Chương 4: Phương Thức

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

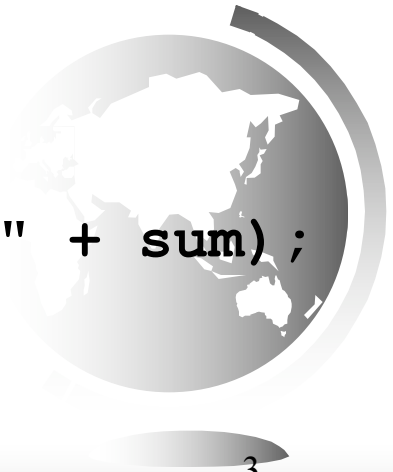
Đặt vấn đề

Tính tổng các số nguyên từ 1 đến 10, từ 20 đến 30 và từ 35 đến 45.

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Đặt vấn đề

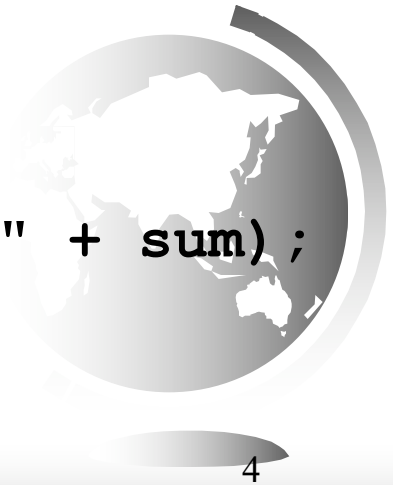
```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
System.out.println("Tổng từ 1 đến 10 là " + sum);

sum = 0;
for (int i = 20; i <= 30; i++)
  sum += i;
System.out.println("Tổng từ 20 đến 30 là " + sum);

sum = 0;
for (int i = 35; i <= 45; i++)
  sum += i;
System.out.println("Tổng từ 35 đến 45 là " + sum);
```



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Đặt vấn đề

```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
System.out.println("Tổng từ 1 đến 10 là " + sum);

sum = 0;
for (int i = 20; i <= 30; i++)
  sum += i;
System.out.println("Tổng từ 20 đến 30 là " + sum);

sum = 0;
for (int i = 35; i <= 45; i++)
  sum += i;
System.out.println("Tổng từ 35 đến 45 là " + sum);
```

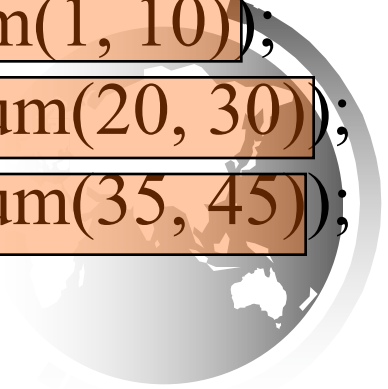


Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Giải pháp

```
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("Tổng từ 1 đến 10 là " + sum(1, 10));
  System.out.println("Tổng từ 20 đến 30 là " + sum(20, 30));
  System.out.println("Tổng từ 35 đến 45 là " + sum(35, 45));
}
```



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Định nghĩa Phương thức

Một phương thức là một tập hợp các câu lệnh được nhóm lại với nhau để giải quyết một vấn đề cụ thể nào đó.

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

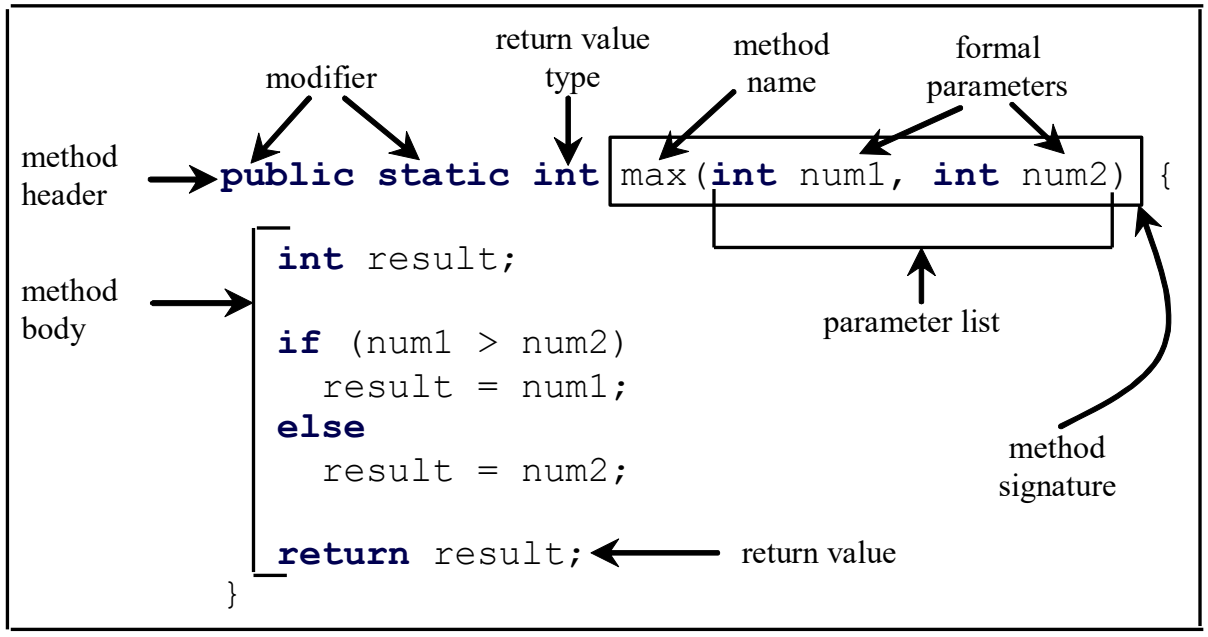
```
int z = max(x, y);
```

actual parameters (arguments)

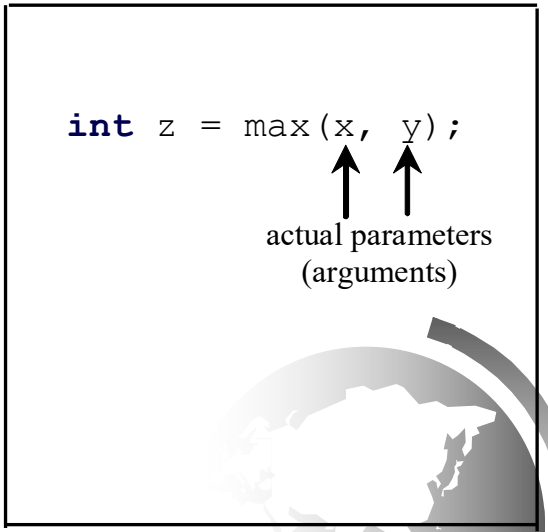
Định nghĩa Phương thức

Một phương thức là một tập hợp các câu lệnh được nhóm lại với nhau để giải quyết một vấn đề cụ thể nào đó.

Define a method



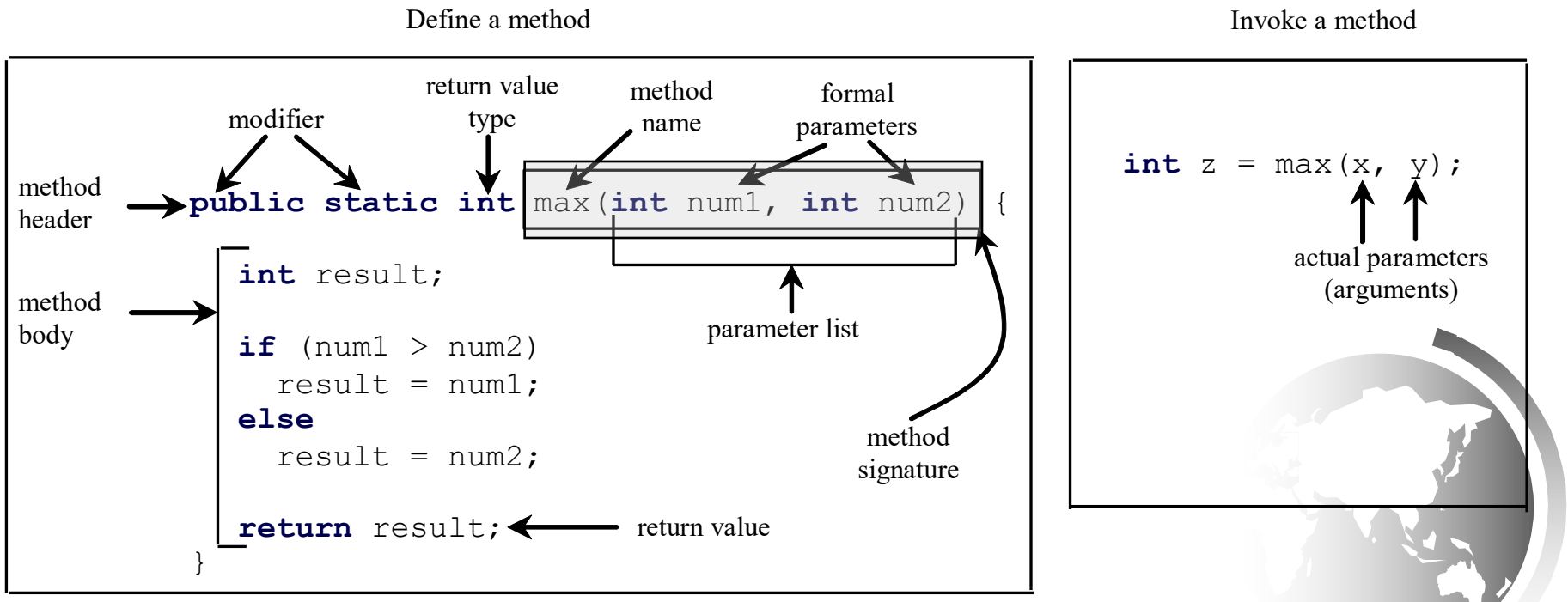
Invoke a method



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Method Signature

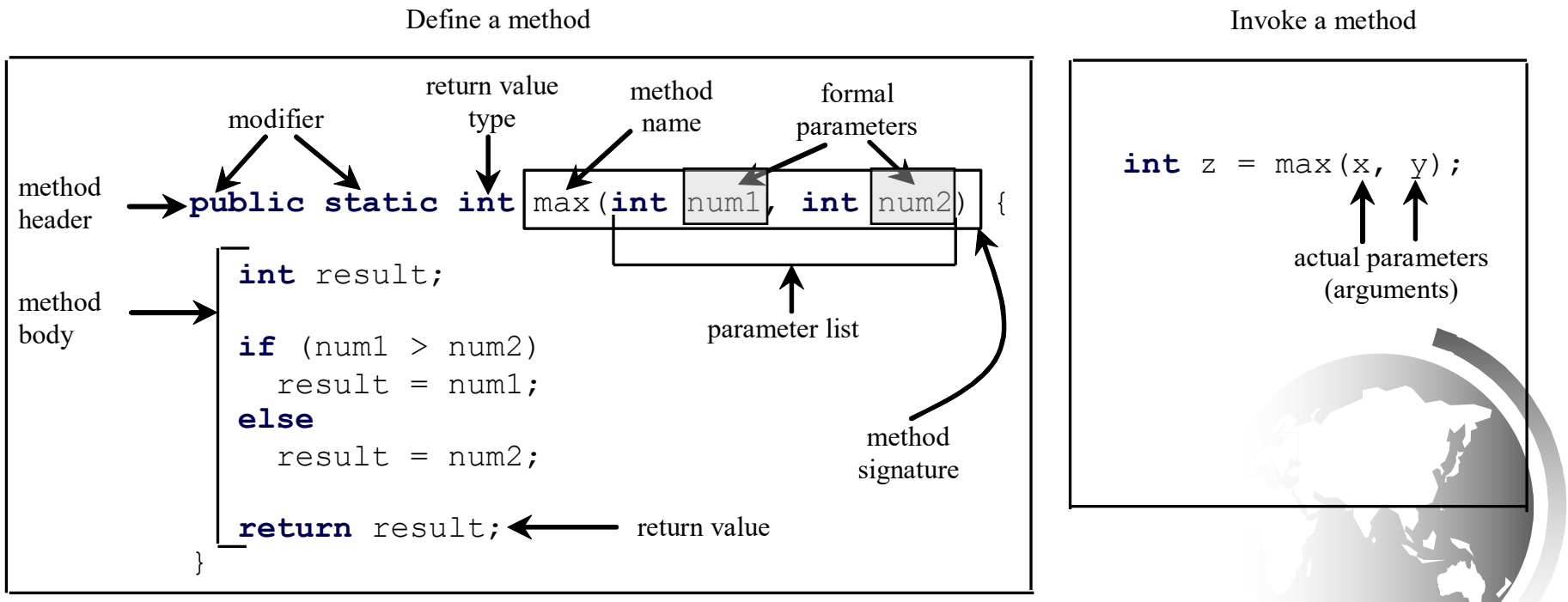
Method signature là sự kết hợp của tên phương thức và danh sách tham số.



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

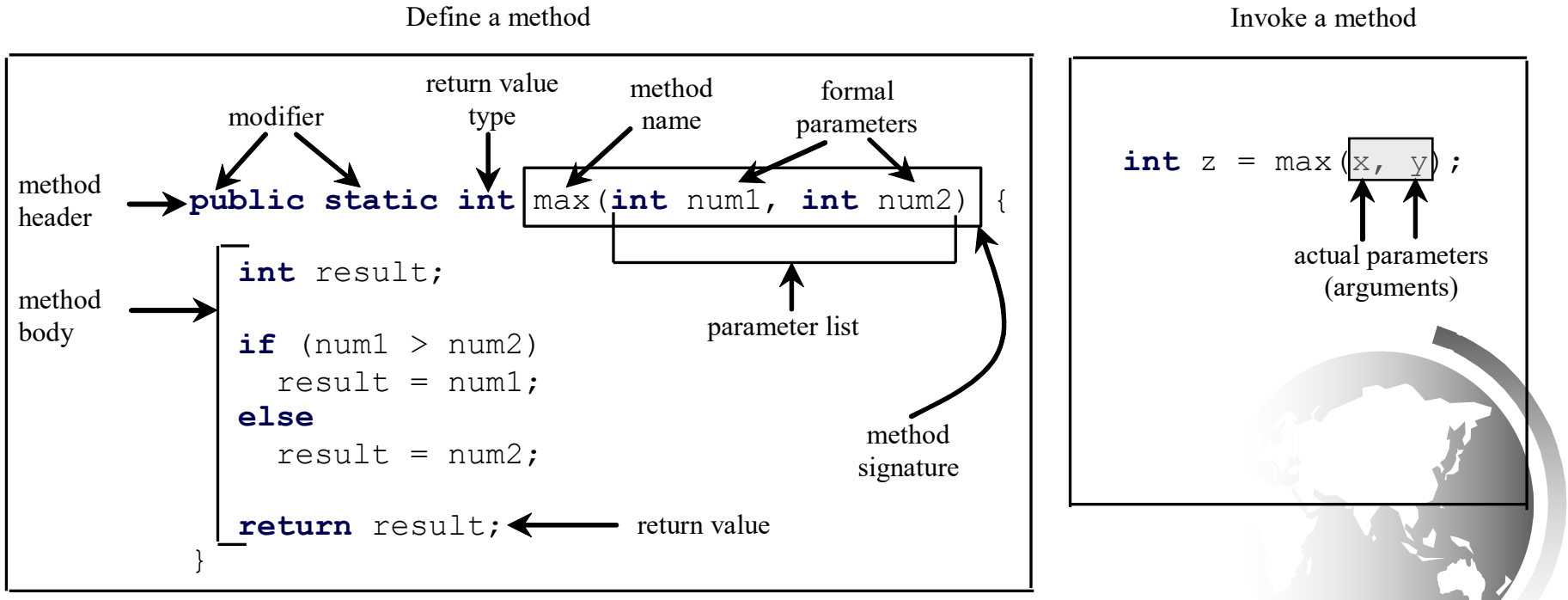
Tham số hình thức

Các biến được định nghĩa trong tiêu đề của phương thức được gọi là các *tham số hình thức*.



Tham số thực

Khi một phương thức được thực thi, chúng ta truyền giá trị vào cho tham số thì giá trị này được gọi là *tham số thực hoặc đối số*.

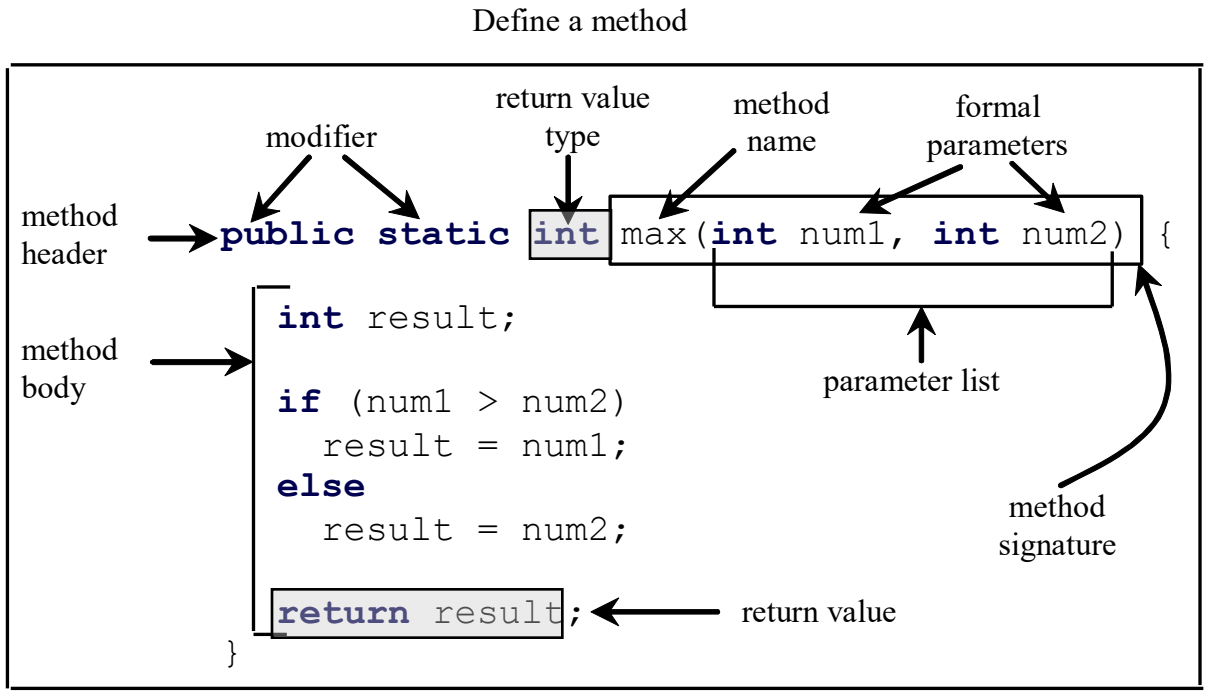


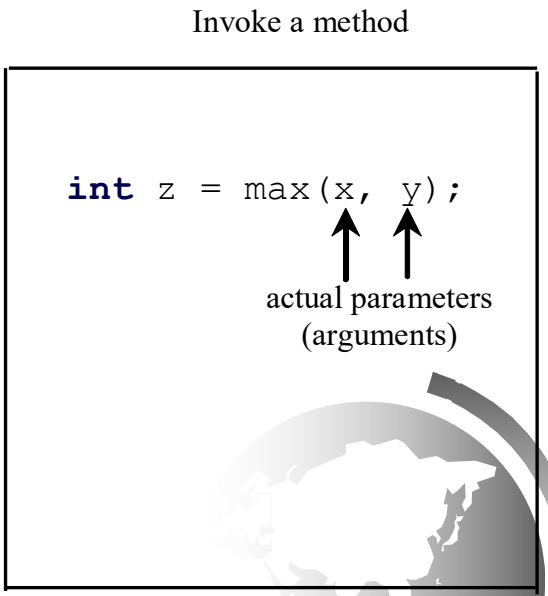
Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Kiểu giá trị trả về

Một phương thức có thể trả về 1 giá trị. Kiểu giá trị trả về là kiểu dữ liệu của giá trị mà phương thức trả về.

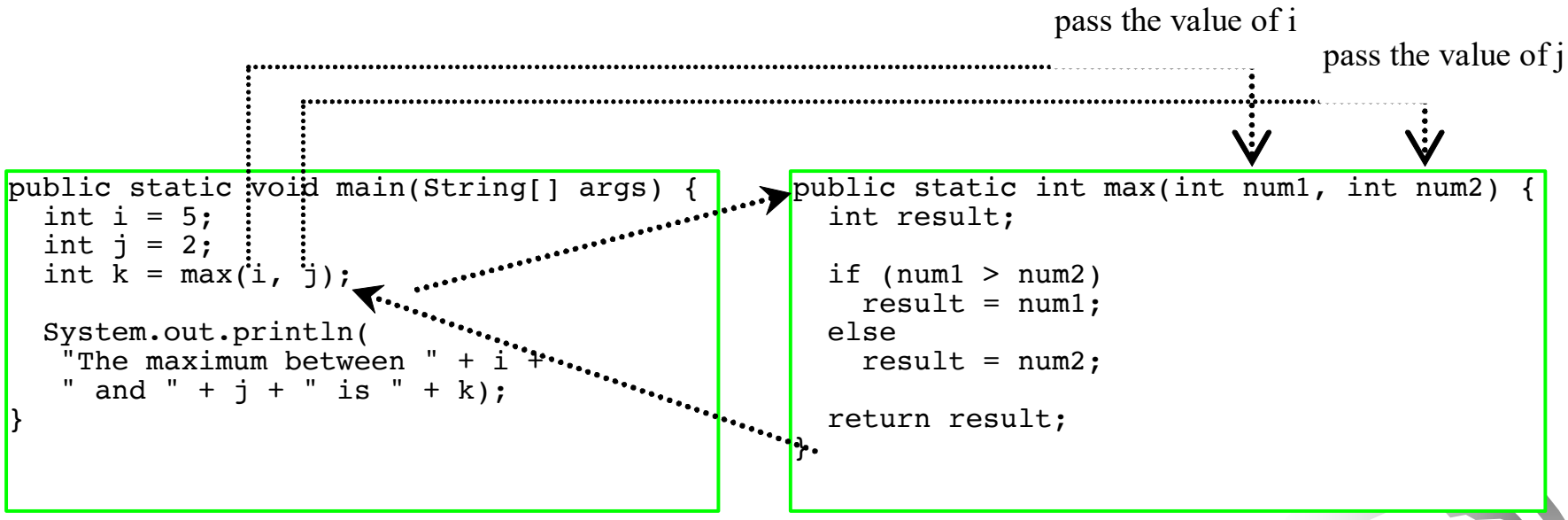
Nếu phương thức không trả về giá trị thì kiểu giá trị trả về là void.





animation

Gọi phương thức



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

animation

Gọi phương thức

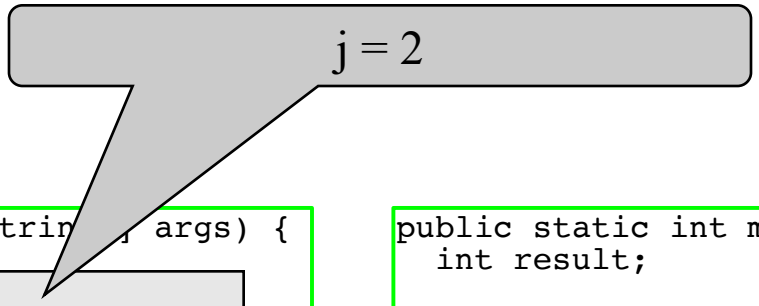
i = 5

```
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;
}
```

animation

Gọi phương thức




```
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;
}
```

animation

Gọi phương thức

Gọi thực thi 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;
}
```

animation

Gọi phương thức

Gọi thực thi max(i, j)
Truyền giá trị của i cho num1
Truyền giá trị của j cho 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;
}
```

animation

Gọi phương thức

Khai báo biến 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;
}
```

animation

Gọi phương thức

(num1 > num2) → true *(vì num1 = 5 và 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;
}
```

animation

Gọi phương thức

result = 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;
}
```

animation

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);
}
```

```
pu      tatic int max(int num1, int num2) {
        esult;

    num1 > num2)
    esult = num1;
  se
    result = num2;

  return result;
}
```

animation

Gọi phương thức

Trở về max(i, j) và gán giá trị trả về cho k

```
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;
}
```

animation

Gọi phương thức

In kết quả ra màn hình

```
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;
}
```

LƯU Ý

Một phương thức có giá trị trả về thì bắt buộc phải có câu lệnh return.

Phương thức (a) đúng về logic, nhưng đó lại là một lỗi biên dịch bởi vì Java compiler nghĩ rằng có thể phương thức này sẽ không trả về giá trị nào cả.

```
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)

Cách xử lý: xóa *if (n < 0)* trong (a), để trình biên dịch có thể thấy một câu lệnh return có thể được thực thi.

Tái sử dụng phương thức

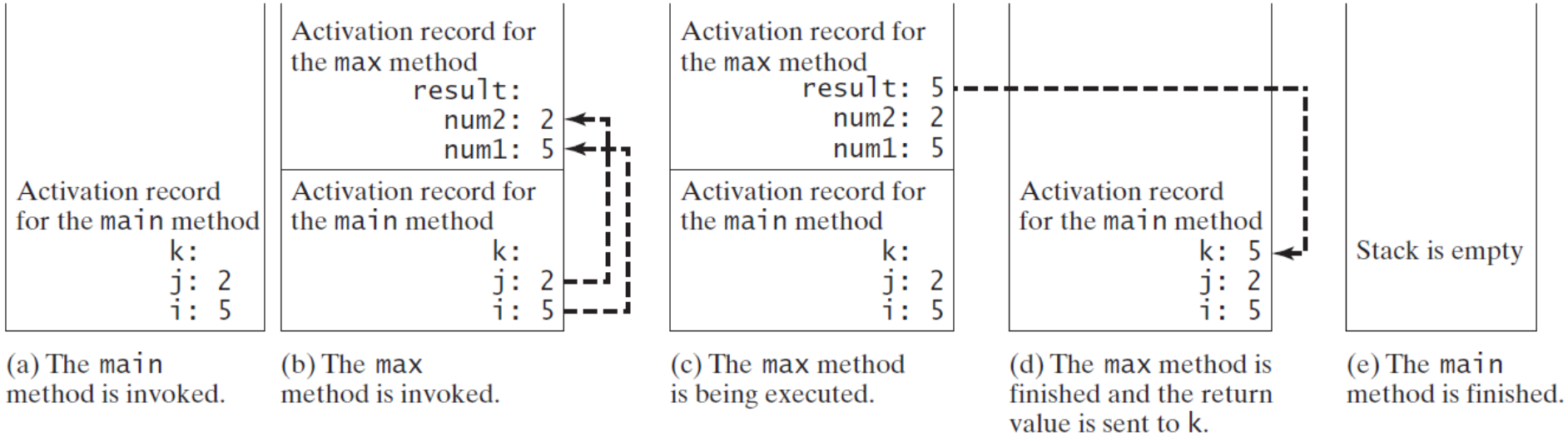
Một trong những lợi ích của phương thức là tái sử dụng.

Phương thức max có thể được gọi từ bất kỳ phương thức nào bên trong lớp TestMax.

Nếu tạo một lớp Test mới thì có thể gọi phương thức max method bằng cách sau ClassName.methodName (vd: TestMax.max).

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Ngăn xếp (Stack) thực thi



(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.

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

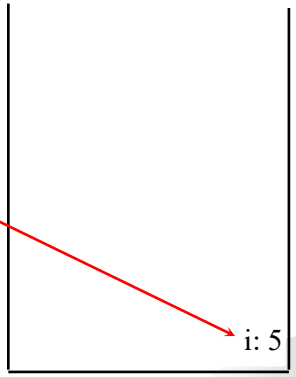
animation

Ngăn xếp thực thi

Khai báo và khởi tạo i

```
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.

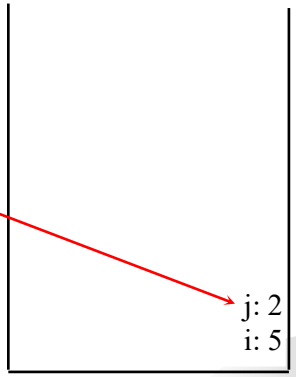
animation

Ngăn xếp thực thi

Khai báo và khởi tạo 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;
}
```



The main method is invoked.

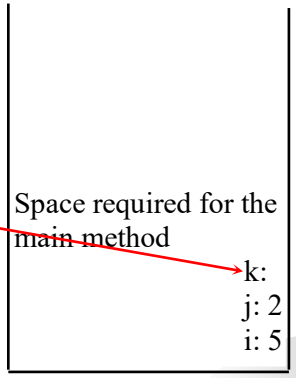
animation

Ngăn xếp thực thi

Khai báo 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;
}
```



The main method is invoked.

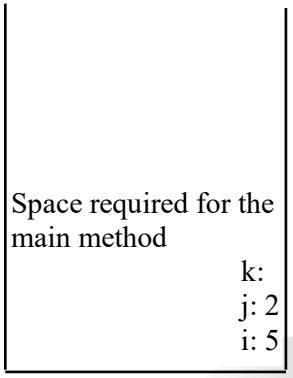
animation

Ngăn xếp thực thi

Gọi 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;
}
```



The main method is invoked.

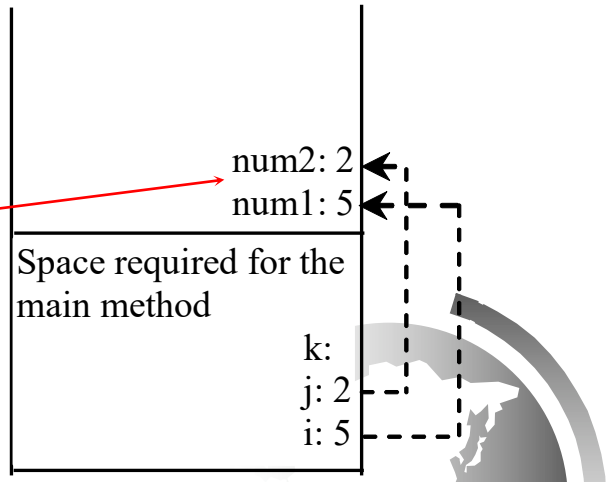
animation

Ngăn xếp thực thi

Truyền giá trị của i và j cho num1 và 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;
}
```



The max method is invoked.

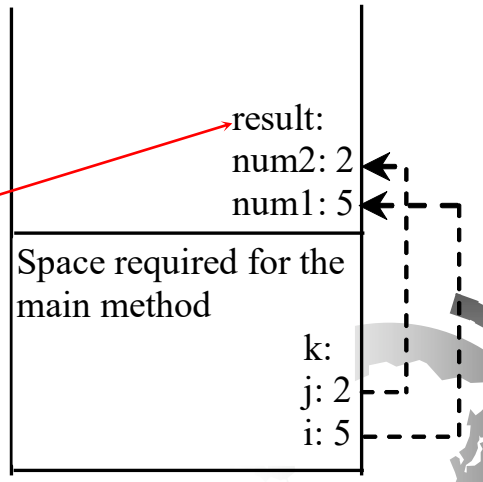
animation

Ngăn xếp thực thi

```
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;
}
```

Khai báo result



The max method is invoked.

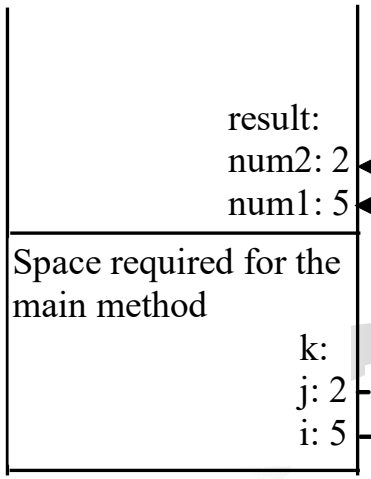
animation

Ngăn xếp thực thi

```
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) → true



The max method is invoked.

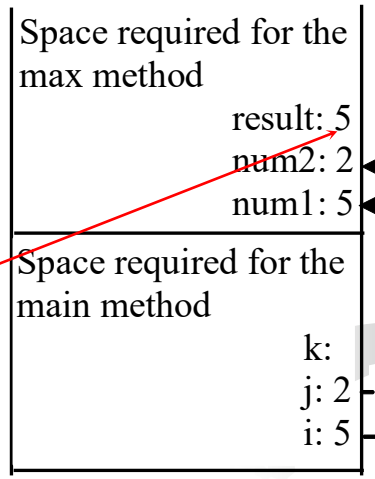
animation

Ngăn xếp thực thi

```
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;
}
```

Gán num1 cho result



The max method is invoked.

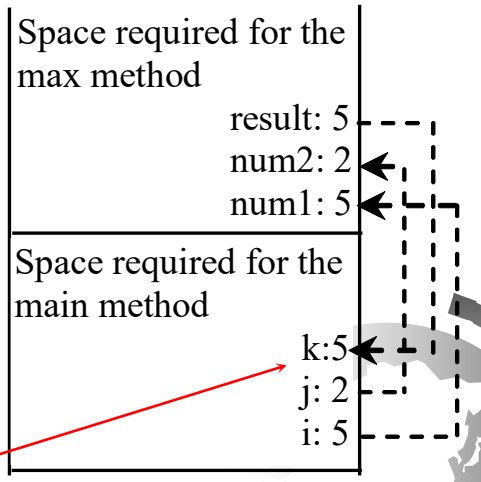
animation

Ngăn xếp thực thi

Trả result về và gán cho 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;
}
```



The max method is invoked.

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

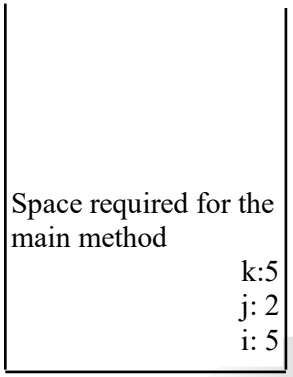
animation

Ngăn xếp thực thi

In kết quả

```
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.

Truyền tham số

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

Giả sử gọi phương thức:

nPrintln("Welcome to Java", 5);

Thì kết quả là gì?

Giả sử gọi phương thức:

nPrintln("Computer Science", 15);

Thì kết quả là gì?

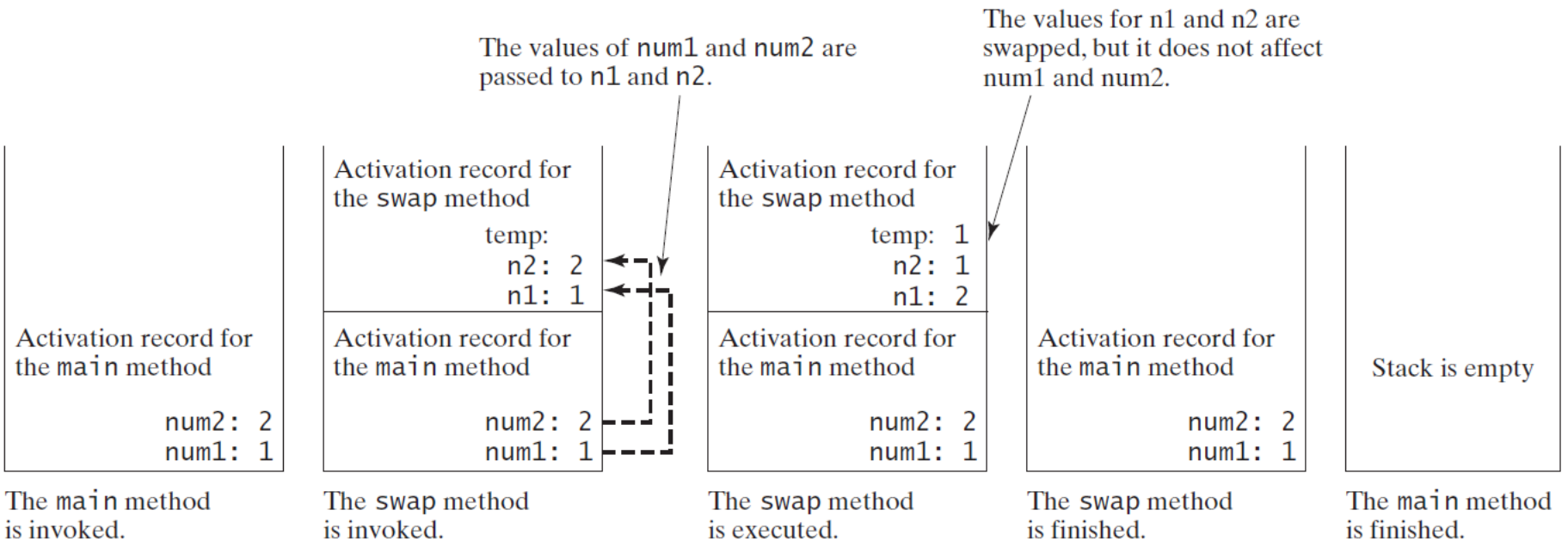
Giả sử gọi phương thức:

nPrintln(15, "Computer Science");

Thì kết quả là gì?

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Truyền giá trị
(Truyền tham trị)



Nạp chồng phương thức

Nạp chồng phương thức `max`

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

Lời gọi hàm mơ hồ
(Ambiguous Invocation)

Đôi khi có thể có nhiều hơn 1 kết quả trùng khớp cho 1 lời gọi hàm → trình biên dịch không thể chọn kết quả trùng khớp nào là cụ thể nhất. Đó gọi là lời *gọi hàm mơ hồ* ➔ là một lỗi biên dịch.

Lời gọi hàm mơ hồ

```
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;
  }
}
```

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Phạm vi của biến cục bộ

Biến cục bộ: là một biến được định nghĩa bên trong một phương thức.

Phạm vi: phần chương trình mà biến có thể được tham chiếu.

Phạm vi của một biến cục bộ bắt đầu từ phần khai báo cho đến cuối khối chứa biến đó.

Biến cục bộ cần được khai báo trước khi sử dụng.

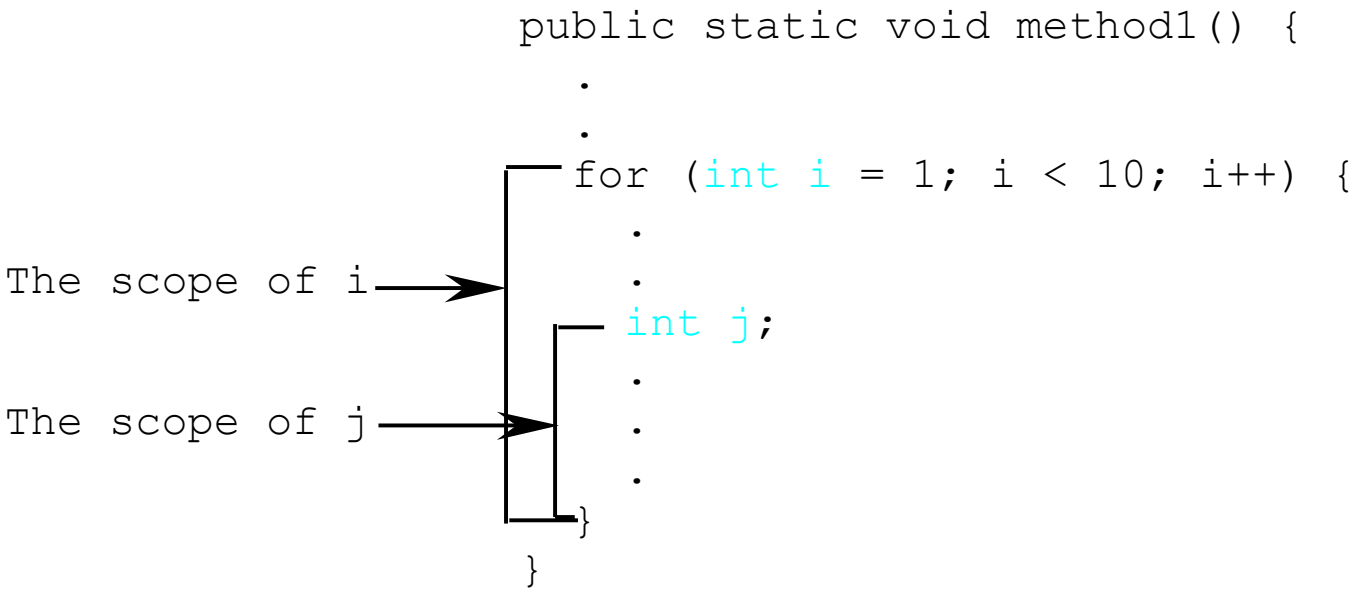
Phạm vi của biến cục bộ

Có thể khai báo các biến cục bộ cùng tên nhiều lần trong các khối riêng rẽ (không lồng nhau) trong một phương thức.

Phạm vi của biến cục bộ

Một biến được khai báo trong biểu thức khởi tạo của vòng lặp for thì có phạm vi toàn bộ vòng lặp đó.

Nhưng một biến được khai báo bên trong thân vòng lặp for thì có phạm vi giới hạn từ vị trí khai báo đến cuối khối chứa biến đó.



Phạm vi của biến cục bộ

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;
  }
}
```

Phạm vi của biến cục bộ

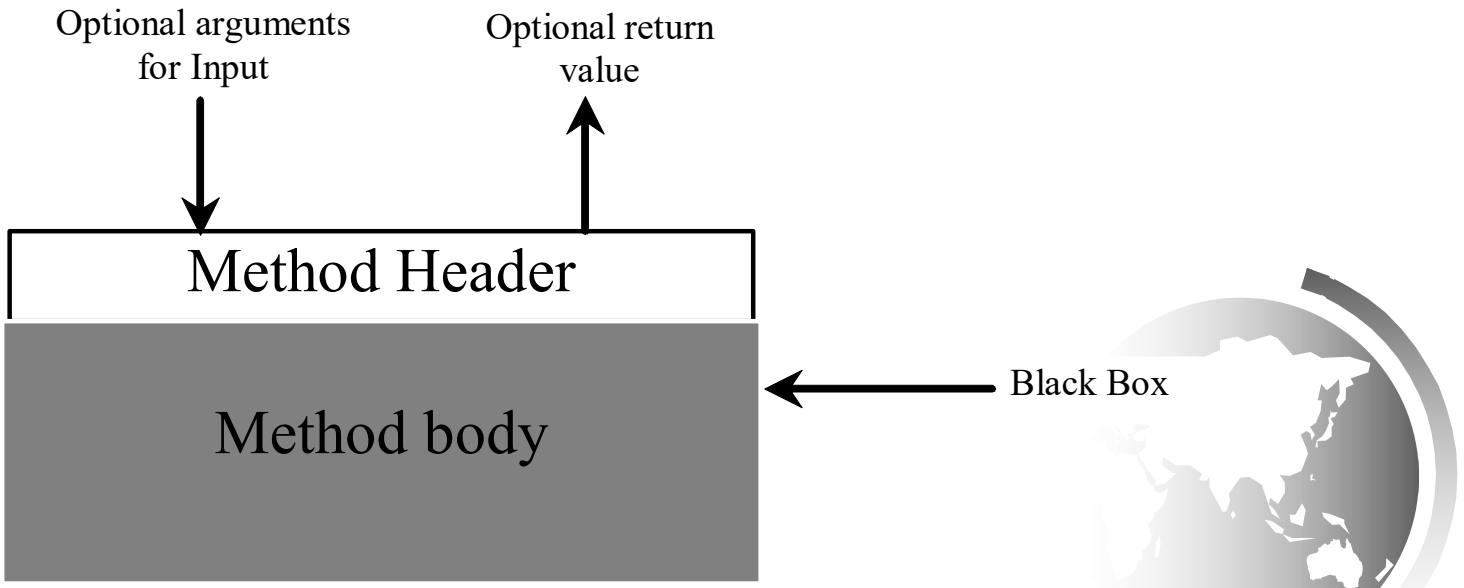
```
// No errors
public static void correctMethod() {
  int x = 1;
  int y = 1;
  // i is declared
  for (int i = 1; i < 10; i++) {
    x += i;
  }
  // i is declared again
  for (int i = 1; i < 10; i++) {
    y += i;
  }
}
```

Phạm vi của biến cục bộ

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

Phương thức trừu tượng

Có thể xem phần thân của phương thức như một hộp đen chứa các cài đặt cụ thể cho phương thức.



Ưu điểm của Phương thức

- Viết một phương thức một lần và tái sử dụng nó ở bất kỳ chỗ nào.
- Ẩn thông tin. Ẩn phần cài đặt với người dùng.
- Giảm độ phức tạp.

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Phương pháp Làm mịn từng bước
Stepwise Refinement

Khái niệm phương thức trừu tượng có thể được áp dụng cho quá trình phát triển phần mềm.

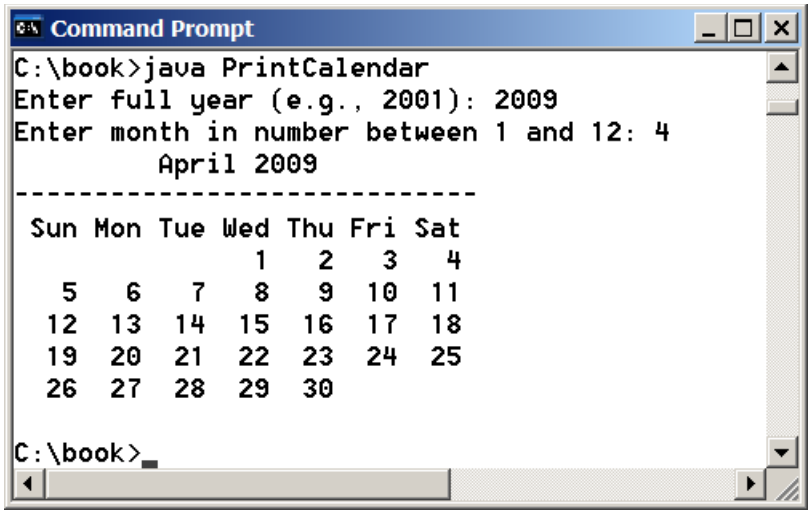
Khi viết một chương trình lớn, chúng ta có thể sử dụng chiến thuật *"chia để trị"* còn được gọi là *làm mịn từng bước* (*stepwise refinement)*, để tách thành các vấn đề nhỏ hơn.

Các vấn đề nhỏ hơn cũng có thể được tách nhỏ hơn nữa.

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

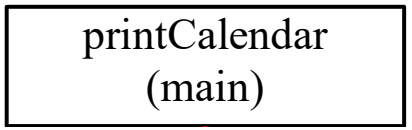
Minh họa phương pháp
Stepwise Refinement

Viết chương trình nhập vào tháng, năm dương lịch. Hãy in lịch của tháng đó.



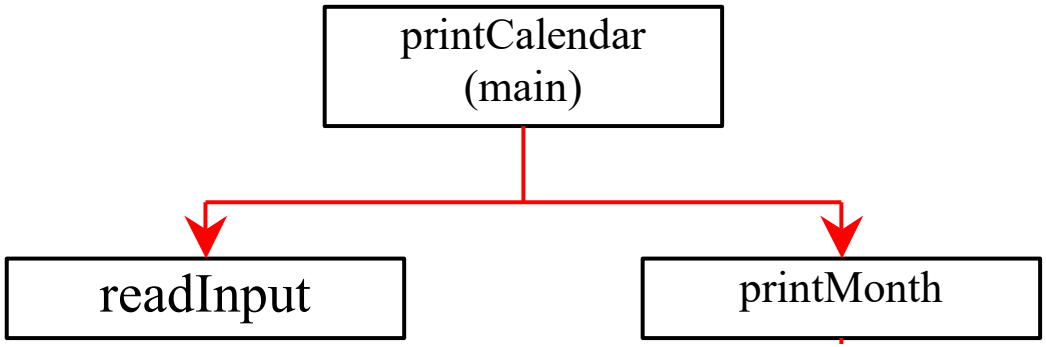
Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Sơ đồ thiết kế



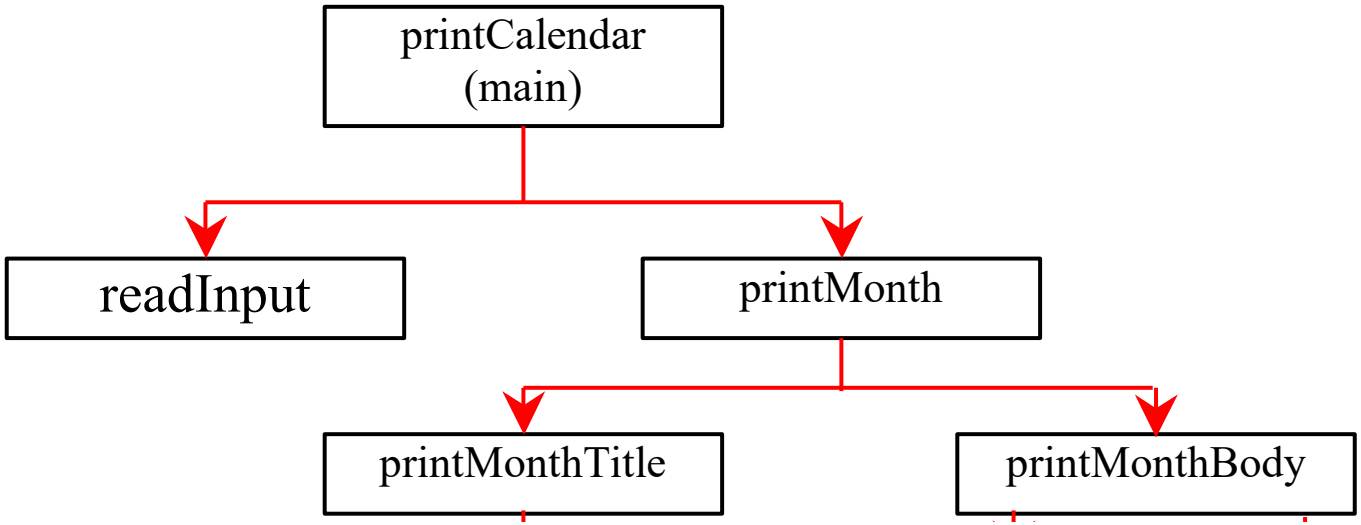
Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Sơ đồ thiết kế

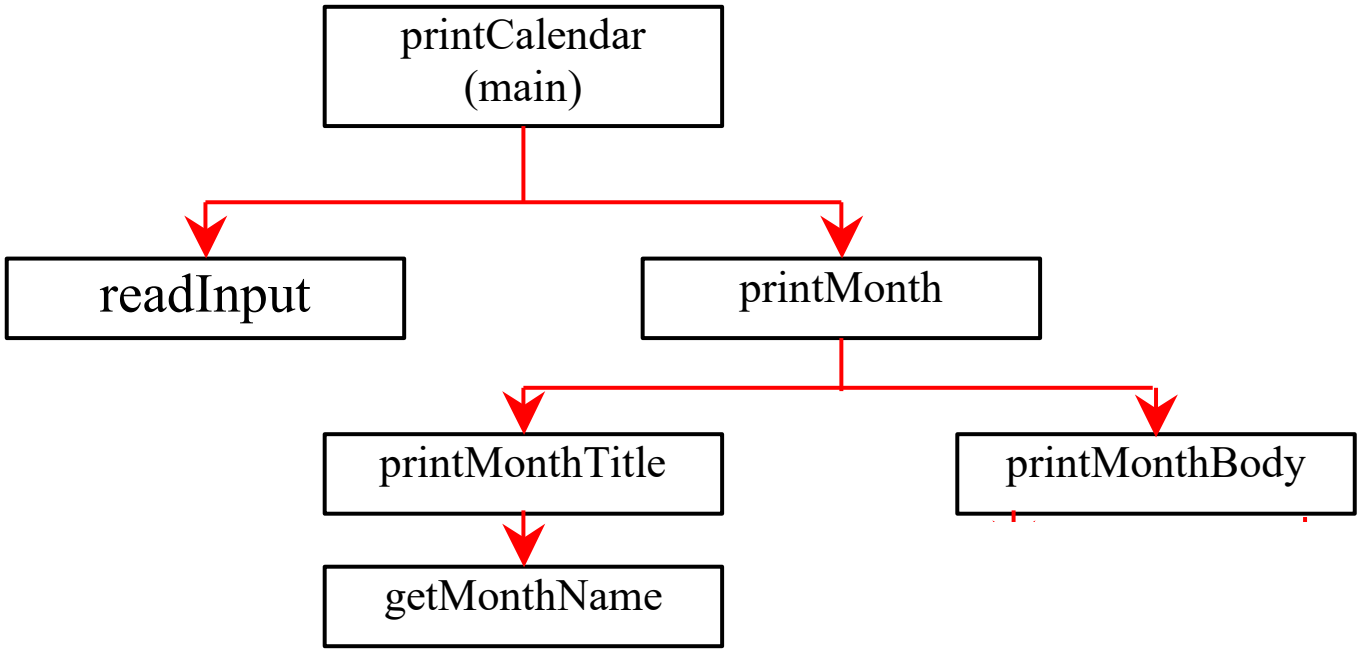


Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

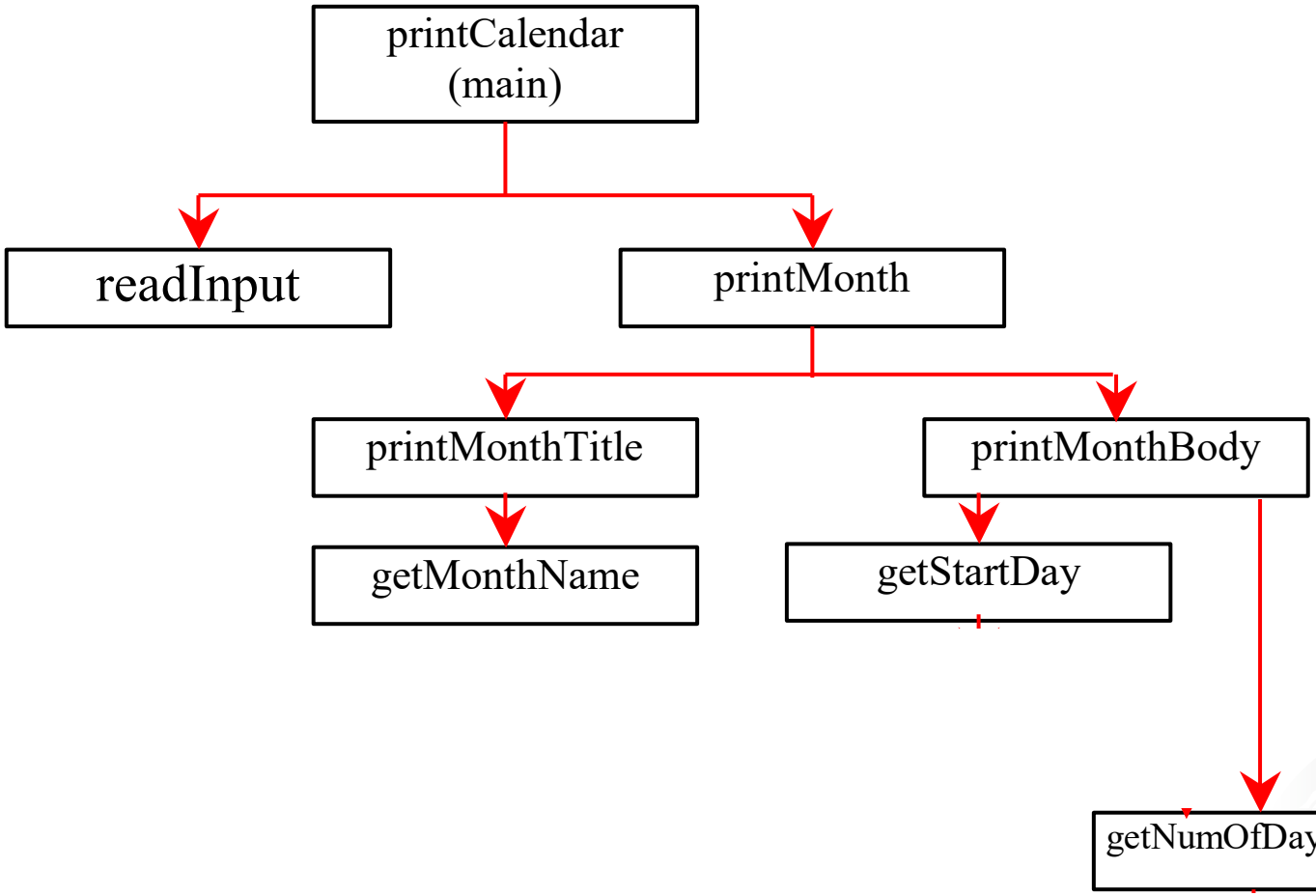
Sơ đồ thiết kế



Sơ đồ thiết kế

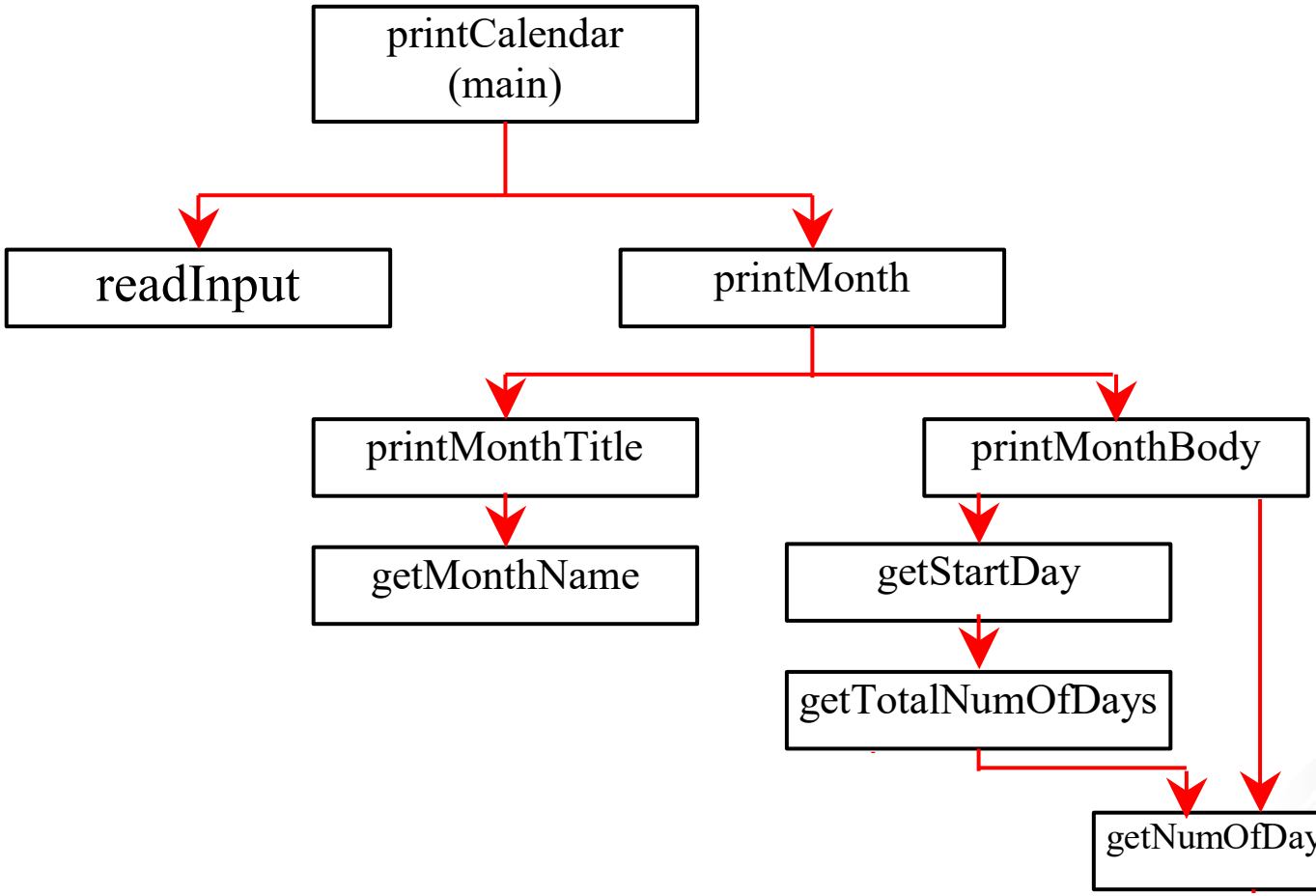


Sơ đồ thiết kế



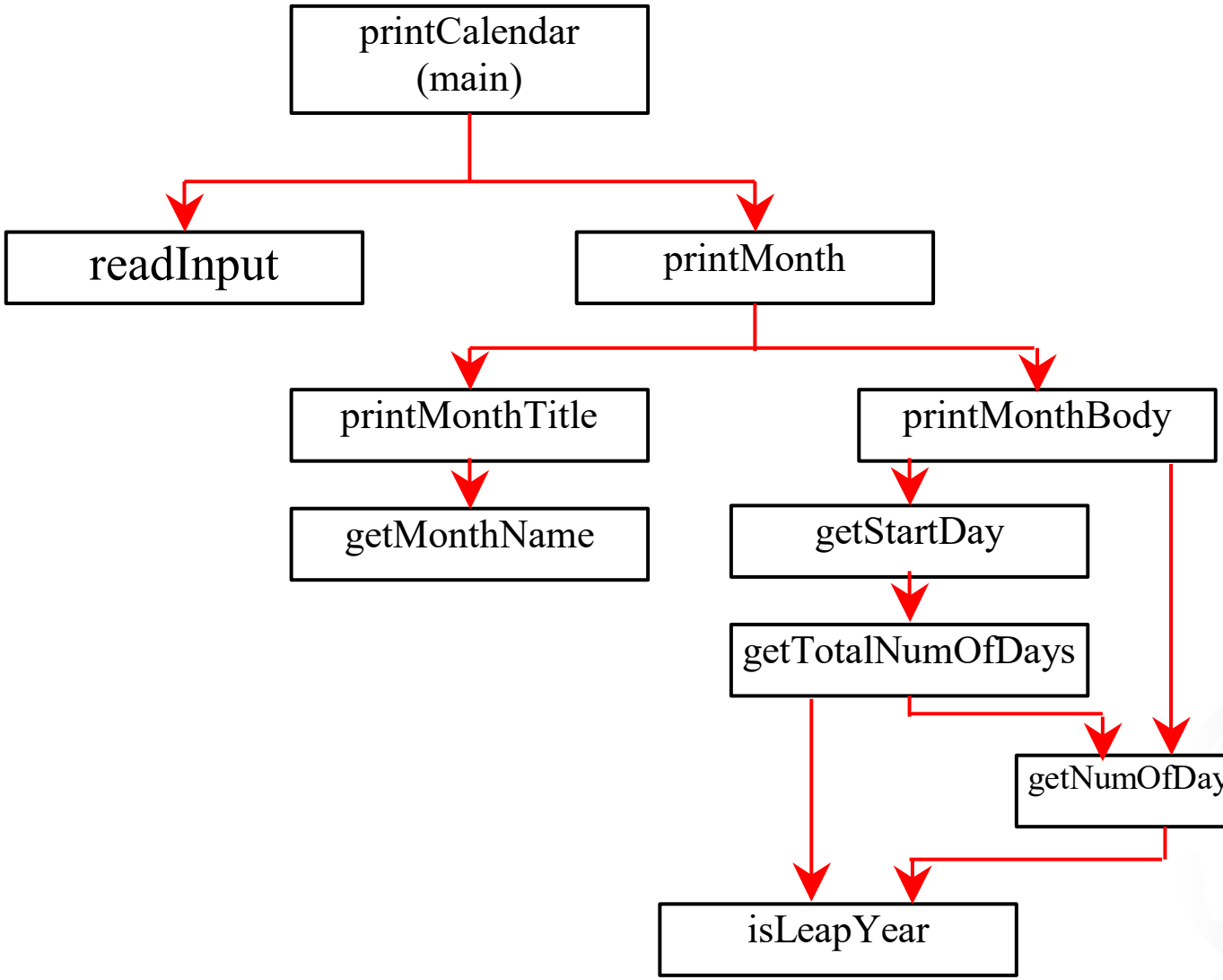
Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Sơ đồ thiết kế



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Sơ đồ thiết kế



Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.

Ưu điểm của

Stepwise Refinement

Chương trình đơn giản hơn

Tái sử dụng các phương thức

Viết mã nguồn, gỡ lỗi và kiểm tra dễ dàng hơn

Better Facilitating Teamwork

Liang, Introduction to Java Programming, Tenth Edition, (c) 2015 Pearson Education, Inc. All rights reserved.