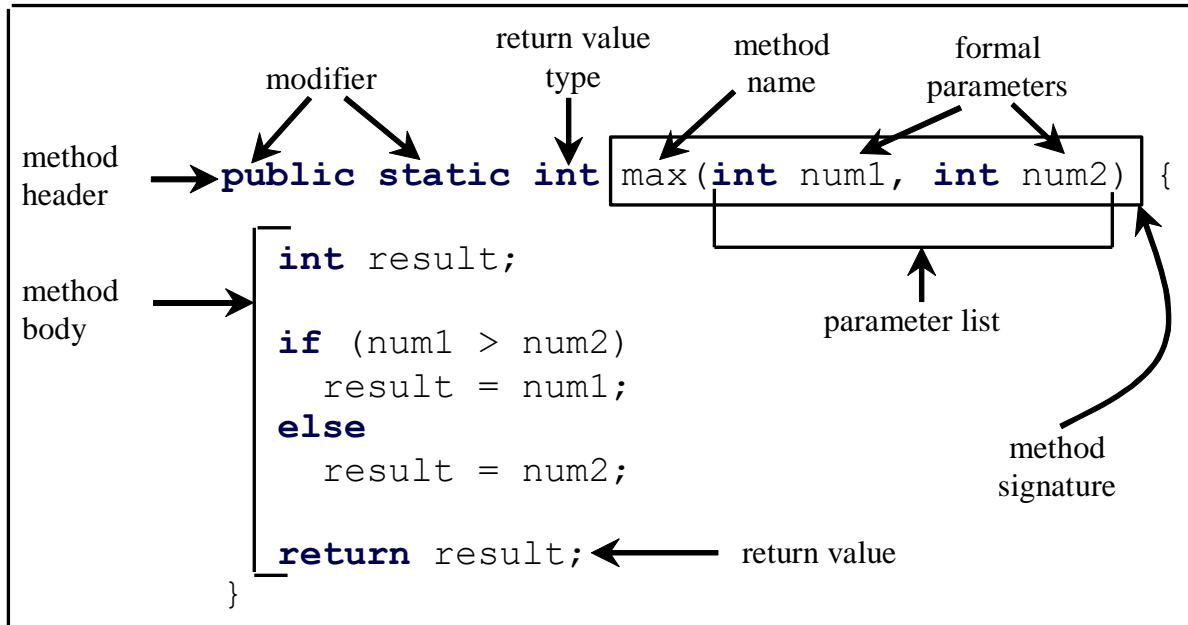
Solution

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

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));
}
```
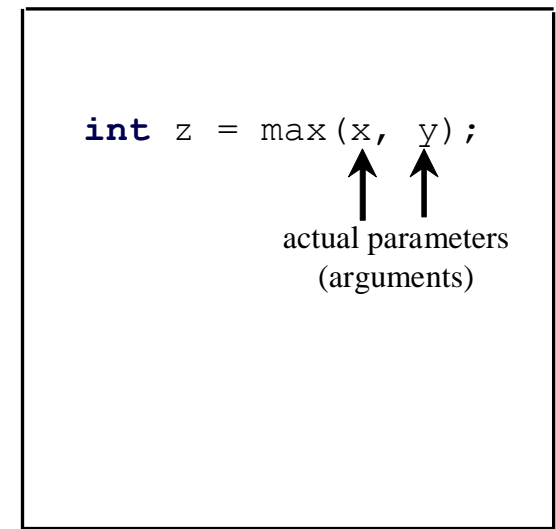
Defining Methods

A method is a collection of statements that are grouped together to perform an operation.

Define a method

```
                        return value      method        formal
        modifier          type            name         parameters

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

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

parameter list

Invoke a method

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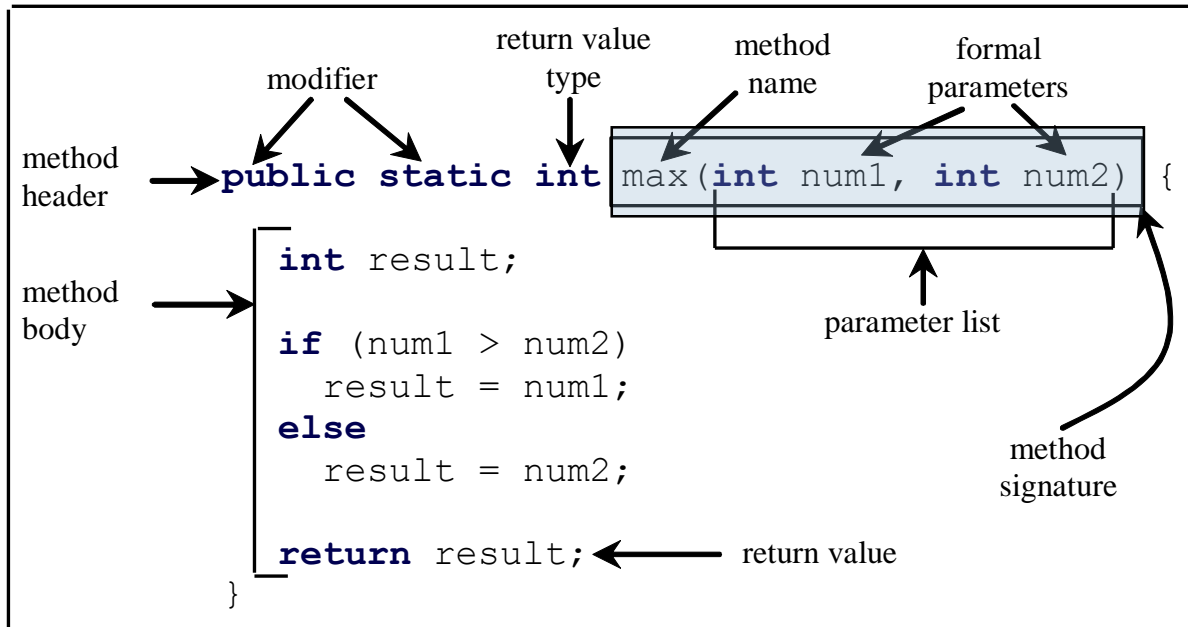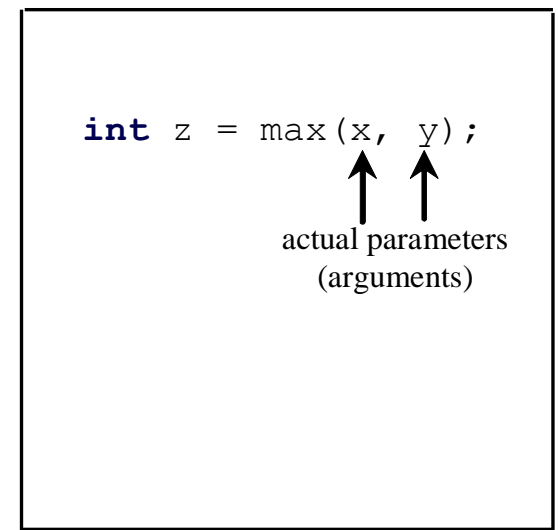
Method Signature

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

Define a method

```
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);
```
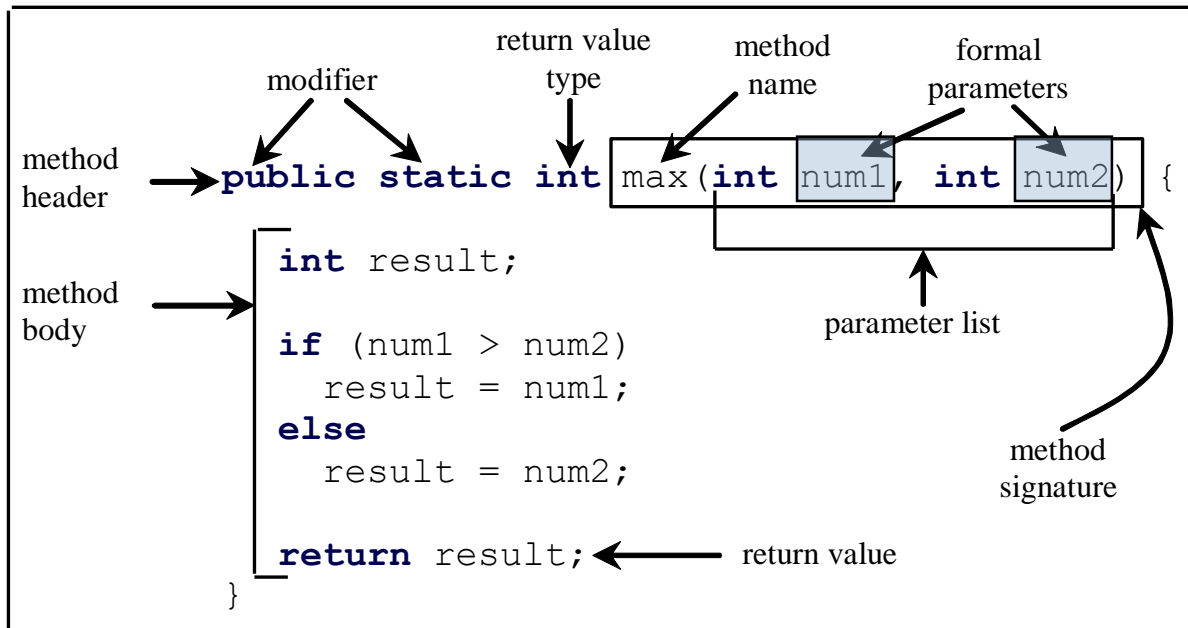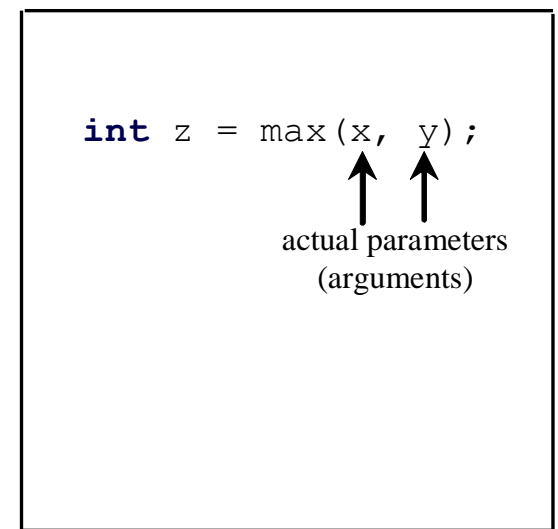
- actual parameters (arguments)

Formal Parameters

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

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
                                                   signature
            return result;  ←  return value
          }
```
parameter list

Invoke a method

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

                        actual parameters
                           (arguments)
```
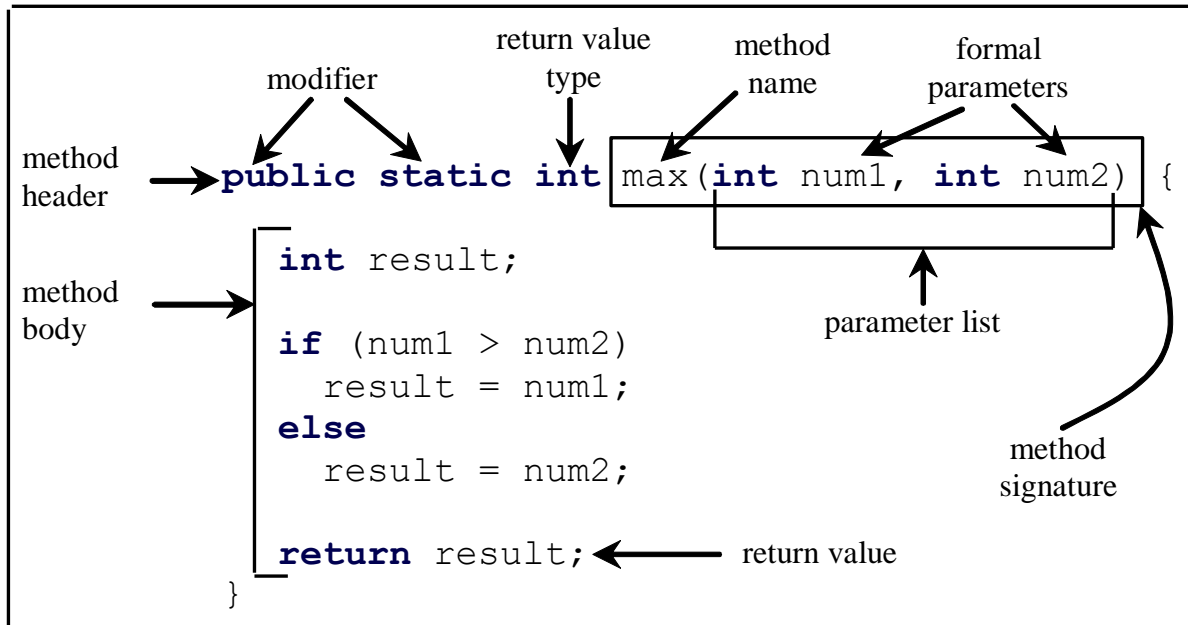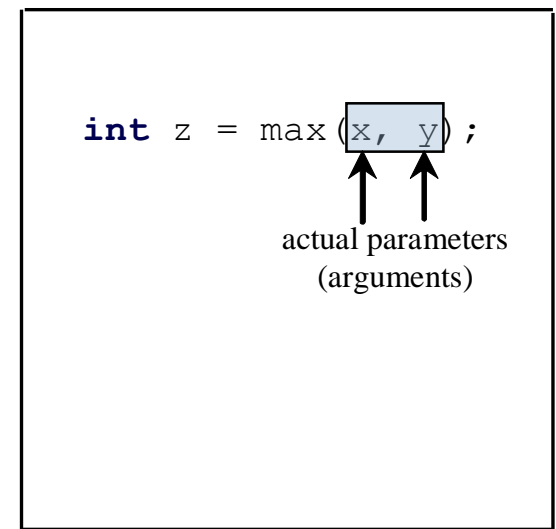
Actual Parameters

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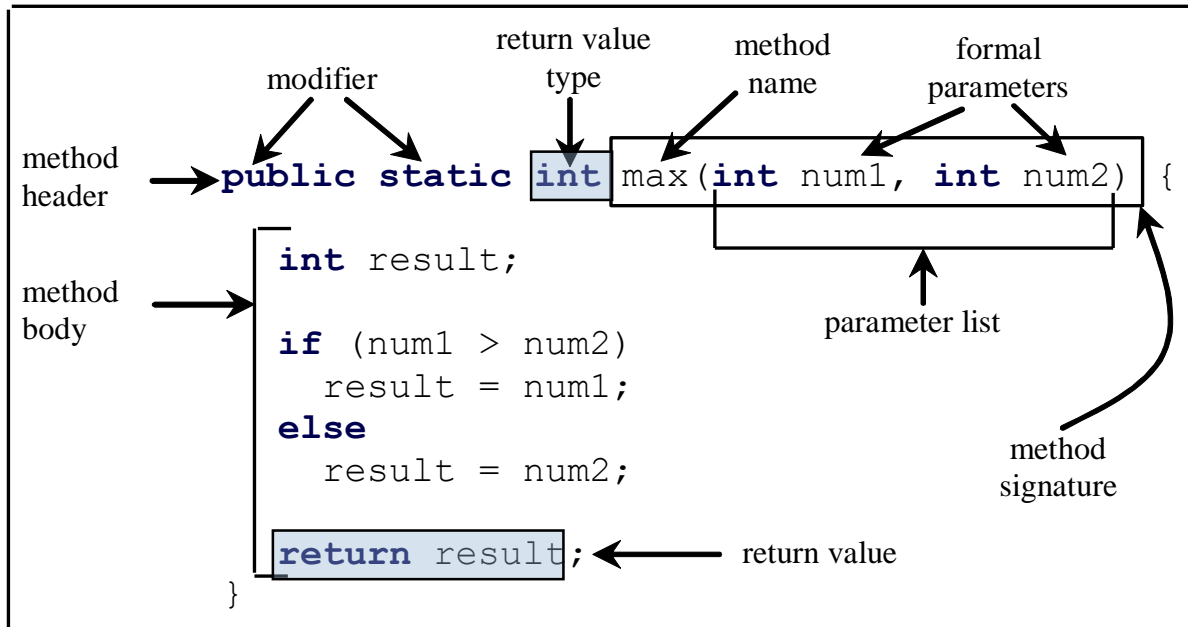
Define a method

Invoke a method


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

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

    return result;
}
```

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

- modifier
- return value type
- method name
- formal parameters
- method header
- parameter list
- method body
- method signature
- return value
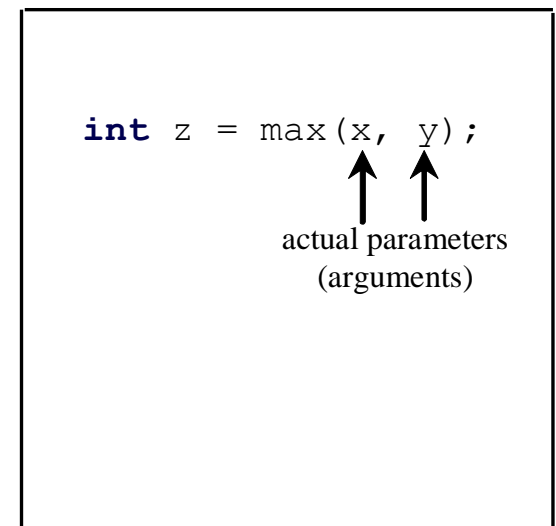- actual parameters (arguments)

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>.
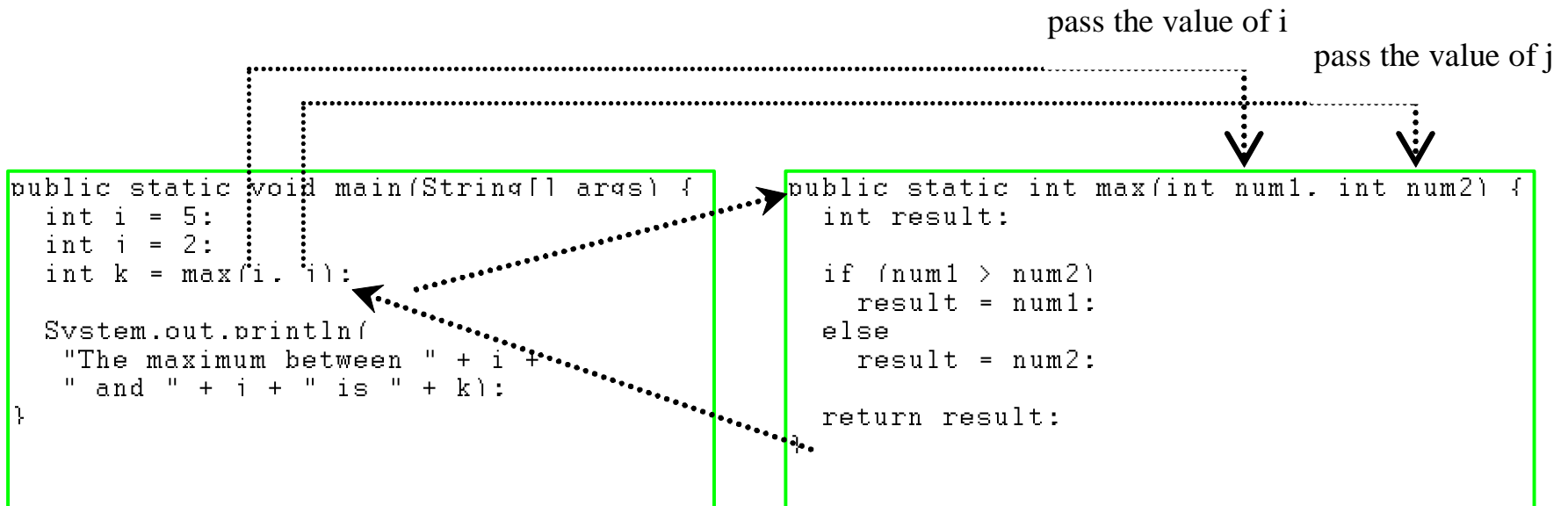
Define a method

Invoke a method

modifier

return value type

method name

formal parameters

method header

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

```
int result;
```

method body

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

parameter list

```
return result;
```
← return value

```
}
```

method signature

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

actual parameters (arguments)

Calling Methods

`max` method:

This program demonstrates calling a method max to return the largest of the `int` values

Calling Methods, cont.

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;
}
```

Trace Method Invocation

i is now 5

```
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;
}
```

Trace Method Invocation

j is now 2

```
public static void main(Strin    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;
}
```

Trace Method Invocation

invoke max(i, j)

```java
public static void main(Strin   args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

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

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

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

  return result;
}
```

Trace Method Invocation

invoke max(i, j)
Pass the value of i to num1
Pass the value of j to num2

```java
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);
}
```

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

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

  return result;
}
```

Trace Method Invocation

declare variable result

```
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;
}
```

Trace Method Invocation

(num1 > num2) is true since num1 is 5 and num2 is 2

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

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

  return result;
}
```

Trace Method Invocation

result is now 5

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

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

  return result;
}
```

Trace Method Invocation

return result, which is 5

```
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;
}
```

Trace Method Invocation

return max(i, j) and assign the
return value to k

```
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;
}
```

30

Trace Method Invocation

Execute the print statement

```
public static void main(String     gs) {
  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;
}
```

31

Example

□ Write a method to find a maximum between 3 integers with the following signature.

public static int max(int num1, int num2, int num3)

CAUTION

A <u>return</u> statement is required for a value-returning method. The method shown below in (a) is logically correct, but it has a compilation error because the Java compiler thinks it is 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;
}
```
(a)

Should be

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

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.

Reuse Methods from Other Classes

NOTE: One of the benefits of methods is for reuse. The <u>max</u> method can be invoked from any class besides <u>TestMax</u>. If you create a new class <u>Test</u>, you can invoke the <u>max</u> method using <u>ClassName.methodName</u> (e.g., <u>TestMax.max</u>).

void Method Example

```java
 1  public class TestVoidMethod {
 2    public static void main(String[] args) {
 3      System.out.print("The grade is ");
 4      printGrade(78.5);
 5
 6      System.out.print("The grade is ");
 7      printGrade(59.5);
 8    }
 9
10    public static void printGrade(double score) {
11      if (score >= 90.0) {
12        System.out.println('A');
13      }
14      else if (score >= 80.0) {
15        System.out.println('B');
16      }
17      else if (score >= 70.0) {
18        System.out.println('C');
19      }
20      else if (score >= 60.0) {
21        System.out.println('D');
22      }
23      else {
24        System.out.println('F');
25      }
26    }
27  }
```

Passing Parameters

```
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("Continuing Education", 15);
What is the output?

Can you invoke the method using
    nPrintln(15, "Continuing Education");

Pass by Value

```java
1  public class Increment {
2    public static void main(String[] args) {
3      int x = 1;
4      System.out.println("Before the call, x is " + x);
5      increment(x);
6      System.out.println("after the call, x is " + x);
7    }
8
9    public static void increment(int n) {
10     n++;
11     System.out.println("n inside the method is " + n);
12   }
13 }
```

Exercise 16-2

☐ Write a program to find a greatest common divisor between 2 integers.

☐ The greatest common divisor of 2 integers is the largest positive integer that divides the numbers without a remainder.

    ☐ For example, the GCD of 8 and 12 is 4.

☐ Your program prompts the user to provide 2 different integers and then display the GCD of these 2 integers by calling the GCD method.