

Chapter 6

Methods

COSC1046

Ping Luo

Opening Problem

- Find the sum of integers from 1 to 10, from 20 to 30, and from 35 to 45, respectively.
- Compute the average points each student obtained during four years' study.

Opening 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 <= 37; i++)  
    sum += i;  
System.out.println("Sum from 20 to 37 is " + sum);
```

```
sum = 0;  
for (int i = 35; i <= 49; i++)  
    sum += i;  
System.out.println("Sum from 35 to 49 is " + sum);
```

Use Method

```
public static int sum(int i1, int i2) {  
    int result = 0;  
    for (int i = i1; i <= i2; i++)  
        result += i;  
  
    return result;  
}  
  
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 37 is " + sum(20, 37));  
    System.out.println("Sum from 35 to 49 is " + sum(35, 49));  
}
```

Define a Method

- The syntax for defining a method is as follows:

```
modifier returnType methodName(list of parameters) {  
    // Method body;  
}
```

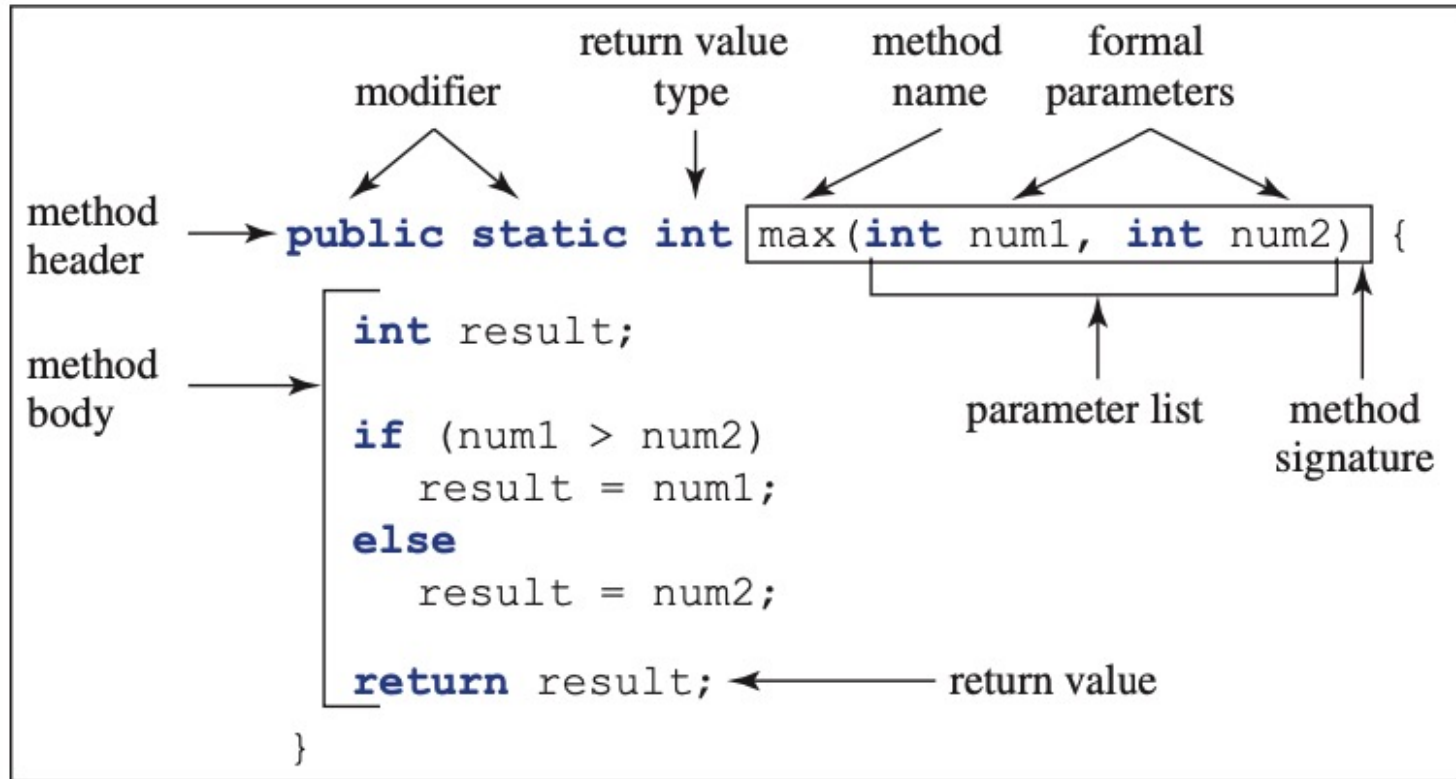
A diagram illustrating the components of the method syntax. Two blue arrows point from the code above to two boxes below. The first arrow points from the word 'modifier' in the code to a box containing the word 'public'. The second arrow points from the words 'returnType' and 'methodName' in the code to a box containing 'void, int, etc.'.

public

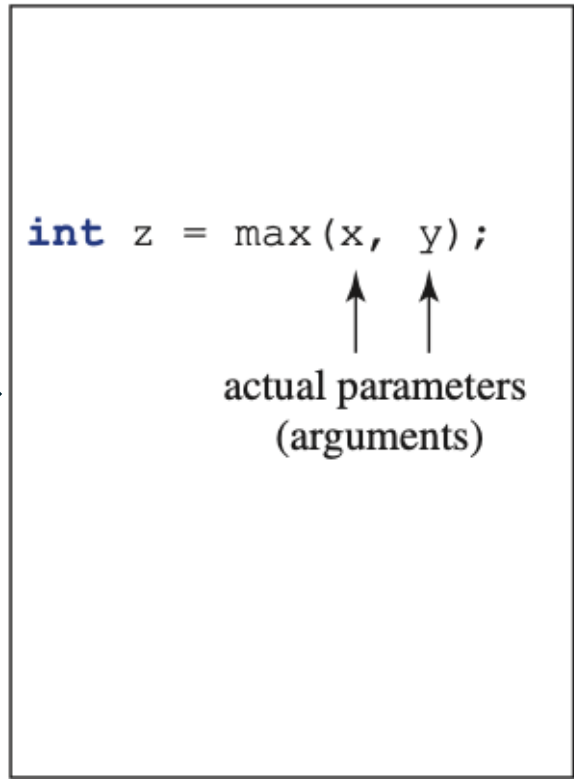
void, int,
etc.

Method

Define a method



Invoke a method



Method

- The **static** modifier is used for all the methods in this chapter. The reason for using it will be discussed in Chapter 9, Objects and Classes.
- The **returnValueType** is the **data type** of the value the method **returns**. Some methods perform desired operations without returning a value. In this case, the returnValueType is the keyword **void**.

Method

- A parameter is like a **placeholder**: when a method is invoked, you pass a value to the parameter
- Parameters are **optional**; that is, a method may contain no parameters. For example, the `Math.random()` method has no parameters.
- In the method header, you need to declare each parameter **separately**. For instance, **`max(int num1, int num2)` is correct, but `max(int num1, num2)` is wrong.**

Note

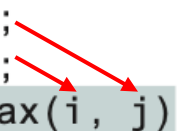
- We say “define a method” and “declare a variable.” We are making a subtle distinction here.
- A definition defines what the defined item is, but a declaration usually involves **allocating memory** to store data for the declared item.

Method Example

```
public class TestMax {  
    /** Main method */  
    public static void main(String[] args) {  
        int i = 5;  
        int j = 2;  
        int k = max(i, j);  
        System.out.println("The maximum of " + i +  
            " and " + j + " is " + k);  
    }  
  
    /** Return the max of two numbers */  
    public static int max(int num1, int num2) {  
        int result;  
  
        if (num1 > num2)  
            result = num1;  
        else  
            result = num2;  
  
        return result;  
    }  
}
```

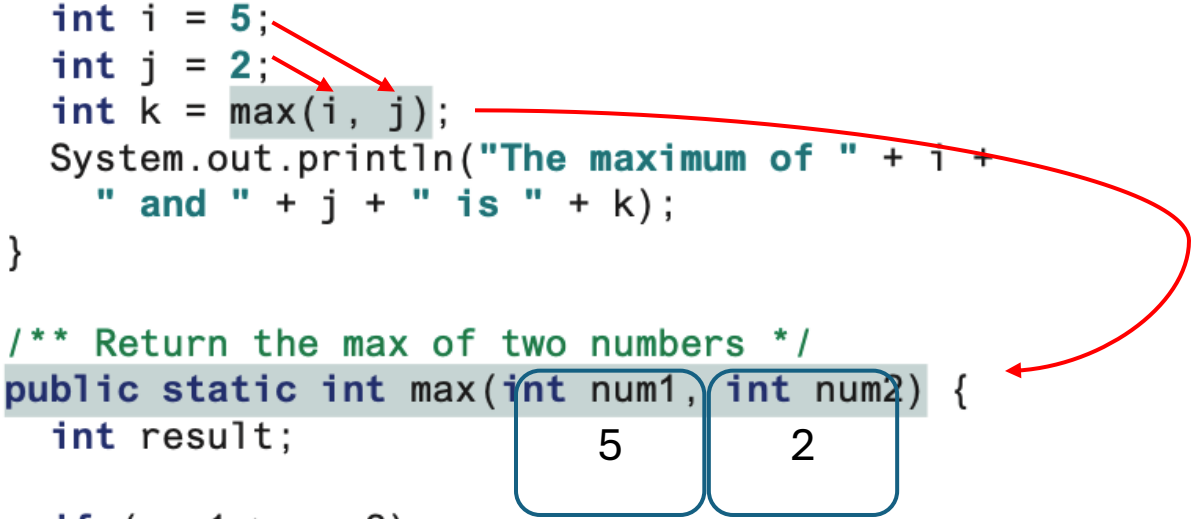
Method Example

```
public class TestMax {  
    /** Main method */  
    public static void main(String[] args) {  
        int i = 5;  
        int j = 2;  
        int k = max(i, j);  
        System.out.println("The maximum of " + i +  
            " and " + j + " is " + k);  
    }  
  
    /** Return the max of two numbers */  
    public static int max(int num1, int num2) {  
        int result;  
  
        if (num1 > num2)  
            result = num1;  
        else  
            result = num2;  
  
        return result;  
    }  
}
```

A diagram with two red arrows. The first arrow starts at the value '5' in the line 'int i = 5;' and points to the first parameter 'num1' in the method call 'max(i, j)'. The second arrow starts at the value '2' in the line 'int j = 2;' and points to the second parameter 'j' in the method call 'max(i, j)'. This illustrates how the values of variables 'i' and 'j' are passed as arguments to the 'max' method.

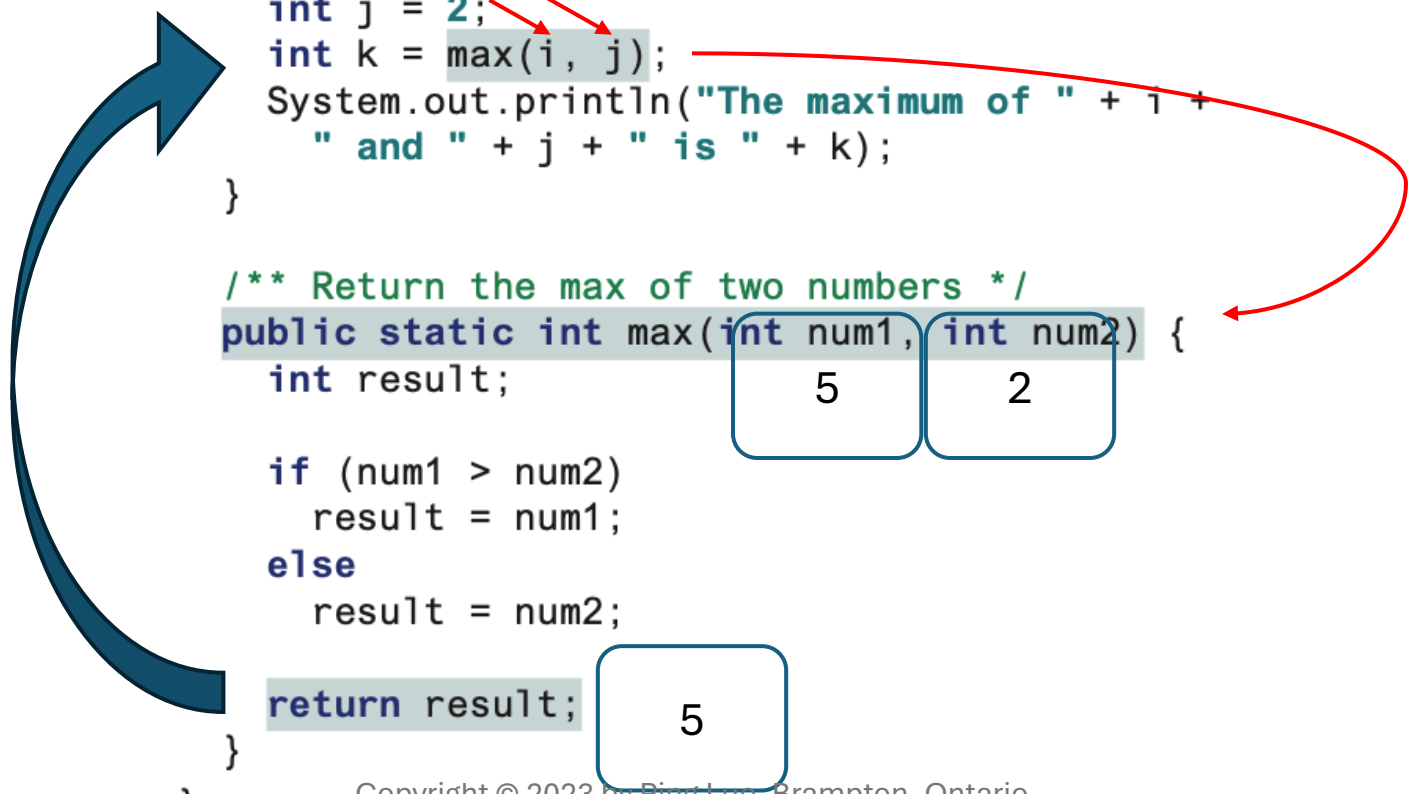
Method Example

```
public class TestMax {  
    /** Main method */  
    public static void main(String[] args) {  
        int i = 5;  
        int j = 2;  
        int k = max(i, j);  
        System.out.println("The maximum of " + i +  
            " and " + j + " is " + k);  
    }  
  
    /** Return the max of two numbers */  
    public static int max(int num1, int num2) {  
        int result;  
  
        if (num1 > num2)  
            result = num1;  
        else  
            result = num2;  
  
        return result;  
    }  
}
```



Method Example

```
public class TestMax {  
    /** Main method */  
    public static void main(String[] args) {  
        int i = 5;  
        int j = 2;  
        int k = max(i, j);  
        System.out.println("The maximum of " + i +  
            " and " + j + " is " + k);  
    }  
  
    /** Return the max of two numbers */  
    public static int max(int num1, int num2) {  
        int result;  
  
        if (num1 > num2)  
            result = num1;  
        else  
            result = num2;  
  
        return result;  
    }  
}
```

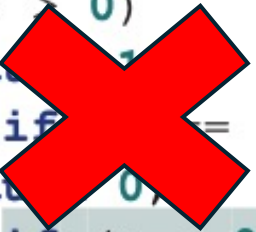


Caution


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

Caution

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



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



Case Study: Grade

```
if (score >= 90.0)
    grade = 'A';
else if (score >= 80.0)
    grade = 'B';
else if (score >= 70.0)
    grade = 'C';
else if (score >= 60.0)
    grade = 'D';
else
    grade = 'F';
```

input: score

getGrade(score)

output: grade

Example

```
public class TestReturnGradeMethod {  
    public static void main(String[] args) {  
        System.out.print("The grade is " + getGrade(78.5));  
        System.out.print("\nThe grade is " + getGrade(59.5));  
    }  
  
    modifier returnType getGrade(parameter) {  
        if (score >= 90.0)  
            return 'A';  
        else if (score >= 80.0)  
            return 'B';  
        else if (score >= 70.0)  
            return 'C';  
        else if (score >= 60.0)  
            return 'D';  
        else  
            return 'F';  
    }  
}
```

Bonus!!!

Example

```
public class TestReturnGradeMethod {  
    public static void main(String[] args) {  
        System.out.print("The grade is " + getGrade(78.5));  
        System.out.print("\nThe grade is " + getGrade(59.5));  
    }  
  
    modifier returnType getGrade(parameter) {  
        if (score >= 90.0)  
            return 'A';  
        else if (score >= 80.0)  
            return 'B';  
        else if (score >= 70.0)  
            return 'C';  
        else if (score >= 60.0)  
            return 'D';  
        else  
            return 'F';  
    }  
}
```

Bonus!!!

Caution

- The arguments must match the parameters in **order, number, and compatible type**, as defined in the method signature.

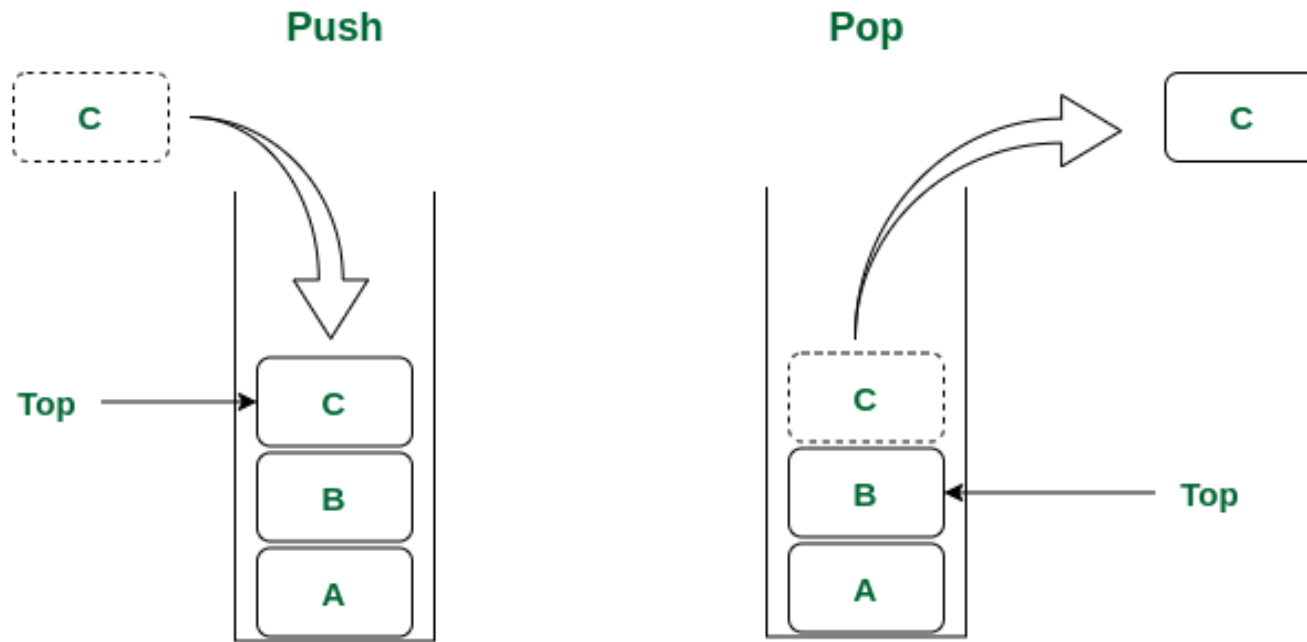
```
public static void nPrintln(String message, int n) {  
    for (int i = 0; i < n; i++)  
        System.out.println(message);  
}
```

- nPrintln("Hello", 3) - **correct**
- nPrintln(3, "Hello") - **wrong**

Stack

- Each time a method is invoked, the system creates an activation record that stores parameters and variables for the method and places the activation record in an area of memory known as a **call stack**.
- A call stack is also known as an execution stack, runtime stack, or machine stack and it is often shortened to just “the stack.”
- A call stack stores the activation records in a **last-in, first-out** fashion.

Stack

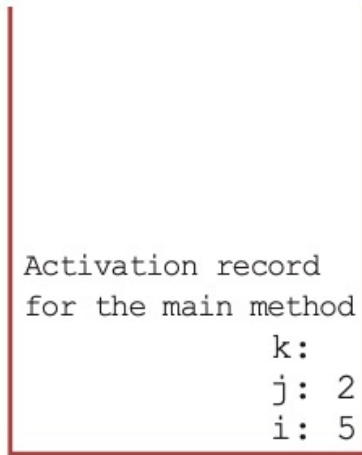


Stack Data Structure

Stack Example

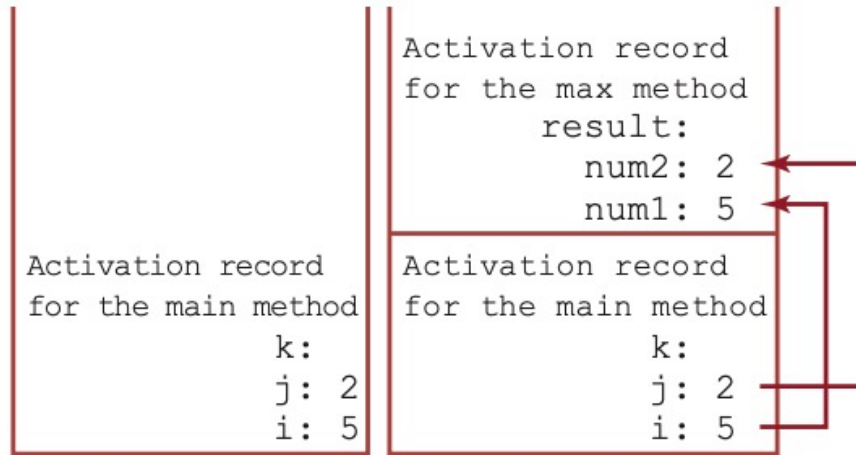
```
public class TestMax {  
    /** Main method */  
    public static void main(String[] args) {  
        int i = 5;  
        int j = 2;  
        int k = max(i, j);  
        System.out.println("The maximum of " + i +  
            " and " + j + " is " + k);  
    }  
  
    /** Return the max of two numbers */  
    public static int max(int num1, int num2) {  
        int result;  
  
        if (num1 > num2)  
            result = num1;  
        else  
            result = num2;  
  
        return result;  
    }  
}
```

Stack



(1) The main
method is invoked.

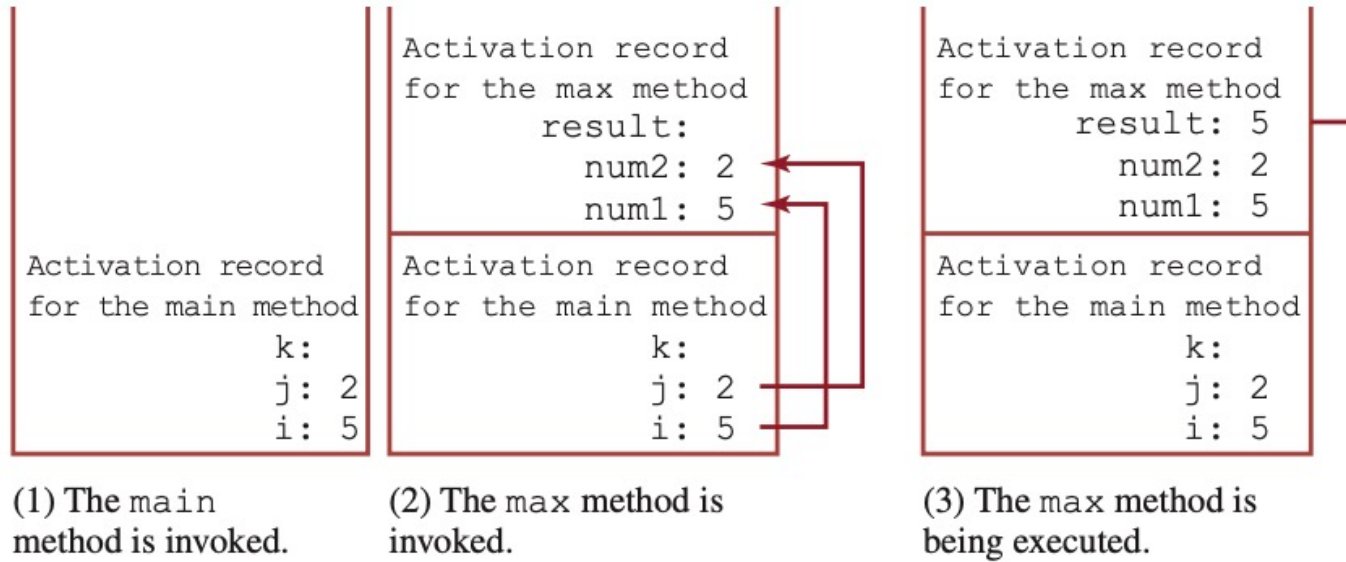
Stack



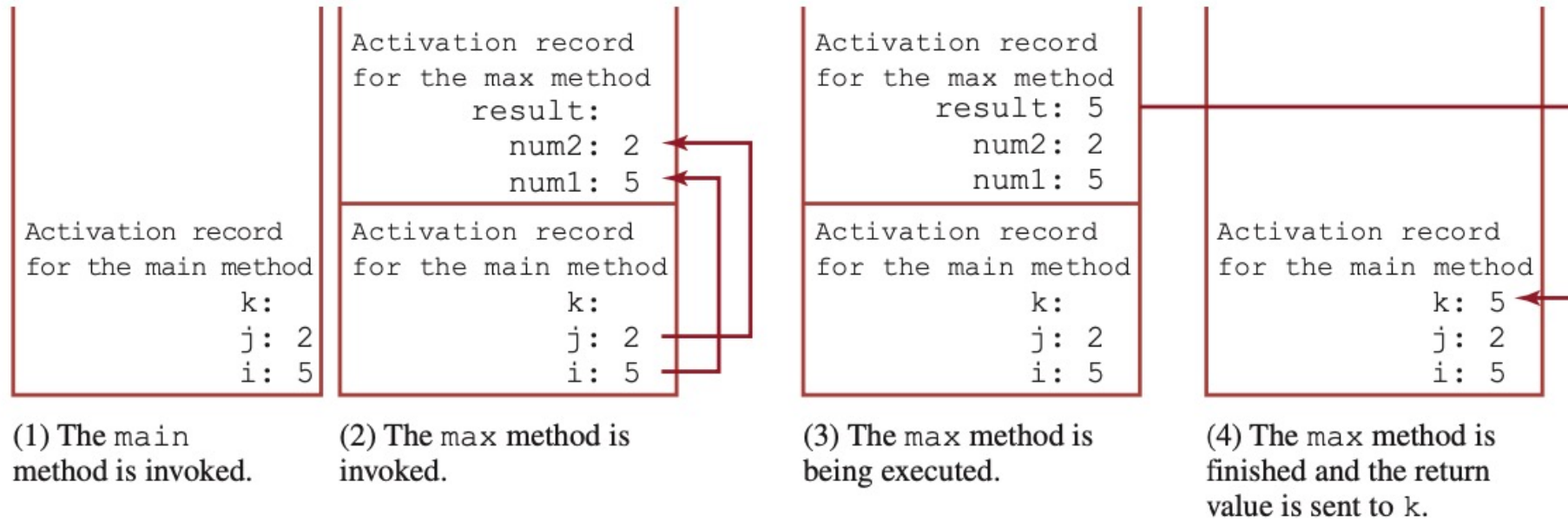
(1) The main method is invoked.

(2) The max method is invoked.

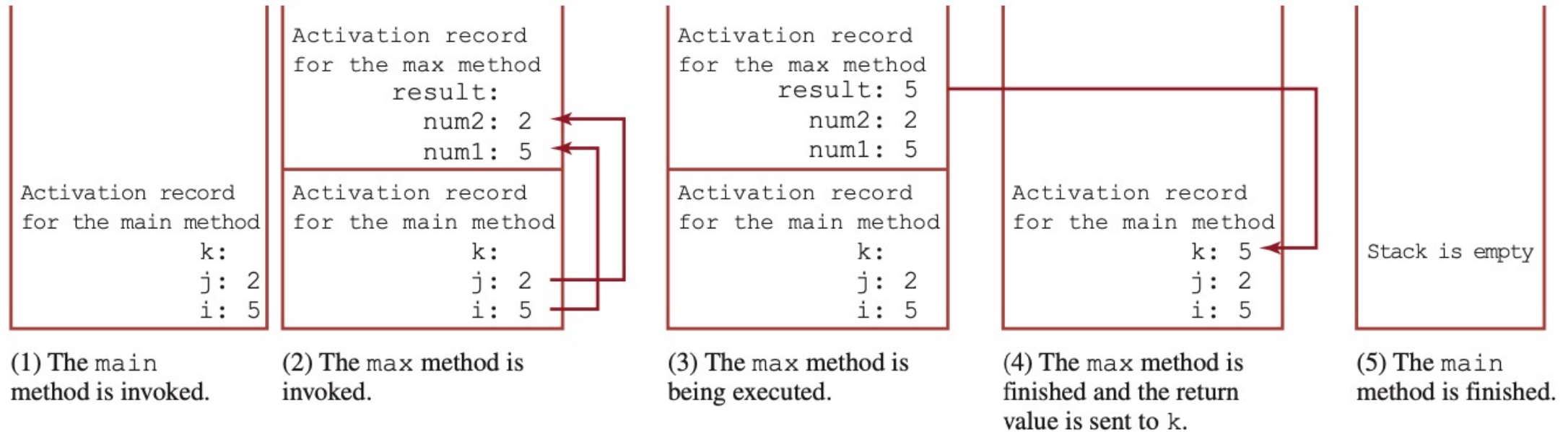
Stack



Stack



Stack



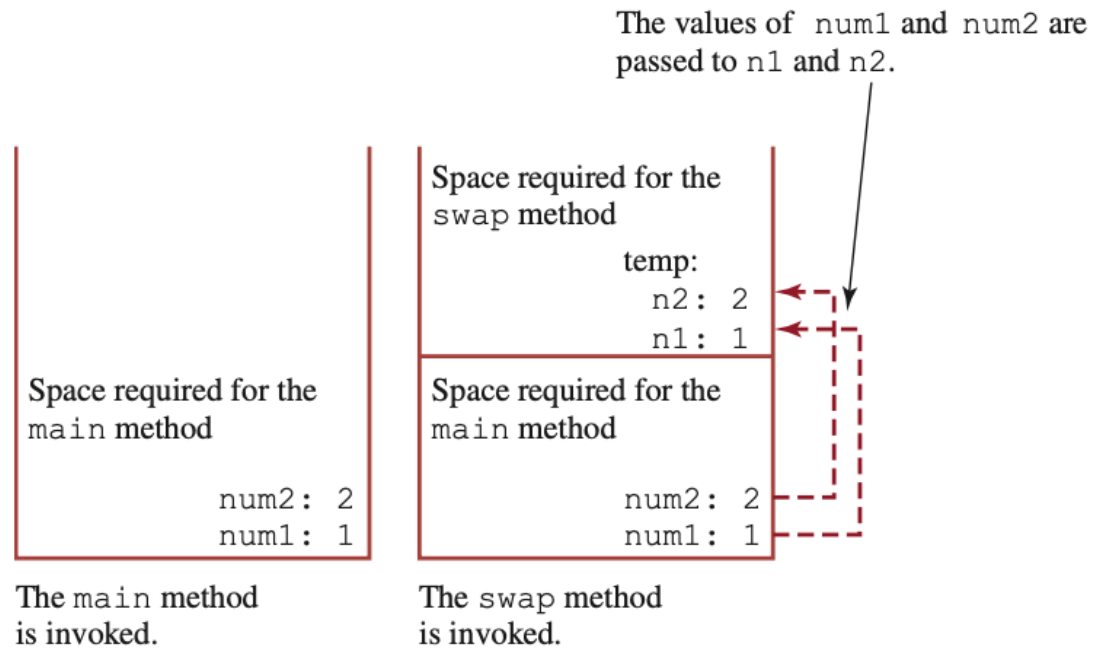
Pass Values

```
public class Increment {  
    Run | Debug  
    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);  
    }  
}
```

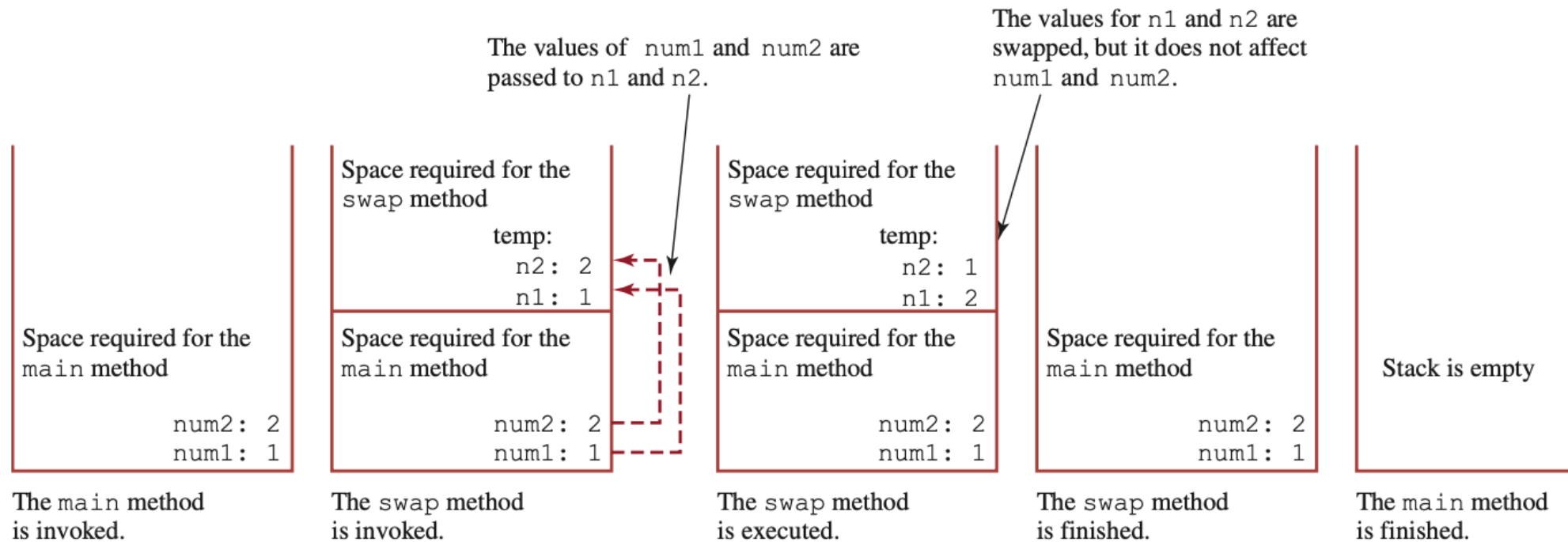
Caution

- When you invoke a method with an argument, the **value** of the argument is passed to the parameter. If the argument is a variable rather than a literal value, the **value of the variable** is passed to the parameter.
- <https://liveexample.pearsoncmg.com/html/TestPassByValue.html>

Stack



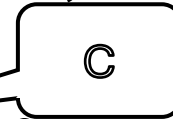
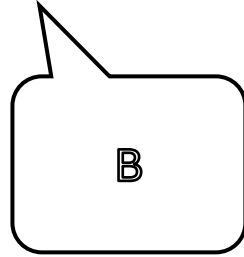
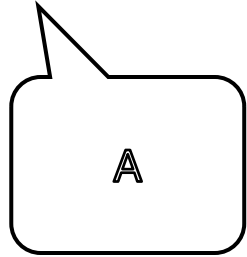
Stack



Case Study: Converting Hexadecimals to Decimals

- Given a hex number, for example, AB8C, its equivalent decimal value is:

- $10 \times 16^3 + 11 \times 16^2 + 8 \times 16^1 + 12 \times 16^0 = 43916$




```

import java.util.Scanner;

public class Hex2Dec {
    /** Main method */
    Run | Debug
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print(s:"Enter a hex number: ");
        String hex = input.nextLine();

        System.out.println("The decimal value for hex number "
            + hex + " is " + hexToDecimal(hex.toUpperCase()));
    }

    // finish the method here
    MODIFIER RETURN TYPE hexToDecimal() {
        int decimalValue = 0;

        |
        return decimalValue;
    }

    public static int hexCharToDecimal(char ch) {
        if ('A' <= ch && ch <= 'F')
            return 10 + ch - 'A';
        else // ch is '0', '1', ..., or '9'
            return ch - '0';
    }
}

```

Mid-Term Exam

- **Online** Test on Moodle
- 8:30pm on Feb 27th to 8:30am on Feb 29th
- 50 single-choice questions, contributes 25 points toward the final grade
- You are expected to allocate approximately 40-60 minutes to complete the exam.
- Students who miss the Exam will receive a grade of 0.

10 Min Break

Method

```
public static void nPrintln(String message, int n) {  
    for (int i = 0; i < n; i++)  
        System.out.println(message);  
}
```

Method

```
public static void nPrintln(String message, int n) {  
    for (int i = 0; i < n; i++)  
        System.out.println(message);  
}
```

```
public static int maxInt(int num1, int num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```

```
public static double maxDouble(double num1, double num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```

Overloading Method

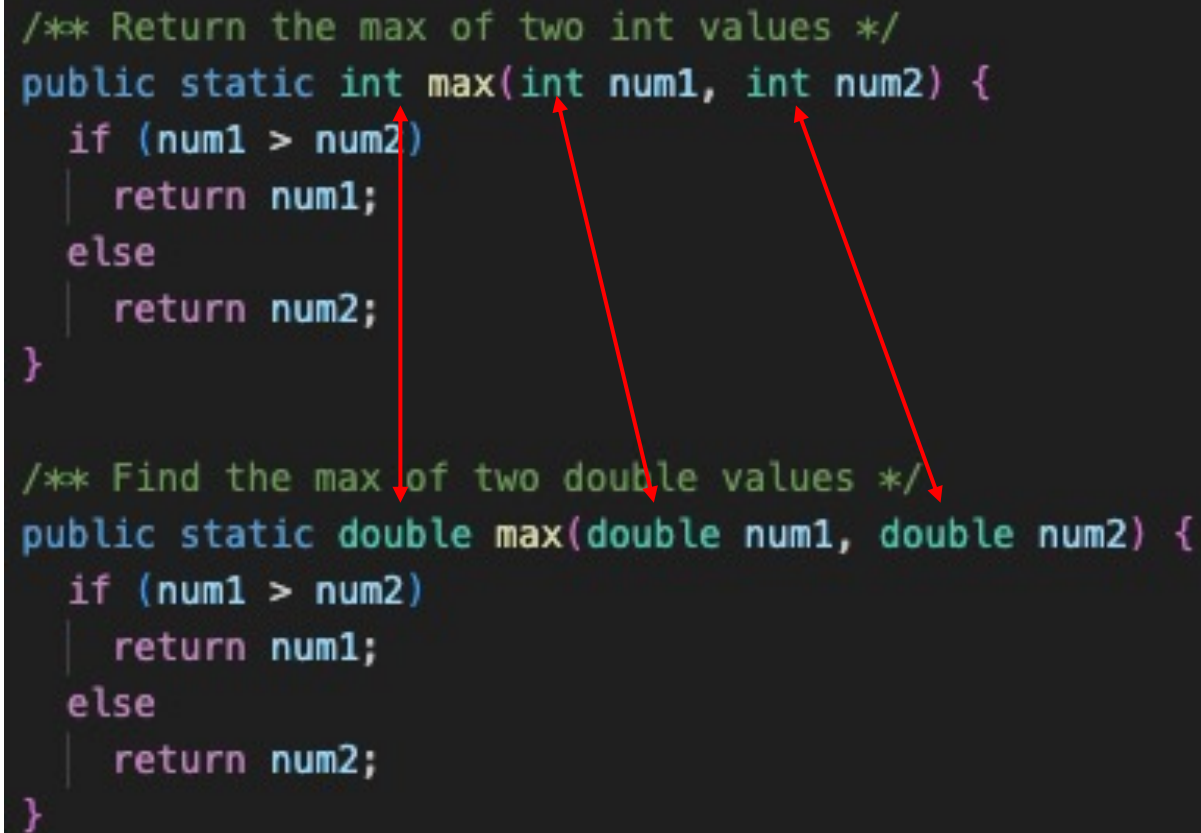
- Overloading methods enable you to define the methods with the **same name** as long as their **parameter lists are different**.

```
public static double max(double num1, double num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```

```
public static int max(int num1, int num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```

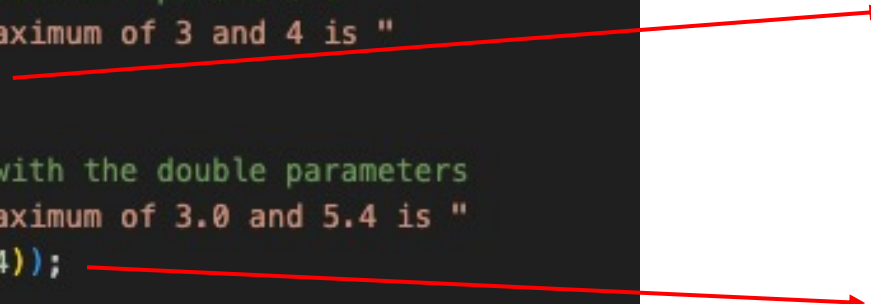
Overloading Method

```
/** Return the max of two int values */  
public static int max(int num1, int num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}  
  
/** Find the max of two double values */  
public static double max(double num1, double num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```

The diagram illustrates method overloading by showing two methods with the same name, 'max', but different parameter lists. Red arrows point from the 'max' identifier in each method signature to its corresponding parameter list: one arrow points from the 'max' in the first method to '(int num1, int num2)', and another arrow points from the 'max' in the second method to '(double num1, double num2)'. This visualizes how the compiler distinguishes between the two methods based on their parameters.

Note

```
public static void main(String[] args) {  
    // Invoke the max method with int parameters  
    System.out.println("The maximum of 3 and 4 is "  
        + max(num1:3, num2:4));  
  
    // Invoke the max method with the double parameters  
    System.out.println("The maximum of 3.0 and 5.4 is "  
        + max(num1:3.0, num2:5.4));  
  
    // Invoke the max method with three double parameters  
    System.out.println("The maximum of 3.0, 5.4, and 10.14 is "  
        + max(num1:3.0, num2:5.4, num3:10.14));  
}
```



```
/** Return the max of two int values */  
public static int max(int num1, int num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}  
  
/** Find the max of two double values */  
public static double max(double num1, double num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```


Note

```
public static void main(String[] args) {  
    // Invoke the max method with int parameters  
    System.out.println("The maximum of 3 and 4 is "  
        + max(num1:3, num2:4));  
  
    // Invoke the max method with the double parameters  
    System.out.println("The maximum of 3.0 and 5.4 is "  
        + max(num1:3.0, num2:5.4));  
  
    // Invoke the max method with three double parameters  
    System.out.println("The maximum of 3.0, 5.4, and 10.14 is "  
        + max(num1:3.0, num2:5.4, num3:10.14));  
}
```

```
/** Return the max of two int values */  
public static int max(int num1, int num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}  
  
/** Find the max of two double values */  
public static double max(double num1, double num2) {  
    if (num1 > num2)  
        return num1;  
    else  
        return num2;  
}
```

```
/** Return the max of three double values */  
public static double max(double num1, double num2, double num3) {  
    return max(max(num1, num2), num3);  
}
```

Ambiguous Invocation

compile error!!!

```
public class AmbiguousOverloading {  
    public static void main(String[] args) {  
        System.out.println(max(1, 2));  
    }  
  
    public static double max(int num1, double num2) {  
        if (num1 > num2)  
            return num1;  
        else  
            return num2;  
    }  
  
    public static double max(double num1, int num2) {  
        if (num1 > num2)  
            return num1;  
        else  
            return num2;  
    }  
}
```

Case Study: Prime Number Method

- Write a program to print out the first **N** prime numbers.
- A prime number is a whole number greater than 1 whose only factors are 1 and itself.
- The smallest prime number is 2.

what do we need?

Case Study: Prime Number Method

- Write a program to print out the first **N** prime numbers.
- A prime number is a whole number greater than 1 whose only factors are 1 and itself.
- The smallest prime number is 2.

what do we need?

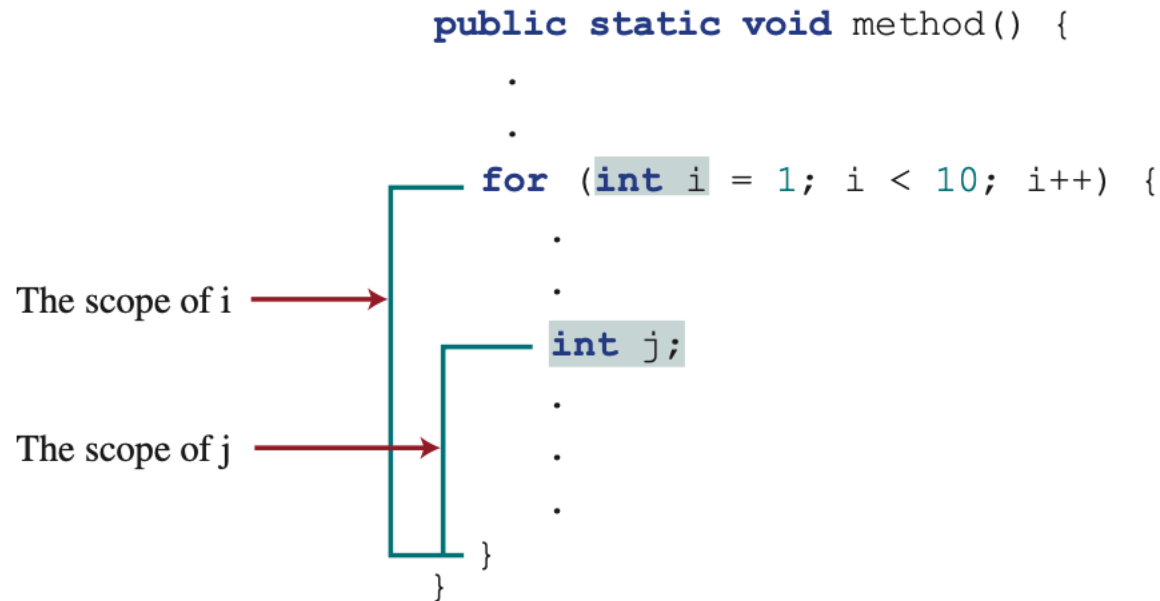
1. count
2. while
3. function to determine if a number is prime number



Bonus!!!

The Scope of Variables

- The scope of a variable is the part of the program where the variable can be referenced.*



The Scope of Variables

```
for (int i = 0; i < 10; i++) {  
}  
System.out.println(i); // Causes a syntax error on i
```

```
public static void method1() {  
    int x = 1;  
    int y = 1;  
  
    for (int i = 1; i < 10; i++) {  
        x += i;  
    }  
  
    for (int i = 1; i < 10; i++) {  
        y += i;  
    }  
}
```

right

```
public static void method2() {  
    int i = 1;  
    int sum = 0;  
  
    for (int i = 1; i < 10; i++) {  
        sum += i;  
    }  
}
```

wrong

The Scope of Variables

```
public class Scope {  
    Run | Debug  
    public static void main(String args[]) {  
        int i = 0;  
  
        for (; i<10; i++) {  
            System.out.println(x:"okay");  
        }  
        System.out.println(i);  
    }  
}
```

```
okay  
okay  
okay  
okay  
okay  
okay  
okay  
okay  
okay  
okay  
okay  
10
```

Study Week

- Feb 19 – 23, no lecture, no lab.

Feb 27 Lecture

1. Method II
2. Review for Chapters 1- 6
3. Exam start at 8:30pm

Q&A