

# Lecture 5

methods



```
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);
```

```
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);
```

```
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));  
}
```

- A method is a collection of statements that are grouped together to perform an operation.
- It is equivalent to a function in python (`def`)

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);
```

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

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

Method header

```
    int result;
```

```
    if (num1 > num2)
```

```
        result = num1;
```

```
    else
```

```
        result = num2;
```

```
    return result;
```

```
}
```

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

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

Modifier

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

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

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

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

Return value  
type

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



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

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

Method name

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

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

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

Parameter list

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

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

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

Formal  
parameters

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

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

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

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

    return result;
}
```

Method signature

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

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

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

```
    int result;
```

```
    if (num1 > num2)
```

```
        result = num1;
```

```
    else
```

```
        result = num2;
```

```
    return result;
```

```
}
```

Method body

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

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

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

Return value



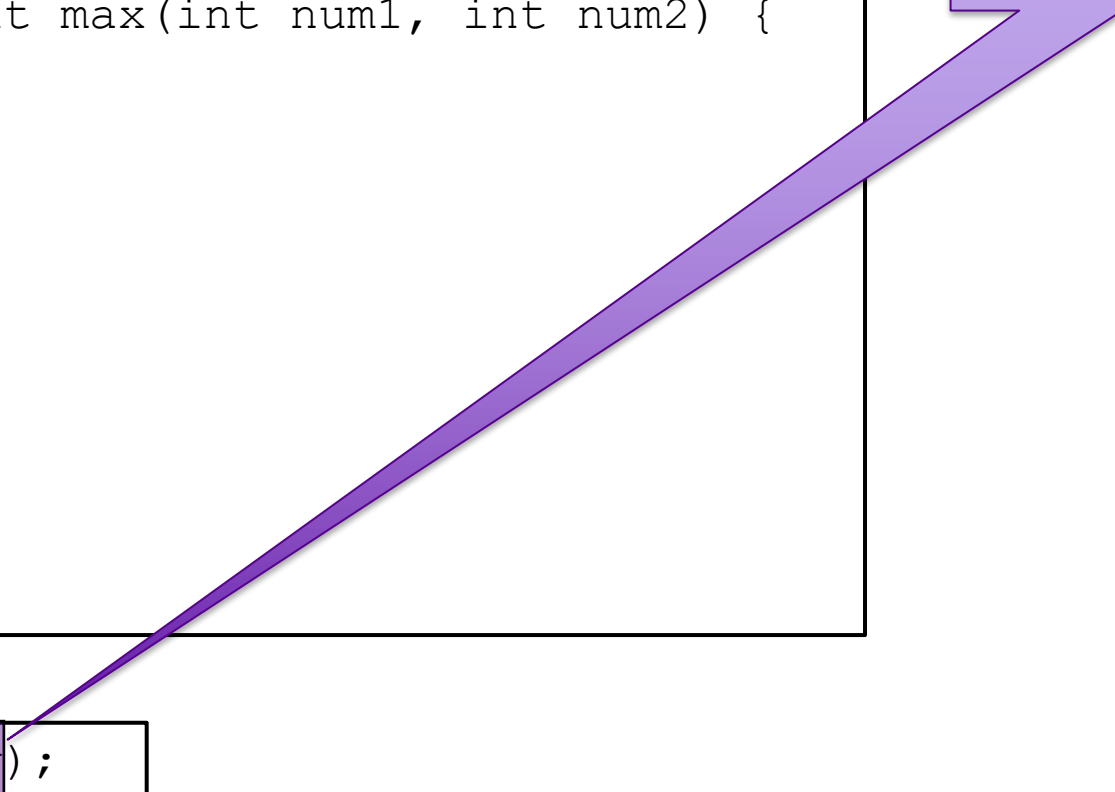
```
int z = max(x, y);
```

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

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

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

Actual parameter  
(arguments)



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

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

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



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

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

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

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

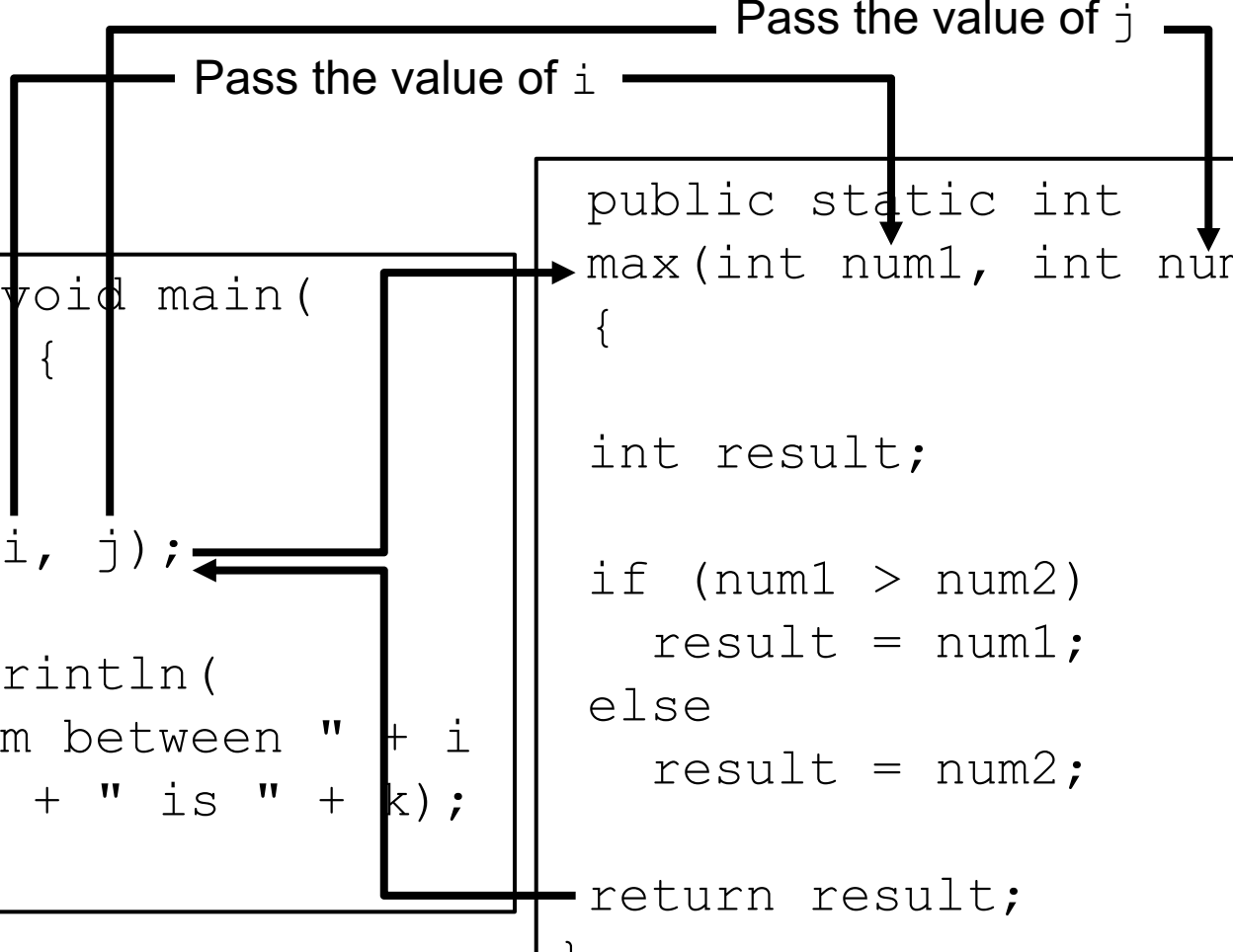
```
public static int max(int num1, int num2) {  
  
    int result;  
  
    if (num1 > num2)  
        result = num1;  
    else  
        result = num2;  
  
    return result;  
}
```

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

- The ***returnValueType*** is the data type of the value the method returns. If the method does not return a value, the ***returnValueType*** is the keyword ***void***.

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

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



```
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;
}
```

Pass the value of i

Pass the value of j



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;  
}
```

j is now 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 int  
max(int num1, int num2)  
{  
  
    int result;  
  
    if (num1 > num2)  
        result = num1;  
    else  
        result = num2;  
  
    return result;  
}
```

invoke max(i, 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;  
}
```

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);  
  
    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;  
}
```



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;  
}
```



(num1 > num2) is true  
since num1=5 and num2= 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 int  
max(int num1, int num2)  
{  
    int result;  
    if (num1 > num2)  
        result = num1;  
    else  
        result = num2;  
  
    return result;  
}
```

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 int  
max(int num1, int num2)  
{  
    int result;  
  
    if (num1 > num2)  
        result = num1;  
    else  
        result = num2;  
  
    return result;  
}
```



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;  
}
```



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;  
}
```

Execute the print statement

```
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;  
}
```

- A return statement is **required** for a value-returning method.
- Example:

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



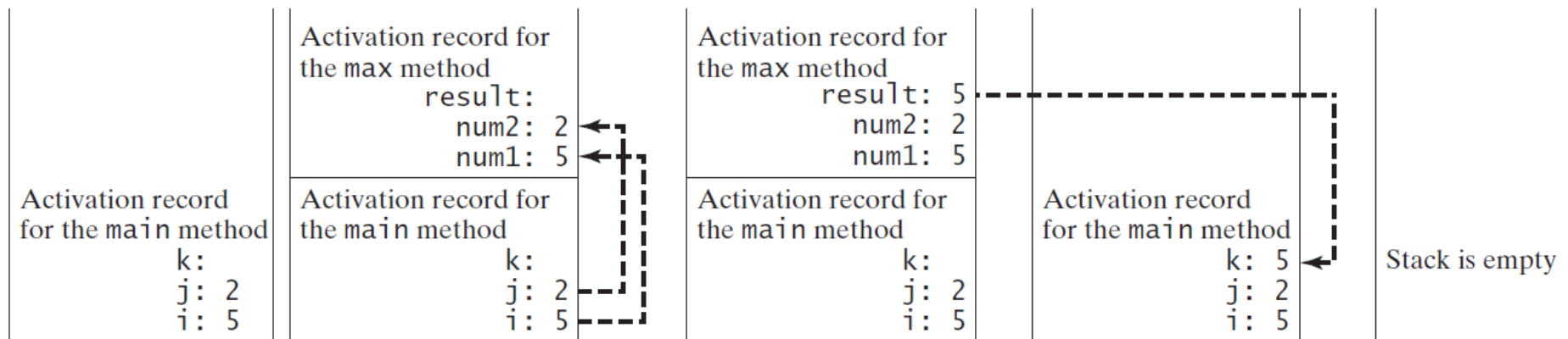
```
public static int sign(int n) {
    if (n > 0)
        return 1;
    else if (n == 0)
        return 0;
    else
        return -1;
}
```

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



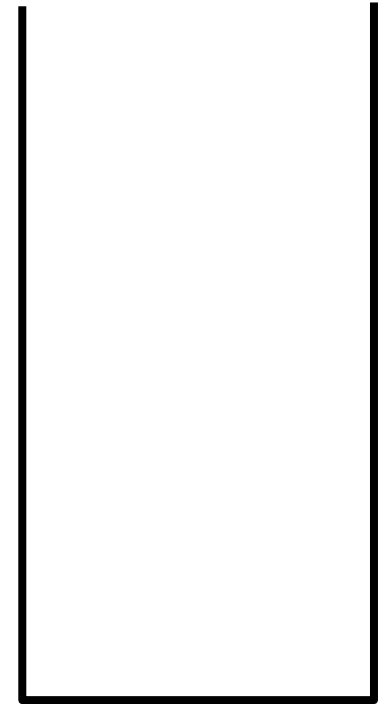
- One of the benefits of methods is for reuse.
- The `max` method can be invoked from any class besides `TestMax`.
- If you create a new class `Test`, you can invoke the `max` method using `ClassName.methodName` (e.g., `TestMax.max`).





```
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;  
}
```



The main method is invoked

```
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);
}
```

i is declared  
and initialized

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

    return result;
}
```

i: 5

The main method is invoked

k is declared

```
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;
}
```

k:  
j: 2  
i: 5

The main method is invoked

```
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);
}
```

Invoke max(i, j)

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

    return result;
}
```

k:  
j: 2  
i: 5

The main method is invoked

```
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;
}
```

pass the values of  
i and j to  
num1 and num2

num2: 2

num1: 5

k:

j: 2

i: 5

The max method is invoked

```
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);
}
```

Declare result

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

    return result;
}
```

result:

num2: 2

num1: 5

k:

j: 2

i: 5

The max method is invoked

```
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;
}
```

(num1 > num2)  
is true

result:

num2: 2

num1: 5

k:

j: 2

i: 5

The max method is invoked



```
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;
}
```

Assign num1 to  
result

result: 5  
num2: 2  
num1: 5

k:

j: 2

i: 5

The max method is invoked

```
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;
}
```

Return `result`  
and assign it to `k`

result: 5
num2: 2
num1: 5
k: 5
j: 2
i: 5

The max method is invoked

```
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);
}
```

Execute print  
statement

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

    return result;
}
```

k: 5  
j: 2  
i: 5

The main method is invoked

- Use `void` for methods that do not return values
- Example:
  - `public static void main(String[] args)`
- This type of method does not return a value.
- The method performs some actions.

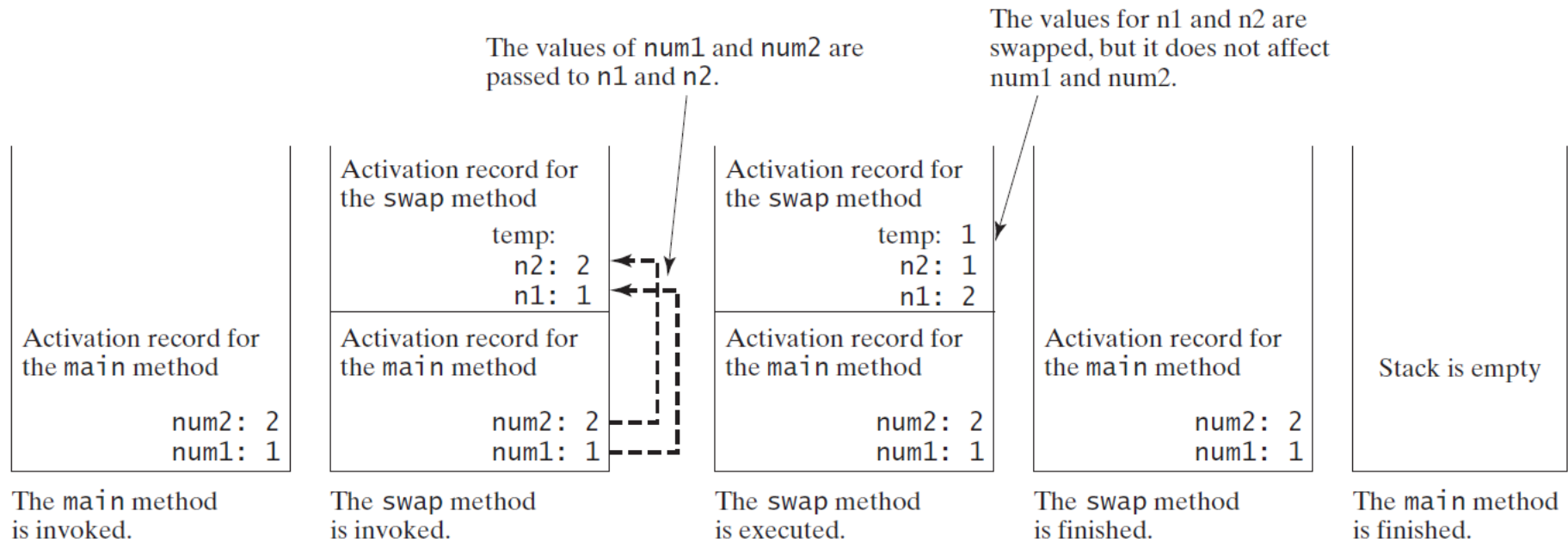
```
public static void nPrintln(String message, int n) {  
    for (int i = 0; i < n; i++)  
        System.out.println(message);  
}
```

- 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");`

```
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);  
    }  
}
```

```
public class TestPassByValue {
    public static void main(String[] args) {
        int num1 = 1;
        int num2 = 2;
        System.out.println("Before, num1 is " + num1 + " and num2 is " + num2);
        swap(num1, num2);
        System.out.println("After, num1 is " + num1 + " and num2 is " + num2);
    }

    public static void swap(int n1, int n2) {
        System.out.println("\tInside the swap method");
        System.out.println("\t\tBefore, n1 is " + n1 + " and n2 is " + n2);
        // Swap n1 with n2
        int temp = n1;
        n1 = n2;
        n2 = temp;
        System.out.println("\t\tAfter, n1 is " + n1 + " and n2 is " + n2);
    }
}
```





- reduce redundant coding.
- enable code reuse.
- modularize code
- improve the quality of the program.

- Overloading the max Method

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

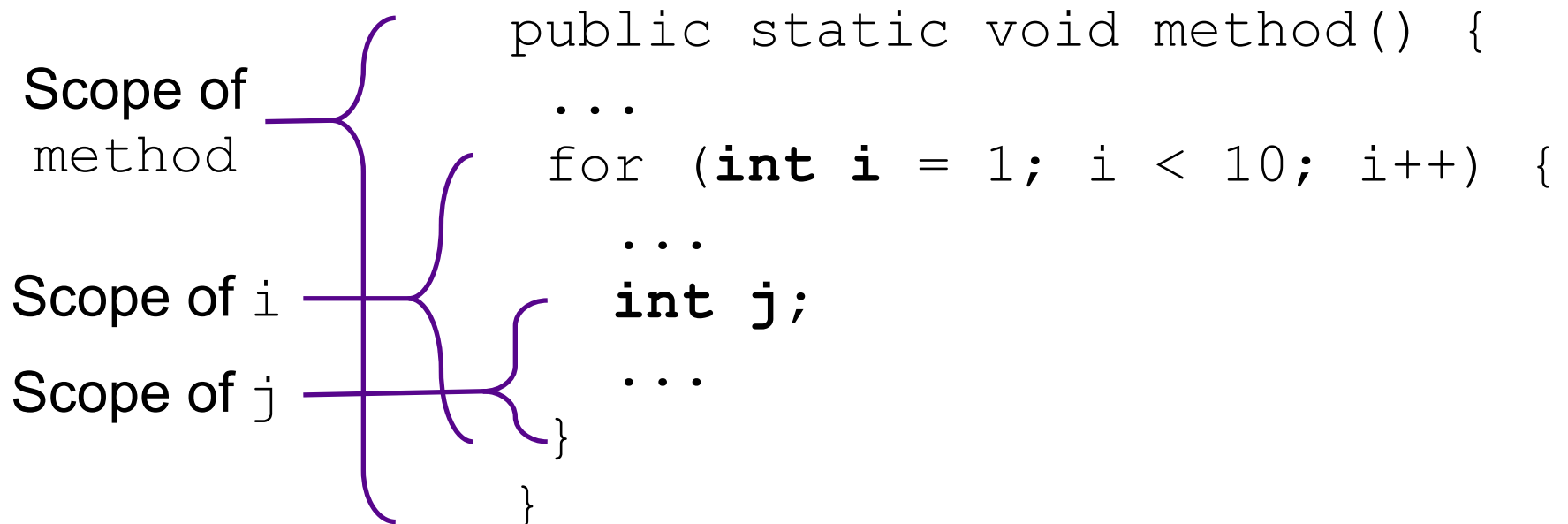
- Sometimes there may be two or more possible matches for an invocation of a method, but the compiler cannot determine the most specific match.
- This is referred to as ambiguous invocation.
- Ambiguous invocation is a compile error.



```
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;  
    }  
}
```

- 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.

- 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.



```

public static void method() {
    ...
    for (int i = 1; i < 10; i++) {
        ...
        int j;
        ...
    }
}

```

Scope of method

Scope of *i*

Scope of *j*



```
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;  
        }  
    }  
}
```



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



```
// Fine with no errors
public static void correctMethod() {
    int x = 1;
    int y = 1;
    // i is declared
    for (int i = 1; i < 10; i++) {
        x += i;
    }
    // i is declared again
    for (int i = 1; i < 10; i++) {
        y += i;
    }
}
```

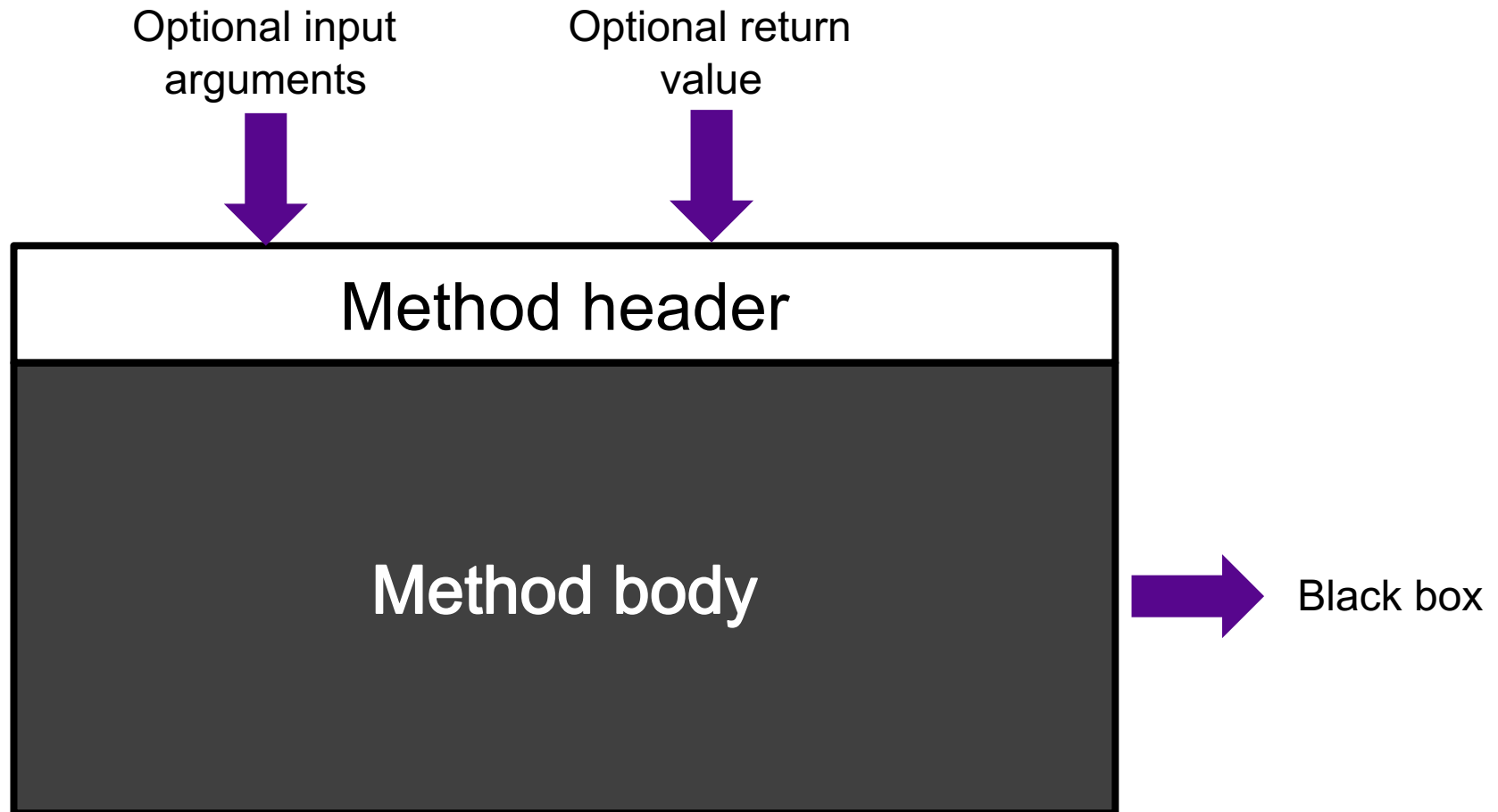
```
// Fine with no errors
public static void correctMethod() {
    int x = 1;
    int y = 1;
    // i is declared
    for (int i = 1; i < 10; i++) {
        x += i;
    }
    // i is declared again
    for (int i = 1; i < 10; i++) {
        y += i;
    }
}
```

```
// With errors
public static void incorrectMethod() {
    int x = 1;
    int y = 1;
    for (int i = 1; i < 10; i++) {
        int x = 0;
        x += i;
    }
}
```



```
// With errors  
public static void incorrectMethod() {  
    int x = 1;  
    int y = 1;  
    for (int i = 1; i < 10; i++) {  
        int x = 0;  
        x += i;  
    }  
}
```

- You can think of the method body as a black box that contains the detailed implementation for the method.



- Write a method once and reuse it anywhere.
- Information hiding. Hide the implementation from the user.
- Reduce complexity.

- The concept of method abstraction can be applied to the process of developing programs.
- When writing a large program, you can use the “divide and conquer” strategy, also known as stepwise refinement, to decompose it into subproblems.
- The subproblems can be further decomposed into smaller, more manageable problems.



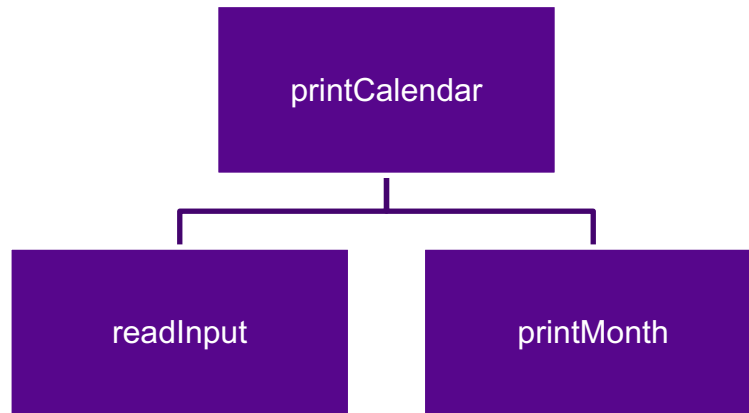
- Let us use the PrintCalendar example to demonstrate the stepwise refinement approach.

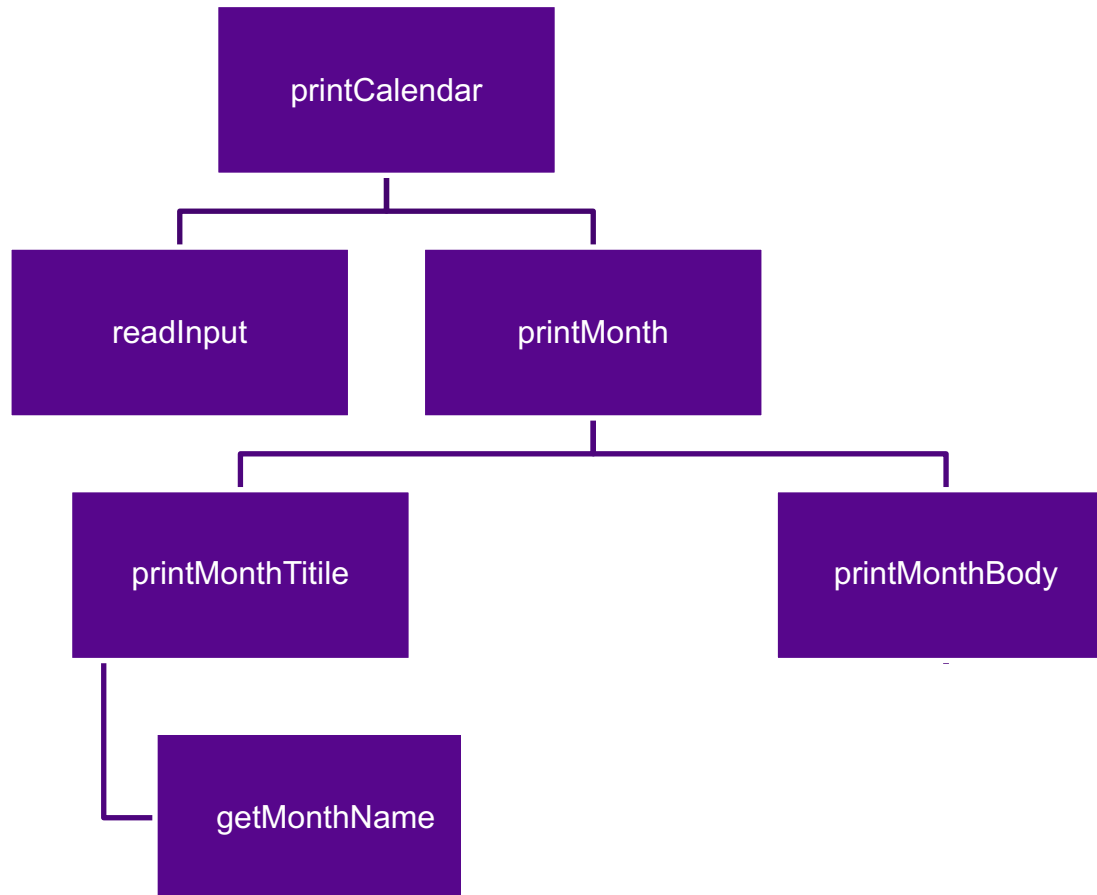
```
C:\book>java PrintCalendar
Enter full year (e.g., 2001): 2009
Enter month in number between 1 and 12: 4
      April 2009
-----
Sun Mon Tue Wed Thu Fri Sat
      1  2  3  4
  5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30

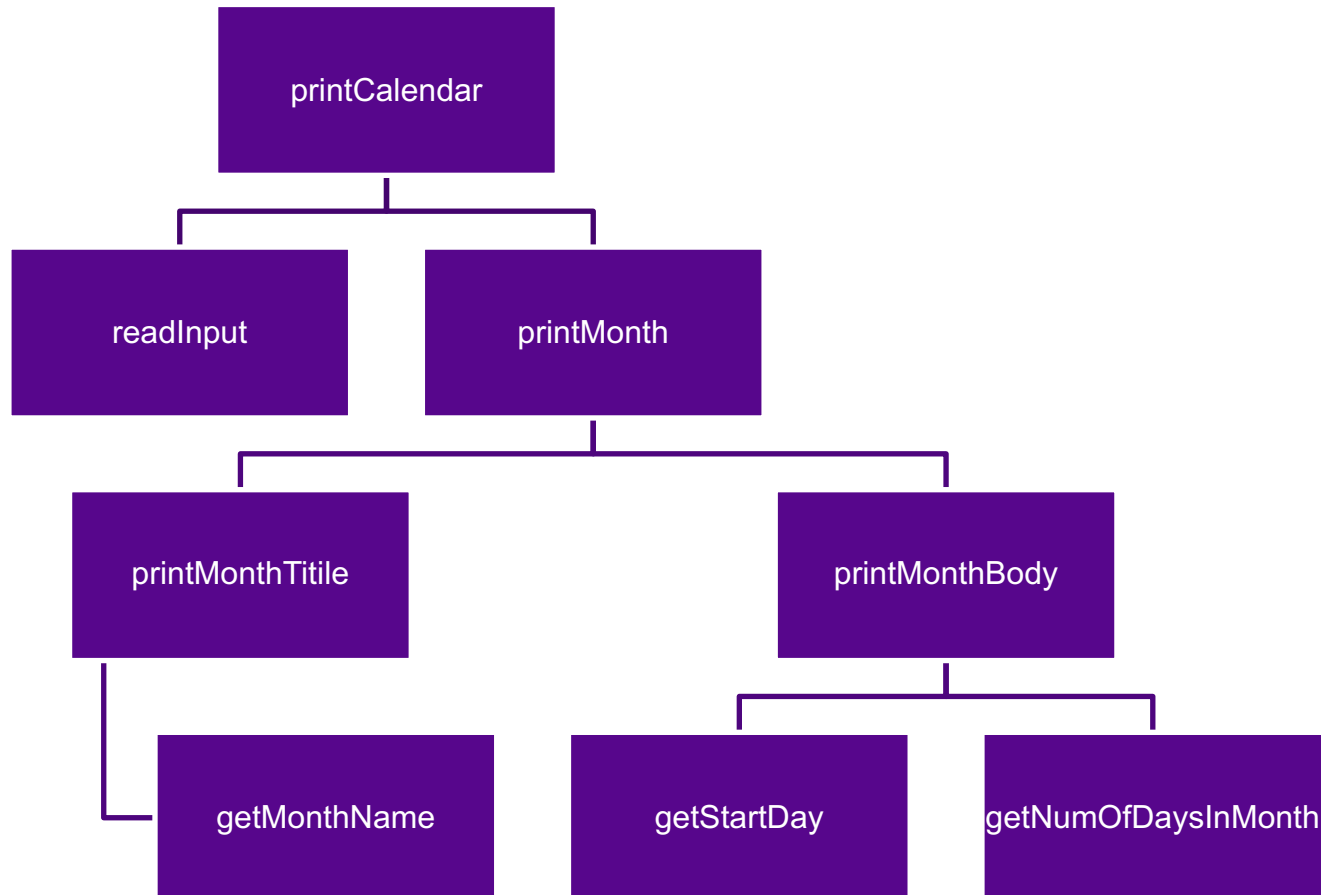
C:\book>
```

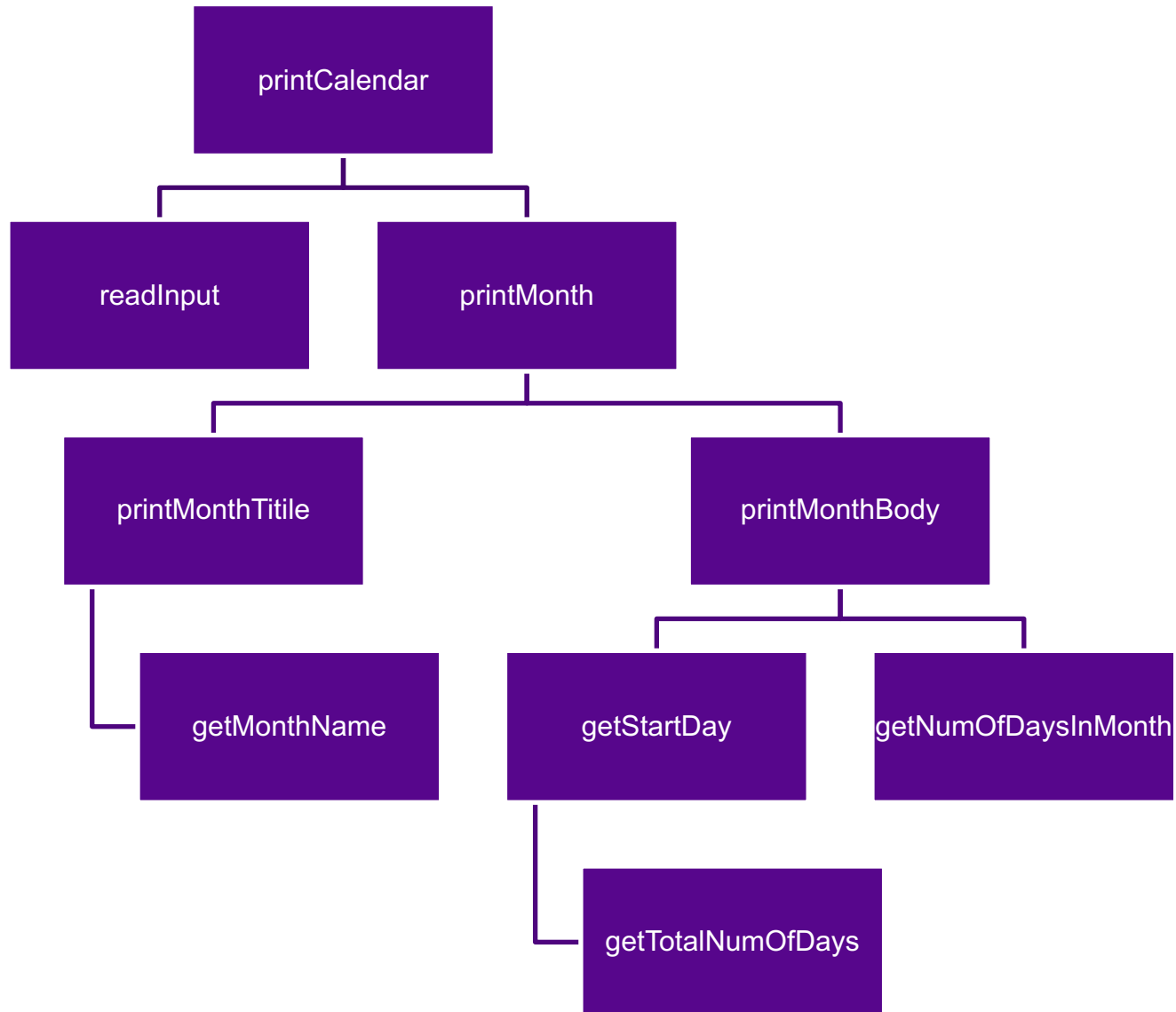
The screenshot shows a Windows Command Prompt window titled "Command Prompt". The user has entered the command `java PrintCalendar`. The program prompts for a full year (2009) and a month (4). It then displays the month name "April 2009" followed by a calendar grid for April 2009. The grid shows days of the week (Sun to Sat) and dates (1 to 30). The prompt `C:\book>` is visible at the bottom.

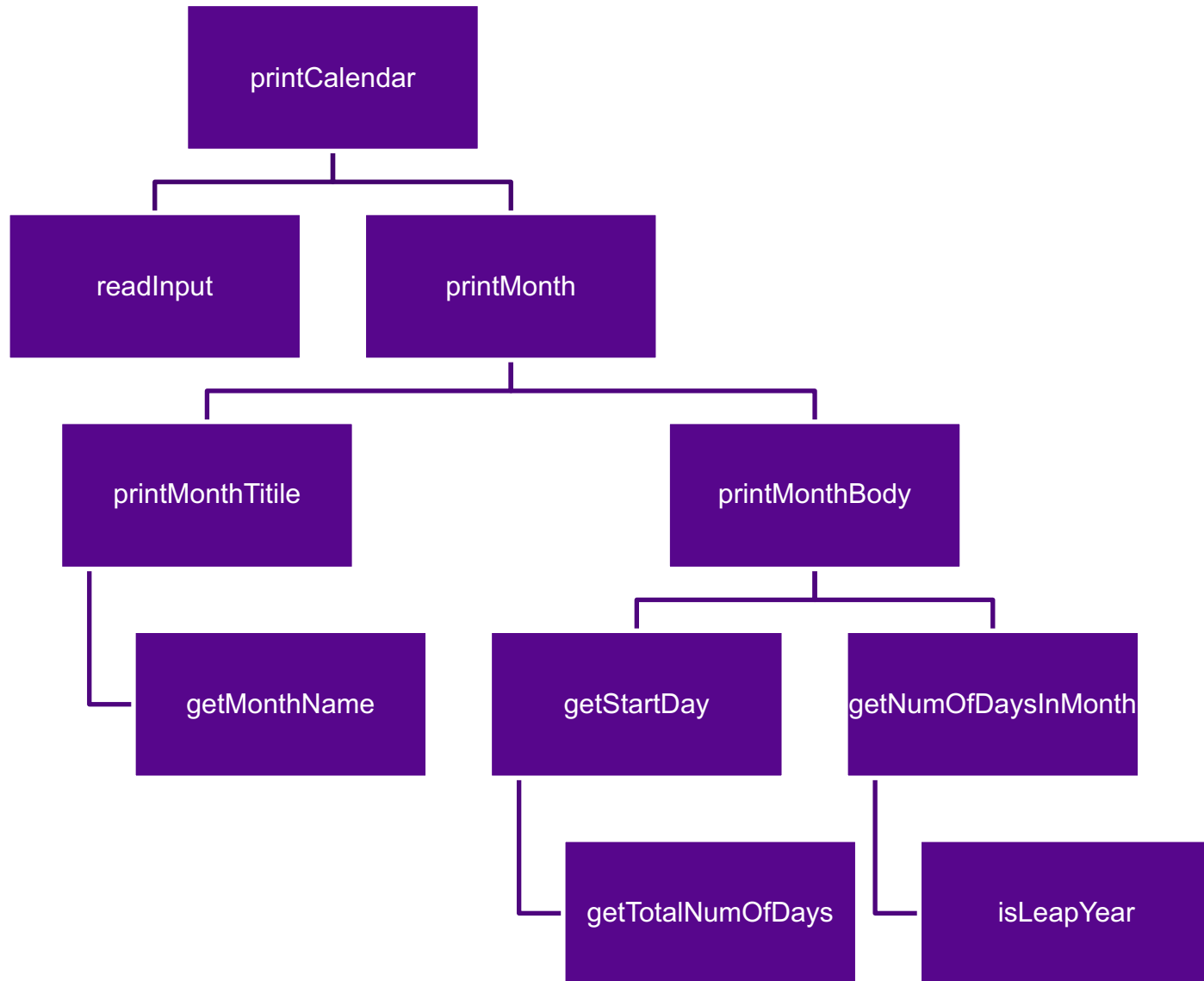












- Top-down approach is to implement one method in the structure chart at a time from the top to the bottom.
- Stubs can be used for the methods waiting to be implemented.
- A stub is a simple but incomplete version of a method.
- The use of stubs enables you to test invoking the method from a caller.
- Implement the main method first and then use a stub for the printMonth method.

- Bottom-up approach is to implement one method in the structure chart at a time from the bottom to the top.
- For each method implemented, write a test program to test it.
- Both top-down and bottom-up methods are fine. Both approaches implement the methods incrementally and help to isolate programming errors and makes debugging easy.
- Sometimes, they can be used together.



- Simpler Program
- Reusing Methods
- Easier Developing, Debugging, and Testing
- Better Facilitating Teamwork