BKACAD®

HỌC VIỆN CÔNG NGHỆ BKACAD

Lập trình Java – Phương thức (Methods)

Ths. Vũ Duy Khương

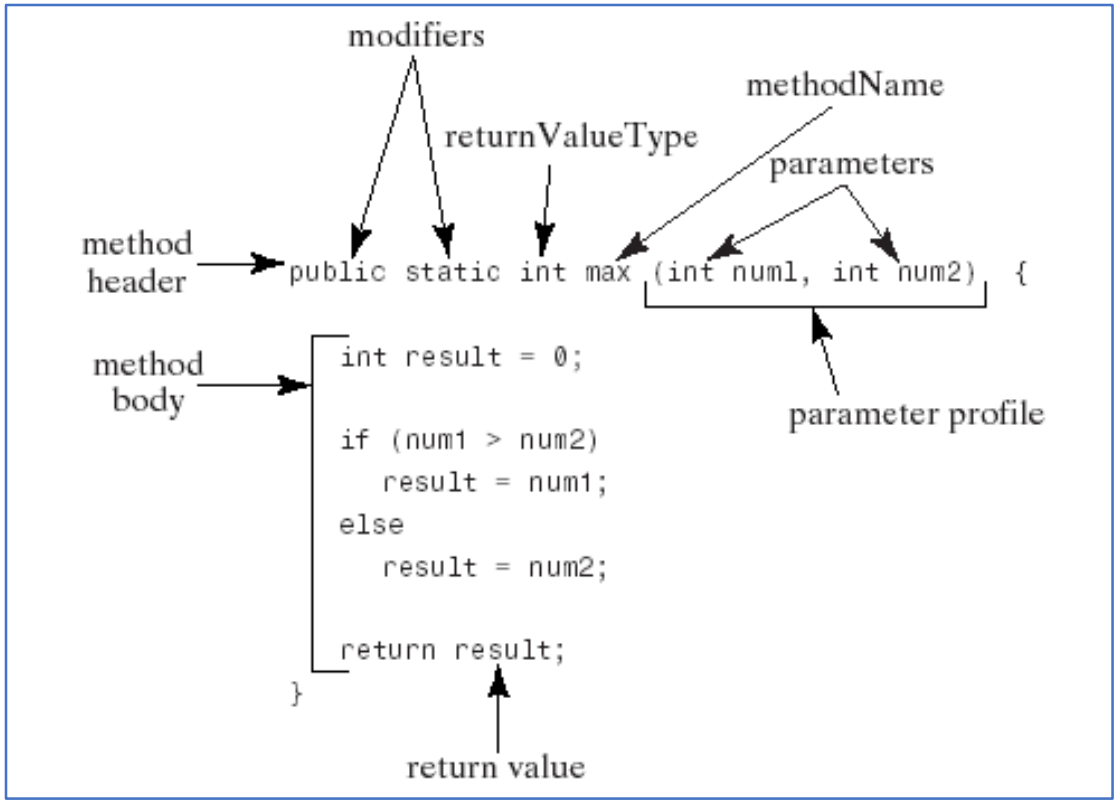
Nội dung

1. Giới thiệu phương thức
2. Truyền tham số
3. Overloading Methods
4. Phạm vi của biến cục bộ
5. Phương thức trừu tượng

Giới thiệu phương thức (method)

Một phương thức là một tập các câu lệnh được nhóm lại với nhau nhằm thực hiện một công việc.

Cấu trúc của phương thức:



Giới thiệu phương thức (tiếp)

- ***parameter profile*** gồm kiểu, thứ tự, và số tham số của một phương thức.
- ***method signature (header)*** gồm tên phương thức và parameter profiles.
- Các tham số khai báo trong method header được gọi là tham số hình thức (*formal parameters*).
- Khi phương thức được gọi, các tham số hình thức được thay thế bởi các biến hoặc dữ liệu, được gọi là các tham số thực sự (*actual parameters*).

Giới thiệu phương thức (tiếp)

- Một phương thức có thể trả về một giá trị. Kiểu của giá trị đó là kiểu dữ liệu của phương thức trả về.
- Nếu phương thức không trả về một giá trị, kiểu của phương thức trả về dùng từ khóa **void**.
- Ví dụ, kiểu giá trị trả về trong phương thức main là void.

Khai báo phương thức

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

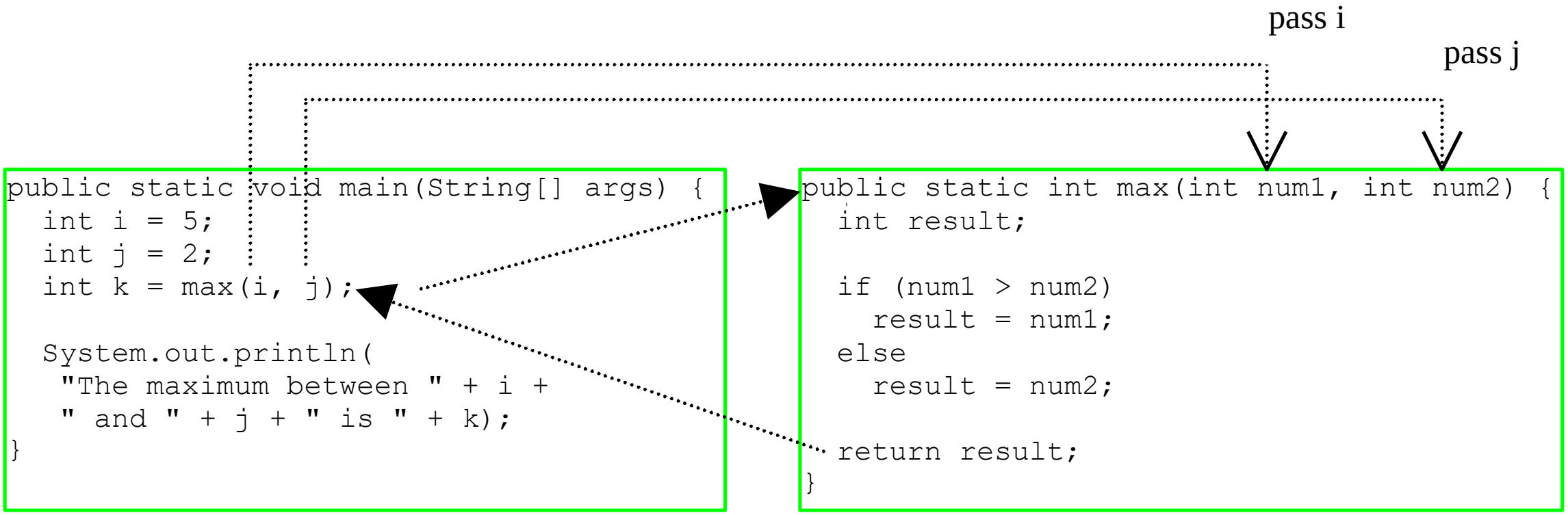
Cách gọi phương thức

Ví dụ 1: Phương thức `max`

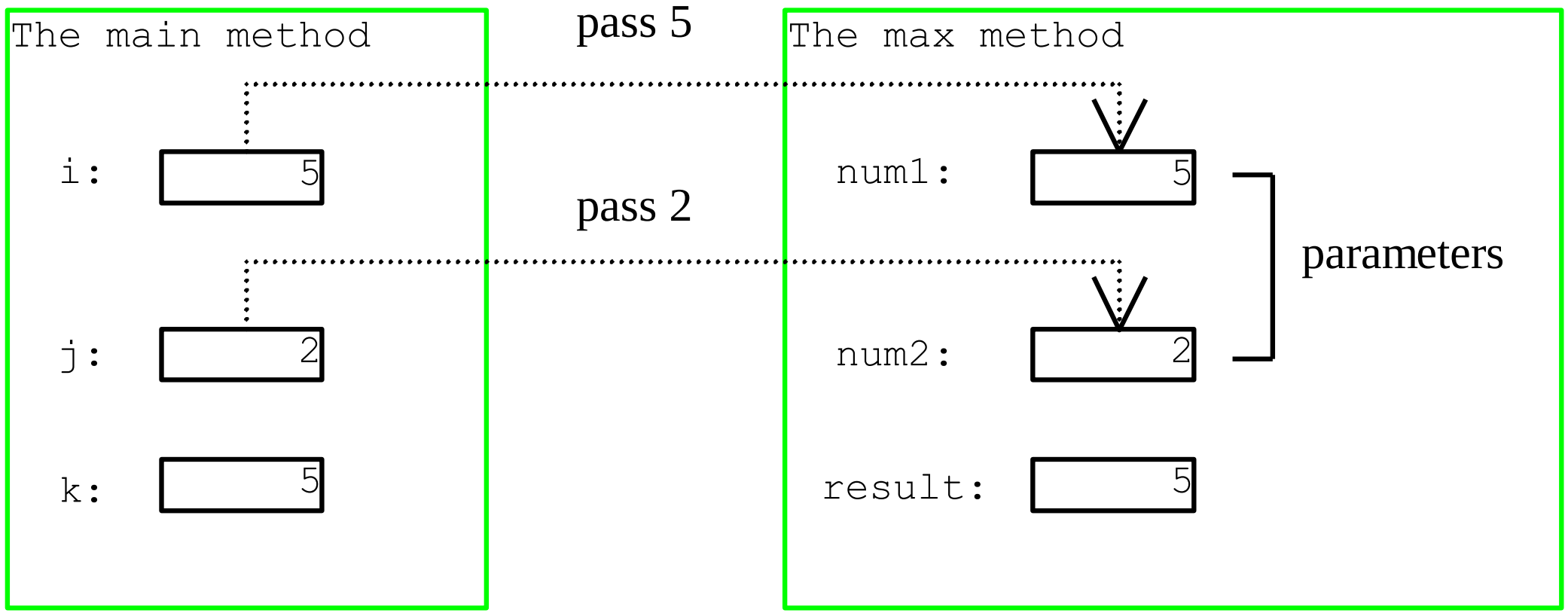
Chương trình minh họa việc gọi phương thức max để trả về giá trị lớn nhất.

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

Cách gọi phương thức (tiếp)



Cách gọi phương thức (tiếp)



Lưu ý

- Câu lệnh trả về giá trị bắt buộc phải có đối với một phương thức non-void.
- Phương thức sau đúng về logic, nhưng có lỗi biên dịch vì trình biên dịch Java nghĩ rằng phương thức này không trả về bất kỳ giá trị nào.

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

?: Hãy sửa lỗi chương trình trên để chạy đúng

Truyền tham số (pass by value)

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

Trong ví dụ này, giá trị n không bị thay đổi sau khi gọi phương thức Println()

```
253 public class Operation1 {
254     int data = 50;
255
256     void change(int data) {
257         data = data + 100;
258     }
259
260     public static void main(String args[]) {
261         Operation1 op = new Operation1();
262
263         System.out.println("Trước khi thay đổi: " + op.data);
264         op.change(500);
265         System.out.println("Sau khi thay đổi: " + op.data);
266     }
267 }
```

Truyền tham chiếu (pass by reference)

Ví dụ 2: Truyền tham chiếu

Trong ví dụ này, giá trị của biến data của đối tượng op bị thay đổi sau khi gọi phương thức change()

```
269 public class Operation2 {
270     int data = 50;
271 
272     void change(Operation2 op) {
273         op.data = op.data + 100;
274     }
275 
276     public static void main(String args[]) {
277         Operation2 op = new Operation2();
278 
279         System.out.println("Trước khi thay đổi: " + op.data);
280         op.change(op); // truyền đối tượng
281         System.out.println("Sau khi thay đổi: " + op.data);
282     }
283 }
```

Overloading Methods

Ví dụ 3: Overloading method `max`

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

Gọi mập mờ

Đôi khi có thể có nhiều hơn một đáp ứng khi gọi một phương thức, nhưng trình biên dịch không thể xác định được đáp ứng thích hợp nhất. Điều này được gọi là "Gọi mập mờ" (*ambiguous invocation)* - đây là một lỗi biên dịch.

Gọi mập mờ

```
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;
  }
}
```

Phạm vi của các biến cục bộ

- Biến cục bộ (local variable): biến được khai báo 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 khi khai báo đến cuối block chứa biến đó. Một biến cục bộ phải được khai báo trước khi sử dụng.

Phạm vi của các biến cục bộ (tiếp)

- Có thể khai báo một biến cục bộ trùng tên nhiều lần trong các khối riêng rẽ không lồng nhau trong một phương thức, nhưng bạn không thể khai báo một biến cục bộ 2 lần trong các khối lồng nhau.

Phạm vi của các biến cục bộ (tiếp)

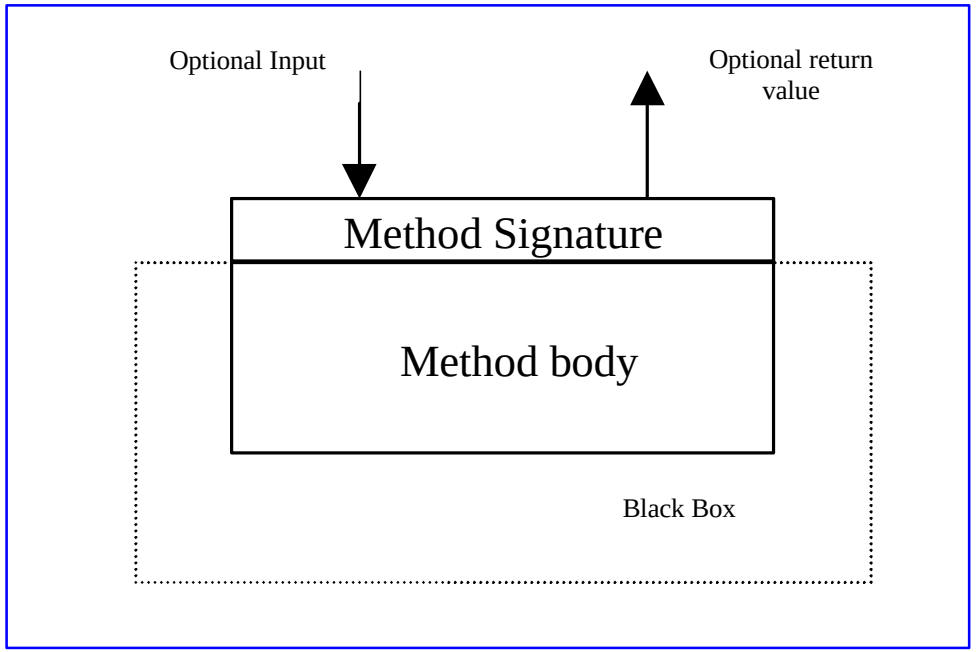
```
// Fine with no errors
public static void correctMethod() {
  int x = 1;
  int y = 1;
  // i is declared
  for (int i = 1; i < 10; i++) {
    x += i;
  }
  // i is declared again
  for (int i = 1; i < 10; i++) {
    y += i;
  }
}
```

Phạm vi của các biến cục bộ (tiếp)

```
// With error
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 - Method Abstraction

Thân phương thức như một hộp đen chứa sự thực hiện chi tiết của phương thức.



Lợi ích của phương thức

- Viết 1 lần, dùng nhiều lần.
- Giấu thông tin. Giấu sự thực hiện đối với user.
- Giảm độ phức tạp.

The Math Class

- Các hằng lớp:
  - PI
  - E
- Các phương thức lớp:
  - Các phương thức lượng giác
  - Các phương thức số mũ
  - Các phương thức làm tròn
  - Các phương thức min, max, abs, và random

Các phương thức lượng giác

- `sin(double rad)`
- `cos(double rad)`
- `tan(double rad)`
- `acos(double rad)`
- `asin(double rad)`
- `atan(double rad)`
- `toRadians(double deg)`
- `toDegrees(double rad)`

Ví dụ

Phương thức	Giá trị trả về
`Math.sin(0)`	`0.0`
`Math.sin(Math.PI/6)`	`0.5`
`Math.cos(0)`	`1.0`
`Math.cos(Math.PI/6)`	`0.866`

Các phương thức số mũ

Phương thức	Giá trị trả về
`exp(double a)`	e^a
`log(double a)`	`ln(a)`
`pow(double a, double b)`	a^b
`sqrt(double a)`	căn bậc hai của `a`

Các phương thức làm tròn

- `double ceil(double x)`

  x được làm tròn lên giá trị nguyên gần nhất. Giá trị nguyên này được trả về như một giá trị thực.

- `double floor(double x)`

  `x` được làm tròn xuống giá trị nguyên gần nhất. Giá trị nguyên này được trả về như một giá trị thực .

- `double rint(double x)`

  x được làm tròn đến giá trị nguyên gần nhất. Nếu phần lẻ của x bằng 0.5 thì giá trị đó là số chẵn.

- `int round(float x)`

  Trả về `(int)Math.floor(x+0.5)`

- `long round(double x)`

  Trả về `(long)Math.floor(x+0.5)`

Ví dụ

- `Math.ceil(2.1)` → `3.0`
- `Math.ceil(-2.1)` → `-2.0`
- `Math.floor(2.1)` → `2.0`
- `Math.floor(-2.1)` → `-3.0`
- `Math.rint(2.1)` → `2.0`
- `Math.rint(-2.1)` → `-2.0`
- `Math.rint(2.5)` → `2.0`
- `Math.round(2.6f)` → `3` (g/t int)
- `Math.round(-2.6)` → `-3` (g/t long)
- `Math.round(2.0)` → `2` (g/t long)

min, max, abs, và random

- `max(a, b)` ; `min(a, b)`

  Trả về giá trị lớn nhất / nhỏ nhất của 2 tham số a, b.

- `abs(a)`

  Trả về giá trị tuyệt đối của a.

- `random()`

  Trả về một giá trị `double` ngẫu nhiên trong khoảng [0.0, 1.0).

Ví dụ

- `Math.max(2, 3)` → 3
- `Math.max(2.5, 3)` → 3.0
- `Math.abs(-2.4)` → 2.4
- `10+(int)(Math.random()*20)`
  Số nguyên thuộc [10, 29]
- `10 + (Math.random()*20)`
  Số thực thuộc [10.0, 30.0)

Ví dụ: Tính trung bình và độ lệch chuẩn

Tạo ngẫu nhiên 10 số rồi tính giá trị trung bình và độ lệch chuẩn theo công thức:

$$mean = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$deviation = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}{n-1}}$$

Ví dụ: Sinh ký tự ngẫu nhiên

Viết các phương thức sinh các ký tự ngẫu nhiên. Chương trình sử dụng các phương thức này để sinh ngẫu nhiên 175 ký tự nằm giữa '!' và '~' rồi hiển thị 25 ký tự trên 1 dòng.
Có thể xem Appendix B, “The ASCII Character Set.”

Appendix B: ASCII Character Set

TABLE B.1 ASCII Character Set in the Decimal Index

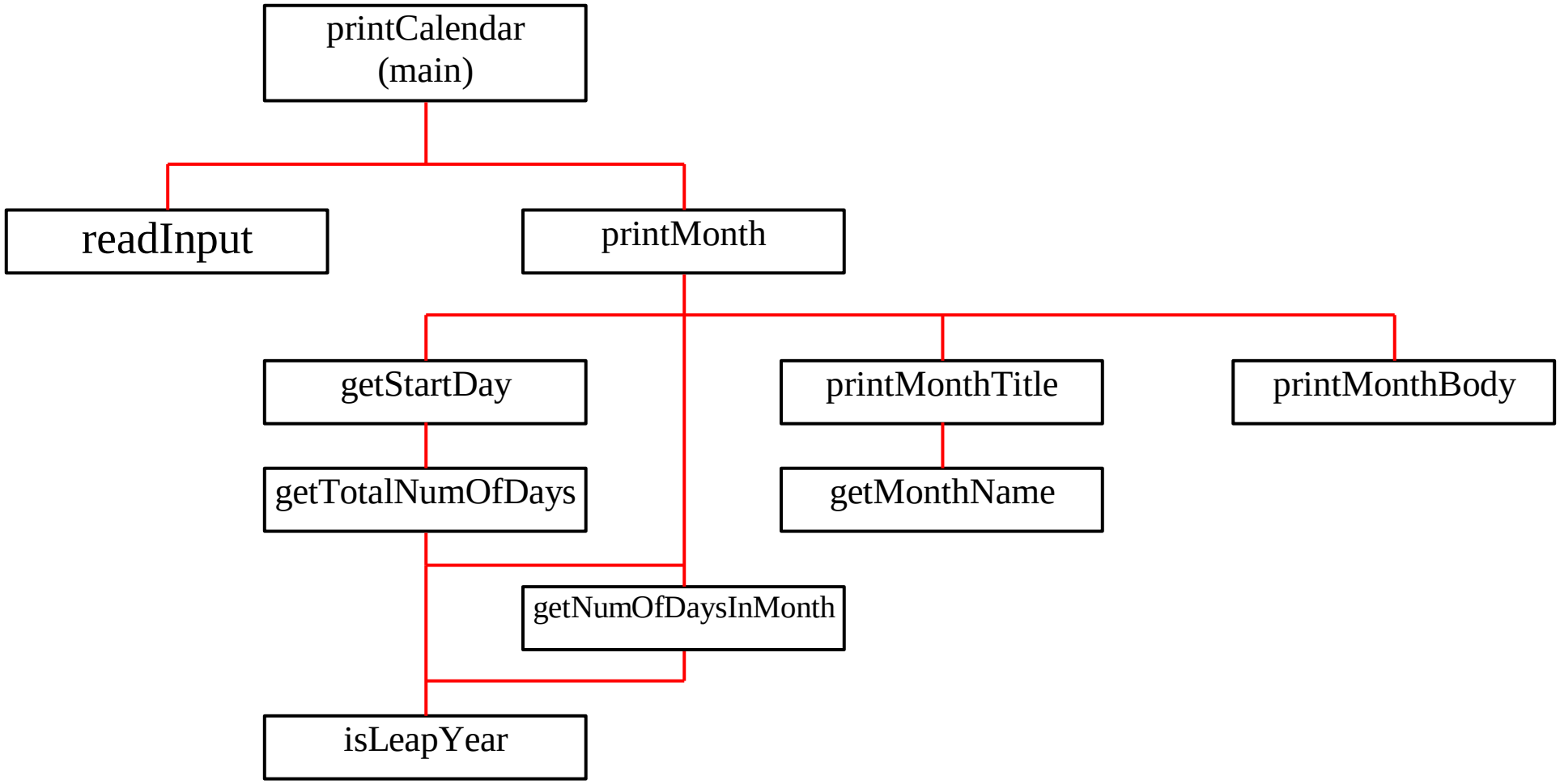
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	″	#	$	%	&	’
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

Case Study

Ví dụ: Hiển thị lịch

Chương trình đọc vào tháng và năm rồi hiển thị lịch của tháng và năm đó.

Design Diagram



Đệ quy

➢ Đệ quy trong java là quá trình trong đó một phương thức gọi lại chính nó một cách liên tiếp. Một phương thức trong java gọi lại chính nó được gọi là phương thức đệ quy.

➢ Cú pháp:

```
returntype methodname() {
    // code
    methodname();
}
```

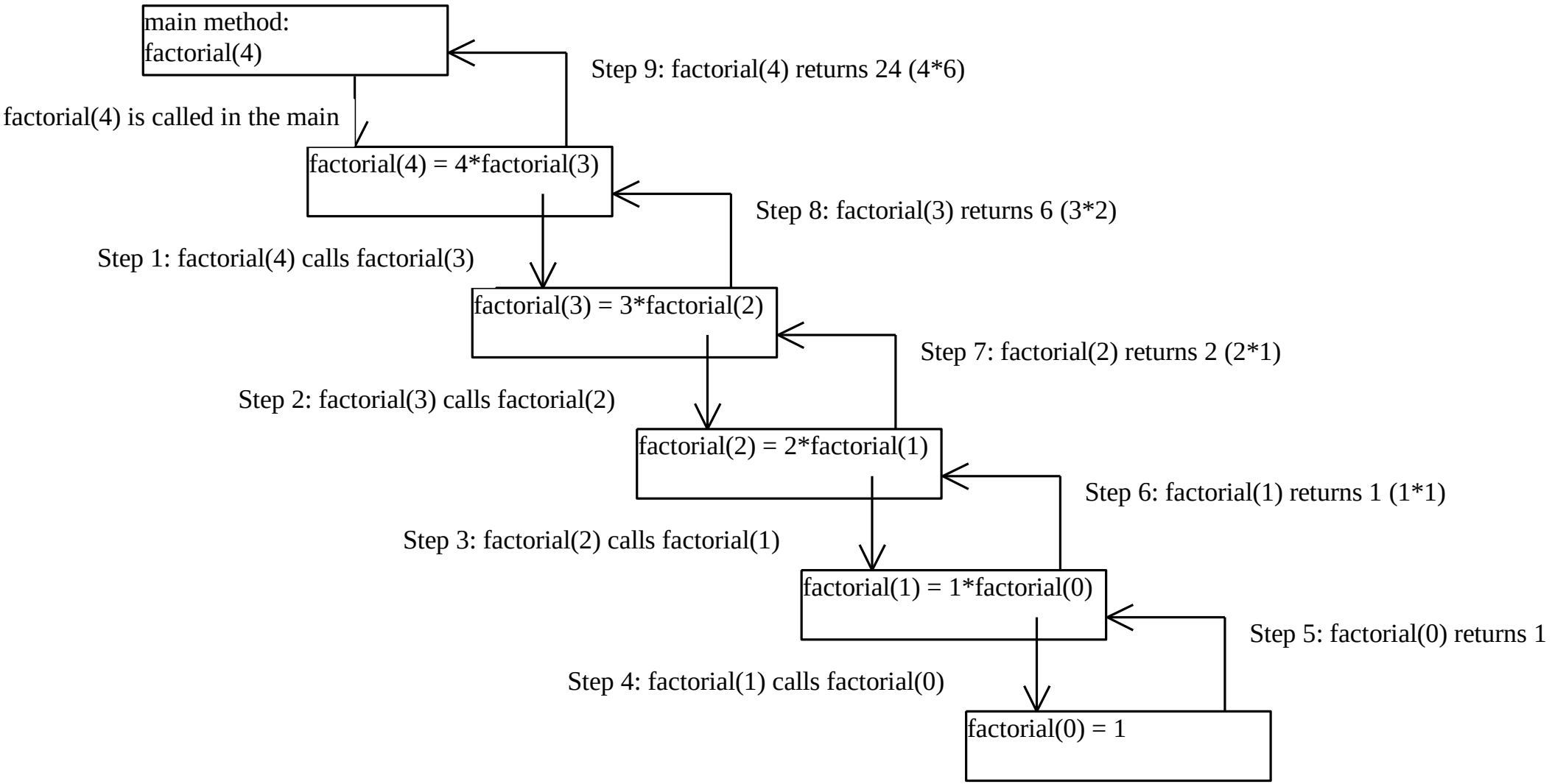
Đệ quy

Ví dụ: Tính giai thừa

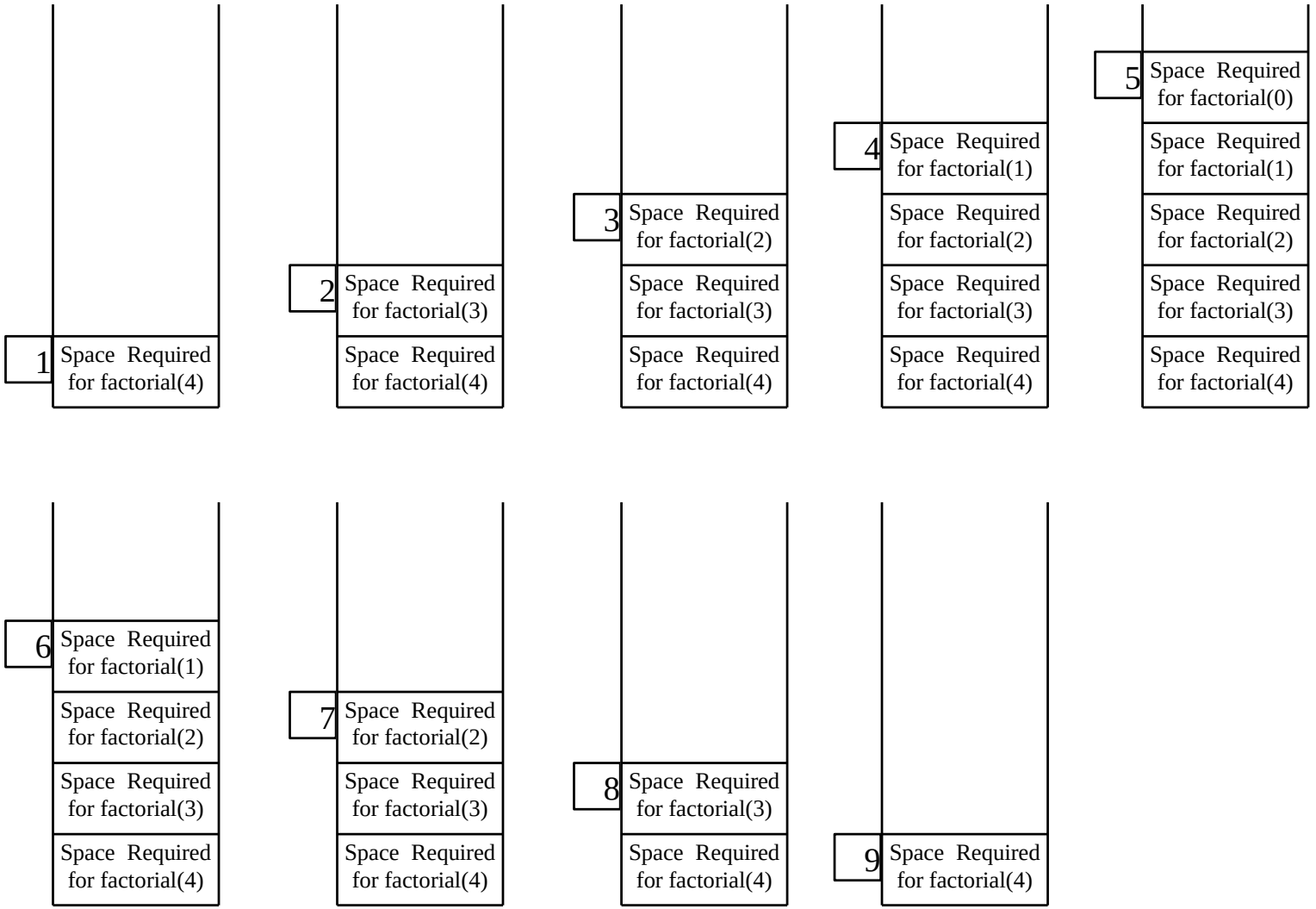
factorial(0) = 1;

factorial(n) = n*factorial(n-1);

Tính giai thừa (tiếp)



Tính giai thừa (tiếp)



Tính giai thừa (tiếp)

```
public class RecursionExample3 {
    static int factorial(int n) {
        if (n == 1)
            return 1;
        else
            return (n * factorial(n - 1));
    }

    public static void main(String[] args) {
        System.out.println("Giai thừa của 5 là: " + factorial(5));
    }
}
```

Dãy số Fibonacci

Ví dụ: Tìm dãy số Finonacci

0 1 1 2 3 5 8 13 21 34 55 89…

f0 f1

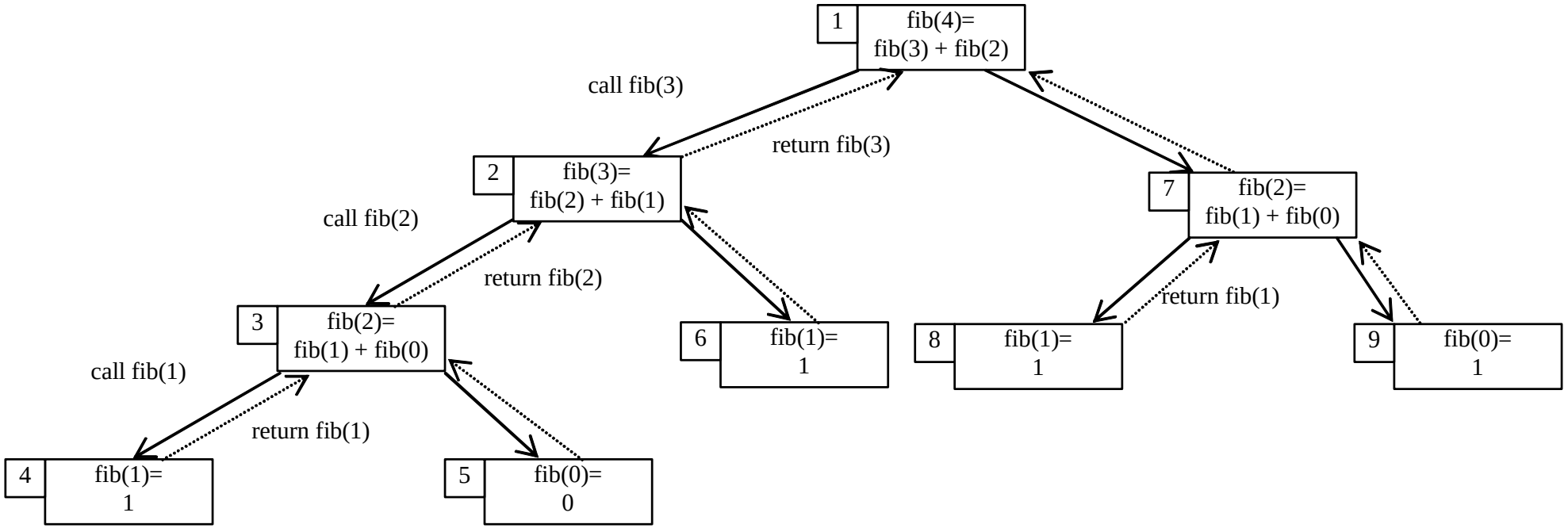
fib(2) = fib(0) + fib(1);

$$fib(0) = 0;$$

$$fib(1) = 1;$$

$$fib(n) = fib(n-2) + fib(n-1); n>=2$$

Dãy số Fibonacci (tiếp)



Dãy số Fibonacci (tiếp)

```
public class RecursionExample4 {
    static int n1 = 0, n2 = 1, n3 = 0;

    static void printFibo(int count) {
        if (count > 0) {
            n3 = n1 + n2;
            n1 = n2;
            n2 = n3;
            System.out.print(" " + n3);
            printFibo(count - 1);
        }
    }

    public static void main(String[] args) {
        int count = 15;
        System.out.print(n1 + " " + n2); // in 0 và 1
        printFibo(count - 2);// n-2 vì 2 số 0 và 1 đã được in ra
    }
}
```

Tháp Hanoi

Ví dụ: Giải bài toán Tháp Hanoi

Tháp Hanoi (tiếp)

Q&A


ĐẠI HỌC
BÁCH KHOA
BKACAD®
HỌC VIỆN CÔNG NGHỆ BKACAD
THANK YOU