

# Bài giảng Lập trình căn bản với Java

Dựa trên sách và bài giảng của Robert Sedgewick và Kevin Wayne

Biên soạn: Trần Quốc Long



# Mục lục

<b>I</b>	<b>Java căn bản</b>	<b>1</b>
1	Chương trình đầu tiên: Hello World	3
2	Các kiểu dữ liệu cơ bản	13
3	Rẽ nhánh và vòng lặp	27
4	Mảng	45
5	Nhập xuất	47
6	Nghiên cứu tình huống: Thuật toán PageRank	49
<b>II</b>	<b>Hàm</b>	<b>51</b>
<b>III</b>	<b>Lập trình hướng đối tượng</b>	<b>53</b>
<b>IV</b>	<b>Cấu trúc dữ liệu và giải thuật</b>	<b>55</b>



# **Phần I**

## **Java căn bản**



# Chương 1

## Chương trình đầu tiên: Hello World

### Mục lục chương

1.1 Cài đặt hệ biên dịch Java JDK . . . . .	3
1.2 Viết chương trình đầu tiên . . . . .	6
1.3 Dịch chương trình . . . . .	7
1.4 Chạy chương trình . . . . .	7
1.5 Các loại lỗi . . . . .	8
1.6 Đầu vào, đầu ra . . . . .	8
1.7 Hỏi đáp . . . . .	8
1.8 Bài tập . . . . .	10

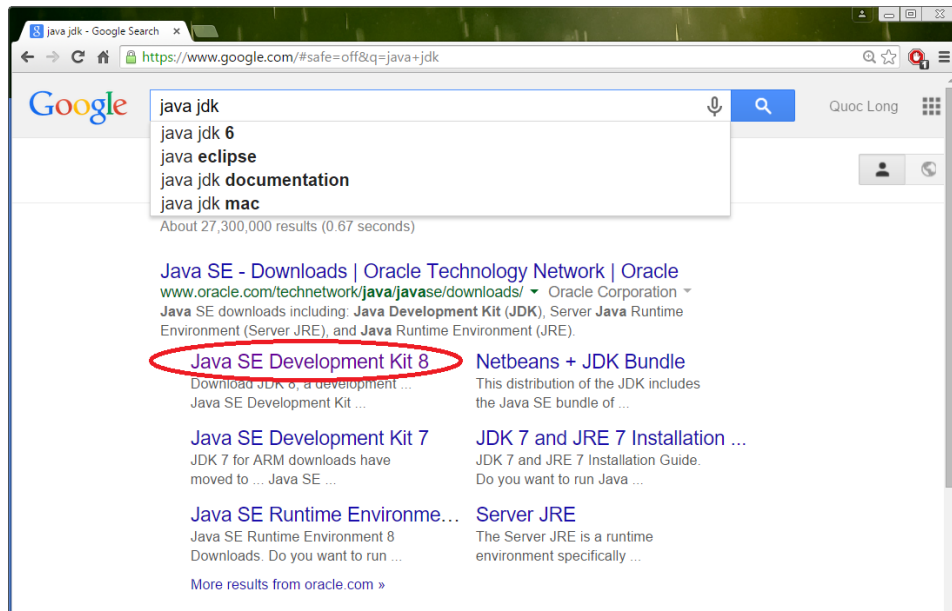
Trong chương này, chúng tôi sẽ dẫn dắt bạn vào thế giới lập trình Java bằng cách đưa bạn qua ba bước cơ bản cần thiết để tạo ra một chương trình đơn giản chạy được. Hệ thống Java bao gồm các ứng dụng không giống như bất kỳ các ứng dụng khác bạn đang sử dụng (như soạn thảo văn bản, e-mail, hoặc trình duyệt internet). Để tạo được một ứng dụng Java, bạn cần cài đặt Java trên máy tính của mình. Bạn cũng cần một trình soạn thảo và một trình terminal để gõ dòng lệnh.

### 1.1 Cài đặt hệ biên dịch Java JDK

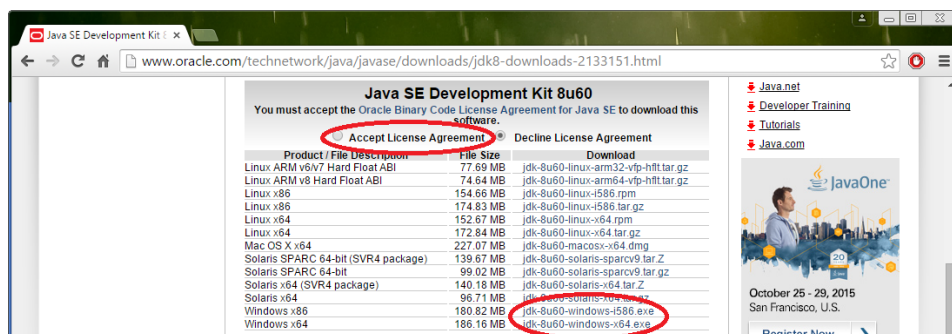
Để biên dịch và chạy một chương trình Java, bạn cần các chương trình sau trong hệ điều hành của bạn

1. Hệ biên dịch Java JDK,
2. Trình terminal để gõ lệnh,
3. Hệ soạn thảo văn bản.

**Hệ biên dịch Java JDK.** Để cài đặt hệ biên dịch Java JDK, bật trình duyệt internet và tìm kiếm với từ khóa "java jdk". Sau đó nhấn vào phiên bản mới nhất của Java JDK. Tại thời điểm này là **Java SE Development Kit 8**.

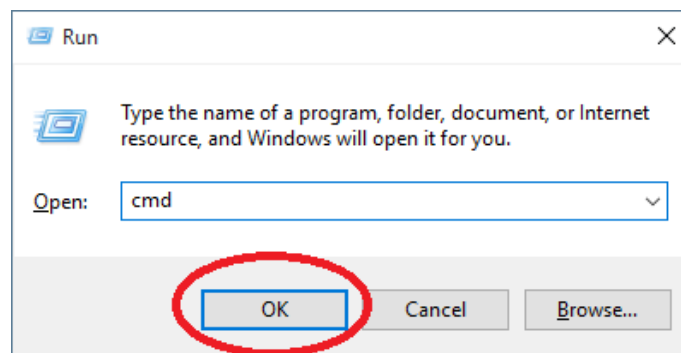


Do Java được Oracle mua lại nên ta sẽ đến trang [oracle.com](http://oracle.com). Ấn "Accept License Agreement" để đồng ý với các điều kiện sử dụng Java của Oracle rồi ấn vào một trong các link để tải Java JDK về máy. Ở đây chúng tôi chọn bản `jdk-8u60-windows-i586.exe` (32-bit) dành cho hệ điều hành Windows.



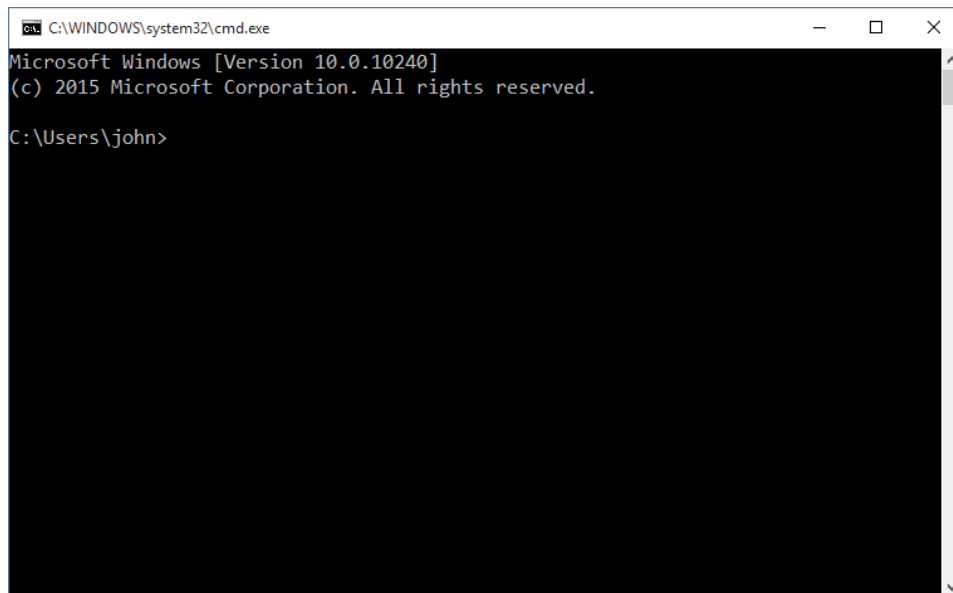
Sau khi Java JDK được tải về máy, ấn vào tập tin vừa tải để chạy chương trình cài đặt Java JDK. Ấn Next để chọn cấu hình mặc định để Java JDK được cài vào thư mục `C:\Program Files (x86)\Java\jdk1.8.0`. Như vậy là bạn đã cài được trình biên dịch JDK.

**Trình Terminal.** Windows đã có sẵn một trình terminal để các bạn gõ các lệnh điều khiển máy tính. Để chạy chương trình này, ấn phím `[Win] + [R]` và gõ vào phần Open đoạn lệnh `cmd` rồi ấn `[Enter]`.



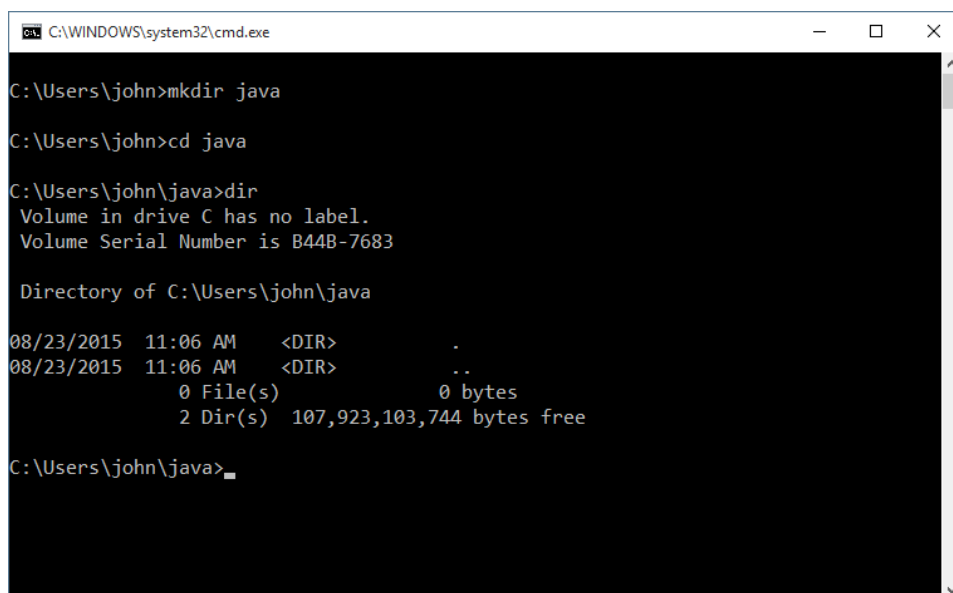
Bạn sẽ thấy một cửa sổ như sau





Đây là trình terminal để các bạn gõ các lệnh điều khiển máy tính. Bạn sẽ thấy **dấu nhắc lệnh** nháy ở bên cạnh thư mục hiện thời của ổ cứng bạn đang sử dụng. Ở đây là `C:\Users\john`. Từ vị trí này bạn có thể thực hiện các lệnh để điều khiển máy tính. Hãy thực hiện các lệnh sau

- `mkdir java`: Tạo thư mục mới có tên `java`.
- `cd java`: Di chuyển đến thư mục vừa mới tạo.
- `dir`: Liệt kê danh sách file trong thư mục hiện thời.



```
C:\Users\john>mkdir java
C:\Users\john>cd java
C:\Users\john\java>dir
Volume in drive C has no label.
Volume Serial Number is B44B-7683

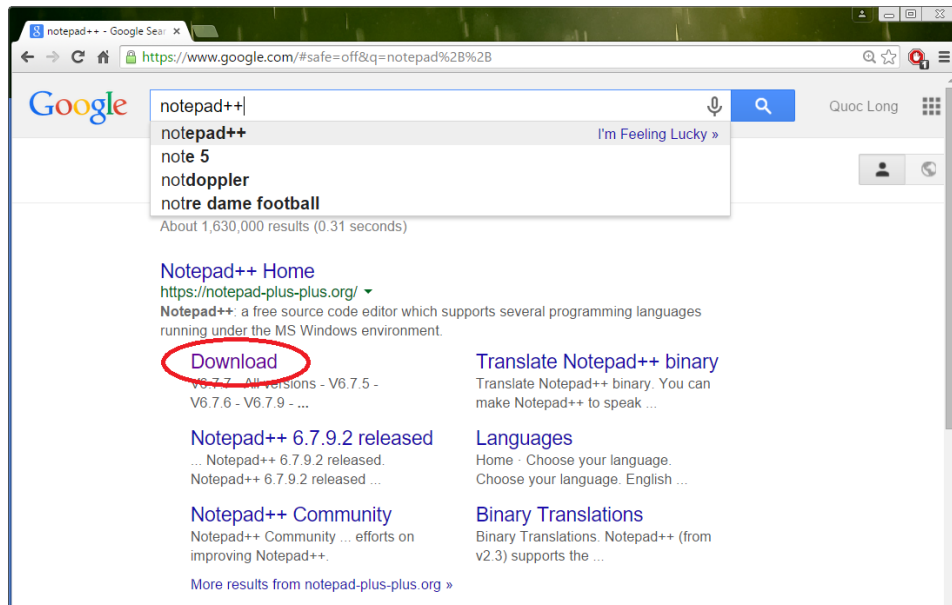
Directory of C:\Users\john\java

08/23/2015  11:06 AM  <DIR>          .
08/23/2015  11:06 AM  <DIR>          ..
               0 File(s)                0 bytes
               2 Dir(s)  107,923,103,744 bytes free

C:\Users\john\java>
```

Như vậy, bạn đã tạo được thư mục `java` trong thư mục người dùng và di chuyển vào thư mục này. Tất nhiên, Windows còn rất nhiều câu lệnh khác và trong quá trình học tập trở thành lập trình viên, bạn sẽ học và tự học vô số lệnh thú vị hơn nữa.

**Hệ soạn thảo văn bản.** Bạn sẽ cần một hệ soạn thảo văn bản hỗ trợ lập trình Java. Ở đây chúng tôi chọn `Notepad++`. Đây là hệ soạn thảo miễn phí có thể tải về từ địa chỉ như hình sau



Sau khi tải Notepad++ về và cài đặt, bạn có thể chạy thử. Dưới đây là một chương trình Java đơn giản hiển thị trong Notepad++.

```
C:\Users\john\Documents\INT1006 - THCS4\lecture\code\ch1\HelloWorld.java - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
HelloWorld.java
1  /* *****
2  * Compilation: javac HelloWorld.java
3  * Execution: java HelloWorld
4  *
5  * Prints "Hello, World". By tradition, this is everyone's first program.
6  *
7  * % java HelloWorld
8  * Hello, World
9  *
10 * These 17 lines of text are comments. They are not part of the program;
11 * they serve to remind us about its properties. The first two lines tell
12 * us what to type to compile and test the program. The next line describes
13 * the purpose of the program. The next few lines give a sample execution
14 * of the program and the resulting output. We will always include such
15 * lines in our programs and encourage you to do the same.
16 *
17 * *****/
18
19 public class HelloWorld {
20
21     public static void main(String[] args) {
22         System.out.println("Hello, World");
23     }
24 }
25
26
Ja length: 923 lines: 26 Ln: 25 Col: 2 Sel: 0|0 UNIX UTF-8 w/o BOM INS
```

## 1.2 Viết chương trình đầu tiên

Một chương trình đơn giản chỉ là một chuỗi ký tự, giống như một câu, một đoạn văn hoặc một bài thơ. Để tạo ra một chương trình, chúng ta chỉ cần sử dụng một trình soạn thảo văn bản để viết giống như viết e-mail. Chương trình HelloWorld.java dưới đây là một ví dụ. Hãy nhập các ký tự sau vào trình soạn thảo văn bản của bạn và lưu nó lại thành một tập tin có tên là HelloWorld.java.

```

1 public class HelloWorld {
2
3     public static void main(String[] args) {
4         System.out.println("Hello, World");
5     }
6
7 }

```

## 1.3 Dịch chương trình

Khi mới lập trình, đối với bạn, ngôn ngữ lập trình Java dường như được thiết kế cho máy tính hiểu. Trên thực tế, ngược lại, ngôn ngữ này được thiết kế để chính các lập trình viên (tức là chính bạn) hiểu họ định hướng dẫn máy tính làm gì. Khi đã có một chương trình bằng ngôn ngữ Java, một trình biên dịch (interpreter, compiler) sẽ dịch chương trình của bạn từ ngôn ngữ Java sang một ngôn ngữ phù hợp hơn để thực hiện (chạy) trên máy tính. Như vậy, từ tập tin văn bản với phần mở rộng `.java` (chương trình của bạn), trình biên dịch sẽ tạo ra một tập tin có phần mở rộng `.class` (gần với ngôn ngữ máy).

Để biên dịch chương trình `HelloWorld.java` hãy gõ đoạn lệnh sau vào trình terminal. Chúng tôi sử dụng các biểu tượng % để biểu thị dấu nhắc lệnh (hệ thống của bạn có thể sử dụng dấu nhắc lệnh khác).

```
% javac HelloWorld.java
```

Nếu bạn gõ văn bản chương trình `HelloWorld.java` chính xác, bạn sẽ thấy không có thông báo lỗi. Nếu có lỗi, hãy mở lại trình soạn thảo để sửa hết lỗi cho chương trình giống với đoạn mã 1.1.

## 1.4 Chạy chương trình

Sau khi bạn biên dịch chương trình, bạn có thể chạy nó. Đây là phần thú vị nhất vì máy tính phải tuân thủ ý chí của bạn. Để chạy chương trình `HelloWorld`, gõ lệnh sau tại trình terminal

```
% java HelloWorld
```

Nếu mọi việc suôn sẻ, bạn sẽ thấy các phản hồi sau trên màn hình

```
Hello, World
```

Nhìn vào chương trình `HelloWorld.java`, dòng lệnh chính là dòng với câu lệnh

```
System.out.println ()
```

hiển thị đoạn văn bản "Hello, World". Khi chúng ta viết các chương trình phức tạp hơn, chúng tôi sẽ thảo luận về ý nghĩa của các từ `public`, `class`, `main`, `String [] args`, `System.out`.

Trong phần đầu của cuốn sách, các chương trình của chúng ta sẽ giống như `HelloWorld.java`, ngoại trừ đoạn lệnh trong hàm `main ()`. Cách dễ nhất để viết một chương trình đơn giản như vậy là:

- Sao chép `HelloWorld.java` thành một file mới có tên là `<Tên chương trình>.java`.
- Trong file mới, thay `HelloWorld` thành `<Tên chương trình>` ở mọi nơi.
- Thay thế câu lệnh `print` bởi chuỗi các câu lệnh khác.

```

1 public class Hi {
2
3     public static void main(String[] args) {
4         System.out.print("Hi, ");
5         System.out.print(args[0]);
6         System.out.println(". How are you?");
7     }
8
9 }

```

## 1.5 Các loại lỗi

Hầu hết các lỗi chương trình đều có thể phát hiện bằng cách cẩn thận kiểm tra chương trình khi chúng ta tạo ra nó, giống như khi chúng ta sửa chữa lỗi chính tả và lỗi ngữ pháp khi chúng ta gõ văn bản hoặc e-mail. Có những loại lỗi sau đây

- Lỗi biên dịch (Compile-time error). Những lỗi này được trình biên dịch thông báo khi dịch chương trình. Khi gặp những lỗi này, trình biên dịch không thể tiếp tục làm việc, do đó nó sinh thông báo giải thích lý tại sao xảy ra lỗi.
- Lỗi khi chạy (Run-time error). Những lỗi này được hệ điều hành máy tính thông báo khi chúng ta chạy chương trình và chương trình thực hiện một hoạt động không hợp lệ (ví dụ, chia cho 0).
- Lỗi logic (Logical error). Các lỗi này xảy ra khi chúng ta chạy chương trình và nó cho kết quả không như mong đợi. Các lỗi này chỉ có thể phát hiện bởi chính các lập trình viên (**đây là lý do chúng ta có việc làm**). Thường các lỗi này rất khó tìm.

Một trong những kỹ năng đầu tiên mà bạn sẽ được học chính là xác định lỗi; một trong số đó chính là phải cẩn thận khi viết chương trình để tránh nhiều loại lỗi.

## 1.6 Đầu vào, đầu ra

Nhiều khi chúng ta muốn cung cấp **đầu vào** cho các chương trình. Đầu vào là dữ liệu mà chương trình xử lý để sản xuất một kết quả tức là **đầu ra**. Cách đơn giản nhất để cung cấp dữ liệu đầu vào được minh họa trong Hi.java dưới đây. Khi chương trình này chạy, nó đọc các đối số trên dòng lệnh mà bạn gõ vào sau tên chương trình và in thông báo ra màn hình. Biên dịch và chạy chương trình

```

% javac Hi.java
% java Hi Alice
Hi, Alice. How are you?
% Java Hi Bob
Hi, Bob. How are you?

```

## 1.7 Hỏi đáp

**Hỏi.** Tại sao là Java?

**Đáp.** Các chương trình chúng ta đang viết rất giống các chương trình trong một số ngôn ngữ khác, vì vậy sự lựa chọn ngôn ngữ thật sự không quan trọng. Chúng ta sử dụng Java vì nó khá phổ biến. Nó bao trùm một bộ đầy đủ các khái niệm trừu tượng trong lập trình hiện đại, và có một loạt các bộ kiểm tra tự động kiểm soát các lỗi có thể xảy ra. Do đó, Java thích hợp cho nhập môn học tập lập trình. Nên nhớ, không có ngôn ngữ nào hoàn hảo, và bạn chắc chắn sẽ lập trình bằng các ngôn ngữ khác trong tương lai.

**Hỏi.** Tôi có cần gõ lại các chương trình và thử chạy chúng?

**Đáp.** Lúc mới học, mọi người nên tự gõ lại các chương trình, biên dịch chúng, chạy và sửa lỗi nếu có. Tuy nhiên, bạn có thể tìm thấy tất cả các đoạn mã trong cuốn sách này trên website

<http://introcs.cs.princeton.edu/java>

**Hỏi.** Quy tắc của Java về tab, khoảng trống và các ký tự xuống dòng là gì?

**Đáp.** Không có nhiều. Trình biên dịch Java coi chúng tương đương nhau. Ví dụ, chúng ta cũng có thể viết chương trình HelloWorld như sau:

```
public class HelloWorld {public static void main (  
    String [] args) {System.out.println ("Hello_World");  }}
```

Nhưng chúng ta sẽ tuân thủ quy ước về khoảng cách và thụt dòng khi chúng ta viết chương trình, giống như chúng ta luôn thụt dòng đầu đoạn khi chúng ta viết văn bản.

**Hỏi.** Các quy tắc về dấu ngoặc kép là gì?

**Đáp.** Các khoảng trống bên trong 2 dấu ngoặc kép là ngoại lệ cho quy tắc trong câu hỏi bên trên: các ký tự bên trong 2 dấu ngoặc kép sẽ được in ra chính xác như khi viết. Bao nhiêu khoảng trắng trong 2 dấu ngoặc kép sẽ được in ra bấy nhiêu. Do đó, nếu bạn quên mất một dấu ngoặc kép, trình biên dịch sẽ không biết đâu là đoạn ký tự cần in, đâu là phần còn lại của chương trình. Để in một dấu ngoặc kép, một dòng mới, hoặc dấu tab, sử dụng "\", \n, hoặc \t trong dấu ngoặc kép.

**Hỏi.** Ý nghĩa của các từ `static` và `void`?

**Đáp.** Các từ khóa này chỉ định một số đặc tính của hàm `main ()` mà bạn sẽ được học trong cuốn sách. Tại thời điểm này, chúng ta cần đưa các từ khóa này vào trong các đoạn mã vì chúng cần thiết.

**Hỏi.** Điều gì xảy ra khi bạn bỏ qua một dấu ngoặc hoặc viết sai một trong các từ khóa, như `static` và `public`?

**Đáp.** Điều đó phụ thuộc vào chính xác những gì bạn làm. Lỗi như vậy được gọi là lỗi cú pháp. Hãy thử và xem điều gì xảy ra.

**Hỏi.** Có thể một chương trình sử dụng nhiều hơn một đối số trên dòng lệnh?

**Đáp.** Có thể, bạn có thể dùng nhiều đối số, mặc dù chúng ta thường sử dụng chỉ một số ít. Bạn có thể dùng đối số thứ hai bằng `args[1]`, thứ ba bằng `args[2]`, và vân vân. Lưu ý rằng chúng ta bắt đầu đếm từ 0 trong Java.

**Hỏi.** Java có thư viện và hàm gì có sẵn cho tôi sử dụng?

**Đáp.** Có hàng ngàn thư viện cho bạn sử dụng nhưng chúng tôi sẽ giới thiệu chúng cẩn thận để bạn không bị choáng ngợp bởi quá nhiều lựa chọn.

**Hỏi.** Tôi nên viết mã Java như thế nào? Tôi nên viết chú thích trong Java như thế nào?

**Đáp.** Lập trình viên sử dụng các chỉ dẫn lập trình để chương trình dễ đọc, dễ hiểu, và dễ bảo trì. Khi bạn có kinh nghiệm, bạn sẽ phát triển một phong cách lập trình riêng của mình, giống như mỗi nhà văn đều có phong cách riêng. Phụ lục B cung cấp một số chỉ dẫn lập trình và cách chú thích. Chúng tôi khuyên bạn nên đọc phụ lục này sau khi bạn đã viết một vài chương trình.

**Hỏi.** Tập tin có đuôi `.class` chính xác là gì?

**Đáp.** Đó là một tập tin nhị phân (chuỗi các số 0 và số 1). Nếu bạn đang sử dụng Unix hay OS X, bạn có thể kiểm tra nội dung của nó bằng cách gõ

```
% od HelloWorld.class -x
```

tại dấu nhắc lệnh. Lệnh này sẽ hiển thị nội dung tập tin hệ thập lục phân (cơ sở 16). Bạn có thể thấy từ đầu tiên của mỗi tập tin `.class` là `cafe`.

```
00000000  cafe  babe  0000  002e  001d  0a00  0600  0f09
00000020  0010  0011  0800  120a  0013  0014  0700  1507
00000040  0016  0100  063c  696e  6974  3e01  0003  2829
00000060  5601  0004  436f  6465  0100  0f4c  696e  654e
00000100  756d  6265  7254  6162  6c65  0100  046d  6169
00000120  6e01  0016  285b  4c6a  6176  612f  6c61  6e67
00000140  2f53  7472  696e  673b  2956  0100  0a53  6f75
00000160  7263  6546  696c  6501  000f  4865  6c6c  6f57
00000200  6f72  6c64  2e6a  6176  610c  0007  0008  0700
00000220  170c  0018  0019  0100  0c48  656c  6c6f  2c20
00000240  576f  726c  6407  001a  0c00  1b00  1c01  000a
00000260  4865  6c6c  6f57  6f72  6c64  0100  106a  6176
00000300  612f  6c61  6e67  2f4f  626a  6563  7401  0010
00000320  6a61  7661  2f6c  616e  672f  5379  7374  656d
00000340  0100  036f  7574  0100  154c  6a61  7661  2f69
00000360  6f2f  5072  696e  7453  7472  6561  6d3b  0100
00000400  136a  6176  612f  696f  2f50  7269  6e74  5374
00000420  7265  616d  0100  0770  7269  6e74  6c6e  0100
00000440  1528  4c6a  6176  612f  6c61  6e67  2f53  7472
00000460  696e  673b  2956  0021  0005  0006  0000  0000
00000500  0002  0001  0007  0008  0001  0009  0000  001d
00000520  0001  0001  0000  0005  2ab7  0001  b100  0000
00000540  0100  0a00  0000  0600  0100  0000  0c00  0900
00000560  0b00  0c00  0100  0900  0000  2500  0200  0100
00000600  0000  09b2  0002  1203  b600  04b1  0000  0001
00000620  000a  0000  000a  0002  0000  000f  0008  0010
00000640  0001  000d  0000  0002  000e
0000652
```

## 1.8 Bài tập

1. Viết chương trình `TenHelloWorlds.java` in ra "Hello, World" mười lần.

2. Điều gì xảy ra nếu bạn bỏ các từ sau trong `HelloWorld.java`

- `public`
- `static`
- `void`
- `args`

3. Điều gì xảy ra nếu bạn viết sai các từ sau trong `HelloWorld.java`

- `public`
- `static`
- `void`
- `args`

4. Điều gì xảy ra nếu bạn gõ các dòng lệnh sau (chương trình `Hi.java`)

- `java Hi`
- `java Hi @!&^%`
- `java Hi 1234`
- `java Hi.class Bob`
- `java Hi.java Bob`
- `java Hi Alice Bob`

5. Sửa `Hi.java` thành chương trình `HiThree.java` sử dụng 3 đối số dòng lệnh là 3 cái tên và in ra một câu chào với các tên đó theo thứ tự đảo ngược. Ví dụ:

```
% java HiThree Alice Bob Carol
Hi Carol, Bob, and Alice.
```

6. Viết chương trình in `Initials.java` in ra các chữ cái đầu của tên bạn bằng 9 dòng các kí tự \* giống như sau.

```
**      ***      *****      **      *      **
**      ***      **      **      **      ***      **
**      ***      **      **      **      **      **
**      ***      **      **      **      **      **
*****      **      **      **      **      **      **
**      ***      **      **      **      **      **
**      ***      **      **      **      **      **
**      ***      **      **      ***      ***
**      ***      **      **      *      *
```

7. Điều gì xảy ra khi biên dịch chương trình sau ? Hãy giải thích tại sao ?

```
public class Hello {
    public static void main() {
        System.out.println("Doesn't execute");
    }
}
```





# Chương 2

## Các kiểu dữ liệu cơ bản

### Mục lục chương

2.1	Định nghĩa cơ bản . . . . .	13
2.2	Kí tự và chuỗi kí tự . . . . .	14
2.3	Số nguyên . . . . .	15
2.4	Số thực . . . . .	15
2.5	Boolean . . . . .	18
2.6	Phép so sánh . . . . .	18
2.7	Chuyển đổi kiểu . . . . .	19
2.8	Hỏi đáp . . . . .	20
2.9	Bài tập . . . . .	21

Một kiểu dữ liệu là một tập hợp các giá trị và một tập hợp các toán tử xác định trên chúng. Ví dụ, chúng ta đã quen thuộc với những con số và với các toán tử xác định trên chúng như phép cộng và phép nhân. Tuy nhiên, ngược lại với toán học, nơi chúng ta quen với suy nghĩ tập hợp các số là vô hạn, trong các chương trình máy tính chúng ta phải làm việc với một tập hữu hạn các khả năng.

Có tám kiểu dữ liệu được dựng sẵn (*build-in*) trong Java. Chủ yếu là các loại số khác nhau. Các kiểu dữ liệu khác được định nghĩa trong *hệ thống thư viện*. Có thể nói việc lập trình trong Java thực sự là việc xây dựng các kiểu dữ liệu riêng của chúng ta. Chúng ta cũng sử dụng kiểu dữ liệu *chuỗi kí tự* thường xuyên nên chúng ta cũng sẽ xem xét nó ở dưới đây.

### 2.1 Định nghĩa cơ bản

Chúng ta sẽ sử dụng bốn câu lệnh Java đoạn sau đây để giới thiệu một số thuật ngữ chúng ta sẽ sử dụng:

```
int a, b, c;  
a = 1234;  
b = 99;  
c = a + b;
```

Lệnh đầu tiên khai báo ba biến với các định *a*, *b*, và *c* là kiểu *int*. Hai lệnh gán tiếp theo thay đổi các giá trị của các biến, sử dụng các hằng số 1234 và 99. Lệnh cuối cùng gán *c* giá trị 1333 bằng cách tính biểu thức *a + b*.

```

1 public class Ruler {
2     public static void main(String[] args) {
3         String ruler1 = " 1 ";
4         String ruler2 = ruler1 + "2" + ruler1;
5         String ruler3 = ruler2 + "3" + ruler2;
6         String ruler4 = ruler3 + "4" + ruler3;
7         String ruler5 = ruler4 + "5" + ruler4;
8
9         System.out.println(ruler1);
10        System.out.println(ruler2);
11        System.out.println(ruler3);
12        System.out.println(ruler4);
13        System.out.println(ruler5);
14    }
15 }
16

```

## 2.2 Kí tự và xâu kí tự

Một `char` là một kí tự hoặc biểu tượng giống như những kí tự mà bạn gõ trên bàn phím. Chúng ta thường chỉ gán giá trị cho các biến kí tự. Một `String` là một xâu các ký tự. Các toán tử nhất mà chúng ta thực hiện trên xâu là toán tử nối: cho hai xâu, nối chúng lại với nhau để tạo ra xâu mới. Ví dụ, hãy xem đoạn mã Java sau đây:

```

String a, b, c;
a = "Hello, ";
b = "Bob";
c = a + b;

```

Lệnh đầu tiên khai báo ba biến kiểu `String`. Ba phát biểu kế tiếp gán giá trị cho chúng, và cuối cùng, `c` có giá trị "Hello, Bob" là kết quả của phép nối 2 xâu `a` và `b`.

Sử dụng phép nối xâu, chương trình `Ruler.java` (đoạn mã 2.1) in ra độ cao tương đối của các vạch trên một cái thước (Thước có  $2^n - 1$  vạch).



*Độ cao các vạch thước với  $n = 4$*

```

% java Ruler
1
1 2 1
1 2 1 3 1 2 1
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

```

```

1 public class IntOps {
2
3     public static void main(String[] args) {
4         int a = Integer.parseInt(args[0]);
5         int b = Integer.parseInt(args[1]);
6         int sum = a + b;
7         int prod = a * b;
8         int quot = a / b;
9         int rem = a % b;
10
11         System.out.println(a + " + " + b + " = " + sum);
12         System.out.println(a + " * " + b + " = " + prod);
13         System.out.println(a + " / " + b + " = " + quot);
14         System.out.println(a + " % " + b + " = " + rem);
15         System.out.println(a + " = " + quot + " * " + b + " + " + rem);
16     }
17 }

```

## 2.3 Số nguyên

Một biến kiểu `int` là một số nguyên (số nguyên) giữa  $-2^{31}$  và  $2^{31}-1$  (-2,147,483,648 đến 2,147,483,647). Chúng ta sử dụng `int` thường xuyên không chỉ vì ta hay gặp chúng trong thế giới thực mà còn do chúng tự nhiên phát sinh khi thể hiện các thuật toán. Các toán tử toán học thường thấy cho số nguyên như cộng, trừ, nhân, chia được tích hợp sẵn trong Java, như minh họa trong `IntOps.java` (đoạn mã 2.2).

```

% java IntOps 1234 99
1234 + 99 = 1333
1234 * 99 = 122166
1234 / 99 = 12
1234 % 99 = 46
1234 = 12 * 99 + 46

% java IntOps 10 -3
10 + -3 = 7
10 * -3 = -30
10 / -3 = -3
10 % -3 = 1
10 = -3 * -3 + 1

```

Kiểu `long` tương tự như kiểu `int` nhưng nó có thể biểu diễn các số nguyên trong một khoảng lớn hơn nhiều, từ  $-2^{63}$  đến  $2^{63}-1$ . Chúng ta thỉnh thoảng sẽ sử dụng `long` khi chúng ta cần làm việc với các số nguyên lớn.

## 2.4 Số thực

Kiểu `double` biểu diễn các số thực *dấu chấm động*, sử dụng trong các ứng dụng tính toán khoa học. Biểu diễn bên trong của các biến kiểu này giống như ký hiệu khoa học, giúp chúng ta có thể tính

### Đoạn mã 2.3: DoubleOps.java

```

1 public class DoubleOps {
2
3     public static void main(String[] args) {
4         double a = Double.parseDouble(args[0]);
5         double b = Double.parseDouble(args[1]);
6         double sum = a + b;
7         double prod = a * b;
8         double quot = a / b;
9         double rem = a % b;
10
11        System.out.println(a + " + " + b + " = " + sum);
12        System.out.println(a + " * " + b + " = " + prod);
13        System.out.println(a + " / " + b + " = " + quot);
14        System.out.println(a + " % " + b + " = " + rem);
15
16        System.out.println();
17        System.out.println("sin(pi/2) = " + Math.sin(Math.PI/2));
18        System.out.println("log(e)      = " + Math.log(Math.E));
19    }
20 }

```

toán với số thực trong phạm vi lớn. Chúng ta có thể khai báo một số thực bằng một chuỗi các chữ số và dấu thập phân, ví dụ như, 3.14159 là xấp xỉ 6 sáu chữ số của hằng số toán học  $\pi$ . Hoặc ta có thể dùng một chuỗi ký hiệu khoa học, ví dụ như, 6.022E23 cho hằng số Avogadro  $6.022 \times 10^{23}$ .

Các toán tử số học như cộng, nhân, chia, cho `double` được xây dựng sẵn trong Java, như minh họa ở chương trình `DoubleOps.java` (đoạn mã 2.3).

```

% java DoubleOps 1234 99
1234.0 + 99.0 = 1333.0
1234.0 * 99.0 = 122166.0
1234.0 / 99.0 = 12.464646464646465
1234.0 % 99.0 = 46.0

sin(pi/2) = 1.0
log(e)     = 1.0

% java DoubleOps 10 -3
10.0 + -3.0 = 7.0
10.0 * -3.0 = -30.0
10.0 / -3.0 = -3.3333333333333335
10.0 % -3.0 = 1.0

sin(pi/2) = 1.0
log(e)     = 1.0

% java DoubleOps Infinity 3
Infinity + 3.0 = Infinity
Infinity * 3.0 = Infinity

```

#### Đoạn mã 2.4: Quadratic.java

```

1 public class Quadratic {
2
3     public static void main(String[] args) {
4         double b = Double.parseDouble(args[0]);
5         double c = Double.parseDouble(args[1]);
6
7         double discriminant = b*b - 4.0*c;
8         double sqroot = Math.sqrt(discriminant);
9
10        double root1 = (-b + sqroot) / 2.0;
11        double root2 = (-b - sqroot) / 2.0;
12
13        System.out.println(root1);
14        System.out.println(root2);
15    }
16 }

```

```

Infinity / 3.0 = Infinity
Infinity % 3.0 = NaN

sin(pi/2) = 1.0
log(e)    = 1.0

```

Chương trình Quadratic.java (đoạn mã 2.4) minh họa việc sử dụng phép tính số thực để tính 2 nghiệm của phương trình bậc hai bằng công thức quen thuộc. Thư viện Math của Java cho ta các hàm lượng giác, logarit, hàm mũ và các phép tính phổ biến khác của số thực.

```

% java Quadratic -3.0 2.0
2.0
1.0

% java Quadratic -1.0 -1.0
1.618033988749895
-0.6180339887498949

% java Quadratic 1.0 1.0
NaN
NaN

```

Chương trình Trig.java (đoạn mã 2.5) minh họa một số hàm lượng giác bao gồm Math.sin(), Math.cos(), và Math.toRadians().

```

% java Trig 30
sin(30.0) = 0.49999999999999994
cos(30.0) = 0.8660254037844387
tan(30.0) = 0.5773502691896257
0.49999999999999994 / 0.8660254037844387 = 0.5773502691896256
0.24999999999999994 + 0.75000000000000001 = 1.0

```

```

1 public class Trig {
2     public static void main(String[] args) {
3         double degrees = Double.parseDouble(args[0]);
4         double radians = Math.toRadians(degrees);
5
6         double s = Math.sin(radians);
7         System.out.println("sin(" + degrees + ") = " + s);
8
9         double c = Math.cos(radians);
10        System.out.println("cos(" + degrees + ") = " + c);
11
12        double t = Math.tan(radians);
13        System.out.println("tan(" + degrees + ") = " + t);
14        System.out.println(s + " / " + c + " = " + s / c);
15
16        double r = s*s + c*c;
17        System.out.println(s*s + " + " + c*c + " = " + r);
18    }
19 }

```

## 2.5 Boolean

Kiểu `boolean` chỉ có hai giá trị: `true` (đúng) hay `false` (sai). Tuy đơn giản nhưng nó là nền tảng của của khoa học máy tính. Các toán tử quan trọng nhất của kiểu `boolean` là AND (và), OR (hoặc) và NOT (ngịch đảo).

- AND: `a && b` bằng `true` nếu cả `a` và `b` đều là `true`, ngược lại nhận giá trị `false`.
- OR: `a || b` bằng `true` nếu ít nhất một trong hai `a` và `b` là `true`, ngược lại nhận giá trị `false`.
- NOT: `!a` là `true` nếu `a` là `false`, ngược lại nhận giá trị `false`.

Một cách khác để biểu diễn các phép toán này là dùng bảng logic

Bảng 2.1: Các toán tử logic

a	b	a && b	a    b	!a
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

## 2.6 Phép so sánh

Các toán tử so sánh là các toán tử trên các kiểu khác nhau, trong đó mỗi toán hạng có thể lấy giá trị trên các kiểu số (ví dụ, `int` hoặc `double`) và tạo ra một kết quả thuộc kiểu `boolean`. Các toán tử này đóng vai trò quan trọng cấu thành trong quá trình phát triển các chương trình phức tạp hơn.

Chương trình `LeapYear.java` (đoạn mã 2.6) kiểm tra xem một số nguyên có tương ứng với một năm nhuận trong lịch Gregorian.

Bảng 2.2: Các toán tử so sánh

Toán tử	Ý nghĩa	a op b	true	false
==	bằng	a bằng b	2 == 2	2 == 3
!=	không bằng	a không bằng b	2 == 3	2 == 2
<	nhỏ hơn	a nhỏ hơn b	3 < 4	5 < 4
<=	không lớn hơn	a không lớn hơn b	4 <= 4	5 <= 3
>	lớn hơn	a lớn hơn b	7 > 4	6 > 8.5
>=	không nhỏ hơn	a không nhỏ hơn b	9 >= 2	5 >= 10

Đoạn mã 2.6: LeapYear.java

```

1 public class LeapYear {
2     public static void main(String[] args) {
3         int year = Integer.parseInt(args[0]);
4         boolean isLeapYear;
5
6         // divisible by 4
7         isLeapYear = (year % 4 == 0);
8
9         // divisible by 4 and not 100
10        isLeapYear = isLeapYear && (year % 100 != 0);
11
12        // divisible by 4 and not 100 unless divisible by 400
13        isLeapYear = isLeapYear || (year % 400 == 0);
14
15        System.out.println(isLeapYear);
16    }
17 }

```

```

% java LeapYear 2004
true

% java LeapYear 1900
false

% java LeapYear 2000
true

```

## 2.7 Chuyển đổi kiểu

Chúng ta sẽ hay thấy mình chuyển đổi dữ liệu từ một kiểu sang kiểu khác bằng một trong các phương pháp sau đây.

- *Chuyển kiểu rõ ràng*: Gọi các hàm như `Math.round()`, `Integer.parseInt()`.
- *Chuyển kiểu tự động*: Đối với các kiểu số cơ bản, Java tự động thực hiện việc chuyển kiểu khi gán các giá trị có phạm vi kiểu lớn hơn kiểu của biến được gán.

```

1 public class RandomInt {
2     public static void main(String[] args) {
3         int N = Integer.parseInt(args[0]);
4
5         // a pseudo-random real between 0.0 and 1.0
6         double r = Math.random();
7
8         // a pseudo-random integer between 0 and N-1
9         int n = (int) (r * N);
10
11         System.out.println("Your random integer is: " + n);
12     }
13 }

```

- Chuyển đổi rõ ràng: Java cung cấp một số hàm chuyển đổi các kiểu khi ta biết rõ việc chuyển kiểu sẽ dẫn đến mất thông tin. Chương trình `RandomInt.java` (đoạn mã 2.7) minh họa cách chuyển đổi này. Chương trình này đọc vào từ dòng lệnh số  $N$  và in ra một số ngẫu nhiên trong khoảng 0 đến  $N-1$ .
- Chuyển đổi tự động cho xâu: Kiểu `String` tuân thủ một số luật đặc biệt. Một trong số đó là bạn có thể chuyển đổi bất kỳ loại dữ liệu nào thành một `String` bằng cách sử dụng toán tử `+`.

Kết quả chạy chương trình `RandomInt.java` (có thể khác kết quả của bạn)

```

% java RandomInt 6
Your random integer is: 3

% java RandomInt 1000
Your random integer is: 764

% java RandomInt 1000
Your random integer is: 140

```

## 2.8 Hỏi đáp

**Hỏi.** Làm thế nào để đánh dấu | ?

**Đáp.** Nhấn  + .

**Hỏi.** Java in ra một tấn của các chữ số khi tôi dùng `System.out.println()` cho số thực. Làm thế nào tôi có thể định dạng để nó chỉ hiển thị 3 chữ số phần thập phân?

**Đáp.** Sử dụng hàm `System.out.printf()` mô tả trong Chương 5.

**Hỏi.** Tại sao thương các số nguyên  $-0/3$  cho kết quả 0, nhưng thương các số thực  $-0.0 / 3.0$  cho kết quả  $-0.0$  ?

**Đáp.** Java biểu diễn số nguyên `int` bằng *phương pháp bù 2* (two's complement notation). Mỗi số nguyên chỉ có một cách biểu diễn. Trong khi đó, Java biểu diễn số thực `double` bằng cách sử dụng chuẩn IEEE, và có 2 cách biểu diễn khác nhau cho số 0 và  $-0$ .

**Hỏi.** Điều gì xảy ra nếu tôi sử dụng `/` và `%` với tử số là số âm ?



**Đáp.** Hãy thử nó xem.  $-47 / 5 = -9$  Và  $-47 \% 5 = -2$ . Các thương luôn làm tròn hướng về số 0. Để đảm bảo tính chất Euclide  $b * (a / b) + (a \% b) = a$ , kết quả của phép lấy số dư có thể âm. Quy ước này được thừa hưởng từ ngôn ngữ trước đó như FORTRAN và C. Một số ngôn ngữ (nhưng không phải Java) có cả phép lấy số dư và phép modulo vì sẽ thuận tiện nếu có một toán tử trả về số dư không âm.

**Hỏi.** Tôi có thể sử dụng % với số thực không ?

**Đáp.** Có. Nếu `angle` không âm, thì `angle % (2 * Math.PI)` chuyển đổi các góc về khoảng giữa 0 và  $2\pi$ .

**Hỏi.** Làm thế nào để in dấu nhảy kép " ?

**Đáp.** Vì " là kí tự đặc biệt khi làm việc với chuỗi kí tự, bạn cần viết nó dưới dạng \". Ví dụ, `System.out.println("The_pig_said_\\"Oink_Oink\\"afterwards");`.

**Hỏi.** Vậy thì làm thế nào để in dấu \ ?

**Đáp.** Sử dụng "\\".

**Hỏi.** Có giới hạn gì cho việc đặt tên biến ?

**Đáp.** Có. Một tên trong Java bắt đầu bằng một kí tự trong bảng chữ cái, tiếp theo là một chuỗi không giới hạn của các chữ cái và chữ số. Các kí tự này có thể lấy trong bảng Unicode nhưng trong cuốn sách này ta chỉ dùng các chữ cái tiếng Anh. Theo quy ước, các biến thường bắt đầu bằng một chữ viết thường. Tên trong Java không được phép dùng các từ dành riêng sau.

<b>abstract</b>	<b>default</b>	<b>goto</b>	<b>package</b>	<b>this</b>
<b>assert</b>	<b>do</b>	<b>if</b>	<b>private</b>	<b>throw</b>
<b>boolean</b>	<b>double</b>	<b>implement</b>	<b>protected</b>	<b>throws</b>
<b>break</b>	<b>else</b>	<b>import</b>	<b>public</b>	<b>transient</b>
<b>byte</b>	<b>enum</b>	<b>instanceof</b>	<b>return</b>	<b>true</b>
<b>case</b>	<b>extends</b>	<b>int</b>	<b>short</b>	<b>try</b>
<b>catch</b>	<b>false</b>	<b>interface</b>	<b>static</b>	<b>void</b>
<b>char</b>	<b>final</b>	<b>long</b>	<b>strictfp</b>	<b>volatile</b>
<b>class</b>	<b>finally</b>	<b>native</b>	<b>super</b>	<b>while</b>
<b>const</b>	<b>float</b>	<b>new</b>	<b>switch</b>	
<b>continue</b>	<b>for</b>	<b>null</b>	<b>synchronized</b>	

**Hỏi.** Thứ tự tính toán của các toán tử Java như thế nào ?

**Đáp.** Xem phụ lục A.

**Hỏi.** Có khác biệt gì giữa `a += b` và `a = a + b`, với `a` và `b` là các biến kiểu số cơ bản ?

**Đáp.** Có nếu `a` và `b` có kiểu khác nhau. Câu lệnh gán `a += b` tương đương với `a = (int) (a + b)` nếu `a` có kiểu `int`. Như vậy nếu `b` là `double` thì `a += b` hợp lệ trong khi `a = a + b` gây lỗi biên dịch.

**Hỏi.** Tại sao tôi cần khai báo kiểu của biến trong Java?

**Đáp.** Bằng cách xác định kiểu, trình biên dịch có thể cảnh báo bạn những lỗi tiềm tàng có thể xảy ra, ví dụ như khi bạn cố gắng để nhân số nguyên với chuỗi kí tự. Tương tự như thế, trong vật lý, ta luôn luôn phải kiểm tra các đại lượng có cùng đơn vị (cùng kiểu). Đối với các chương trình nhỏ, điều này có vẻ như không quan trọng; nhưng với các chương trình lớn, việc này cực kì quan trọng. Các tên lửa Ariane 5 đã phát nổ 40 giây sau khi cất cánh chỉ vì một lỗi không chuyển đổi chính xác một số thực 64 bit thành một số nguyên 16 bit trong phần mềm của chúng.

**Hỏi.** Tại sao là kiểu số thực được gọi là `double`?

**Đáp.** Trong lịch sử, kiểu số dấu chấm động là `float`, nhưng chúng có độ chính xác hạn chế. Kiểu `double` được đưa vào với độ chính xác gấp đôi.

## 2.9 Bài tập

1. Giả sử `a` và `b` là các giá trị `int`. Đoạn mã sau làm việc gì ?

```
int t = a;
b = t;
a = b;
```

2. Viết một chương trình sử dụng `Math.sin()` và `Math.cos()` để kiểm tra đẳng thức  $\sin^2(\theta) + \cos^2(\theta) = 1$  với  $\theta$  nhập từ dòng lệnh. Tại sao kết quả không phải lúc nào cũng chính xác bằng 1?
3. Giả sử rằng `a` và `b` là các giá trị `boolean`. Chứng minh các biểu thức `(!(a && b) && (a || b)) || ((a && b) || !(a || b))` tương đương với `true`.
4. Giả sử rằng `a` và `b` là các giá trị `int`. Đơn giản hóa biểu thức sau đây: `(!(a < b) && !(a > b))`.
5. Toán tử XOR cho kiểu `boolean`: `a ^ b` chỉ bằng `true` nếu chỉ một trong hai `a` hoặc `b` là `true`. Hãy viết bảng logic của toán tử này.
6. Tại sao `10 / 3` cho kết quả là 3 chứ không phải 3.33333 ?
7. Các lệnh sau đây in ra cái gì ?
  - `System.out.println(2 + "bc");`
  - `System.out.println(2 + 3 + "bc");`
  - `System.out.println((2+3) + "bc");`
  - `System.out.println("bc" + (2+3));`
  - `System.out.println("bc" + 2 + 3);`
8. Hãy sử dụng `Quadratic.java` để tính căn bậc hai của một số.
9. Các lệnh sau đây in ra cái gì ?
  - `System.out.println('b');`
  - `System.out.println('b' + 'c');`
  - `System.out.println((char) ('a' + 4));`
10. Giả sử `a = 2147483647` (tương đương với `Integer.MAX_VALUE`). Mỗi lệnh sau đây làm gì?
  - `System.out.println(a);`
  - `System.out.println(a + 1);`
  - `System.out.println(2 - a);`
  - `System.out.println(-2 - a);`
  - `System.out.println(2 * a);`
  - `System.out.println(4 * a);`

Giải thích cho từng kết quả.

11. Còn nếu có lệnh `double a = 3.14159;` thì sao ?
  - `System.out.println(a);`
  - `System.out.println(a + 1);`

- `System.out.println(8 / (int) a);`
- `System.out.println(8 / a);`
- `System.out.println((int) (8 / a));`

12. Giải thích điều gì xảy ra nếu thay `Math.sqrt` bằng `sqrt` trong chương trình `Quadratic.java`.

13. Biểu thức `(Math.sqrt(2) * Math.sqrt(2) == 2)` có giá trị thế nào ?

14. Viết chương trình lấy vào hai số nguyên dương và trả về `true` nếu một trong hai số là bội số của số kia.

15. Viết chương trình lấy vào ba số thực dương và trả về `true` nếu một trong số đó lớn hơn tổng hai số kia. Như vậy chương trình này có thể kiểm tra tính hợp lệ của ba cạnh tam giác.

16. Một sinh viên vật lý được kết quả không mong đợi khi viết

```
F = G * mass1 * mass2 / r * r;
```

để tính  $F = \frac{G \times \text{mass1} \times \text{mass2}}{r^2}$ . Hãy giải thích lỗi và sửa lỗi.

17. Tính giá trị của `a` sau các đoạn mã sau

<code>int a = 1;</code>	<code>boolean a = true;</code>	<code>int a = 2;</code>
<code>a = a + a;</code>	<code>a = !a;</code>	<code>a = a * a;</code>
<code>a = a + a;</code>	<code>a = !a;</code>	<code>a = a * a;</code>
<code>a = a + a;</code>	<code>a = !a;</code>	<code>a = a * a;</code>

18. Giả sử `x` và `y` có kiểu `double` biểu diễn tọa độ Đề-Các  $(x, y)$  của một điểm trên mặt phẳng. Tính khoảng cách từ điểm đó đến gốc tọa độ.

19. Giả sử `a` và `b` là 2 số nguyên. Viết chương trình sinh một số nguyên ngẫu nhiên trong khoảng  $[a, b]$ .

20. Viết chương trình `SumOfTwoDice.java` tính tổng giá trị hai con xúc sắc (có giá trị ngẫu nhiên từ 1 đến 6).

21. Viết chương trình đọc số thực  $t$  từ dòng lệnh và in ra kết quả  $\sin(2t) + \sin(3t)$ .

22. Viết chương trình đọc 3 số thực  $x_0$ ,  $v_0$  và  $t$  từ dòng lệnh và in ra kết quả  $x_0 + v_0 t + \frac{1}{2}gt^2$  với  $g = 9.800722m/s^2$  là gia tốc trọng trường (đây là độ cao vật thể được ném thẳng đứng từ độ cao ban đầu  $x_0$  với vận tốc ban đầu  $v_0$ ).

23. Viết chương trình đọc 2 số nguyên  $m$  (tháng) và  $d$  (ngày) từ dòng lệnh và in ra kết quả `true` nếu đây là ngày thuộc mùa Xuân. Mùa Xuân kéo dài từ 20 tháng 3 đến 20 tháng 6 hàng năm.

24. **(Trả nợ hàng tháng).** Viết chương trình tính số tiền hàng tháng phải trả cho khoản nợ ban đầu  $P$ , lãi suất năm  $r$  (phần trăm), và thời hạn của khoản nợ  $t$  (số năm). Công thức tính như sau

$$c = \frac{\hat{r}P}{1 - (1 + \hat{r})^{-N}},$$

trong đó  $c$  là số tiền trả hàng tháng,  $\hat{r}$  là lãi suất tháng ( $= r/12/100$ ) còn  $N$  là số tháng phải trả nợ ( $= t \times 12$ ).

25. Viết chương trình `WindChill.java` tính nhiệt độ ta cảm nhận được khi nhiệt độ ngoài trời là  $T_a$  (độ C) và tốc độ gió tại độ cao 10m là  $V$  (km/h). Công thức tính như sau

$$T_{wc} = 13.12 + 0.6215T_a - 11.37V^{+0.16} + 0.3965T_aV^{+0.16}.$$

26. Viết chương trình `CartesianToPolar.java` biến đổi tọa độ  $(x, y)$  thành tọa độ cực  $(r, \theta)$ . Sử dụng hàm `Math.atan2(y, x)` để tính  $\arctan(y/x)$  trong khoảng  $[-\pi, \pi]$ .

27. Viết chương trình `StdGaussian.java` sinh ra số ngẫu nhiên theo phân bố Gauss. Công thức tính như sau

$$Z = \sin(2\pi v) \sqrt{-2 \ln u},$$

với  $u$  và  $v$  là các số ngẫu nhiên phân bố đều trong khoảng  $[0, 1]$  (dùng hàm `Math.random()` để sinh).

28. Viết chương trình trả về `true` nếu 3 số nhập vào từ dòng lệnh tăng dần hoặc giảm dần (`false` nếu ngược lại).

29. Viết chương trình `DayOfWeek.java` tính thứ của ngày nhập vào từ dòng lệnh (3 số  $d$  (ngày),  $m$  (tháng),  $y$  (năm)). Dùng công thức sau

```
y0 = y - (14 - m) / 12
x = y0 + y0/4 - y0/100 + y0/400
m0 = m + 12 * ((14 - m) / 12) - 2
d0 = (d + x + (31*m0)/ 12) mod 7
```

Kết quả 0: Chủ nhật, 1: thứ Hai, và vân vân.

30. Viết chương trình `Stats5.java` sinh ra 5 số thực ngẫu nhiên thuộc khoảng  $[0, 1]$ . Sau đó in ra giá trị lớn nhất, nhỏ nhất, và giá trị trung bình của các số này (Sử dụng các hàm `Math.random()`, `Math.min()`, `Math.max()`).

31. **(Phép chiếu Mecator).** Đây là phép chiếu tọa độ giữ nguyên góc biến cặp vĩ tuyến  $\varphi$  và kinh tuyến  $\lambda$  thành một điểm  $(x, y)$  trên mặt phẳng. Phép chiếu này được dùng rất nhiều trong việc vẽ bản đồ. Phép chiếu như sau

$$x = \lambda - \lambda_0$$

$$y = \frac{1}{2} \ln \frac{1 + \sin \varphi}{1 - \sin \varphi}$$

Viết chương trình lấy  $\lambda_0$ ,  $\varphi$  và  $\lambda$  từ dòng lệnh và in kết quả chiếu ra màn hình.

32. Viết chương trình `RGBtoCMYK.java` chuyển đổi màu từ định dạng RGB (đỏ - red, xanh lá - green, xanh nước biển - blue) sang dạng CMYK (xanh lơ - cyan, đỏ tím - magenta, vàng - yellow, đen - black). Định dạng RGB thường dùng trong camera với các giá trị red, green, blue là các số nguyên trong khoảng  $[0, 255]$ . Định dạng CMYK thường dùng trong xuất bản ấn phẩm với các giá trị cyan, magenta, yellow, black là các số thực trong khoảng  $[0, 1]$ . Công thức chuyển như sau

```
white = max {red / 255, green / 255, blue / 255}
cyan = (white - red / 255) / white
magenta = (white - green / 255) / white
yellow = (white - blue / 255) / white
black = 1 - white
```

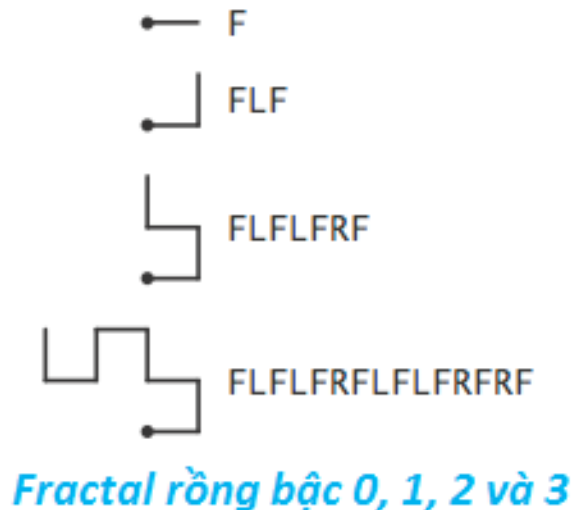
33. Viết chương trình `GreatCircle.java` tính khoảng cách thực (theo hình cầu) giữa 2 điểm có vĩ độ, kinh độ  $(x_1, y_1)$  và  $(x_2, y_2)$  (nhập từ dòng lệnh). Công thức tính như sau

$$d = 60 \arccos(\sin x_1 \sin x_2 + \cos x_1 \cos x_2 \cos(y_1 - y_2)),$$

trong đó  $d$  tính theo đơn vị hải lý (1 hải lý bằng 1.892 km). Công thức trên có sai số khoảng 0.5%. Công thức chính xác hơn như sau

```
double delta = Y1 - Y2;
double p1 = cos(X2) * sin(delta);
double p2 = cos(X1) * sin(X2) - sin(X1) * cos(X2) * cos(delta);
double p3 = sin(X1) * sin(X2) + cos(X1) * cos(X2) * cos(delta);
distance = 60 * Math.atan2(Math.sqrt(p1*p1 + p2*p2), p3);
```

34. Viết chương trình `ThreeSort.java` nhập 3 số từ dòng lệnh và in chúng ra theo thứ tự tăng dần.
35. (**Fractal rồng**). Viết chương trình `Dragon.java` mô tả các bước vẽ nên hình *fractal rồng* có bậc từ 0 đến 5 bằng các kí tự: F - tiến 1 bước, L - quay sang trái, R - quay sang phải.



Fractal rồng bậc  $n$  bằng fractal rồng bậc  $n - 1$ , thêm chữ L, và thêm một fractal rồng bậc  $n - 1$  nữa nhưng viết ngược lại và đảo R với L.



# Chương 3

## Rẽ nhánh và vòng lặp

### Mục lục chương

3.1	Lệnh If . . . . .	27
3.2	Vòng lặp While . . . . .	27
3.3	Vòng lặp For . . . . .	28
3.4	Các vòng lặp và cấu trúc rẽ nhánh khác . . . . .	30
3.5	Ví dụ . . . . .	30
3.6	Hỏi đáp . . . . .	34
3.7	Bài tập . . . . .	34

### 3.1 Lệnh If

. Hầu hết tính toán đều đòi hỏi ta xử lý các trường hợp khác nhau của đầu vào. Chương trình Flip.java (đoạn mã 3.1) sử dụng một lệnh `if-else` in ra kết quả tung một đồng xu.

Bảng 3.1 tóm tắt một số tình huống điển hình mà bạn có thể cần sử dụng lệnh `if` hoặc lệnh `if-else`.

### 3.2 Vòng lặp While

Nhiều phép tính cần được lặp đi lặp lại. Vòng lặp `while` cho phép chúng ta thực hiện một khối lệnh nhiều lần. Như vậy chúng ta có thể thực hiện các tính toán dài dòng mà không cần viết quá nhiều mã.

Đoạn mã 3.1: Flip.java

```
1 public class Flip {
2
3     public static void main(String[] args) {
4         // Math.random() returns a value between 0.0 and 1.0
5         // so it is heads or tails 50% of the time
6         if (Math.random() < 0.5) System.out.println("Heads");
7         else System.out.println("Tails");
8     }
9 }
```

Bảng 3.1: Các trường hợp sử dụng câu lệnh rẽ nhánh

Giá trị tuyệt đối	<code>if (x &lt; 0) x = -x;</code>
Giá trị lớn nhất	<code>if (x &gt; y) max = x; else max = y;</code>
Chia trường hợp tính thuế	<code>if (income &lt; 47450) rate = .22; else if (income &lt; 114650) rate = .25; else if (income &lt; 174700) rate = .28; else if (income &lt; 311950) rate = .33; else rate = .35;</code>
Kiểm tra bắt lỗi chia cho 0	<code>if (d == 0)     System.out.println("Division by zero"); else     System.out.println("Quotient = " + n / d);</code>
Kiểm tra các trường hợp giải phương trình bậc 2	<code>double discriminant = b*b - 4*a*c; if (discriminant &lt; 0)     System.out.println("No real root"); else {     double d = Math.sqrt(discriminant);     System.out.println((-b + d) / (2*a));     System.out.println((-b - d) / (2*a)); }</code>

- Chương trình `TenHellos.java` (đoạn mã 3.2) in câu "Hello" 10 lần.
- Chương trình `PowersOfTwo.java` (đoạn mã 3.3) có đối số dòng lệnh  $N$  và in ra tất cả số mũ của 2 nhỏ hơn hoặc bằng  $2^N$ .

### 3.3 Vòng lặp For

Đa số vòng lặp sau này các bạn viết sẽ tuân thủ các bước cơ bản sau :

- Khởi tạo một biến chỉ số bằng một giá trị nào đó,
- Sử dụng một vòng lặp `while` để kiểm tra một điều kiện dừng vòng lặp
- Sử dụng câu lệnh cuối cùng trong vòng lặp `while` để thay đổi biến chỉ số.

Vòng lặp `for` trong Java là một cách trực tiếp để viết vòng lặp như vậy. Ví dụ, hai dòng mã sau đây tương đương với các dòng tương ứng của mã trong `TenHellos.java`.

```
for (int i = 4; i <= 10; i = i + 1)
    System.out.println(i + "th Hello");
```

- Phép toán `i++` tương đương với phép toán `i += 1`.
- Phạm vi (scope): biến `i` chỉ có nghĩa bên trong vòng lặp `for`.



Đoạn mã 3.2: TenHellos.java

```

1 public class TenHellos {
2     public static void main(String[] args) {
3
4         // print out special cases whose ordinal doesn't end in th
5         System.out.println("1st Hello");
6         System.out.println("2nd Hello");
7         System.out.println("3rd Hello");
8
9         // count from i = 4 to 10
10        int i = 4;
11        while (i <= 10) {
12            System.out.println(i + "th Hello");
13            i = i + 1;
14        }
15
16    }
17 }

```

Đoạn mã 3.3: PowersOfTwo.java

```

1 public class PowersOfTwo {
2     public static void main(String[] args) {
3
4         // read in one command-line argument
5         int N = Integer.parseInt(args[0]);
6
7         int i = 0;                // count from 0 to N
8         int powerOfTwo = 1;       // the ith power of two
9
10        // repeat until i equals N
11        while (i <= N) {
12            System.out.println(i + " " + powerOfTwo);    // print out
13                                                         the power of two
14            powerOfTwo = 2 * powerOfTwo;                // double to
15                                                         get the next one
16            i = i + 1;
17        }
18    }
19 }

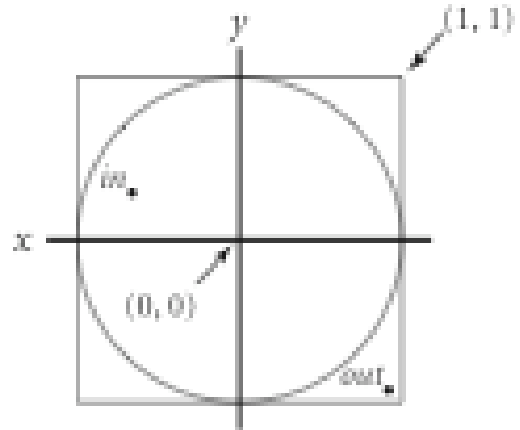
```

## 3.4 Các vòng lặp và cấu trúc rẽ nhánh khác

Sau này chủ yếu ta sẽ sử dụng các cấu trúc `if`, `if-else`, `while` và `for`. Tuy nhiên, Java còn các cấu trúc điều khiển khác mà ta cần biết (mặc dù sẽ ít sử dụng hơn).

**Cấu trúc do-while.** Cấu trúc này giống cấu trúc `while` chỉ có điều nó bỏ qua việc kiểm tra điều kiện trước vòng lặp đầu tiên. Ví dụ đoạn mã sau cho một điểm ngẫu nhiên trong vòng tròn đơn vị.

```
double x, y, r;
do {
    x = 2.0 * Math.random() - 1.0;
    y = 2.0 * Math.random() - 1.0;
    r = x*x + y*y;
} while (r > 1);
```



**Lệnh break.** Lệnh này sử dụng khi ta muốn thoát khỏi vòng lặp ngay lập tức. Chương trình `Prime.java` (đoạn mã 3.4) trả về `true` nếu số  $N$  là số nguyên tố, `false` nếu  $N$  không phải số nguyên tố.

**Lệnh continue.** Lệnh này bỏ qua các lệnh phía sau trong vòng lặp để nhảy sang lần lặp tiếp theo.

**Phép toán lựa chọn.** Đây là phép toán có 3 toán hạng phân cách bởi dấu `?` và dấu `:`. Phép toán có giá trị bằng toán hạng thứ hai nếu toán hạng thứ nhất bằng `true`, ngược lại nó lấy giá trị của toán hạng thứ ba. Ví dụ

```
int min = (x < y) ? x : y;
```

## 3.5 Ví dụ

Chương trình `Loops.java` (đoạn mã 3.5 – 3.10) minh họa một loạt các vòng lặp đơn giản in ra các kết quả khác nhau.

Khả năng lập trình kết hợp vòng lặp và điều kiện rẽ nhánh mở ra cho chúng ta thế giới tính toán vô cùng phong phú.

**Dãy Harmonic.** Chương trình `Harmonic.java` (đoạn mã 3.11) tính chuỗi số Harmonic

$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}$$

Đoạn mã 3.4: Prime.java

```

1 public class Prime {
2
3     public static void main(String[] args) {
4         long N = Long.parseLong(args[0]);
5         boolean isPrime = true;
6         if (N < 2) isPrime = false;
7
8         // try all possible factors i of N
9         // if N has a factor, then it has one less than or equal to
10        sqrt(N),
11        // so for efficiency we only need to check i <= sqrt(N) or
12        // equivalently i*i <= N
13        for (long i = 2; i*i <= N; i++) {
14
15            // if i divides evenly into N, N is not prime, so break
16            // out of loop
17            if (N % i == 0) {
18                isPrime = false;
19                break;
20            }
21
22        // print out whether or not N is prime
23        if (isPrime) System.out.println(N + " is prime");
24        else System.out.println(N + " is not prime");
25    }
26 }

```

Đoạn mã 3.5: Loops.java in các số mũ của 2

```

1 // print powers of two
2 int v = 1;
3 for (int i = 0; i <= N; i++) {
4     System.out.println(v);
5     v = 2 * v;
6 }
7 System.out.println();

```

Đoạn mã 3.6: Loops.java in số mũ của 2 không vượt quá N

```

1 // print largest power of two less than or equal to N
2 v = 1;
3 while (v <= N/2) {
4     v = 2 * v;
5 }
6 System.out.println(v);
7 System.out.println();

```

Đoạn mã 3.7: Loops.java in tổng  $1 + 2 + \dots + N$

```
1 // compute a finite sum (1 + 2 + ... + N)
2 int sum = 0;
3 for (int i = 1; i <= N; i++) {
4     sum += i;
5 }
6 System.out.println(sum);
7 System.out.println();
```

Đoạn mã 3.8: Loops.java tính giai thừa

```
1 // compute a finite product (N!)
2 int product = 1;
3 for (int i = 1; i <= N; i++) {
4     product *= i;
5 }
6 System.out.println(product);
7 System.out.println();
```

Đoạn mã 3.9: Loops.java chia vòng tròn lượng giác thành  $N$  phần

```
1 // print a table of values (2 pi i / N)
2 for (int i = 0; i <= N; i++) {
3     System.out.println(i + " " + 2 * Math.PI * i / N);
4 }
5 System.out.println();
```

Đoạn mã 3.10: Loops.java độ cao vạch thước kẻ

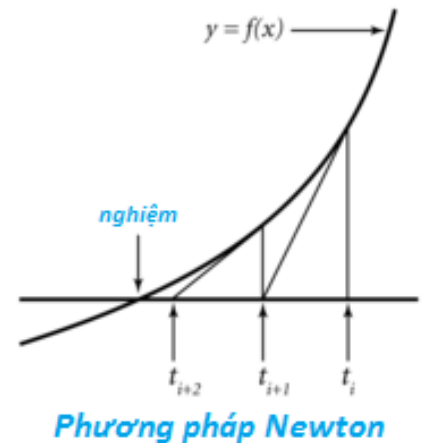
```
1 // print the ruler function
2 String ruler = " ";
3 for (int i = 1; i <= N; i++) {
4     ruler = ruler + i + ruler;
5 }
6 System.out.println(ruler);
7 System.out.println();
```

```

1 public class Harmonic {
2     public static void main(String[] args) {
3
4         // command-line argument
5         int N = Integer.parseInt(args[0]);
6
7         // compute 1/1 + 1/2 + 1/3 + ... + 1/N
8         double sum = 0.0;
9         for (int i = 1; i <= N; i++) {
10             sum += 1.0 / i;
11         }
12
13         // print out Nth harmonic number
14         System.out.println(sum);
15     }
16 }
17 }

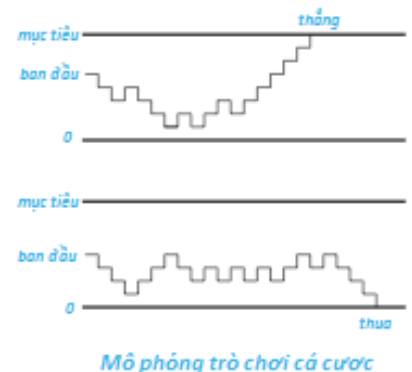
```

**Phương pháp Newton.** Chương trình Sqrt.java (đoạn mã 3.12) tính căn bậc 2 của một số thực  $x$ : bắt đầu với ước lượng  $t$ , so sánh  $t$  với  $x/t$ , nếu 2 số này bằng nhau (trong độ chính xác cho phép) thì báo kết quả, ngược lại, thay ước lượng bằng trung bình cộng của 2 số này.



**Chuyển hệ cơ số.** Chương trình Binary.java (đoạn mã 3.13) in số ở dạng hệ cơ số 2.

**Mô phỏng trò cá cược.** Một người chơi cá cược bắt đầu với một khoản tiền (\$50), mỗi lần anh ta cược \$1. Nếu thắng anh ta được \$1, nếu thua anh ta mất \$1 (xác suất thắng thua bằng nhau). Cuộc chơi dừng khi người chơi hết tiền hoặc đạt một số tiền nào đó (ví dụ \$250). Chương trình Gambler.java (đoạn mã 3.14) mô phỏng trò chơi này với đầu vào là số tiền ban đầu, số tiền cần đạt được và số lần chơi.



**Phân tích số.** Chương trình Factors.java (đoạn mã 3.15) phân tích số  $N$  (từ dòng lệnh) thành tích các số nguyên tố.



```

1 public class Sqrt {
2     public static void main(String[] args) {
3
4         // read in the command-line argument
5         double c = Double.parseDouble(args[0]);
6         double epsilon = 1e-15;    // relative error tolerance
7         double t = c;              // estimate of the square root of c
8
9         // repeatedly apply Newton update step until desired precision
           is achieved
10        while (Math.abs(t - c/t) > epsilon*t) {
11            t = (c/t + t) / 2.0;
12        }
13
14        // print out the estimate of the square root of c
15        System.out.println(t);
16    }
17
18 }

```

## 3.6 Hỏi đáp

**Hỏi.** Chương trình của tôi mắc kẹt trong vòng lặp vô hạn. Làm thế nào để thoát khỏi nó?

**Đáp.** Nhấn tổ hợp  + .

**Hỏi.** Làm thế nào để kiểm tra 2 chuỗi ký tự có bằng nhau.

**Đáp.** Đây là điểm khác biệt giữa chuỗi ký tự và các kiểu cơ bản khác như `int`, `double`, `boolean`. Xem phần 3.

**Hỏi.** Tại sao lệnh `if (a <= b <= c)` không chạy ?

**Đáp.** Hãy dùng `if (a <= b && b <= c)`

**Hỏi.** Có ví dụ nào mà ta không thể bỏ dấu ngoặc nhọn sau vòng `for`.

**Đáp.** Xem các đoạn mã sau đây, đoạn mã đầu tiên là hợp lệ (nhưng vô nghĩa), đoạn thứ hai gây lỗi biên dịch. Chính xác mà nói, dòng thứ hai trong đoạn thứ hai là một khai báo chứ không phải là một câu lệnh.

```

// legal
for (int i = 0; i <= N; i++) {
    int x = 5;
}

// illegal
for (int i = 0; i <= N; i++)
int x = 5;

```

## 3.7 Bài tập

- Viết một chương trình nhận 3 số nguyên từ dòng lệnh và in ra "Equal" nếu cả ba số đều bằng nhau, và in ra "Not equal" nếu ngược lại.

Đoạn mã 3.13: Binary.java

```
1 public class Binary {
2     public static void main(String[] args) {
3
4         // read in the command-line argument
5         int n = Integer.parseInt(args[0]);
6
7         // set v to the largest power of two that is <= n
8         int v = 1;
9         while (v <= n/2) {
10             v = v * 2;
11         }
12
13         // check for presence of powers of 2 in n, from largest to
14         // smallest
15         while (v > 0) {
16
17             // v is not present in n
18             if (n < v) {
19                 System.out.print(0);
20             }
21
22             // v is present in n, so remove v from n
23             else {
24                 System.out.print(1);
25                 n = n - v;
26             }
27
28             // next smallest power of 2
29             v = v / 2;
30         }
31
32         System.out.println();
33     }
34 }
35 }
```

Đoạn mã 3.14: Gambler.java

```

1 public class Gambler {
2
3     public static void main(String[] args) {
4         int stake = Integer.parseInt(args[0]);    // gambler's starting
           bankroll
5         int goal  = Integer.parseInt(args[1]);    // gambler's desired
           bankroll
6         int T     = Integer.parseInt(args[2]);    // number of trials
           to perform
7
8         int bets = 0;        // total number of bets made
9         int wins = 0;        // total number of games won
10
11         // repeat T times
12         for (int t = 0; t < T; t++) {
13
14             // do one gambler's ruin simulation
15             int cash = stake;
16             while (cash > 0 && cash < goal) {
17                 bets++;
18                 if (Math.random() < 0.5) cash++;    // win $1
19                 else cash--;    // lose $1
20             }
21             if (cash == goal) wins++;    // did gambler go
           achieve desired goal?
22         }
23
24         // print results
25         System.out.println(wins + " wins of " + T);
26         System.out.println("Percent of games won = " + 100.0 * wins /
           T);
27         System.out.println("Avg # bets          = " + 1.0 * bets / T);
28     }
29
30 }

```



Đoạn mã 3.15: Factors.java

```
1 public class Factors {
2
3     public static void main(String[] args) {
4
5         // command-line argument
6         long n = Long.parseLong(args[0]);
7
8         System.out.print("The prime factorization of " + n + " is: ");
9
10        // for each potential factor i
11        for (long i = 2; i*i <= n; i++) {
12
13            // if i is a factor of N, repeatedly divide it out
14            while (n % i == 0) {
15                System.out.print(i + " ");
16                n = n / i;
17            }
18        }
19
20        // if biggest factor occurs only once, n > 1
21        if (n > 1) System.out.println(n);
22        else      System.out.println();
23    }
24 }
```

2. Viết lại chương trình `Quadratic.java` (đoạn mã 2.4) in ra nghiệm của phương trình  $ax^2 + bx + c = 0$ , in ra thông báo lỗi thích hợp nếu  $\Delta$  âm và xử lý cả trường hợp  $a = 0$ .
3. Các lệnh sau đây có lỗi gì ?
  - `if (a > b) then c = 0;`
  - `if a > b { c = 0; }`
  - `if (a > b) c = 0;`
  - `if (a > b) c = 0 else b = 0;`
4. Viết một đoạn mã trả về `true` nếu cả hai biến thực `x` và `y` nằm trong khoảng  $(0, 1)$ , `false` nếu ngược lại.
5. Viết thêm vào lời giải của bài tập 25 chương 2 đoạn mã kiểm các điều kiện để công thức có nghĩa, in thông báo lỗi nếu các biến không thỏa mãn các điều kiện này.
6. Giả sử `i` và `j` đều có kiểu `int`. Giá trị của `j` là gì sau mỗi câu lệnh sau đây được thực hiện?
  - `for (i = 0, j = 0; i < 10; i++) j += i;`
  - `for (i = 0, j = 1; i < 10; i++) j += j;`
  - `for (j = 0; j < 10; j++) j += j;`
  - `for (i = 0, j = 0; i < 10; i++) j += j++;`
7. Viết lại `TenHellos.java` (đoạn mã 3.2) thành chương trình `Hellos.java` có thể nhập số lần chào từ dòng lệnh. Gợi ý: kiểm tra `i % 10` và `i % 100` xem lúc nào cần dùng `"st"`, `"nd"`, hay `"rd"`.
8. Viết chương trình `FivePerLine.java` sử dụng một vòng lặp `for` và một câu lệnh `if`, in các số nguyên từ 1000 đến 2000 với năm số nguyên trên mỗi dòng. Gợi ý: sử dụng toán tử `%`.
9. Viết một chương trình nhập số nguyên  $N$  từ dòng lệnh và sử dụng `Math.random()` để in  $N$  số thực ngẫu nhiên thuộc  $[0, 1]$ , sau đó in giá trị trung bình của chúng. (xem bài tập 2.30).
10. Mô tả điều gì xảy ra nếu bạn gọi `RulerN.java` (đoạn mã 3.16) với số  $N$  quá lớn, chẳng hạn như

```
% java RulerN 100.
```

11. Viết chương trình `FunctionGrowth.java` in một bảng các giá trị của  $\log N$ ,  $N$ ,  $N \log N$ ,  $N^2$ ,  $N^3$ , và  $2^N$  cho  $N = 16, 32, 64, \dots, 2048$ . Sử dụng kí tự tab (`' '`) để giống hàng các cột.
12. Giá trị của `m` và `n` sau khi thực hiện đoạn mã sau đây là gì ?

```
int n = 123456789;
int m = 0;
while (n != 0) {
    m = (10 * m) + (n % 10);
    n = n / 10;
}
```

13. Đoạn mã sau in ra những gì ?

```

1 public class RulerN {
2     public static void main(String[] args) {
3         int N = Integer.parseInt(args[0]);
4
5         // ruler of order 0
6         String ruler = " ";
7
8         // repeat N times
9         for (int i = 1; i <= N; i++) {
10
11             // concatenate a ruler of order 0, the number i, and a
12             // ruler of order 0
13             ruler = ruler + i + ruler;
14
15             // print out the final result
16             System.out.println(ruler);
17         }
18     }
19 }

```

```

int f = 0, g = 1;
for (int i = 0; i <= 15; i++) {
    System.out.println(f);
    f = f + g;
    g = f - g;
}

```

14. Mở rộng lời giải bài tập 2.24 để in ra tổng số tiền đã thanh toán và số tiền còn lại sau mỗi lần thanh toán hàng tháng.
15. Không giống như dãy Harmonic, tổng  $1/1 + 1/4 + 1/9 + 1/16 + \dots + 1/N^2$  hội tụ về một hằng số khi  $N$  tiến đến vô cùng (hằng số này là  $\pi^2/6$  nên có thể dùng tổng này để xấp xỉ số  $\pi$ ). Vòng lặp nào trong các vòng lặp sau tính tổng trên? Biết rằng có khai báo `int N = 1000000;` `double sum = 0.0;`

```

(a) for (int i = 1; i <= N; i++)
    sum = sum + 1 / (i * i);

(b) for (int i = 1; i <= N; i++)
    sum = sum + 1.0 / i * i;

(c) for (int i = 1; i <= N; i++)
    sum = sum + 1.0 / (i * i);

(d) for (int i = 1; i <= N; i++)
    sum = sum + 1 / (1.0 * i * i);

```

16. Tiếp tuyến của đồ thị  $f(x)$  tại  $x = t$  có độ dốc là  $f'(t)$ . Hãy xây dựng phương trình đường tiếp tuyến của đồ thị tại điểm  $(t, f(t))$  và tính giao điểm của tiếp tuyến này với trục  $x$ . Từ đó ta có phương pháp Newton giải phương trình  $f(x) = 0$  bằng cách lặp như sau: bắt đầu với ước lượng  $t$ , thay thế ước lượng này bằng  $t - f(t)/f'(t)$ . Sử dụng công thức này và  $(x^2)' = 2x$  để chứng minh chương trình `Sqrt.java` (đoạn mã 3.12) cài đặt phương pháp Newton để tính căn bậc 2.
17. Sử dụng các công thức của phương pháp của Newton trong bài tập trước để phát triển một chương trình `Root.java` sử dụng hai đối số từ dòng lệnh  $c$  và  $k$ , in ra căn bậc  $k$  của  $c$ . Giả sử  $k$  là một số nguyên dương. Bạn có thể sử dụng `Math.pow()`, nhưng nhớ phải dùng số mũ là một số nguyên không âm.
18. Giả sử rằng  $x$  và  $t$  là các biến kiểu `double` và  $N$  là một biến kiểu `int`. Viết đoạn mã để tính  $x^N/N!$ .
19. Sửa đổi `Binary.java` (3.13) để được chương trình `Kary.java` sử dụng thêm đối số thứ hai từ dòng lệnh  $K$  và chuyển đổi đối số đầu tiên sang hệ cơ số  $K$ . Giả sử cơ số  $K$  nằm giữa 2 và 16. Đối với các hệ cơ số lớn hơn 10, sử dụng các chữ cái từ A đến F đại diện cho các con số từ 11 đến con số 16.
20. Viết một đoạn mã đặt biểu diễn nhị phân của số nguyên `int N` vào một xâu `String s`.
21. Viết một phiên bản của `Gambler.java` (đoạn mã 3.14) sử dụng một vòng lặp `for` thay cho vòng lặp `while`.
22. Viết chương trình `GamblerPlot.java` mô phỏng số tiền của một con bạc khi chơi cá cược bằng cách in một dòng sau mỗi lần đặt cược với có một dấu hoa thị tương ứng với mỗi đô-la con bạc đang có.
23. Sửa đổi `Gambler.java` sử dụng thêm một tham số dòng lệnh chỉ định xác suất (cố định) mà các con bạc thắng mỗi lần cược. Sử dụng chương trình của bạn tìm hiểu xem xác suất này ảnh hưởng đến cơ hội chiến thắng và số lượng lần đặt cược trung bình.
24. Sửa đổi `Gambler.java` sử dụng thêm một tham số dòng lệnh chỉ định cụ thể số lượng lần đặt cược mà các con bạc sẵn sàng chơi, do đó có ba cách có thể cho trò chơi để kết thúc: con bạc thắng, con bạc thua, hoặc hết giờ. Thêm vào giá trị trung bình số tiền con bạc có được khi trò chơi kết thúc.
25. Sửa đổi `Factors.java` (đoạn mã 3.15) để in chỉ một bản sao của từng ước số nguyên tố.
26. Thử nghiệm nhanh xác định tác động của việc sử dụng các điều kiện vòng lặp ( $i \leq N / i$ ) thay cho ( $i * i \leq N$ ) trong `Factors.java`. Đối với mỗi điều kiện, tìm số nguyên  $M$  lớn nhất sao cho khi bạn nhập số  $M$ , chương trình chắc chắn hoàn thành trong vòng 10 giây.
27. Viết chương trình `Checkerboard.java` sử dụng đối số dòng lệnh  $N$  và in ra bảng cờ  $N \times N$  bằng các dấu cách và dấu `*` như ví dụ  $4 \times 4$  sau đây.

```
* * * *
 * * * *
* * * *
 * * * *
```

28. Viết chương trình `GCD.java` chương trình tìm ước số chung lớn nhất (UCLN) của hai số nguyên  $x$  và  $y$  bằng cách sử dụng thuật toán Euclid, dựa trên quan sát sau đây: Nếu  $x > y$ , và  $x$  chia hết cho  $y$  thì UCLN của  $x$  và  $y$  là  $y$ ; nếu không thì UCLN của  $x$  và  $y$  là bằng UCLN của  $x \% y$  và  $y$ .
29. Viết `RelativelyPrime` chương trình mà phải mất một đối số dòng lệnh  $N$  và in ra một bảng  $N$ -by- $N$  như vậy mà có một  $*$  trong hàng  $i$ , cột  $j$  nếu UCLN của  $i$  và  $j$  là 1 ( $i$  và  $j$  là tương đối nguyên tố) và một không gian tại vị trí đó bằng cách khác.
30. Viết chương trình `PowersOfK.java` lấy đối số dòng lệnh  $k$  và in tất cả các số mũ của  $k$  trong kiểu `long`. Lưu ý: Giá trị `Long.MAX_VALUE` là giá trị kiểu `long` lớn nhất có thể có.
31. Tạo một điểm ngẫu nhiên  $(x, y, z)$  trên bề mặt quả cầu đơn vị bằng cách sử dụng phương pháp của Marsaglia: Chọn một điểm ngẫu nhiên  $(a, b)$  trong hình tròn đơn vị như đã nói ở trên. Sau đó đặt

$$x = 2a\sqrt{1 - a^2 - b^2}, y = 2b\sqrt{1 - a^2 - b^2}, z = 1 - 2(a^2 + b^2).$$

32. **(Taxi của Ramanujan).** Ramanujan là nhà toán học người Ấn Độ nổi tiếng với trực giác của ông với các con số. Một ngày, khi nhà toán học người Anh G.H. Hardy đến thăm ông tại bệnh viện, Hardy kể số xe taxi của mình là 1729, một con số khá buồn tẻ. Nhưng Ramanujan trả lời: "Không, Hardy! Không, Hardy! Đó là một con số rất thú vị. Đó là số nhỏ nhất là tổng hai số mũ 3 theo hai cách khác nhau." Kiểm chứng điều này bằng cách viết chương trình `Ramanujan.java` dùng đối số dòng lệnh  $N$  và in ra tất cả các số nguyên nhỏ hơn hoặc bằng  $N$  là tổng của hai số mũ 3 theo hai cách khác nhau - tức là tìm các số nguyên dương phân biệt  $a, b, c$  và  $d$  như vậy mà  $a^3 + b^3 = c^3 + d^3$ . Sử dụng bốn vòng lặp nhau.

Sau đó hãy chứng minh biển số xe 87539319 không phải là một con số buồn tẻ.

33. **(Tổng kiểm tra).** Số hiệu sách chuẩn quốc tế (ISBN) là mã số gồm 10 chữ số duy nhất cho mỗi cuốn sách. Chữ số tận cùng bên phải là chữ số **tổng kiểm tra** (*checksum*) được xác định duy nhất từ 9 chữ số còn lại từ điều kiện là

$$d_1 + 2d_2 + 3d_3 + \dots + 10d_{10} \equiv 0 \pmod{11}$$

Ở đây các chữ số được đánh số từ bên phải sang. Như vậy  $d_1$  có thể nhận giá trị từ 0 đến 10 (quy ước sử dụng ký tự X thay cho con số 10). Ví dụ: chữ số checksum tương ứng với 9 chữ số 020131452 số  $d_1$  sao cho

$$d_1 + 2 \times 2 + 3 \times 5 + 4 \times 4 + 5 \times 1 + 6 \times 3 + 7 \times 1 + 8 \times 0 + 9 \times 2 + 10 \times 0 \equiv 0 \pmod{11}$$

Viết chương trình `ISBN.java` sử dụng số nguyên có 9 chữ số từ dòng lệnh và in ra mã ISBN 10 chữ số có thêm chữ số checksum (0,1,...,9,X).

34. **(Lịch).** Viết chương trình `Calendar.java` lấy hai đối số dòng lệnh  $m$  (tháng) và  $y$  (năm) và in ra lịch hàng tháng cho tháng  $m$  của năm  $y$ . Ví dụ, đầu ra của bạn cho `java Calendar 2 2009` là

February 2009						
S	M	Tu	W	Th	F	S
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

Gợi ý: Xem chương trình `LeapYear.java` (đoạn mã 2.6) và `DayOfWeek.java` (bài tập 2.29).

35. **(Đếm số số nguyên tố)** Viết chương trình `PrimeCounter.java` sử dụng đối số trên dòng lệnh và in ra số các số nguyên tố nhỏ hơn  $N$  ( $< 10^7$ ). Lưu ý: Nếu bạn không cẩn thận, chương trình có thể không hoàn thành trong một khoảng thời gian hợp lý. Trong chương 5, chúng ta sẽ tìm hiểu một thuật toán hiệu quả hơn để thực hiện tính toán này, gọi là sàng *Eratosthenes*.
36. **(Fractal rồng)**. Viết chương trình `Dragon.java` sử dụng đối số dòng lệnh  $N$  và in ra hướng dẫn để vẽ một fractal rồng bậc  $N$ . Xem bài tập 2.35.

37. Điều gì xảy ra khi biên dịch đoạn mã sau đây?

```
double x;
if (a >= 0) x = 3.14;
if (a < 0) x = 2.71;
System.out.println(x);
```

38. **(Hàm mũ)**. Giả sử rằng  $x$  và  $sum$  là các biến kiểu `double`. Viết một đoạn mã để sử dụng khai triển Taylor để tính giá trị hàm mũ

$$e^x = 1 + x + x^2/2! + x^3/3! + x^4/4! + \dots$$

39. **(Hàm lượng giác)**. Viết hai chương trình `Sin.java` và `Cos.java` rằng tính  $\sin x$  và  $\cos x$  bằng cách sử dụng những khai triển Taylor

$$\sin x = x - x^3/3! + x^5/5! - x^7/7! + \dots$$

$$\cos x = 1 - x^2/2! + x^4/4! - x^6/6! + \dots$$

40. **(Thí nghiệm)**. Chạy thử nghiệm sau để so sánh thời gian chạy của hàm `Math.exp()` và ba phương pháp sau đây từ tập 3.38 để tính  $e^x$ : hai vòng lặp `for` lồng nhau, cải tiến với một vòng lặp `for`, cải tiến thêm với điều kiện dừng các phương pháp cải tiến với một đơn cho vòng lặp, và sau này với các điều kiện chấm dứt ( $\text{hạn} > 0$ ). Đối với mỗi phương pháp, thử xem chương trình có thể thực hiện bao nhiêu tính toán trong 10 giây.
41. **(Đi bộ ngẫu nhiên trên mặt phẳng)**. Viết chương trình mô phỏng hành vi của một hạt chuyển động trong một lưới ô vuông. Tại mỗi bước, hạt có thể di chuyển về phía bắc, nam, đông hoặc tây với xác suất bằng nhau và bằng  $1/4$ , độc lập với các bước đi trước đó. Xác định khoảng cách trung bình của hạt đó với điểm khởi đầu sau  $N$  bước. (So sánh với lý thuyết: tỉ lệ thuận với  $\sqrt{N}$ .)
42. **(Mô phỏng trò chơi)**. Trong chương trình năm 1970, trò chơi "Let's Make A Deal", một thí sinh được chọn trong ba cửa ra vào. Đằng sau một cánh cửa là một giải thưởng có giá trị, phía sau hai cánh cửa kia là món quà bất ngờ. Sau khi các thí sinh chọn một cánh cửa, người dẫn chương trình mở ra một trong hai cánh cửa khác (tất nhiên không tiết lộ giải thưởng). Sau đó các thí sinh được quyền chuyển sang cánh cửa chưa mở khác. Các thí sinh có nên làm như vậy? Bằng trực giác, có thể nói rằng cánh cửa thí sinh lựa chọn lúc đầu và cánh cửa chưa mở còn lại có khả năng chứa giải thưởng như nhau, vì vậy sẽ không có động lực để chuyển đổi. Viết một chương trình `MonteHall.java` để kiểm tra trực giác này bằng cách mô phỏng. Chương trình của bạn sử dụng tham số  $N$  từ dòng lệnh, chơi trò chơi  $N$  lần bằng cách sử dụng một trong hai chiến lược (chuyển đổi hoặc không chuyển đổi) và in các xác suất thành công của mỗi chiến lược.

43. **(Hỗn độn).** Viết chương trình nghiên cứu mô hình đơn giản tăng dân số đơn giản sau, mô hình này có thể được áp dụng để nghiên cứu về cá ở các ao hồ, vi khuẩn trong ống nghiệm, hoặc một loạt các tình huống tương tự. Giả sử dân số nằm trong khoảng từ 0 (tuyệt chủng) đến 1 (dân số tối đa có thể). Nếu dân số tại thời điểm  $t$  là  $x$ , thì dân số tại thời điểm  $t + 1$  là  $rx(1 - x)$ , trong đó  $r$  là tham số, đôi khi được gọi là tham số khả năng sinh sản, nó kiểm soát tốc độ tăng trưởng. Bắt đầu với một số nhỏ dân số ví dụ  $x = 0.01$  - hãy nghiên cứu kết quả của mô hình sau nhiều lần lặp, với các giá trị khác nhau của  $r$ . Với giá trị nào của  $r$  thì dân số ổn định ở  $x = 1 - 1/r$  ? Bạn có thể nói gì về dân số khi  $r$  là 3.5 ? 3.8 ? 5 ?
44. **(Giả thuyết Euler).** Năm 1769, Leonhard Euler đưa ra phiên bản tổng quát của Định lý Fermat, rằng cần ít nhất  $n$  số mũ bậc  $n$  để có tổng cũng là số mũ bậc  $n$ , khi  $n > 2$ . Viết chương trình bác bỏ giả thuyết Euler (tồn tại đến năm 1967), bằng cách sử dụng bốn vòng lặp lồng nhau để tìm bốn số nguyên dương có tổng các số mũ bậc 5 của chúng cũng là số mũ bậc năm. Tức là tìm  $a, b, c, d$ , và  $e$  sao cho  $a^5 + b^5 + c^5 + d^5 = e^5$ . Nhớ sử dụng kiểu long.





# Chương 4

## Mảng



## Chương 5

### Nhập xuất



## Chương 6

### Nghiên cứu tình huống: Thuật toán PageRank



# Phần II

## Hàm





## Phần III

### Lập trình hướng đối tượng



## Phần IV

### Cấu trúc dữ liệu và giải thuật

