Middlesex University London

Week 7

Methods

REVISION TASK

- Write a program counting the number of digit in the password below.
- ”Password123.mdx.ac.uk.NW4”

THIS WEEK:

- Main method
- Void method
- Return method

WHAT IS METHOD

- A Java method is a collection of statements that are grouped together to perform an operation.
- Methods can be use to define reusable code and organize and simplify code
- Java methods are where you put the operations on data (variables) in your Java code.
- When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.
- We already are very well familiar with the **main** method

MAIN METHOD

```
public class Welcome {
    /**
     * main method begins execution of Java application
     */
    public static void main(String[] args) {


    } //end method main

}///end class Welcome
```

DEFINING METHODS

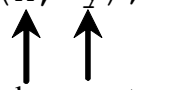
- A method is a collection of statements that are grouped together to perform an operation.

Define a method

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

   int result;

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

   return result;
}
```

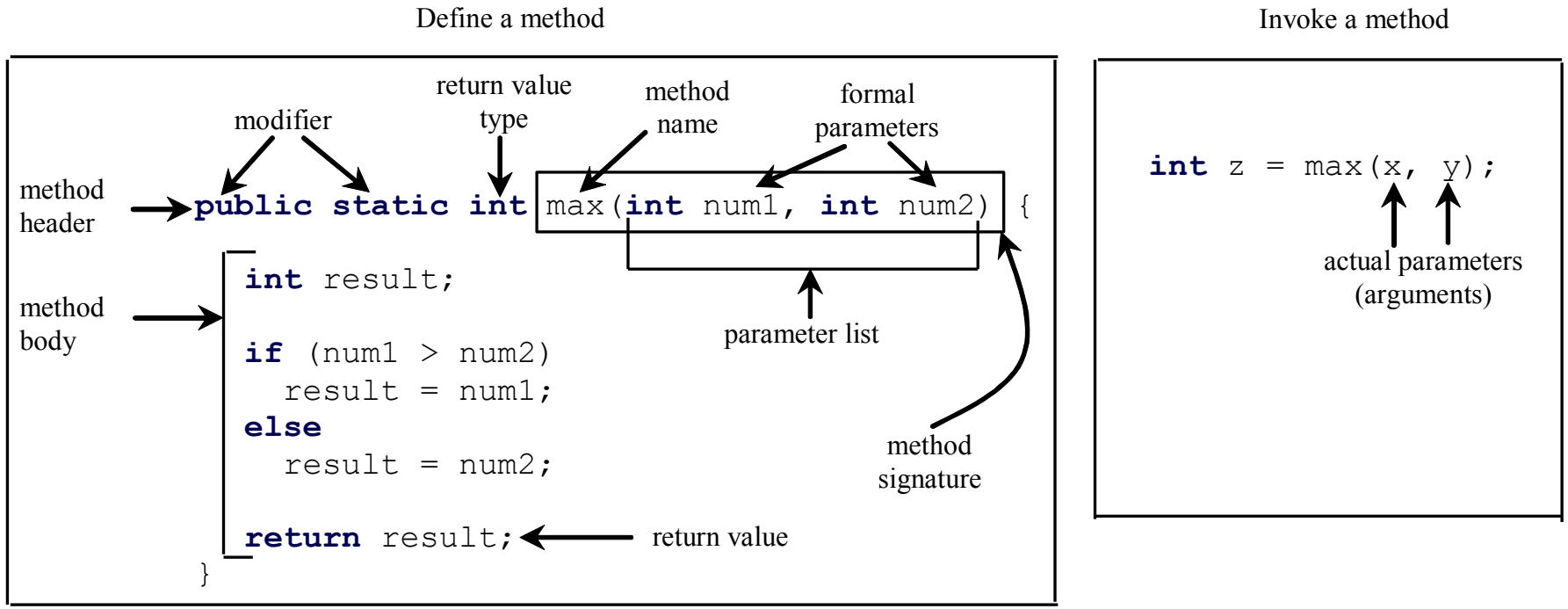
Invoke a method

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

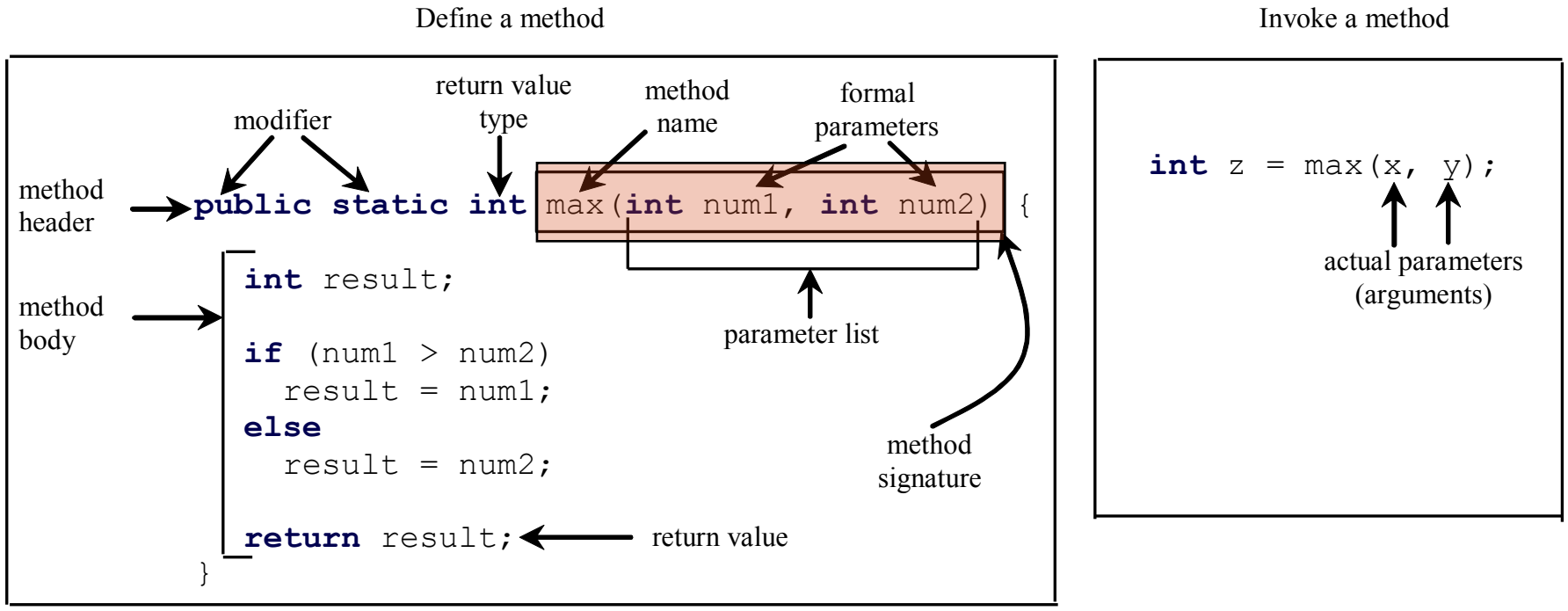


actual parameters
(arguments)

DEFINING METHODS

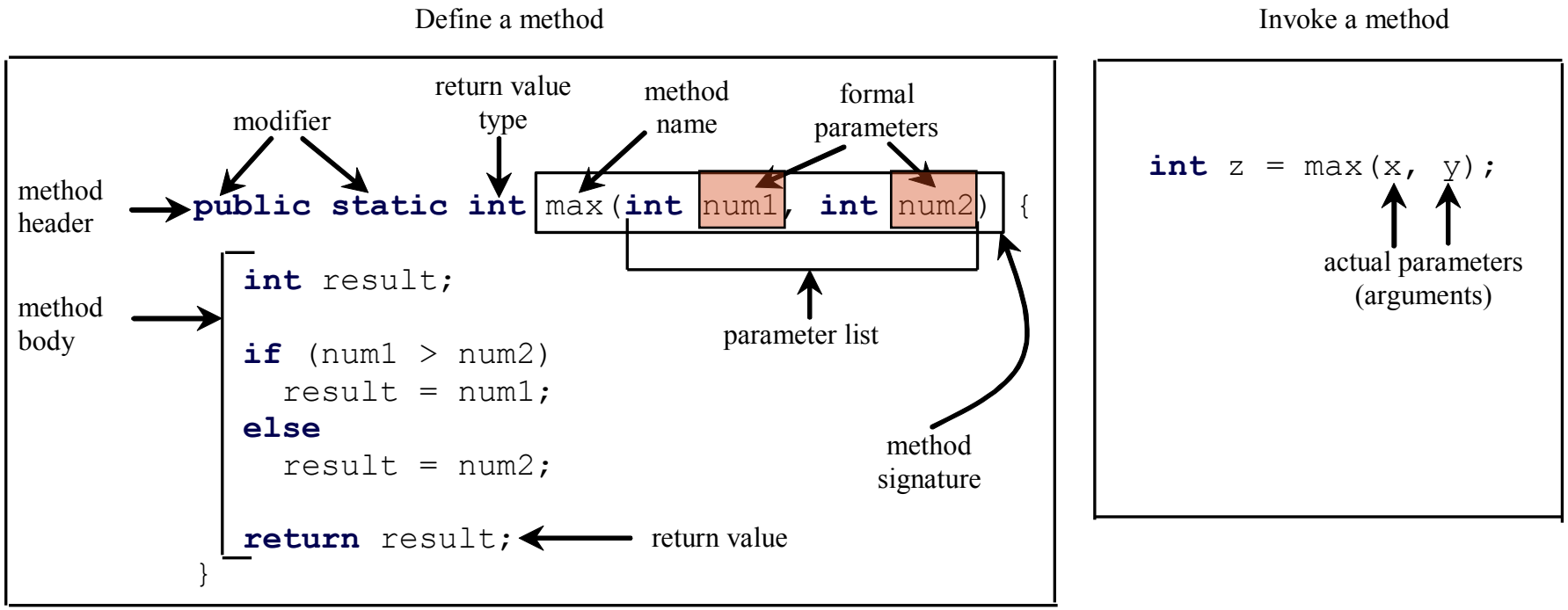


METHOD SIGNATURE



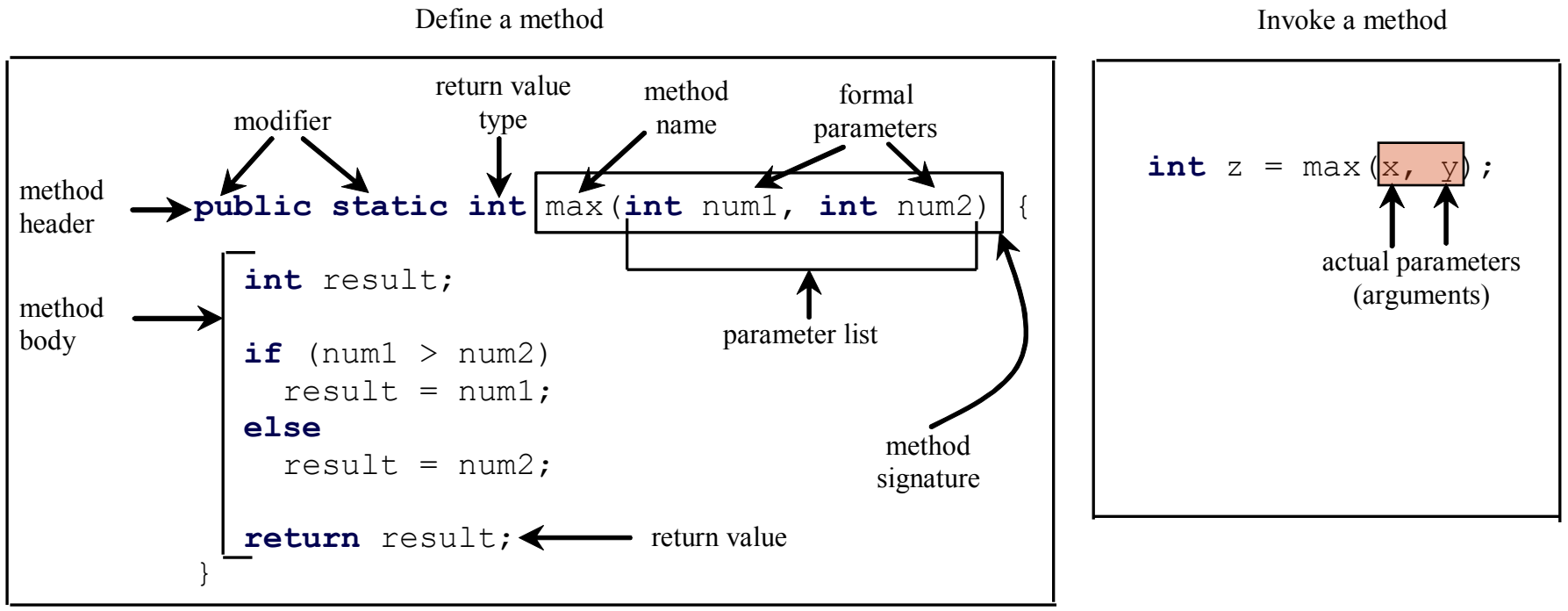
FORMAL PARAMETERS

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



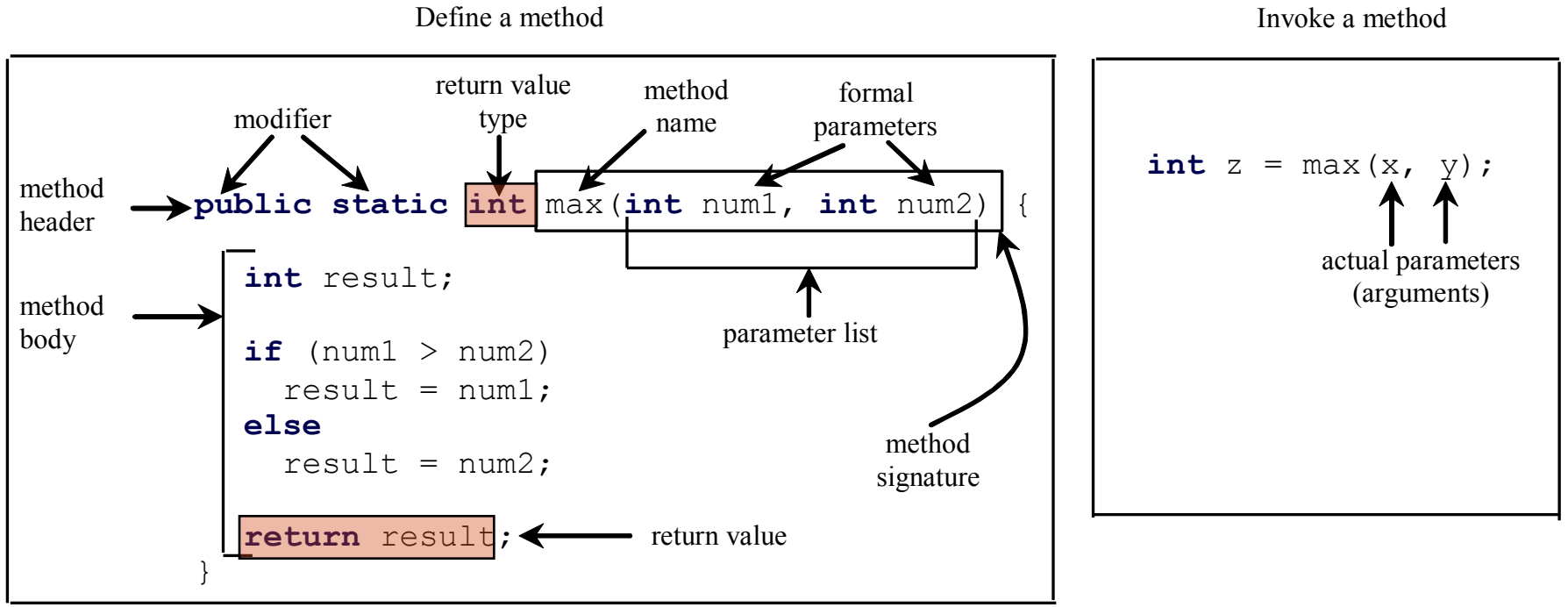
ACTUAL PARAMETERS

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



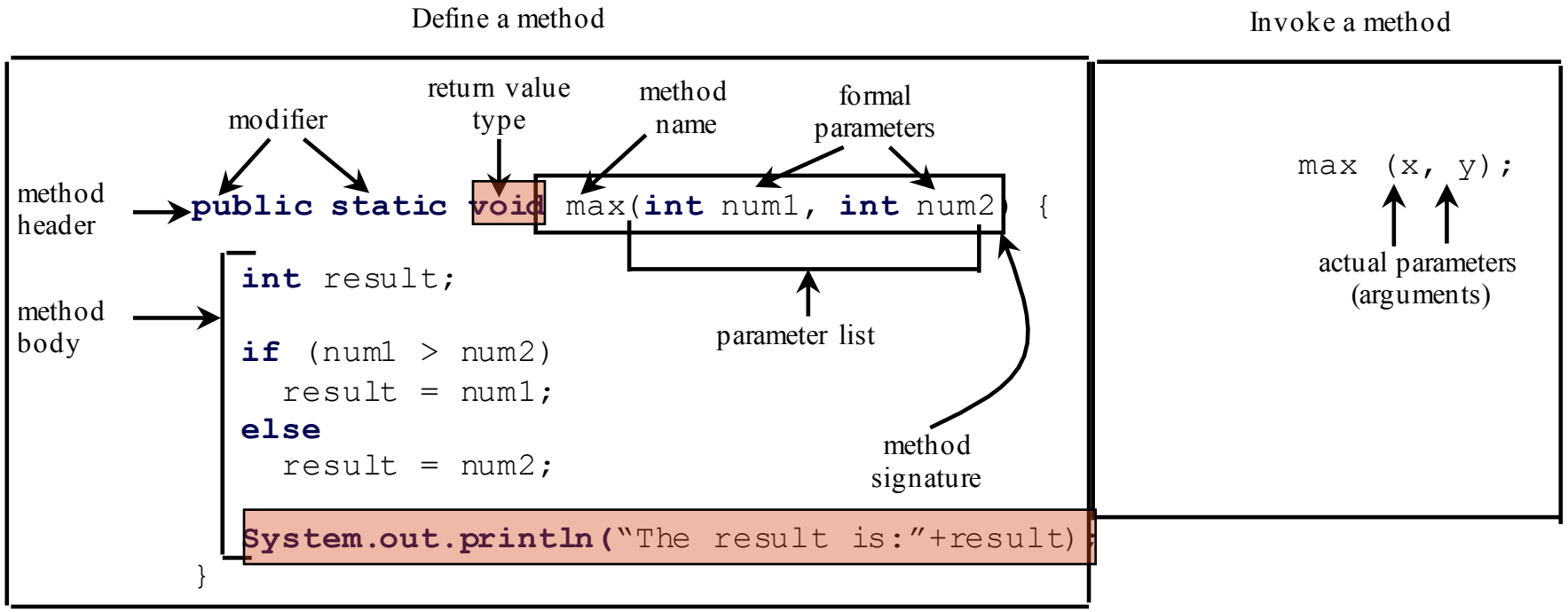
RETURN VALUE TYPE

- A method may return a value. The returnValueType is the **data type of the value** the method returns.



RETURN VALUE TYPE

- Method does not return a value, the returnValueType is the keyword void.



CALLING METHODS

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i,j);//invoke max method
    System.out.println("The maximum between "+i+" and "+j +" is: "+k);
}
 /** Return the max between two numbers */
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// Return result
}
```

CALLING METHODS, CONT.

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i,j);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// Return result
}
```

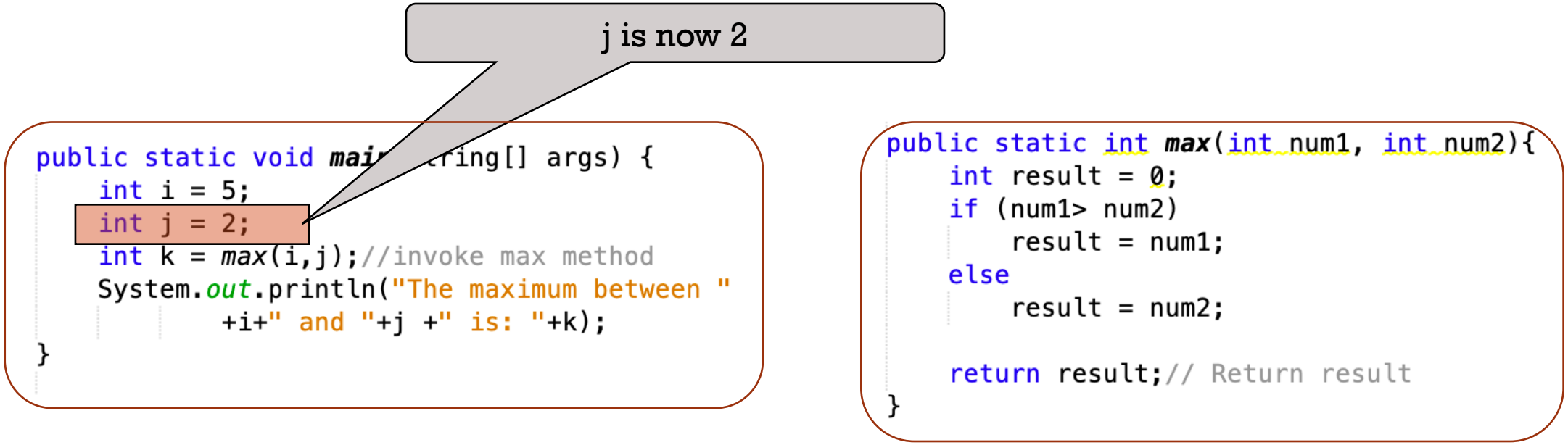
TRACE METHOD INVOCATION

i is now 5

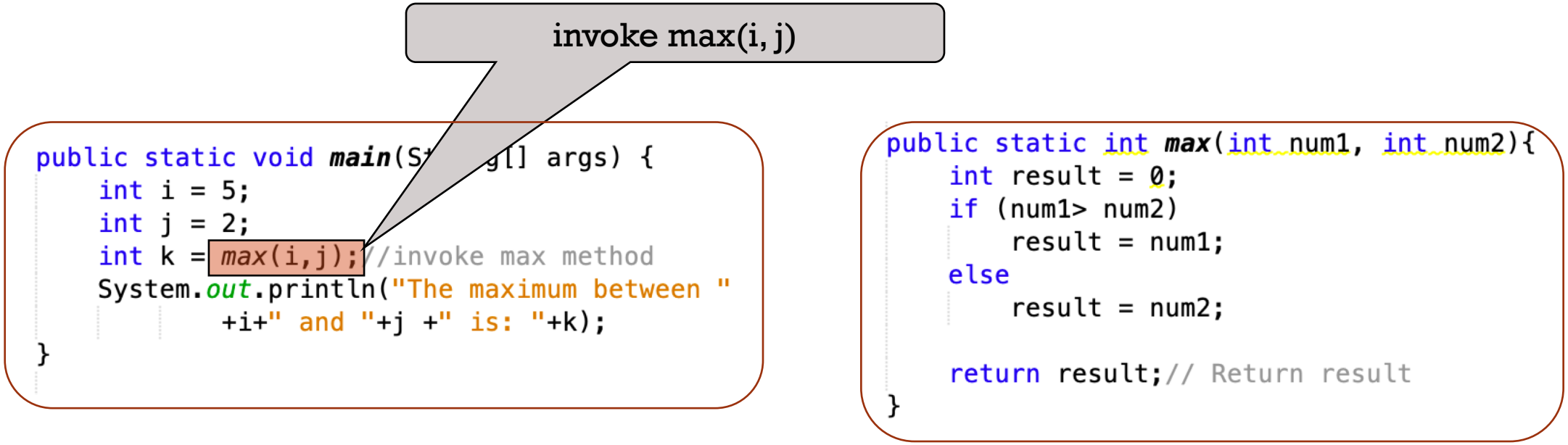
```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i,j);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// Return result
}
```

TRACE METHOD INVOCATION



TRACE METHOD INVOCATION



TRACE METHOD INVOCATION

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

```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i,j);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// 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);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// 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);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// 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);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// 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);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
blic static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// 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);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// Return result
}
```

TRACE METHOD INVOCATION

Execute the print statement

```
public static void main(Stri    args) {
    int i = 5;
    int j = 2;
    int k = max(i,j);//invoke max method
    System.out.println("The maximum between "
            +i+" and "+j +" is: "+k);
}
```

```
public static int max(int num1, int num2){
    int result = 0;
    if (num1> num2)
        result = num1;
    else
        result = num2;

    return result;// Return result
}
```

TASK

- Rewrite the revision task in such way that the ‘counter of digit in a password’ is defined in method. The method takes as an argument String and return integer variable.
- In main method create Scanner which will ask the user for a password.

CAUTION

- A return 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 possible that this method does not return any value.

```
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)

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

OVERLOADING METHODS

- Overloading the `max` Method

```
public static double max(double num1, double num2) {
    double result = 0.0;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
```

VOID VS RETURNING VALUE METHOD

- This type of method return a value. The method declaration type has to match the return value type

```
public static char getGrade(double score){
      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';
}
```

- Does not return any value. Void. The method performs some actions.

```
public static void printGrade(double score) {
    if (score >= 90.0) {
      System.out.println('A');
    }
    else if (score >= 80.0) {
      System.out.println('B');
    }
    else if (score >= 70.0) {
      System.out.println('C');
    }
    else if (score >= 60.0) {
      System.out.println('D');
    }
    else {
      System.out.println('F');
  }
}
```

PASSING PARAMETERS

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");`

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

SCOPE OF LOCAL VARIABLES

A local variable: a variable defined inside a method.

Scope: the part of the program where the variable can be referenced.

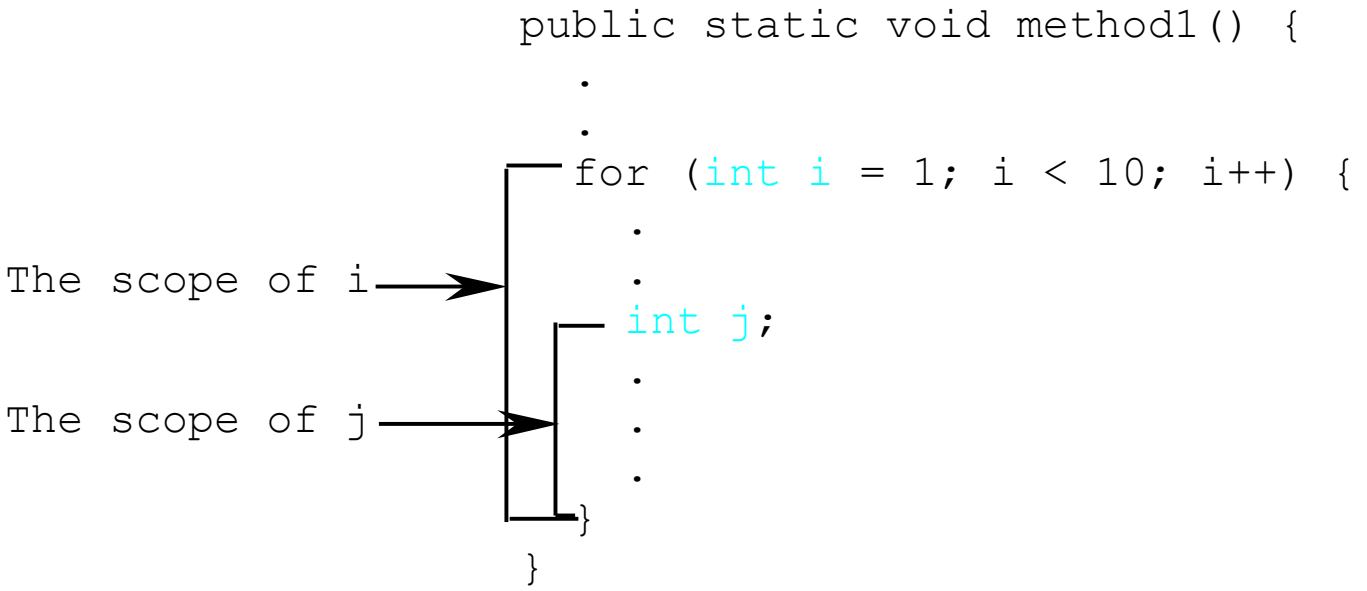
The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable.

A local variable must be declared before it can be used.

- You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks.

SCOPE OF LOCAL VARIABLES, CONT.

- A variable declared in the initial action part of a for loop header has its scope in the entire loop. But a variable declared inside a for loop body has its scope limited in the loop body from its declaration and to the end of the block that contains the variable.



SCOPE OF LOCAL VARIABLES, CONT.

It is fine to declare i in two non-nesting blocks

```
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;
  }
}
```

It is wrong to declare i in two nesting blocks

```
public static void method2() {

  int i = 1;
  int sum = 0;

  for (int i = 1; i < 10; i++)
    sum += i;
  }
}
```

KEY THINGS YOU SHOULD UNDERSTAND AFTER WEEK 6

- Types of methods
- Methods definition, functionality, arguments of methods, calling methods.
- Local variable and scope
- Overloading of methods

MASTERING YOUR SKILLS

- Read chapter 6 from the Liang book and finish all the exercises from the end of the book