

Bài giảng Lập trình căn bản với Java

Dựa trên sách và bài giảng của Robert Sedgewick và Kevin Wayne
Người dịch: Trần Quốc Long

Mục lục

I	Java căn bản	1
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	63
6	Nghiên cứu tình huống: Thuật toán PageRank	91
II	Hàm	93
III	Lập trình hướng đối tượng	95
IV	Cấu trúc dữ liệu và giải thuật	97

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

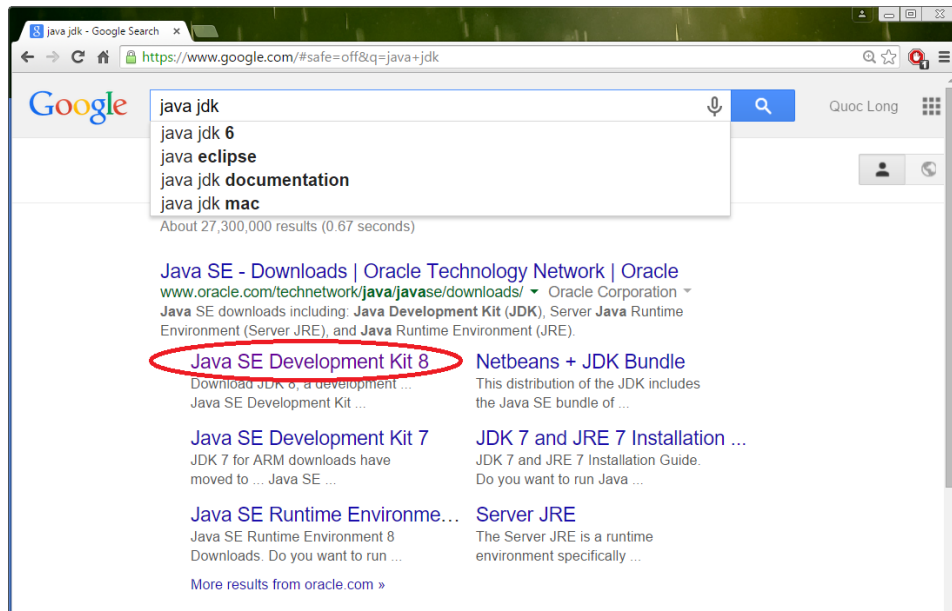
Trong chương này, chúng tôi sẽ dẫn dắt bạn vào thế giới lập trình Java bằng cách đưa bạn qua ba bước cơ bản cần thiết để tạo ra một chương trình đơn giản chạy được. Hệ thống Java bao gồm các ứng dụng không giống như bất kỳ các ứng dụng khác bạn đang sử dụng (như soạn thảo văn bản, e-mail, hoặc trình 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. 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

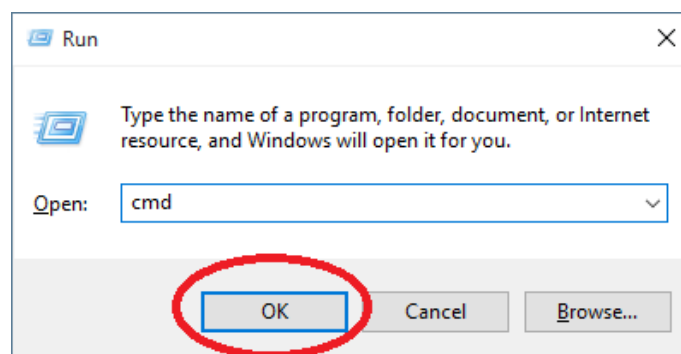
`C:\Program Files (x86)\Java\jdk1.8.0_60`. Như vậy là bạn đã cài được trình biên dịch JDK.

Đường dẫn PATH. Để sử dụng Java dễ dàng, bạn cần đưa thư mục

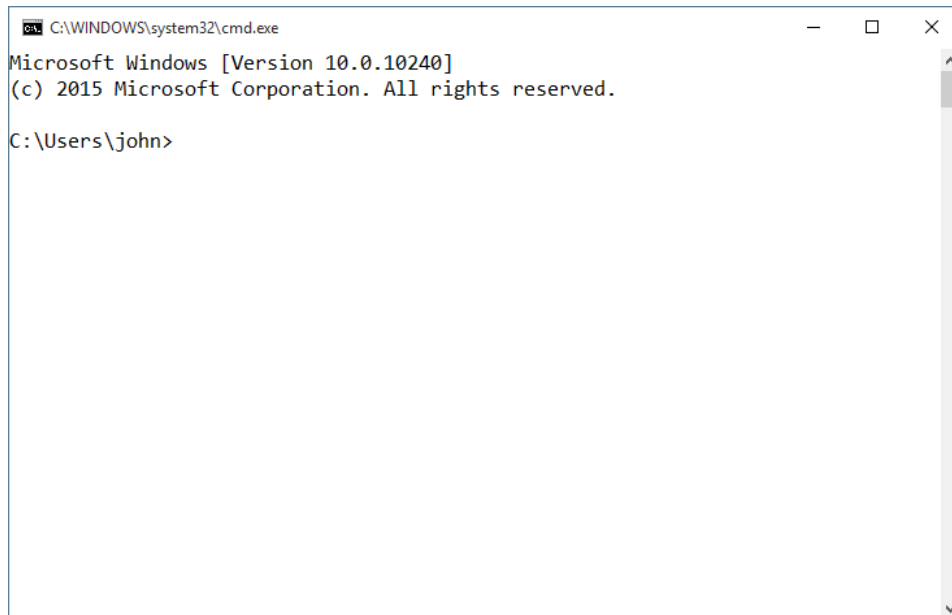
`C:\Program Files (x86)\Java\jdk1.8.0_60\bin` vào biến môi trường PATH của Windows đồng thời đặt biến môi trường `JAVA_HOME` bằng đường dẫn đến Java, `C:\Program Files (x86)\Java\jdk1.8.0_60`.

Tùy theo phiên bản Windows bạn đang sử dụng sẽ có cách riêng để thay đổi các biến môi trường này. Hãy tra cứu qua Internet !!!

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  +  và gõ vào phần Open đoạn lệnh `cmd` rồi ấn .



Bạn sẽ thấy một cửa sổ như sau

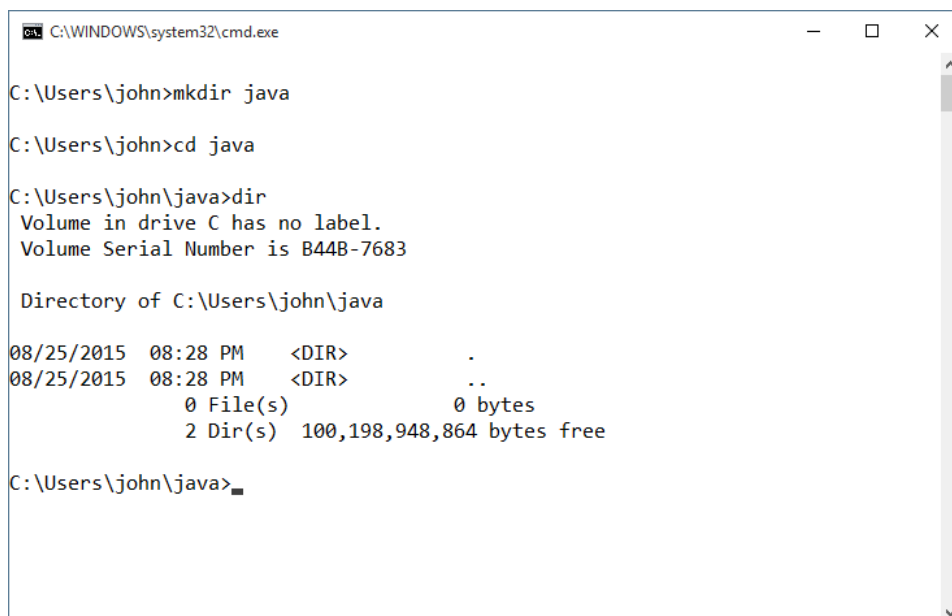


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\john>
```

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. Ổ 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:\WINDOWS\system32\cmd.exe

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

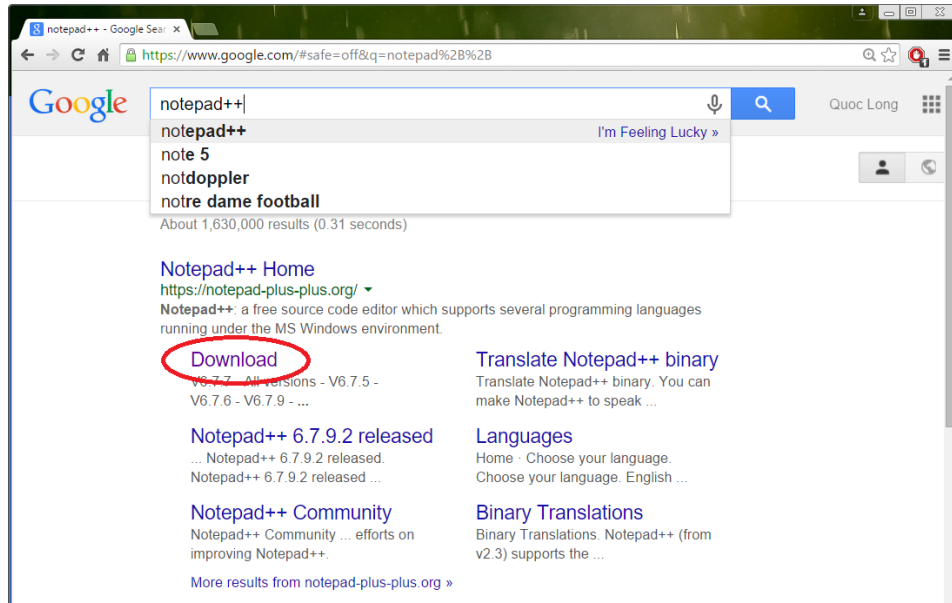
Directory of C:\Users\john\java

08/25/2015  08:28 PM    <DIR>          .
08/25/2015  08:28 PM    <DIR>          ..
               0 File(s)                0 bytes
               2 Dir(s) 100,198,948,864 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 π . Hoặc ta có thể dùng một chuỗi ký hiệu khoa học, ví dụ như, 6.022E23 cho hằng số Avogadro 6.022×10^{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 2π .

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.

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

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 θ 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, θ) . 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 φ và kinh tuyến λ 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 λ_0 , φ và λ 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
```

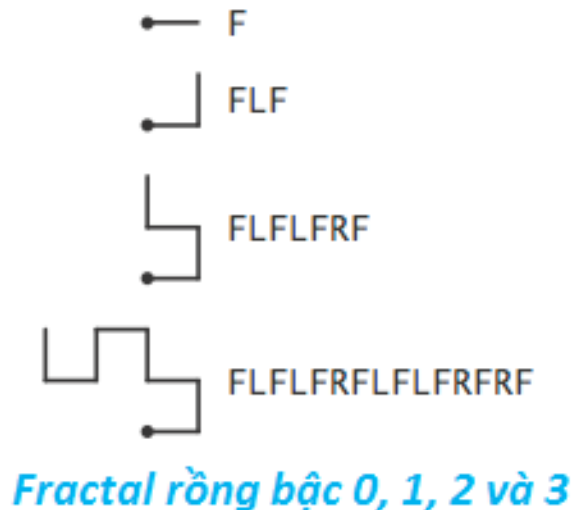
33. Viết chương trình `GreatCircle.java` tính khoảng cách thực (theo hình cầu) giữa 2 điểm có vĩ độ, kinh độ (x_1, y_1) và (x_2, y_2) (nhập từ dòng lệnh). Công thức tính như sau

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

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 < 0) x = -x;</code>
Giá trị lớn nhất	<code>if (x > y) max = x; else max = y;</code>
Chia trường hợp tính thuế	<code>if (income < 47450) rate = .22; else if (income < 114650) rate = .25; else if (income < 174700) rate = .28; else if (income < 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 < 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

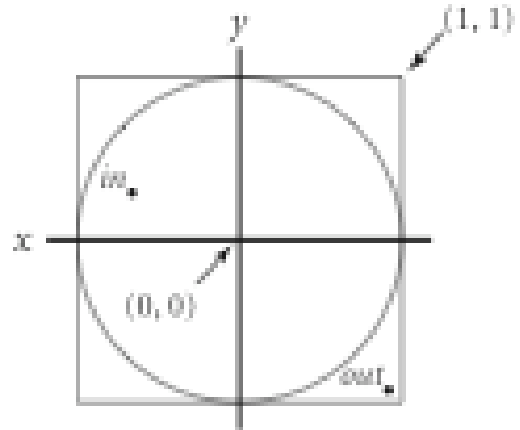
```
1 public class PowersOfTwo {
2     public static void main(String[] args) {
3
4         // read in one command-line argument
5         int N = Integer.parseInt(args[0]);
6
7         int i = 0;                // count from 0 to N
8         int powerOfTwo = 1;       // the ith power of two
9
10        // 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 nãyay 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

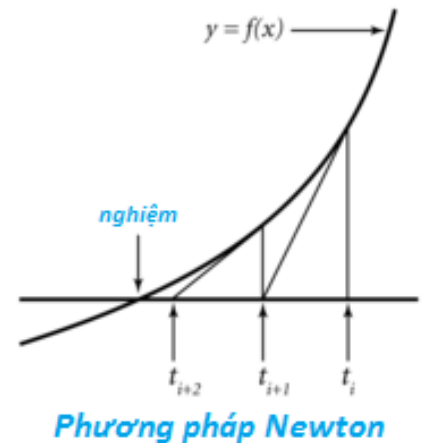
```
1 // print the ruler function
2 String ruler = " ";
3 for (int i = 1; i <= N; i++) {
4     ruler = ruler + i + ruler;
5 }
6 System.out.println(ruler);
7 System.out.println();
```

```

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 }

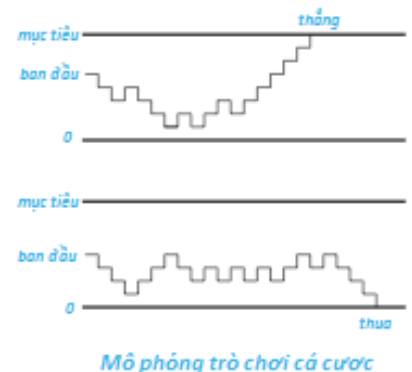
```

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 `Ctrl` + `C`.

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 Δ 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 π). 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 4×4 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.

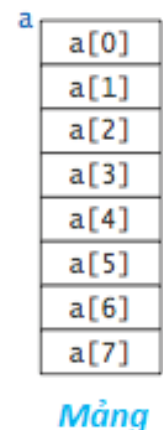
Chương 4

Mảng

Mục lục chương

4.1	Đoạn mã sử dụng mảng điển hình	45
4.2	Lập trình với mảng	46
4.3	Ví dụ	48
4.4	Hỏi đáp	56
4.5	Bài tập	56

Trong chương này, chúng ta tìm hiểu một cấu trúc cơ bản được gọi là mảng (array). Một mảng lưu trữ một chuỗi các giá trị có cùng kiểu. Chúng ta không chỉ lưu trữ các giá trị mà còn muốn nhanh chóng truy cập từng giá trị đơn lẻ. Cách mà chúng ta truy cập các giá trị đơn lẻ là đánh số và sử dụng chỉ số. Trong Java, các chỉ số mảng được đánh số từ 0, nếu mảng có N giá trị, các chỉ số chạy từ 0 đến N-1.



4.1 Đoạn mã sử dụng mảng điển hình

Hai véc-tơ có cùng độ dài, tích vô hướng của chúng là tổng các tích của các giá trị tương ứng của chúng. Nếu chúng ta biểu diễn 2 véc-tơ bằng các mảng 1 chiều `x[]` và `y[]` có độ dài N với giá trị kiểu `double`, tích vô hướng của chúng có thể tính như sau

```
double sum = 0.0;
for (int i = 0; i < N; i++)
    sum += x[i]*y[i];
```

Ví dụ sau chạy đoạn mã trên cho 2 véc-tơ có độ dài bằng 3

i	x[i]	y[i]	x[i]*y[i]	sum
				0
0	.30	.50	.15	.15
1	.60	.10	.06	.21
2	.10	.40	.04	.25
				.25

Đoạn mã 4.1: Arrays.java sinh N số ngẫu nhiên

```

1 // initialize to random values between 0 and 1
2 double[] a = new double[N];
3 for (int i = 0; i < N; i++) {
4     a[i] = Math.random();
5 }

```

Đoạn mã 4.2: Arrays.java in mỗi số trên 1 dòng

```

1 // print array values, one per line
2 System.out.println("a[]");
3 System.out.println("-----");
4 for (int i = 0; i < N; i++) {
5     System.out.println(a[i]);
6 }
7 System.out.println();
8 System.out.println("a = " + a);
9 System.out.println();

```

Các đoạn mã 4.1 – 4.6 minh họa một loạt cách dùng mảng hay gặp.

4.2 Lập trình với mảng

Trước khi xem xét thêm ví dụ, chúng ta hãy xem xét các đặc điểm quan trọng của lập trình với mảng.

- *Chỉ số bắt đầu từ 0.* Phần tử đầu tiên của mảng `a[]` là luôn `a[0]`, phần tử thứ hai là `a[1]`, v.v...Có vẻ tự nhiên hơn nếu `a[1]` là phần tử đầu tiên, phần tử thứ hai là `a[2]`, v.v..., nhưng bắt đầu với chỉ số 0 có một số lợi thế và đã trở thành quy ước được sử dụng trong hầu hết các ngôn ngữ lập trình hiện đại.
- *Độ dài.* Khi chúng ta tạo một mảng, kích thước của nó đã cố định. Bạn có thể truy xuất chiều dài của mảng `a[]` bằng `a.length`.
- *Cấp phát bộ nhớ.* Khi bạn sử dụng `new` để tạo ra mảng mới, Java dành không gian trong bộ nhớ cho nó (và khởi tạo các giá trị). Quá trình này được gọi là cấp phát bộ nhớ.
- *Kiểm tra biên.* Khi lập trình với mảng, bạn phải cẩn thận. Trách nhiệm của bạn là sử dụng các chỉ số hợp lệ trong khoảng `[0, a.length-1]` để truy xuất một phần tử của mảng `a[]`.

Đoạn mã 4.3: Arrays.java tìm giá trị lớn nhất

```

1 // find the maximum
2 double max = Double.NEGATIVE_INFINITY;
3 for (int i = 0; i < N; i++) {
4     if (a[i] > max) max = a[i];
5 }
6 System.out.println("max = " + max);

```

Đoạn mã 4.4: Arrays.java tìm giá trị trung bình

```
1 // average
2 double sum = 0.0;
3 for (int i = 0; i < N; i++) {
4     sum += a[i];
5 }
6 System.out.println("average = " + sum / N);
```

Đoạn mã 4.5: Arrays.java sao chép mảng

```
1 // copy to another array
2 double[] b = new double[N];
3 for (int i = 0; i < N; i++) {
4     b[i] = a[i];
5 }
```

Đoạn mã 4.6: Arrays.java đảo ngược mảng

```
1 // reverse the order
2 for (int i = 0; i < N/2; i++) {
3     double temp = b[i];
4     b[i] = b[N-i-1];
5     b[N-i-1] = temp;
6 }
```

- *Đặt giá trị mảng tại thời gian biên dịch.* Khi chúng ta có một số lượng nhỏ các giá trị lưu trữ trong mảng, chúng ta có thể khởi tạo mảng bằng cách liệt kê các giá trị trong dấu ngoặc nhọn, cách nhau bởi dấu phẩy. Ví dụ, chúng ta có thể sử dụng đoạn mã sau đây trong chương trình quản lý các quân bài tứ lơ khơ.

```
String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
String[] rank = { "2", "3", "4", "5", "6", "7", "8", "9", "10",
    "Jack", "Queen", "King", "Ace"
};
```

Sau khi khởi tạo 2 mảng trên, đoạn mã sau in ra một quân bài ngẫu nhiên

```
int i = (int) (Math.random() * rank.length);
int j = (int) (Math.random() * suit.length);
System.out.println(rank[i] + " of " + suit[j]);
```

- *Đặt giá trị mảng tại thời gian chạy.* Một tình huống điển hình hơn là khi chúng ta muốn lưu trữ các giá trị tính toán được trong một mảng. Ví dụ, chúng ta có thể sử dụng đoạn mã sau để khởi tạo một mảng có kích thước 52 đại diện cho một bộ bài chơi, sử dụng các mảng `rank[]` và `suit[]` ở trên.

```
String[] deck = new String[ranks.length * suits.length];
for (int i = 0; i < ranks.length; i++)
    for (int j = 0; j < suits.length; j++) {
        deck[suits.length*i + j] = rank[i] + " of " + suit[j];
        System.out.println(rank[i] + " of " + suit[j]);
    }
```

4.3 Ví dụ

Tráo đổi và lấy mẫu trên mảng. Bây giờ chúng ta mô tả một số thuật toán thường dùng để sắp xếp lại các phần tử trong mảng.

- *Tráo đổi.* Chúng tôi sẽ hay phải tráo đổi hai giá trị trong một mảng. Tiếp tục ví dụ của chúng ta với các quân bài, đoạn mã sau đây sẽ trao đổi quân bài ở vị trí `i` với quân bài ở vị trí `j`.

```
String t = deck[i];
deck[i] = deck[j];
deck[j] = t;
```

- *Xáo trộn ngẫu nhiên.* Đoạn mã sau đây tráo các quân bài trong mảng `deck[]`.

```
int N = deck.length;
for (int i = 0; i < N; i++)
{
    int r = i + (int) (Math.random() * (N-i));
    String t = deck[r];
    deck[r] = deck[i];
    deck[i] = t;
}
```

Đoạn mã này chạy từ trái qua phải mảng `deck[]`, với mỗi vị trí `i` ta chọn một quân bài ngẫu nhiên `deck[r]` nằm trong khoảng từ `deck[i]` đến `deck[N-1]` rồi tráo đổi với quân bài `deck[i]`.

- *Sử dụng hằng số.* Các giá trị như 4, 13 và 52 nên được đưa vào các biến hằng số vì các giá trị này có xu hướng rải khắp chương trình của bạn, làm cho chương trình khó bảo trì. Ta nên ra một hằng có tên đầy đủ ý nghĩa cho mỗi con số và sử dụng chúng nhất quán trong chương trình. Chương trình `Deck.java` (đoạn mã 4.7) viết lại các ví dụ trước, nhưng dùng hằng số.
- *Lấy mẫu không trả lại.* Trong nhiều tình huống, chúng ta muốn lấy một mẫu ngẫu nhiên các phần tử của mảng sao cho mỗi phần tử xuất hiện nhiều nhất một lần. Chương trình `Sample.java` (đoạn mã 4.8) nhận hai đối số dòng lệnh `M` và `N`, và tạo ra một chỉnh hợp chập `M` của `N` số trong một mảng cho trước.

Rút gọn các đoạn mã lặp đi lặp lại. Một ứng dụng đơn giản của mảng là rút ngắn và đơn giản hóa các đoạn mã lặp đi lặp lại. Ví dụ trong đoạn mã sau đó in ra tên rút gọn của một tháng nhất định sử dụng con số chỉ tháng (1 là Jan, 2 là Feb, v.v...).

```
if (m == 1) System.out.println("Jan");
else if (m == 2) System.out.println("Feb");
else if (m == 3) System.out.println("Mar");
else if (m == 4) System.out.println("Apr");
else if (m == 5) System.out.println("May");
else if (m == 6) System.out.println("Jun");
else if (m == 7) System.out.println("Jul");
else if (m == 8) System.out.println("Aug");
else if (m == 9) System.out.println("Sep");
else if (m == 10) System.out.println("Oct");
else if (m == 11) System.out.println("Nov");
else if (m == 12) System.out.println("Dec");
```

Chúng ta có thể rút gọn đoạn mã trên bằng cách sử dụng mảng các xâu chứa tên của tháng

```
String[] months = { "", "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
};
System.out.println(months[m]);
```

Kỹ thuật này đặc biệt hữu ích nếu bạn cần truy xuất vào tên của một tháng bằng con số của nó ở nhiều nơi khác nhau trong chương trình của bạn. Lưu ý rằng chúng ta cố ý lãng phí một phần tử trong mảng (phần tử 0) để `months[1]` tương ứng với tháng "Jan", v.v...

Lưu trữ giá trị đã tính toán. Một ứng dụng khác của mảng là lưu trữ các giá trị bạn đã tính toán, để sử dụng sau. Ví dụ, giả sử rằng bạn đang viết một chương trình tính toán có sử dụng các giá trị Harmonic: $1/1 + 1/2 + 1/3 + \dots$. Bạn có thể lưu trữ những giá trị ban đầu của chuỗi trong mảng như sau

```
double[] H = new double[N];
for (int i = 1; i < N; i++)
    H[i] = H[i-1] + 1.0/i;
```

và sau đó chỉ cần sử dụng `H[i]` để truy xuất giá trị chuỗi số bất kỳ. Tính toán trước giá trị theo cách trên là một ví dụ về việc đánh đổi không gian lấy thời gian tính toán: bằng cách đầu tư không gian (mảng để lưu các giá trị), chúng ta tiết kiệm thời gian (vì chúng ta không cần phải tính toán lại chúng). Phương pháp này không hiệu quả nếu chúng ta chỉ cần các giá trị chuỗi với `N` rất lớn, nhưng nó rất hiệu quả nếu chúng ta cần một số lượng lớn các giá trị của chuỗi.

Đoạn mã 4.7: Deck.java

```
1 public class Deck {
2     public static void main(String[] args) {
3         String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
4         String[] rank = { "2", "3", "4", "5", "6", "7", "8", "9",
5             "10", "Jack", "Queen", "King", "Ace" };
6
7         // avoid hardwired constants
8         int SUITS = suit.length;
9         int RANKS = rank.length;
10        int N = SUITS * RANKS;
11
12        // initialize deck
13        String[] deck = new String[N];
14        for (int i = 0; i < RANKS; i++) {
15            for (int j = 0; j < SUITS; j++) {
16                deck[SUITS*i + j] = rank[i] + " of " + suit[j];
17            }
18        }
19
20        // shuffle
21        for (int i = 0; i < N; i++) {
22            int r = i + (int) (Math.random() * (N-i));
23            String t = deck[r];
24            deck[r] = deck[i];
25            deck[i] = t;
26        }
27
28        // print shuffled deck
29        for (int i = 0; i < N; i++) {
30            System.out.println(deck[i]);
31        }
32    }
33 }
```

Đoạn mã 4.8: Sample.java

```
1 public class Sample {
2     public static void main(String[] args) {
3         int M = Integer.parseInt(args[0]);    // choose this many
4         int N = Integer.parseInt(args[1]);    // from 0, 1, ..., N-1
5
6         // create permutation 0, 1, ..., N-1
7         int[] perm = new int[N];
8         for (int i = 0; i < N; i++)
9             perm[i] = i;
10
11        // create random sample in perm[0], perm[1], ..., perm[M-1]
12        for (int i = 0; i < M; i++) {
13
14            // random integer between i and N-1
15            int r = i + (int) (Math.random() * (N-i));
16
17            // swap elements at indices i and r
18            int t = perm[r];
19            perm[r] = perm[i];
20            perm[i] = t;
21        }
22
23        // print results
24        for (int i = 0; i < M; i++)
25            System.out.print(perm[i] + " ");
26        System.out.println();
27    }
28 }
```

```

1 public class CouponCollector {
2     public static void main(String[] args) {
3         int N = Integer.parseInt(args[0]);    // number of card types
4         boolean[] found = new boolean[N];    // found[i] = true if
5         card i has been collected            // total number of cards
6         int cardcnt = 0;                      collected
7         int valcnt = 0;                      // number of distinct
8         cards
9         // repeatedly choose a random card and check whether it's a
10        new one
11        while (valcnt < N) {
12            int val = (int) (Math.random() * N);    // random card
13            between 0 and N-1
14            cardcnt++;                             // we collected one
15            more card
16            if (!found[val]) valcnt++;             // it's a new card
17            type
18            found[val] = true;                     // update found[]
19        }
20
21        // print the total number of cards collected
22        System.out.println(cardcnt);
23    }
24 }

```

Sưu tập phiếu trúng thưởng. Giả sử bạn có một bộ bài úp xuống, lần lượt bạn lật từng quân bài lên. Hỏi bạn cần lật bao nhiêu thẻ để mỗi chất đều có một quân bài được lật. Bài toán tổng quát chính là việc sưu tập phiếu trúng thưởng (ví dụ, dưới nắp chai nước ngọt). Giả sử một công ti phát hành phiếu trúng thưởng với N loại phiếu, hỏi bạn cần phải thu thập bao nhiêu phiếu để mỗi loại bạn có ít nhất một phiếu. Chương trình `CouponCollector.java` (đoạn mã 4.9) là một ví dụ mô phỏng quá trình này.



Sưu tập phiếu

Sàng Eratosthenes. Hàm đếm số số nguyên tố $\pi(N)$ là số các số nguyên tố nhỏ hơn hoặc bằng N . Ví dụ $\pi(17) = 7$ vì bảy số nguyên tố đầu tiên là 2, 3, 5, 7, 11, 13, và 17. Chương trình `PrimeSieve.java` (đoạn mã 4.10) dùng đối số N từ dòng lệnh và tính $\pi(N)$ bằng cách sử dụng sàng Eratosthenes.

```

% java PrimeSieve 25
The number of primes <= 25 is 9

% java PrimeSieve 100

```


Đoạn mã 4.10: PrimeSieve.java

```
1 public class PrimeSieve {
2     public static void main(String[] args) {
3         int N = Integer.parseInt(args[0]);
4
5         // initially assume all integers are prime
6         boolean[] isPrime = new boolean[N + 1];
7         for (int i = 2; i <= N; i++) {
8             isPrime[i] = true;
9         }
10
11        // mark non-primes <= N using Sieve of Eratosthenes
12        for (int i = 2; i*i <= N; i++) {
13
14            // if i is prime, then mark multiples of i as nonprime
15            // suffices to consider multiples i, i+1, ..., N/i
16            if (isPrime[i]) {
17                for (int j = i; i*j <= N; j++) {
18                    isPrime[i*j] = false;
19                }
20            }
21        }
22
23        // count primes
24        int primes = 0;
25        for (int i = 2; i <= N; i++) {
26            if (isPrime[i]) primes++;
27        }
28        System.out.println("The number of primes <= " + N + " is " +
29                           primes);
30    }
```

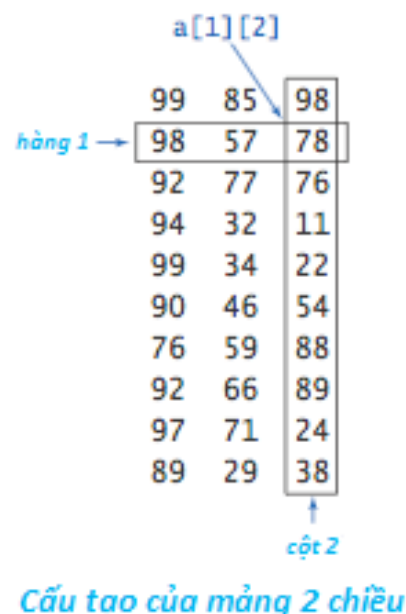
```
The number of primes <= 100 is 25

% java -Xmx100m PrimeSieve 100000000
The number of primes <= 100000000 is 5761455

% java PrimeSieve -Xmx1100m 1000000000
The number of primes <= 1000000000 is 50847534
```

Các câu lệnh `java -Xmx100m` và `java -Xmx1100m` xin cấp phát bộ nhớ 100MB và 1100MB cho chương trình `PrimeSieve`.

Mảng 2 chiều. Trong nhiều ứng dụng, ta cần tổ chức thông tin dưới dạng một bảng hình chữ nhật các con số và truy xuất đến các hàng và cột trong bảng. Trong toán học, bảng số tương ứng với **ma trận** và cấu trúc tương ứng trong mảng hai chiều.



- Khai báo mảng 2 chiều

```
double [][] a = new double[M][N];
```

- Truy xuất phần tử dòng i cột j , ta dùng `a[i][j]`.
- Khởi tạo (mặc định bằng 0), hoặc sử dụng vòng lặp lồng nhau

```
double [][] a;
a = new double[M][N];
for (int i = 0; i < M; i++)
    for (int j = 0; j < N; j++)
        a[i][j] = 0;
```

- Biểu diễn trong bộ nhớ: Java biểu diễn mảng là mảng của các mảng. Mảng 2 chiều có M dòng, N cột là M mảng, mỗi phần tử là một mảng gồm N phần tử. Như vậy ta có thể dùng `a[i]` để trỏ đến dòng thứ i của mảng 2 chiều. Hơn nữa, cách biểu diễn này cho phép các dòng không cần có độ dài bằng nhau.
- Đặt giá trị khi biên dịch: Đoạn mã sau khởi tạo mảng `a[][]` gồm các phần tử đã biết giá trị trước

```
int[][] a = {
    { 99, 85, 98 },
    { 98, 57, 78 },
    { 92, 77, 76 },
    { 94, 32, 11 },
    { 99, 34, 22 },
    { 90, 46, 54 },
    { 76, 59, 88 },
    { 92, 66, 89 },
    { 97, 71, 24 },
    { 89, 29, 38 }
};
```

- Mảng nhiều chiều có độ dài các dòng khác nhau, khi đó ta sử dụng `a[i].length` để truy xuất số phần tử của dòng thứ `i`.

```
for (int i = 0; i < a.length; i++) {
    for (int j = 0; j < a[i].length; j++) {
        System.out.print(a[i][j] + " ");
    }
    System.out.println();
}
```

- Mảng nhiều hơn 2 chiều: Tổng quát hóa lên ta có mảng nhiều chiều với số ngoặc vuông tương ứng.

```
double[][][] a = new double[N][N][N];
```

và ta truy xuất phần tử với mã `a[i][j][k]`.

Các phép toán với ma trận

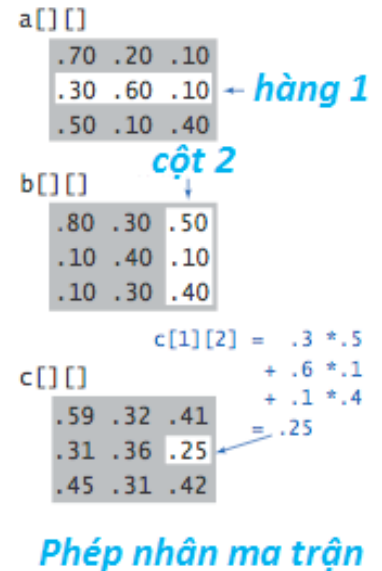
- Phép cộng ma trận.* Ứng dụng điển hình trong tính toán khoa học là phép cộng ma trận như sau

```
double[][] c = new double[N][N];
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        c[i][j] = a[i][j] + b[i][j];
    }
}
```

- Phép nhân ma trận.*

Tích của hai ma trận `a[] []` và `b[] []` là ma trận `c[] []` có `c[i][j]` là tích vô hướng của dòng `i` của `a[] []` với cột của `b[] []`.

```
double[] [] c = new double[N][N];
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        for (int k = 0; k < N; k++) {
            c[i][j] += a[i][k]*b[k][j];
        }
    }
}
```



Bước nhảy ngẫu nhiên không lặp lại trên bảng. Hãy nghiên cứu chương trình `SelfAvoidingWalk.java` (đoạn mã 4.11). Chương trình này tính tỉ lệ số lần `N` bước nhảy ngẫu nhiên trên mặt phẳng không bị lặp lại. Để hiểu rõ hơn hãy làm bài tập 3.41.

4.4 Hỏi đáp

Hỏi. Trong Java tôi thấy hai cách khai báo mảng `int a[]` và `int []a`. Có gì khác biệt không ?

Đáp. Cả hai cách đều hợp lệ và tương đương nhau. Cách đầu bắt nguồn từ ngôn ngữ C. Cách thứ hai của Java chỉ rõ đây là khai báo mảng.

Hỏi. Tại sao chỉ số của mảng bắt đầu từ 0 thay vì 1?

Đáp. Quy ước này có nguồn gốc từ các chương trình ngôn ngữ máy với địa chỉ của phần tử mảng được tính bằng cách cộng chỉ số với địa chỉ của phần tử đầu tiên. Nếu ta bắt đầu từ chỉ số 1 sẽ lãng phí phần tử ở đầu mảng hoặc mất thời gian trừ đi 1.

Hỏi. Điều gì xảy ra nếu tôi sử dụng một số âm làm chỉ số một mảng?

Đáp. Cũng giống như khi bạn sử dụng chỉ số quá lớn. Bất cứ khi nào chương trình dùng chỉ số mảng không nằm giữa 0 và chiều dài mảng trừ đi một, Java sẽ sinh ngoại lệ `ArrayIndexOutOfBoundsException` và chấm dứt chương trình.

Hỏi. Còn lưu ý nào khác khi tôi sử dụng mảng?

Đáp. Bạn nên nhớ Java luôn khởi tạo mảng khi bạn tạo ra chúng, do đó tạo ra một mảng cần thời gian tỉ lệ thuận với kích thước của mảng.

Hỏi. Nếu `a[]` là một mảng, tại sao `System.out.println(a)` in ra một số nguyên hệ thập lục phân, ví dụ như `@f62373`, thay vì các phần tử của mảng?

Đáp. Câu hỏi hay. Java in địa chỉ trong bộ nhớ của mảng. Tuy nhiên, khác với C, trong Java hiếm khi bạn cần địa chỉ này.

4.5 Bài tập

- Viết chương trình khai báo và khởi tạo một mảng `a[]` kích thước 1000 và truy cập phần tử `a[1000]`. Có lỗi gì khi biên dịch chương trình không ? Điều gì xảy ra khi bạn chạy nó ?
- Mô tả và giải thích những gì sẽ xảy ra khi bạn biên dịch chương trình `HugeArray.java` (đoạn mã 4.12).

```

1 public class SelfAvoidingWalk {
2     public static void main(String[] args) {
3         int N = Integer.parseInt(args[0]);           // lattice size
4         int T = Integer.parseInt(args[1]);           // number of trials
5         int deadEnds = 0;                             // trials resulting in a dead end
6
7         // simulate T self-avoiding walks
8         for (int t = 0; t < T; t++) {
9             boolean[][] a = new boolean[N][N];       // intersections
10             visited
11             int x = N/2, y = N/2;                     // current position
12
13             // repeatedly take a random step, unless you've already
14             // escaped
15             while (x > 0 && x < N-1 && y > 0 && y < N-1) {
16
17                 // dead-end, so break out of loop
18                 if (a[x-1][y] && a[x+1][y] && a[x][y-1] && a[x][y+1]) {
19                     deadEnds++;
20                     break;
21                 }
22
23                 // mark (x, y) as visited
24                 a[x][y] = true;
25
26                 // take a random step to unvisited neighbor
27                 double r = Math.random();
28                 if (r < 0.25) {
29                     if (!a[x+1][y])
30                         x++;
31                 }
32                 else if (r < 0.50) {
33                     if (!a[x-1][y])
34                         x--;
35                 }
36                 else if (r < 0.75) {
37                     if (!a[x][y+1])
38                         y++;
39                 }
40                 else if (r < 1.00) {
41                     if (!a[x][y-1])
42                         y--;
43                 }
44             }
45
46             System.out.println(100*deadEnds/T + "% dead ends");
47         }
48     }
49 }

```

```

1 public class HugeArray {
2
3     public static void main(String[] args) {
4         int N = 1000;
5         int[] a = new int[N*N*N*N];
6         System.out.println(a.length);
7     }
8 }

```

3. Cho hai vectơ có chiều dài N biểu diễn bằng mảng một chiều, viết một đoạn mã tính khoảng cách Euclide giữa chúng (căn bậc hai của tổng bình phương các phần tử tương ứng).
4. Viết đoạn mã đảo ngược thứ tự của mảng một chiều `a[]` gồm các giá trị `double`. Cố gắng không tạo thêm mảng để giữ được kết quả. Gợi ý: Sử dụng đoạn mã trao đổi hai phần tử.
5. Có gì sai với đoạn mã sau đây ?

```

int[] a;
for (int i = 0; i < 10; i++)
    a[i] = i * i;

```

6. Viết đoạn mã đó in nội dung của một mảng boolean hai chiều, sử dụng `*` đại diện cho `true` và dấu cách đại diện cho `false`. Nhớ in cả chỉ số hàng và cột.
7. Đoạn mã sau đây in ra những gì?

```

int [] a = new int[10];
for (int i = 0; i < 10; i++)
    a[i] = 9 - i;
for (int i = 0; i < 10; i++)
    a[i] = a[a[i]];
for (int i = 0; i < 10; i++)
    System.out.println(a[i]);

```

8. Đoạn mã sau đây đưa giá trị nào vào mảng `a[]`?

```

int N = 10;
int[] a = new int[N];
a[0] = 0;
a[1] = 1;
for (int i = 2; i < N; i++)
    a[i] = a[i-1] + a[i-2];

```

9. Đoạn mã sau đây in ra những gì?

```

int[] a = { 1, 2, 3 };
int[] b = { 1, 2, 3 };
System.out.println(a == b);

```

10. Viết chương trình `Deal.java` chia bài, dùng đối số dòng lệnh `N` và in `N` bộ bài ngẫu nhiên (5 quân mỗi bộ) từ một bộ tứ lơ khơ.

11. Viết đoạn mã tạo ra một mảng hai chiều `b[][]` là bản sao của một mảng hai chiều hiện có `a[][]`, theo các giả định sau đây:
- `a[][]` là mảng vuông.
 - `a[][]` là mảng chữ nhật.
 - `a[][]` có các dòng có độ dài khác nhau.
12. Viết đoạn mã in chuyển vị (thay đổi hàng thành cột) của một bảng điểm (mỗi môn học có một số điểm thành phần). Ví dụ sau (mỗi môn học có 3 điểm thành phần)

```
int [][] a = {
    { 99, 85, 98 },
    { 98, 57, 78 },
    { 92, 77, 76 },
    { 94, 32, 11 },
    { 99, 34, 22 },
    { 90, 46, 54 },
    { 76, 59, 88 },
    { 92, 66, 89 },
    { 97, 71, 24 },
    { 89, 29, 38 }
};
```

mã của bạn sẽ phải in ra như sau:

```
99 98 92 94 99 90 76 92 97 89
85 57 77 32 34 46 59 66 71 29
98 78 76 11 22 54 88 89 24 38
```

13. Viết đoạn mã chuyển vị mảng vuông tại chỗ mà không cần tạo một mảng thứ hai.
14. Viết chương trình sử dụng số nguyên `N` từ dòng lệnh và tạo ra một mảng `boolean` `N x N` `a[][]` như vậy mà `a[i][j]` là `true` nếu i và j là nguyên tố cùng nhau (không có ước số chung lớn hơn 1), và `false` nếu ngược lại.
15. Viết chương trình tính tích của hai ma trận `boolean` vuông, sử dụng phép toán OR (`||`) thay cho phép cộng (+) và phép AND (`&&`) thay cho phép nhân (*).
16. Xem bài tập 12, hãy viết đoạn mã tính điểm trung bình các môn học từ bảng điểm. Biết rằng các điểm thành phần có trọng số ghi trong mảng `weights[]`. Với ví dụ trên, nếu ta có
- ```
double[] weights = { .25, .25, .50 };
```
- thì điểm cuối cùng sẽ có trọng số gấp đôi các điểm còn lại.
17. Viết đoạn mã tính tích 2 ma trận chữ nhật. Để phép nhân có nghĩa, số cột của ma trận đầu phải bằng số dòng của ma trận thứ hai để các phép tính tích vô hướng hợp lệ. In thông báo lỗi nếu hai ma trận không hợp lệ.
18. Sửa chương trình `SelfAvoidingWalk.java` (đoạn mã 4.11) để tính độ dài trung bình cũng như xác suất không thành công (đường cụt, không thể đi hết `N` bước).
19. Sửa chương trình `SelfAvoidingWalk.java` (đoạn mã 4.11) để tính toán và in ra diện tích trung bình của hình chữ nhật nhỏ nhất có cạnh song song với các hàng và cột và chứa đường đi vừa đi qua. Thống kê riêng cho đường đi thành công đầy đủ `N` bước và đường cụt.

20. **Mô phỏng xúc sắc.** Đoạn mã sau tính toán phân bố xác suất tổng điểm của 2 lần tung

```
double[] dist = new double[13];
for (int i = 1; i <= 6; i++)
 for (int j = 1; j <= 6; j++)
 dist[i+j] += 1.0;

for (int k = 1; k <= 12; k++)
 dist[k] /= 36.0;
```

Giá trị `dist[k]` là xác suất tổng 2 con xúc sắc bằng `k`. Hãy viết chương trình mô phỏng `N` lần tung 2 con xúc sắc (`N` từ dòng lệnh) và tính tần suất xuất hiện của tổng 2 lần tung. So sánh các con số thu được với mảng `dist[k]` ở trên.

21. **Vùng cao nguyên dài nhất.** Cho một mảng các số nguyên, tìm ra vị trí và độ dài của đoạn số liên tục dài nhất sao cho: (1) Các số trong đoạn bằng nhau và (2) các số sát bên trái và bên phải nhỏ hơn các số trong đoạn.
22. Viết chương trình `ShufflingTest.java` kiểm tra đoạn mã tráo bài ở trên bằng các mô phỏng `N` lần tráo bộ bài có `M` quân bài. Sau đó tính mảng `p[][]` có kích thước `M x M` sao cho `p[i][j]` là số lần quân bài `i` ở vị trí `j`. Giá trị này phải gần bằng `N / M`.
23. **Tráo bài tồi.** Giả sử rằng bạn chọn một số nguyên ngẫu nhiên giữa 0 và `N-1` trong đoạn mã tráo bài thay vì một số giữa `i` và `N-1`. Hãy cho thấy xác suất các kết quả tráo bài không bằng nhau.
24. **Nghe nhạc chế độ ngẫu nhiên.** Bạn đặt máy nghe nhạc của bạn sang chế độ chọn bài ngẫu nhiên. Nó sẽ phát ngẫu nhiên `N` bài hát trước khi lặp lại một cách bất kỳ. Viết một chương trình để ước lượng khả năng bạn sẽ không nghe thấy một cặp bài hát theo thứ tự. Ví dụ, bài hát 3 không theo sau bài hát 2, bài hát 10 không theo sau bài hát 9.
25. **Hoán vị nghịch đảo.** Viết chương trình `InversePermutation.java` đọc một hoán vị của các số nguyên từ 0 đến `N-1` với `N` lấy từ dòng lệnh và in ra hoán vị nghịch đảo của nó. Nếu hoán vị là một mảng `a[]`, nghịch đảo của nó là mảng `b[]` sao cho `a[b[i]] = b[a[i]] = i`. Nhớ kiểm tra đầu vào là một hoán vị hợp lệ.
26. **Ma trận Hadamard.** Ma trận Hadamard  $H(N)$  là ma trận  $N \times N$  các giá trị boolean (bit) tính chất rất hay là hai hàng kế tiếp khác nhau trong chính xác  $N / 2$  bit. (Tính chất này rất hữu ích trong việc thiết kế mã sửa lỗi).  $H(1)$  là một ma trận  $1 \times 1$  - với một giá trị `true`, và với  $N > 1$ ,  $H(2N)$  thu được bằng cách sắp xếp bốn bản sao của  $H(N)$  thành ma trận cỡ  $2N \times 2N$ , sau đó đảo ngược tất cả giá trị trong ma trận  $N \times N$  ở phía dưới bên phải, như ví dụ sau đây (với `T = true` và `F = false`).

| H(1) | H(2) | H(4)    |
|------|------|---------|
| T    | T T  | T T T T |
|      | T F  | T F T F |
|      |      | T T F F |
|      |      | T F F T |

Viết chương trình `Hadamard.java` lấy đối số dòng lệnh `N` và in ra  $H(N)$ . Giả sử `N` là số mũ của 2.



27. **Tin đồn.** Alice tổ chức một bữa tiệc với  $N$  người khách, bao gồm cả Bob. Bob loan truyền một tin đồn về Alice bằng cách nói với nó vào một trong những người khác. Một người nghe tin đồn này lần đầu tiên ngay lập tức sẽ kể nó vào một vị khách khác, chọn ngẫu nhiên từ tất cả những người ở bên ngoại trừ Alice và người kể cho họ. Nếu một người (bao gồm cả Bob) nghe tin đồn cho một lần thứ hai, anh ta hoặc cô ấy sẽ không tuyên truyền tiếp nữa. Viết chương trình để ước tính xác suất mà tất cả mọi người tại buổi tiệc (trừ Alice) nghe thấy tin đồn trước khi nó thôi lan truyền. Đồng thời ước tính trung bình số người nghe thấy tin đồn.
28. **Tìm số bị lặp.** Cho một mảng  $N$  phần tử với mỗi phần tử có giá trị từ 1 đến  $N$ , viết chương trình xác định xem có sự trùng lặp trong mảng không.
29. **Đếm số nguyên tố.** So sánh `PrimeSieve.java` với phương pháp chúng ta sử dụng ở cuối chương 3. Đây là một ví dụ điển hình về sự cân bằng thời gian - không gian: `PrimeSieve` nhanh, nhưng đòi hỏi một mảng `boolean` kích thước  $N$ ; cách tiếp cận kia chỉ sử dụng hai biến số nguyên, nhưng lại chậm hơn đáng kể. Ước tính độ lớn của sự khác biệt này bằng cách tìm giá trị của  $N$  mà cách tiếp cận thứ hai này có thể hoàn thành trong khoảng thời gian tương tự như câu lệnh `java PrimeSieve 1000000`.
30. **Trò chơi Minesweeper.** Viết chương trình `Minesweeper.java` lấy 3 đối số dòng lệnh  $M$ ,  $N$ , và  $p$  và tạo ra một mảng `boolean`  $M \times N$  trong đó mỗi vị trí có mìn với xác suất  $p$ . Sau đó in ra bảng với dấu `*` thể hiện mìn và dấu chấm thể hiện ô an toàn. Sau đó, thay thế các ô an toàn với số lượng mìn lân cận (trên, dưới, trái, phải, hoặc chéo).

```
* * . . . * * 1 0 0
. 3 3 2 0 0
. * . . . 1 * 1 0 0
```

Cố gắng viết mã càng gọn càng tốt. (Gợi ý: sử dụng bảng  $(M+2) \times (N + 2)$ ).

31. **Độ dài đi bộ ngẫu nhiên.** Giả sử bảng không có giới hạn. Viết chương trình mô phỏng tính quãng đường trung bình của người đi bộ ngẫu nhiên (không lặp lại ô đã đi).
32. **Đi bộ tự tránh trong 3 chiều.** Viết chương trình mô phỏng để xác minh rằng xác suất đi vào ngõ cụt là 0 với bài toán *đi bộ ngẫu nhiên ba chiều* và tính toán chiều dài đi bộ trung bình cho các giá trị khác nhau của  $N$  (số bước).
33. **Người đi bộ ngẫu nhiên.** Giả sử có  $N$  người đi bộ ngẫu nhiên, bắt đầu ở trung tâm của một lưới  $N \times N$ , di chuyển một bước tại một thời điểm sang trái, phải, lên, hoặc xuống với xác suất như nhau. Viết một chương trình `RandomWalkers.java` mô phỏng và tính số lượng các bước cần để tất cả các ô đều có người đi qua.
34. **Chia bài.** Trong trò chơi tú lơ khơ, bốn người chơi mỗi người được chia 13 quân bài. Một con số thống kê quan trọng là phân phối số lượng quân bài mỗi chất (rô, cơ, bích, tép). Bài nào có khả năng nhất, 5-3-3-2, 4-4-3-2, hay 4-3-3-3?
35. **Sinh nhật.** Giả sử rằng từng người đi vào một căn phòng trống cho đến khi có 2 người có cùng sinh nhật. Tính trung bình, bao nhiêu người sẽ vào phòng? Viết một chương trình `Birthday.java` để mô phỏng thí nghiệm này nhiều lần và ước tính giá trị trung bình. Giả sử ngày sinh nhật là số nguyên ngẫu nhiên thống nhất giữa 0 và 364.
36. **Sưu tập coupon.** Viết chương trình mô phỏng xác nhận kết quả toán học cổ điển rằng số coupon trung bình phải thu thập để có được  $N$  loại coupon khác nhau là  $N * (1 + 1/2 + 1/3 + \dots + 1/N)$ .

37. **Phân phối nhị thức.** Viết chương trình `BinomialDistribution.java` xây dựng và in các giá trị của mảng `a[] []` sao cho `a[N][k]` chứa xác suất tung `N` đồng xu được chính xác `k` mặt ngửa (`N` là đối số dòng lệnh). Để tính toán các con số này, bắt đầu với `a[N][0] = 0` với mọi `N` và `a[1][1] = 1`, sau đó tính giá trị trong hàng kế tiếp, từ trái sang phải, với `a[N][k] = (a[N-1][k] + a[N-1][k-1]) / 2`.

| Tam giác Pascal | Phan phối nhị thức    |
|-----------------|-----------------------|
| 1               | 1                     |
| 1 1             | 1/2 1/2               |
| 1 2 1           | 1/4 1/2 1/4           |
| 1 3 3 1         | 1/8 3/8 3/8 1/8       |
| 1 4 6 4 1       | 1/16 1/4 3/8 1/4 1/16 |

# Chương 5

## Nhập xuất

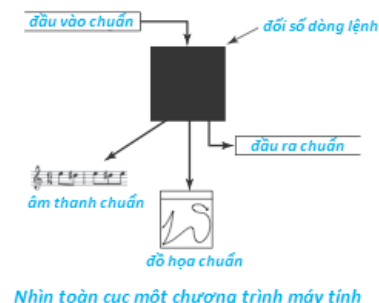
### Mục lục chương

|     |                                                   |    |
|-----|---------------------------------------------------|----|
| 5.1 | Cái nhìn toàn cục một chương trình Java . . . . . | 63 |
| 5.2 | Đầu ra chuẩn . . . . .                            | 64 |
| 5.3 | Đầu vào chuẩn . . . . .                           | 65 |
| 5.4 | Điều hướng và đường ống . . . . .                 | 68 |
| 5.5 | Đồ họa chuẩn . . . . .                            | 69 |
| 5.6 | Âm thanh chuẩn . . . . .                          | 77 |
| 5.7 | Hỏi đáp . . . . .                                 | 84 |
| 5.8 | Bài tập . . . . .                                 | 87 |

Trong chương này chúng ta mở rộng tập hợp các khái niệm trừu tượng (đầu vào dòng lệnh và đầu ra chuẩn) mà chúng ta vẫn dùng làm giao diện giữa các chương trình Java của chúng ta và thế giới bên ngoài. Các khái niệm mới bao gồm đầu vào chuẩn, đồ họa (vẽ) chuẩn và đầu ra âm thanh chuẩn. Đầu vào chuẩn giúp chúng ta viết chương trình xử lý một lượng tùy ý các đầu vào và tương tác với chương trình khác; đồ họa chuẩn giúp chúng tôi vẽ hình; và đầu ra âm thanh chuẩn bổ sung thêm âm thanh.

### 5.1 Cái nhìn toàn cục một chương trình Java

Nhìn toàn cảnh, một chương trình Java có giá trị đầu vào từ dòng lệnh và in ra một chuỗi các ký tự ra đầu ra. Mặc định, cả hai đầu vào từ dòng lệnh và đầu ra chuẩn có liên quan đến các ứng dụng terminal (mà bạn gõ các lệnh java và javac ở đó).



- *Đầu vào từ dòng lệnh.* Đầu vào này có thể lấy từ đối số kiểu mảng `String[]` của hàm `main()`. Mảng này là chuỗi các đối số dòng lệnh mà chúng ta đánh vào trên ứng dụng terminal, cung cấp cho Java bởi hệ điều hành. Theo quy ước, cả Java và các hệ điều hành dùng các đối số như các xâu, vì vậy nếu chúng ta có ý định coi đối số là một con số, chúng tôi sử dụng một lệnh như `Integer.parseInt()` để chuyển đổi nó từ `String` sang kiểu thích hợp.

```

1 public class RandomSeq {
2 public static void main(String[] args) {
3
4 // command-line argument
5 int N = Integer.parseInt(args[0]);
6
7 // generate and print N numbers between 0 and 1
8 for (int i = 0; i < N; i++) {
9 System.out.println(Math.random());
10 }
11 }
12 }

```

- Đầu ra chuẩn. Để in các giá trị ra đầu ra từ các chương trình, chúng ta đã sử dụng `System.out.println()`. Java gửi kết quả đến một luồng trừu tượng của kí tự gọi là đầu ra chuẩn. Mặc định, hệ điều hành kết nối đầu ra chuẩn ở các cửa sổ terminal - vì thế tất cả đầu ra từ các chương trình của chúng ta cho đến nay đều xuất hiện trong cửa sổ terminal.

Chương trình `RandomSeq.java` (đoạn mã 5.1) sử dụng mô hình này: Nó lấy từ dòng lệnh số `N` và tạo ra dãy các số ngẫu nhiên giữa 0 và 1.

Để hoàn thiện mô hình chương, chúng tôi thêm các thư viện sau:

- Đầu vào chuẩn: Đọc số và chuỗi từ người dùng.
- Đồ họa chuẩn: Vẽ đồ họa.
- Âm thanh chuẩn: Tạo âm thanh.

Để sử dụng các thư viện, hãy tải các thư viện sau về máy tính và dịch (gọi `javac`) để tạo các file `.class` trong thư mục làm việc của bạn

|                            |                                                                                                                                     |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <code>StdIn.java</code>    | <a href="http://introcs.cs.princeton.edu/java/15inout/StdIn.java">http://introcs.cs.princeton.edu/java/15inout/StdIn.java</a>       |
| <code>StdOut.java</code>   | <a href="http://introcs.cs.princeton.edu/java/15inout/StdOut.java">http://introcs.cs.princeton.edu/java/15inout/StdOut.java</a>     |
| <code>StdDraw.java</code>  | <a href="http://introcs.cs.princeton.edu/java/15inout/StdDraw.java">http://introcs.cs.princeton.edu/java/15inout/StdDraw.java</a>   |
| <code>StdAudio.java</code> | <a href="http://introcs.cs.princeton.edu/java/15inout/StdAudio.java">http://introcs.cs.princeton.edu/java/15inout/StdAudio.java</a> |

## 5.2 Đầu ra chuẩn

Trong Java, lệnh `print()` và lệnh `println()`, được gọi với `System.out`, trừu tượng hóa đầu ra chuẩn mà chúng ta vẫn dùng trong các chương trước, nhưng từ chương này, để thống nhất cách dùng đầu vào chuẩn và đầu ra chuẩn, chúng tôi sử dụng quy định giao diện lệnh (Application Program Interface - API) như sau:

|                                          |                                 |
|------------------------------------------|---------------------------------|
| <code>public class StdOut</code>         |                                 |
| <code>void print(String s)</code>        | <i>In chuỗi s</i>               |
| <code>void println(String s)</code>      | <i>In chuỗi s và xuống dòng</i> |
| <code>void println()</code>              | <i>Xuống dòng</i>               |
| <code>void printf(String f, ... )</code> | <i>In có định dạng</i>          |

### Giao diện lệnh của StdOut

Lệnh `print()` và lệnh `println()` có cách dùng như các bạn vẫn dùng. Lệnh mới `printf()` cho phép chúng ta điều khiển định dạng khi in ra đầu ra chuẩn.

- *Định dạng.* Trong dạng đơn giản nhất, `printf()` có hai đối số. Đối số đầu tiên, một chuỗi kí tự, chứa định dạng mô tả cách các đối số tiếp theo chuyển đổi thành một chuỗi ở đầu ra.

### Cách dùng lệnh StdOut.printf

- *Chuỗi định dạng.* Đối số đầu tiên của `printf()` được gọi là chuỗi định dạng. Chuỗi định dạng có 2 phần: phần định dạng và phần không định dạng. Phần không định dạng được in trực tiếp ra đầu ra. Phần định dạng sẽ được thay thế bởi các đối số tiếp theo.
- *Nhiều đối số.* Các lệnh `printf()` có thể dùng nhiều hơn hai đối số. Mỗi đối số ở sau đối số thứ nhất sẽ được định dạng bởi một phần định dạng trong đối số thứ nhất.

## 5.3 Đầu vào chuẩn

Thư viện đầu vào chuẩn của chúng ta cài đặt một luồng dữ liệu trừu tượng có thể rỗng hoặc có chứa một dãy các giá trị cách nhau bởi khoảng trắng. Mỗi giá trị là một chuỗi hoặc một giá trị thuộc các kiểu cơ bản Java. Một trong những tính năng chính của đầu vào chuẩn là chương trình của bạn sẽ tiêu thụ các giá trị khi đọc chúng. Sau khi chương trình của bạn đọc một giá trị, đầu vào chuẩn không thể sao lưu và đọc nó một lần nữa. Thư viện này được xác định bởi các API sau:

## public class StdIn

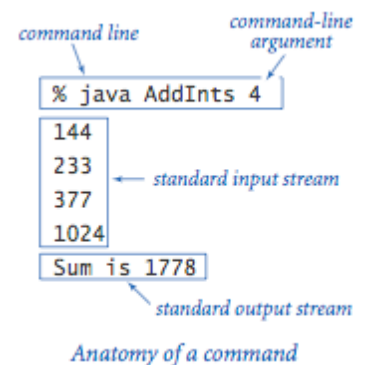
|         |               |                                         |
|---------|---------------|-----------------------------------------|
| boolean | isEmpty()     | true if no more values, false otherwise |
| int     | readInt()     | read a value of type int                |
| double  | readDouble()  | read a value of type double             |
| long    | readLong()    | read a value of type long               |
| boolean | readBoolean() | read a value of type boolean            |
| char    | readChar()    | read a value of type char               |
| String  | readString()  | read a value of type String             |
| String  | readLine()    | read the rest of the line               |
| String  | readAll()     | read the rest of the text               |

*API for our library of static methods for standard input*

Bây giờ chúng ta xem xét một số ví dụ cụ thể.

- *Đọc bàn phím.* Khi bạn sử dụng lệnh `java` để gọi một chương trình Java từ dòng lệnh, bạn thực sự đang làm ba điều: (i) phát lệnh bắt đầu thực hiện chương trình, (ii) xác định giá trị của các đối số dòng lệnh và (iii) bắt đầu xác lập đầu vào tiêu chuẩn. Chuỗi ký tự mà bạn gõ vào cửa sổ terminal sau dòng lệnh là đầu vào chuẩn. Khi bạn gõ ký tự, bạn đang tương tác với chương trình của bạn. Chương trình chờ đợi bạn để tạo ra các dòng đầu vào tiêu chuẩn. Ví dụ sau, `AddInts.java` lấy đối số `N` từ dòng lệnh, sau đó đọc `N` số từ đầu vào tiêu chuẩn và cộng chúng lại:

```
public class AddInts {
 public static void main(String[] args) {
 int N = Integer.parseInt(args[0]);
 int sum = 0;
 for (int i = 0; i < N; i++)
 sum += StdIn.readInt();
 StdOut.println("Sum is " + sum);
 }
}
```



- *Định dạng đầu vào.* Nếu bạn gõ `abc` hoặc `12.2` hoặc `true` khi dùng `StdIn.readInt()` để đọc một số nguyên `int`, lệnh này sẽ sinh một ngoại lệ (lỗi) `NumberFormatException`. `StdIn` coi các khoảng trắng liên tiếp giống như một dấu cách và cho phép bạn để phân biệt các giá trị bằng khoảng trắng.
- *Tương tác với người dùng.* Chương trình `TwentyQuestions.java` (đoạn mã 5.2) chơi một trò chơi đoán số đơn giản như sau: Nó chọn một số, sau đó bạn đoán bằng cách gõ vào một con số, chương trình sẽ trả lời bạn đã đoán chính xác, quá cao hay quá thấp. Bạn được đoán 20 lần.
- *Đầu vào vô hạn.* Chương trình của bạn có thể đọc từ đầu vào miễn là còn giá trị chưa đọc. Ví dụ, chương trình `Average.java` (đoạn mã 5.3) đọc một chuỗi các số thực từ đầu vào chuẩn và in trung bình của chúng.

Đoạn mã 5.2: TwentyQuestions.java

```

1 public class TwentyQuestions {
2
3 public static void main(String[] args) {
4
5 // Generate a number and answer questions
6 // while the user tries to guess the value.
7 int N = 1 + (int) (Math.random() * 1000000);
8
9 StdOut.print("I'm thinking of a number ");
10 StdOut.println("between 1 and 1,000,000");
11 int m = 0;
12 while (m != N) {
13
14 // Solicit one guess and provide one answer
15 StdOut.print("What's your guess? ");
16 m = StdIn.readInt();
17 if (m == N) StdOut.println("You win!");
18 if (m < N) StdOut.println("Too low ");
19 if (m > N) StdOut.println("Too high");
20 }
21 }
22 }

```

Đoạn mã 5.3: Average.java

```

1 public class Average {
2 public static void main(String[] args) {
3 int count = 0; // number input values
4 double sum = 0.0; // sum of input values
5
6 // read data and compute statistics
7 while (!StdIn.isEmpty()) {
8 double value = StdIn.readDouble();
9 sum += value;
10 count++;
11 }
12
13 // compute the average
14 double average = sum / count;
15
16 // print results
17 StdOut.println("Average is " + average);
18 }
19 }

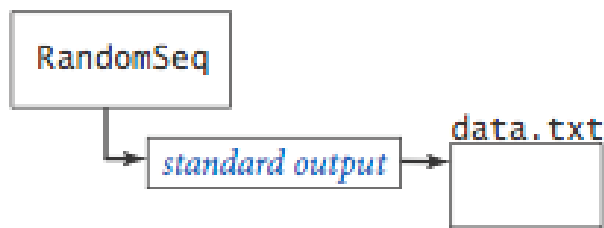
```

## 5.4 Điều hướng và đường ống

Đối với nhiều ứng dụng, việc gõ dữ liệu đầu vào từ bàn phím và cửa sổ terminal giới hạn sức mạnh xử lý của chương trình của chúng ta bởi số lượng dữ liệu chúng ta có thể gõ. Tương tự như vậy, chúng ta thường muốn lưu các thông tin đã in ra đầu ra chuẩn để sử dụng sau này. Chúng ta có thể sử dụng cơ chế của hệ điều hành để giải quyết cả hai vấn đề.

- *Chuyển hướng đầu ra chuẩn vào tập tin.* Bằng cách thêm một chỉ thị đơn giản cho các lệnh gọi chương trình, chúng ta có thể chuyển hướng đầu ra chuẩn vào một tập tin, để lưu trữ vĩnh viễn hoặc làm đầu vào của chương trình khác. Ví dụ, lệnh

```
java RandomSeq 1000 > data.txt
```

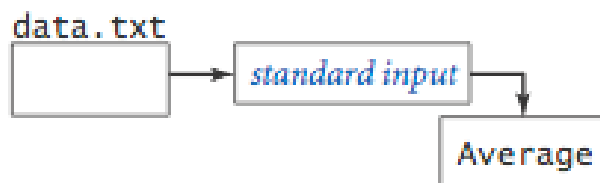


*Redirecting standard output to a file*

chỉ thị rằng đầu ra chuẩn không phải là cửa sổ terminal mà là tập tin `data.txt`. Mỗi lệnh `StdOut.print()` hoặc `StdOut.println()` nối một dòng vào cuối file `data.txt`. Trong ví dụ này, tập tin sẽ chứa 1000 số ngẫu nhiên. Bạn sẽ không thấy gì được in ra cửa sổ terminal mà các số này sẽ được chuyển thẳng ra tập tin sau dấu `>`.

- *Chuyển hướng đầu vào chuẩn từ tập tin.* Tương tự như vậy, chúng ta có thể chuyển hướng đầu vào tiêu chuẩn để `StdIn` đọc dữ liệu từ một tập tin thay vì từ cửa sổ terminal. Ví dụ, lệnh

```
java Average < data.txt
```



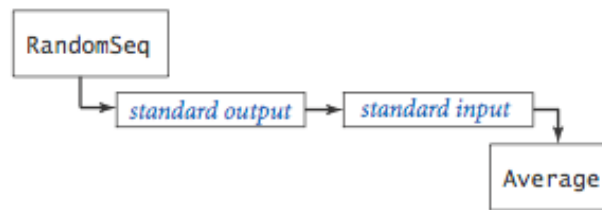
*Redirecting from a file to standard input*

đọc một dãy số từ file `data.txt` và tính giá trị trung bình của chúng. Như vậy, dấu `<` chuyển hướng đầu vào chuẩn sang tập tin đứng sau nó thay vì đợi người dùng gõ vào cửa sổ terminal. Lệnh `StdIn.readDouble()` sẽ đọc một số từ tập tin này. Như vậy, việc chuyển hướng đầu vào cho phép chương trình tiếp cận nguồn dữ liệu bất kì, chỉ bị giới hạn bởi kích thước tập tin ta có thể lưu trữ.

- *Kết nối hai chương trình.* Cách linh hoạt nhất để tạo ra đầu vào chuẩn và đầu ra chuẩn trườ chính là sinh ra chúng bởi chính các chương trình của chúng ta! Cơ chế này được gọi là **đường ống**. Ví dụ, lệnh sau đây



```
java RandomSeq 1000 | java Average
```



*Piping the output of one program to the input of another*

chỉ rằng đầu ra chuẩn của chương trình `RandomSeq` là đầu vào chuẩn của chương trình `Average`. Câu lệnh trên cho kết quả như 2 câu lệnh sau

```
% java RandomSeq 1000 > data.txt
% java Average < data.txt
```

- *Bộ lọc*. Đối với nhiều tác vụ thông thường, ta có thể nghĩ rằng mỗi chương trình trong đây là một bộ lọc chuyển đổi đầu vào chuẩn thành đầu ra chuẩn theo một cách nào đó, sử dụng cơ chế đường ống để kết nối chương trình với nhau.

Hệ điều hành của bạn cũng cung cấp một số bộ lọc. Ví dụ, bộ lọc `sort` sắp xếp đầu vào chuẩn của nó:

```
% java RandomSeq 5 | sort
0.035813305516568916
0.14306638757584322
0.348292877655532103
0.5761644592016527
0.9795908813988247
```

Bạn hãy thử với các bộ lọc `more`, `less`.

## 5.5 Đồ họa chuẩn

Bây giờ chúng tôi giới thiệu một khái niệm trừu tượng đơn giản xuất đồ họa ở đầu ra. Chúng ta hãy tưởng tượng một thiết bị vẽ trừu tượng có khả năng vẽ đường và các điểm trên một phong hai chiều. Chúng ta sẽ vẽ lên phong này bằng cách gọi đến các phương thức tĩnh trong `StdDraw`. Giao diện chính của nó bao gồm hai loại phương pháp: lệnh vẽ mà điều khiển thiết bị (chẳng hạn như vẽ một đường hay vẽ một điểm) và các lệnh điều khiển thiết lập các thông số như kích thước bút hoặc cỡ của phong vẽ. Toàn bộ các lệnh của `StdDraw` có thể đọc ở link sau

<http://introcs.cs.princeton.edu/java/15inout/javadoc/StdDraw.html>

- *Các lệnh vẽ cơ bản*. Xem các lệnh vẽ đường và điểm sau

```
public class StdDraw (basic drawing commands)
 void line(double x0, double y0, double x1, double y1)
 void point(double x, double y)
```

Những phương pháp này gần nói rõ chúng định làm gì: `StdDraw.line(x0, y0, x1, y1)` vẽ một đoạn thẳng nối các điểm  $(x_0, y_0)$  tới điểm  $(x_1, y_1)$  còn `StdDraw.point(x, y)` vẽ một điểm ở vị trí  $(x, y)$ . Hệ tọa độ mặc định là từ  $(0,0)$  đến  $(1,1)$ . Các đoạn thẳng và điểm

mặc định sẽ có màu đen trên nền trắng. *Hình vẽ đầu tiên của bạn.* Chương trình **HelloWorld** cho lập trình đồ họa với StdDraw là vẽ một hình tam giác với một điểm bên trong. Để vẽ hình tam giác, chúng ta vẽ 3 đoạn thẳng.

```
double t = Math.sqrt(3.0)/2.0;
StdDraw.line(0.0, 0.0, 1.0, 0.0);
StdDraw.line(1.0, 0.0, 0.5, t);
StdDraw.line(0.5, t, 0.0, 0.0);
StdDraw.point(0.5, t/3.0);
```

- *Các lệnh thiết lập thông số.* Chúng ta có thể thay đổi hệ tọa độ (hiển thị) trên thiết bị cũng như độ dày của nét vẽ. StdDraw cung cấp các lệnh sau

```
public class StdDraw (basic control commands)
{
 void setXscale(double x0, double x1) reset x range to (x0, x1)
 void setYscale(double y0, double y1) reset y range to (y0, y1)
 void setPenRadius(double r) set pen radius to r
}
```

*Note: Methods with the same names but no arguments reset to default values.*

Ví dụ, nếu ta muốn hệ tọa độ hiển thị từ  $(x_0, y_0)$  đến  $(x_1, y_1)$ , ta dùng 2 lệnh

```
StdDraw.setXscale(x0, x1);
StdDraw.setYscale(y0, y1);
```

- *Lọc dữ liệu ra đồ họa chuẩn.* Chương trình PlotFilter.java (đoạn mã 5.4) đọc một dãy các điểm có tọa độ  $(x, y)$  từ đầu vào chuẩn và vẽ nó ra thành 1 điểm trên đồ họa chuẩn. Chương trình coi 4 số đầu tiên là giới hạn của khung hình. Bạn có thể dùng lệnh

```
% java PlotFilter < USA.txt
```

để in các điểm trong tập tin USA.txt. Tập tin này có thể tải về từ địa chỉ

```
http://introcs.cs.princeton.edu/java/15inout/USA.txt
```



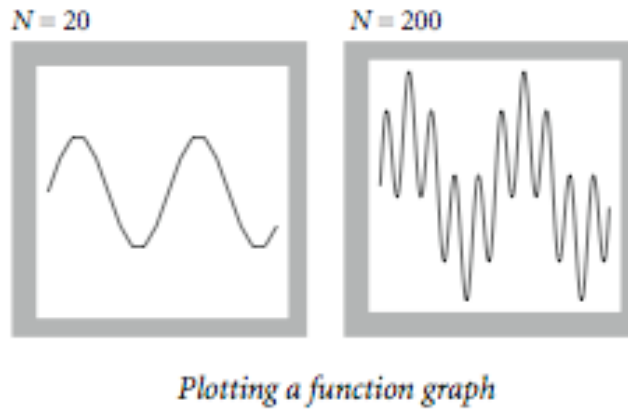
- *Vẽ đồ thị hàm.* Chương trình FunctionGraph.java (đoạn mã 5.5) vẽ đồ thị hàm  $y = \sin(4x) + \sin(20x)$  trong đoạn  $[0, \pi]$ . Do có vô hạn số điểm trong khoảng đó, chúng ta chỉ có thể tính giá trị một số hữu hạn điểm trong đó. Chúng ta sẽ vẽ các điểm tại một số vị trí  $(x, y)$  rồi nối chúng lại với nhau. Số các điểm  $N$  là đối số dòng lệnh.

Đoạn mã 5.4: PlotFilter.java

```
1 public class PlotFilter {
2
3 public static void main(String[] args) {
4
5 // read in bounding box and rescale
6 double x0 = StdIn.readDouble();
7 double y0 = StdIn.readDouble();
8 double x1 = StdIn.readDouble();
9 double y1 = StdIn.readDouble();
10 StdDraw.setXscale(x0, x1);
11 StdDraw.setYscale(y0, y1);
12
13 // turn on animation mode to defer displaying all of the points
14 // StdDraw.show(0);
15
16 // plot points, one at a time
17 while (!StdIn.isEmpty()) {
18 double x = StdIn.readDouble();
19 double y = StdIn.readDouble();
20 StdDraw.point(x, y);
21 }
22
23 // display all of the points now
24 // StdDraw.show(0);
25
26 }
27 }
```

Đoạn mã 5.5: FunctionGraph.java

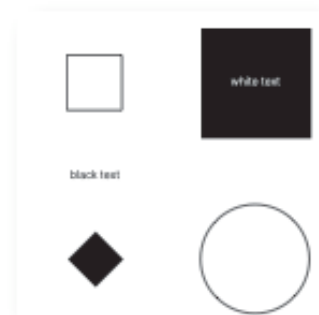
```
1 public class FunctionGraph {
2 public static void main(String[] args) {
3
4 // number of line segments to plot
5 int N = Integer.parseInt(args[0]);
6
7 // the function $y = \sin(4x) + \sin(20x)$, sampled at N points
8 // between $x = 0$ and $x = \pi$
9 double[] x = new double[N+1];
10 double[] y = new double[N+1];
11 for (int i = 0; i <= N; i++) {
12 x[i] = Math.PI * i / N;
13 y[i] = Math.sin(4*x[i]) + Math.sin(20*x[i]);
14 }
15
16 // rescale the coordinate system
17 StdDraw.setXscale(0, Math.PI);
18 StdDraw.setYscale(-2.0, +2.0);
19
20 // plot the approximation to the function
21 for (int i = 0; i < N; i++) {
22 StdDraw.line(x[i], y[i], x[i+1], y[i+1]);
23 }
24 }
25 }
```



- *Vẽ các hình hình học.* **StdDraw** bao gồm các lệnh vẽ vòng tròn, hình chữ nhật, và đa giác. Mỗi một hình dạng cho ta một đường viền bao quanh hình hoặc điền đầy phần bên trong bằng màu sắc.

```
StdDraw.square(.2, .8, .1);
StdDraw.filledSquare(.8, .8, .2);
StdDraw.circle(.8, .2, .2);
double[] xd = { .1, .2, .3, .2 };
double[] yd = { .2, .3, .2, .1 };
StdDraw.filledPolygon(xd, yd);
StdDraw.text(.2, .5, "black text");
StdDraw.setPenColor(StdDraw.WHITE);
StdDraw.text(.8, .8, "white text");
```

```
public class StdDraw (shapes)
void circle(double x, double y, double r)
void filledCircle(double x, double y, double r)
void square(double x, double y, double r)
void filledSquare(double x, double y, double r)
void polygon(double[] x, double[] y)
void filledPolygon(double[] x, double[] y)
```



*Shape and text examples*

- *Văn bản và màu sắc.* Để chú thích hoặc làm nổi bật các yếu tố khác nhau trong bản vẽ, **StdDraw** cung cấp các lệnh vẽ văn bản, thiết lập font chữ, và thiết lập màu mực bút vẽ.

```
public class StdDraw (text and color commands)
void text(double x, double y, String s)
void setFont(Font f)
void setPenColor(Color c)
```

Trong các lệnh này, `java.awt.Font` và `java.awt.Color` là các kiểu không phải kiểu cơ bản (ta sẽ học sau). Màu bút mặc định là màu đen; font chữ mặc định là Serif 16 điểm.

- *Hoạt hình.* Thư viện **StdDraw** có thêm hai lệnh có thể dùng để tạo ra hiệu ứng chuyển động trên màn hình.

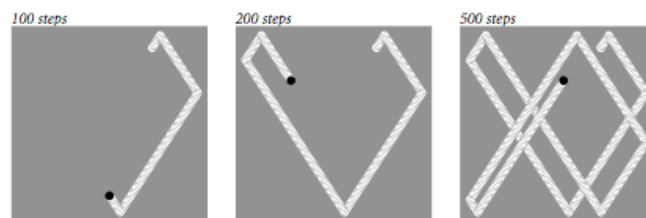
|                                               |                                                                                |
|-----------------------------------------------|--------------------------------------------------------------------------------|
| <code>public class StdDraw</code>             | <i>(advanced control commands)</i>                                             |
| <code>void setCanvasSize(int w, int h)</code> | <i>create canvas in screen window of width from w and height h (in pixels)</i> |
| <code>void clear()</code>                     | <i>clear the canvas to white (default)</i>                                     |
| <code>void clear(Color c)</code>              | <i>clear the canvas; color it c</i>                                            |
| <code>void show(int dt)</code>                | <i>draw, then pause dt milliseconds</i>                                        |
| <code>void show()</code>                      | <i>draw, turn off pause mode</i>                                               |

Chương trình hoạt hình đầu tiên của chúng ta là vẽ một quả bóng đen chuyển động trên phong vẽ. Giả sử quả bóng đang ở vị trí  $(x, y)$  và chúng ta muốn tạo ra cảm giác nó di chuyển đến vị trí mới, tại  $(x + 0,01, y + 0,02)$ . Chúng ta có các bước sau:

- Tính vị trí của quả bóng;
- Xóa phong vẽ;
- Vẽ quả bóng ở vị trí mới;
- Dừng hình trong khoảng thời gian ngắn (để mắt có cảm giác quả bóng chuyển sang vị trí mới)

Để tạo ra cảm giác chuyển động, chúng ta lặp lại các bước trên qua một chuỗi các vị trí (có thể tạo thành một đường thẳng trong trường hợp này). Theo đó, `StdDraw` có một phương thức `show()` cho phép chúng ta kiểm soát các lệnh vẽ trên màn hình. Bạn có thể tưởng tượng rằng `show()` thu thập tất cả các đoạn thẳng, điểm, hình, và văn bản mà chúng ta cung cấp cho nó và vẽ tất cả chúng ngay lập tức khi chúng ta ra lệnh `show()`. Để kiểm soát tốc độ, `show()` có đối số để `StdDraw` chờ đợi trong vài mili-giây sau khi vẽ. Mặc định, `StdDraw` gọi `show()` sau mỗi lệnh vẽ `line()`, `point()`, hoặc các lệnh vẽ khác; chúng ta tắt tùy chọn này khi chúng ta gọi `StdDraw.show(t)` và bật nó trở lại khi chúng ta gọi `StdDraw.show()` không có đối số.

Chương trình `BouncingBall.java` (đoạn mã 5.6) thực hiện các bước sau để tạo ra cảm giác một quả bóng di chuyển trong một cái hộp, gốc tọa độ ở chính giữa. Vị trí hiện tại của quả bóng là  $(rx, ry)$  và vị trí mới tại mỗi bước được tính toán bằng cách thêm  $vx$  vào  $rx$  và  $vy$  vào  $ry$ . Như vậy  $(vx, vy)$  là khoảng cách cố định mà quả bóng di chuyển trong mỗi đơn vị thời gian, nó đại diện cho vận tốc theo các hướng. Để giữ bóng trong hộp, chúng ta mô phỏng quả bóng nảy ra khỏi các bức tường theo quy luật của va chạm đàn hồi. Hiệu ứng này rất dễ thực hiện: khi bóng chạm bức tường thẳng đứng (chiều  $y$ ), chúng ta chỉ cần thay đổi vận tốc hướng  $x$  từ  $vx$  sang  $-vx$  và khi bóng chạm bức tường ngang, chúng ta thay đổi vận tốc hướng  $y$  từ  $vy$  sang  $-vy$ . Những hình ảnh dưới đây cho thấy đường đi của quả bóng (bài tập ??).



- *Hình ảnh.* Thư viện `StdDraw` của chúng ta hỗ trợ vẽ hình bằng lệnh `StdDraw.picture(x, y, filename)` vẽ hình ảnh trong tập tin `filename` (định dạng JPEG, GIF hoặc PNG) tại vị trí  $(x, y)$ . Chương trình `DeluxeBouncingBall.java` (đoạn mã 5.7) minh họa ví dụ trên những quả bóng được thay thế bằng hình ảnh trái đất.
- *Tương tác với người dùng.* Thư viện vẽ chuẩn của chúng ta còn có các lệnh để người dùng có thể tương tác với cửa sổ bằng cách sử dụng chuột.

Đoạn mã 5.6: BouncingBall.java

```
1 public class BouncingBall {
2 public static void main(String[] args) {
3
4 // set the scale of the coordinate system
5 StdDraw.setXscale(-1.0, 1.0);
6 StdDraw.setYscale(-1.0, 1.0);
7
8 // initial values
9 double rx = 0.480, ry = 0.860; // position
10 double vx = 0.015, vy = 0.023; // velocity
11 double radius = 0.05; // radius
12
13 // main animation loop
14 while (true) {
15
16 // bounce off wall according to law of elastic collision
17 if (Math.abs(rx + vx) > 1.0 - radius) vx = -vx;
18 if (Math.abs(ry + vy) > 1.0 - radius) vy = -vy;
19
20 // update position
21 rx = rx + vx;
22 ry = ry + vy;
23
24 // clear the background
25 StdDraw.clear(StdDraw.GRAY);
26
27 // draw ball on the screen
28 StdDraw.setPenColor(StdDraw.BLACK);
29 StdDraw.filledCircle(rx, ry, radius);
30
31 // display and pause for 20 ms
32 StdDraw.show(20);
33 }
34 }
35 }
```

Đoạn mã 5.7: DeluxeBouncingBall.java

```
1 public class DeluxeBouncingBall {
2 public static void main(String[] args) {
3 double rx = .480, ry = .860; // position
4 double vx = .015, vy = .023; // velocity
5 double radius = .03; // a hack since "earth.gif"
6 // is in pixels
7
8 // set the scale of the coordinate system
9 StdDraw.setXscale(-1.0, 1.0);
10 StdDraw.setYscale(-1.0, 1.0);
11
12 // main animation loop
13 while (true) {
14 if (Math.abs(rx + vx) + radius > 1.0) {
15 vx = -vx;
16 StdAudio.play("laser.wav");
17 }
18 if (Math.abs(ry + vy) + radius > 1.0) {
19 vy = -vy;
20 StdAudio.play("pop.wav");
21 }
22 rx = rx + vx;
23 ry = ry + vy;
24 StdDraw.clear();
25 StdDraw.picture(rx, ry, "earth.gif");
26 StdDraw.show(20);
27 }
28 }
29 }
```



```

1 public class MouseFollower {
2 public static void main(String[] args) {
3 while (true) {
4
5 // mouse click
6 if (StdDraw.mousePressed())
7 StdDraw.setPenColor(StdDraw.CYAN);
8 else
9 StdDraw.setPenColor(StdDraw.BLUE);
10
11 // mouse location
12 StdDraw.clear();
13 double x = StdDraw.mouseX();
14 double y = StdDraw.mouseY();
15 StdDraw.filledCircle(x, y, .05);
16 StdDraw.show(10);
17 }
18 }
19 }

```

|                        |                              |
|------------------------|------------------------------|
| double mouseX()        | Tọa độ x của chuột           |
| double mouseY()        | Tọa độ y của chuột           |
| boolean mousePressed() | Có phải phím chuột đang ấn ? |

Các ví dụ

- Chương trình `MouseFollower.java` (đoạn mã 5.8) vẽ một quả bóng màu xanh tại vị trí con chuột, quả bóng đổi màu nếu ta ấn phím chuột.
- Chương trình `OneSimpleAttractor.java` (đoạn mã 5.9) mô phỏng một quả bóng bị hút vào vị trí có con chuột.
- Chương trình `SimpleAttractors.java` (đoạn mã 5.10) mô phỏng 20 quả bóng bị hút vào vị trí có con chuột. Khi người dùng click chuột, các quả bóng sẽ được phân bố ngẫu nhiên.
- Chương trình `Springs.java` (đoạn mã 5.11-5.13) mô phỏng chuyển động của một hệ thống lò xo gắn vào con chuột.

## 5.6 Âm thanh chuẩn

`StdAudio` là thư viện mà bạn có thể sử dụng để chơi và thao tác trên các tập tin âm thanh. Nó cho phép bạn chơi các tập tin `.wav`, viết các chương trình tạo và thao tác các mảng `double[]`, đọc và viết chúng ra dưới dạng file `.wav`:

Đoạn mã 5.9: OneSimpleAttractor.java

```

1 public class OneSimpleAttractor {
2 public static void main(String[] args) {
3 double rx = 0.0, ry = 0.0; // position
4 double vx = 0.0, vy = 0.0; // velocity
5 double mass = 1.0; // mass
6 double dt = 0.5; // time quantum
7 double drag = 0.1; // mess around with this a bit
8 double attractionStrength = 0.01;
9
10 // do the animation
11 while (true) {
12
13 // compute the attractive force to the mouse, accounting
14 // for drag
15 double dx = StdDraw.mouseX() - rx;
16 double dy = StdDraw.mouseY() - ry;
17 double fx = (dx * attractionStrength) - (drag * vx);
18 double fy = (dy * attractionStrength) - (drag * vy);
19
20 // Euler step: update velocity, then position
21 vx += fx * dt / mass;
22 vy += fy * dt / mass;
23 rx += vx * dt;
24 ry += vy * dt;
25
26 // draw particle
27 StdDraw.clear();
28 StdDraw.setPenColor(StdDraw.BLUE);
29 StdDraw.filledCircle(rx, ry, .02);
30 StdDraw.show(10);
31 }
32 }

```

```

1 public class SimpleAttractors {
2 public static void main(String[] args) {
3 int N = Integer.parseInt(args[0]);
4 double[] rx = new double[N];
5 double[] ry = new double[N];
6 double[] vx = new double[N];
7 double[] vy = new double[N];
8 double dt = 0.5;
9 double mass = 1.0;
10 double drag = 0.05; // try changing this other values 0.1
11 double attractionStrength = 0.01;
12
13 StdDraw.setPenColor(StdDraw.BLUE); // initialize drawing
14
15 while (true) { // do the animation
16 if (StdDraw.mousePressed()) { // if the mouse is pressed
17 for (int i = 0; i < N; i++) {
18 vx[i] = .2 * Math.random() - .1; // add random
19 vy[i] = .2 * Math.random() - .1; // velocity
20 }
21 }
22
23 double[] fx = new double[N]; // clear all the forces
24 double[] fy = new double[N];
25 for (int i = 0; i < N; i++) {
26 double dx = StdDraw.mouseX() - rx[i]; // add forces
27 double dy = StdDraw.mouseY() - ry[i]; // for attraction
28 fx[i] += attractionStrength * dx; // to the mouse
29 fy[i] += attractionStrength * dy;
30 }
31
32 for (int i = 0; i < N; i++) { // add drag forces
33 fx[i] += -drag * vx[i]; // drag ~ velocity
34 fy[i] += -drag * vy[i]; // in the opposite direction
35 }
36
37 for (int i = 0; i < N; i++) { // update positions
38 vx[i] += fx[i] * dt / mass; // euler steps
39 vy[i] += fy[i] * dt / mass;
40 rx[i] += vx[i] * dt;
41 ry[i] += vy[i] * dt;
42 }
43
44 StdDraw.clear();
45
46 for (int i = 0; i < N; i++) { // draw a filled circle
47 StdDraw.filledCircle(rx[i], ry[i], .01);
48 }
49
50 StdDraw.show(10);
51 }
52 }
53 }

```

Đoạn mã 5.11: Springs.java

```

1 public class Springs {
2 public static void main(String[] args) {
3 // mess around with this, try 7, 8, 9, 10, 11, 12, 15
4 // probably have to turn down the spring force to keep it
 stable after that...
5 int N = Integer.parseInt(args[0]);
6
7 double[] rx = new double[N];
8 double[] ry = new double[N];
9 double[] vx = new double[N];
10 double[] vy = new double[N];
11 double particleMass = 1.0;
12 double drag = 0.1;
13 double springStrength = 0.1;
14 double springLength = 30;
15 double gravity = 1.0;
16 double timeStep = 0.5;
17
18 // set up the drawing area
19 StdDraw.setXscale(0, 100);
20 StdDraw.setYscale(0, 100);
21 StdDraw.setPenColor(StdDraw.BLUE);
22 StdDraw.setPenRadius(0.0025);
23
24 // initialize the particle positions randomly
25 for (int i = 0; i < N; i++) {
26 rx[i] = 100 * Math.random();
27 ry[i] = 100 * Math.random();
28 }
29
30
31 // do the animation
32 while (true) {
33
34 // clear all the forces
35 double[] fx = new double[N];
36 double[] fy = new double[N];
37
38 // spring forces act between every pairing of particles
39 // spring force is proportional to the difference between
 the rest length of the spring
40 // and the distance between the 2 particles it's acting on

```

Đoạn mã 5.12: Springs.java (tiếp tục)

```

41 for (int i = 0; i < N; i++) {
42 for (int j = 0; j < N; j++) {
43
44 if (i == j) continue;
45
46 // calculate distance between particles i and j
47 double dx = rx[j] - rx[i];
48 double dy = ry[j] - ry[i];
49 double length = Math.sqrt(dx*dx + dy*dy);
50
51 // figure out the force
52 double force = springStrength * (length -
 springLength);
53 double springForceX = force * dx / length;
54 double springForceY = force * dy / length;
55
56 // update the force
57 fx[i] += springForceX;
58 fy[i] += springForceY;
59 }
60 }
61
62 // add drag force
63 // drag is proportional to velocity but in the opposite
 direction
64 for (int i = 0; i < N; i++) {
65 fx[i] += -drag * vx[i];
66 fy[i] += -drag * vy[i];
67 }
68
69 // add gravity forces
70 // just add some force pointing down to all of them
71 for (int i = 0; i < N; i++) {
72 fy[i] += -gravity;
73 }
74
75 // fix particle 1 at the mouse position
76 rx[0] = StdDraw.mouseX();
77 ry[0] = StdDraw.mouseY();
78 vx[0] = 0.0;
79 vy[0] = 0.0;
80 fx[0] = 0.0;

```

Đoạn mã 5.13: Springs.java (tiếp tục)

```
81 fy[0] = 0.0;
82
83 // update positions using approximation
84 for (int i = 0; i < N; i++) {
85 vx[i] += fx[i] * timeStep/particleMass;
86 vy[i] += fy[i] * timeStep/particleMass;
87 rx[i] += vx[i] * timeStep;
88 ry[i] += vy[i] * timeStep;
89 }
90
91
92 // clear
93 StdDraw.clear();
94
95 // draw everything
96 for (int i = 0; i < N; i++) {
97 // draw a circle for each node
98 StdDraw.filledCircle(rx[i], ry[i], 1.0);
99
100 // draw the connections between every 2 nodes
101 for (int j = 0; j < i; j++) {
102 StdDraw.line(rx[i], ry[i], rx[j], ry[j]);
103 }
104 }
105
106 // show and wait
107 StdDraw.show(10);
108 }
109 }
110 }
```

```
public class StdAudio
{
 void play(String file) play the given .wav file
 void play(double[] a) play the given sound wave
 void play(double x) play sample for 1/44100 second
 void save(String file, double[] a) save to a .wav file
 double[] read(String file) read from a .wav file
}
```

API for our library of static methods for standard audio

Chúng ta sẽ bắt đầu với việc giới thiệu một số khái niệm cơ bản đằng sau một trong những lĩnh vực lâu đời nhất và quan trọng nhất của khoa học máy tính và khoa học tính toán, đó là *xử lý tín hiệu số*.

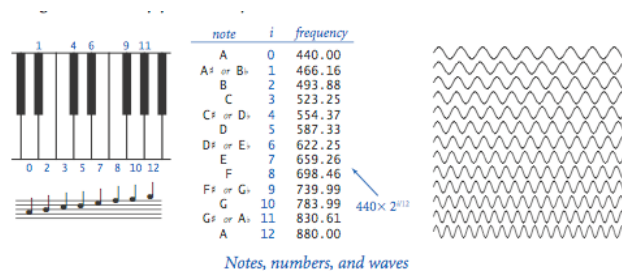
- *Nốt La (A)*. Âm thanh là dao động vật chất mà chúng ta nhận thức được, đặc biệt là sự rung động của màng nhĩ, vì vậy dao động là chìa khóa để hiểu âm thanh. Ta sẽ bắt đầu với nốt La (ký hiệu A). Nốt này chỉ đơn giản là một sóng hình *sin*, dao động ở tần số 440 lần mỗi giây. Hàm  $\sin(t)$  lặp lại sau mỗi  $2\pi$  đơn vị trên trục  $x$ , vì vậy nếu chúng ta đo  $t$  tính bằng giây và vẽ đồ thị hàm  $\sin(2\pi t \times 440)$ , chúng ta sẽ có được một đường cong dao động 440 lần mỗi giây hay 440 hertz (chu kỳ mỗi giây). Khi bạn tăng gấp đôi hoặc giảm một nửa tần số này, bạn di chuyển lên hoặc xuống một quãng tám trên phổ nhạc. Ví dụ 880 hertz là nốt La cao trên một quãng tám và 110 hertz là nốt La thấp dưới hai quãng tám. Tai người có thể nghe trong dải tần số từ 20 đến 20,000 hertz. Cường độ hay biên độ tương ứng với độ to của âm thanh. Chúng ta giả sử biên độ dao động giữa -1 và 1. Đoạn mã phát ra nốt nhạc A trong 1 giây.

```
int SAMPLE_RATE = 44100;
double[] a = new double[SAMPLE_RATE + 1];
for (int i = 0; i <= SAMPLE_RATE; i++) {
 a[i] = Math.sin(2 * Math.PI * i * 440 / SAMPLE_RATE);
}
StdAudio.play(a);
```

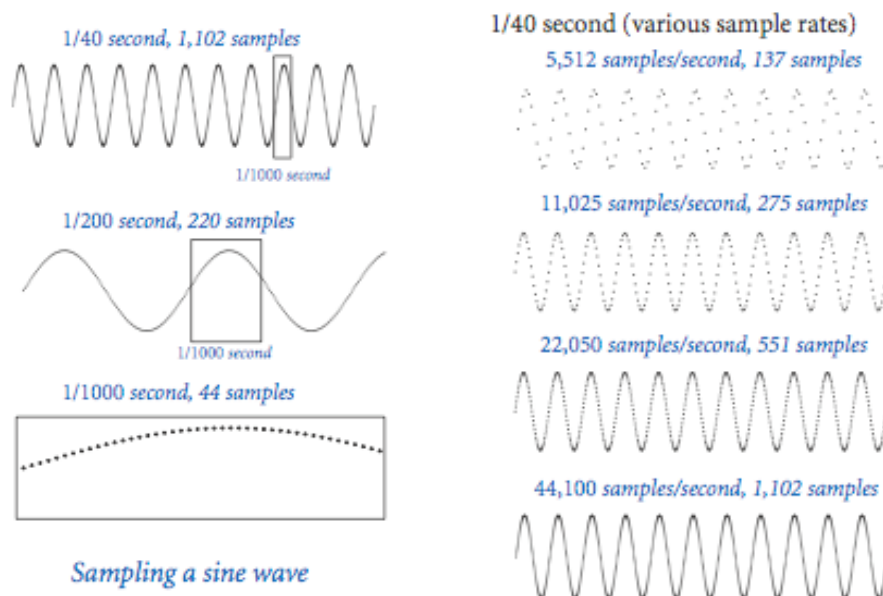
- *Các nốt nhạc khác*. Công thức tính các nốt nhạc khác rất đơn giản, bạn chỉ cần lấy nốt nhạc trước nhân với  $\sqrt[12]{2}$  để tính tần số của nốt nhạc sau. Ví dụ nốt La# sẽ có tần số

$$440 \times \sqrt[12]{2} \approx 466.16$$

Với các công thức này, ta hoàn toàn có thể tạo ra âm nhạc bằng máy tính.



- *Lấy mẫu (sampling)*. Đối với âm thanh số, chúng ta biểu diễn một đường cong bằng cách lấy mẫu nó tại các điểm đều nhau, giống hệt như khi ta muốn vẽ đồ thị hàm. Ta sẽ lấy mẫu đủ dày để chúng ta có biểu diễn chính xác của đường cong - tốc độ lấy mẫu được sử dụng rộng rãi cho âm thanh số là 44,100 mẫu mỗi giây. Đối với nốt La, tốc độ này tương ứng với 100 điểm mẫu mỗi chu kỳ của sóng *sin*. Do ta lấy mẫu tại các điểm đều nhau, ta chỉ cần tính tọa độ  $y$  tại các điểm lấy mẫu. Như vậy, chúng ta biểu diễn âm thanh chỉ đơn giản bằng một mảng số thực `double[]` có giá trị trong khoảng -1 đến 1.



- *Chơi nốt nhạc.* Chương trình `PlayThatTune.java` (đoạn mã 5.14) cho thấy chúng ta có thể dễ dàng tạo ra âm nhạc với `StdAudio`. Chương trình đọc từ đầu vào chuẩn nốt nhạc (đánh số kể từ nốt La) và trường độ rồi chơi nốt nhạc đó trên đầu ra âm thanh chuẩn.

## 5.7 Hỏi đáp

**Hỏi.** `StdIn` khác `java.util.Scanner` ở điểm nào ?

**Đáp.** Có một số khác biệt.

- Nó là một thư viện các *phương thức tĩnh* (sẽ học sau), do đó bạn không cần phải tạo ra các đối tượng để sử dụng nó.
- Nó sử dụng các ký tự mã hóa theo UTF-8, tuân theo mã hóa tiêu chuẩn Unicode.
- Nó bắt buộc định dạng địa phương theo `Locale.US`. Điều này có nghĩa rằng chương trình của bạn sẽ có thể đọc các tập tin dữ liệu trong cuốn sách ngay cả khi mặc định hệ thống của bạn ở địa phương (ví dụ, một số ngôn ngữ có thể dùng dấu phẩy làm dấu thập phân).
- Nó có thêm phương pháp đọc một ký tự.
- `StdIn` gọi các lệnh trong `java.util.Scanner`.

**Hỏi.** `StdOut` khác `System.out.println()` ở điểm nào?

**Đáp.** Có một số khác biệt.

- Nó sử dụng các ký tự mã hóa theo UTF-8, tuân theo mã hóa tiêu chuẩn Unicode.
- Nó bắt buộc định dạng địa phương theo `Locale.US` để thống nhất với `StdIn`.
- Nó đẩy các giá trị ra đầu ra chuẩn ngay sau lệnh `print()`.

Tóm lại, chúng tôi muốn `StdIn` và `StdOut` hoạt động như nhau trên tất cả các hệ thống, bao gồm cả đọc và ghi các tập tin.

**Hỏi.** Tôi có thể đọc lại dữ liệu từ đầu vào chuẩn không ?

**Đáp.** Không, bạn chỉ được đọc một lần.

**Hỏi.** Các định dạng số khi dùng `printf()`.



Đoạn mã 5.14: PlayThatTune.java

```
1 public class PlayThatTune {
2
3 public static void main(String[] args) {
4
5 // repeat as long as there are more integers to read in
6 while (!StdIn.isEmpty()) {
7
8 // read in the pitch, where 0 = Concert A (A4)
9 int pitch = StdIn.readInt();
10
11 // read in duration in seconds
12 double duration = StdIn.readDouble();
13
14 // build sine wave with desired frequency
15 double hz = 440 * Math.pow(2, pitch / 12.0);
16 int N = (int) (StdAudio.SAMPLE_RATE * duration);
17 double[] a = new double[N+1];
18 for (int i = 0; i <= N; i++) {
19 a[i] = Math.sin(2 * Math.PI * i * hz /
20 StdAudio.SAMPLE_RATE);
21 }
22
23 // play it using standard audio
24 StdAudio.play(a);
25 }
26 }
```

**Đáp.** Với số nguyên, dùng `o` cho hệ bát phân, `x` cho hệ thập lục phân. Với số thực, dùng `e` hoặc `g` để dùng định dạng khoa học. Ngoài ra còn có rất nhiều định dạng cho ngày tháng, thời gian. Cụ thể hãy vào link này

<http://docs.oracle.com/javase/6/docs/api/java/util/Formatter.html#syntax>

**Hỏi.** Làm sao để in ký hiệu `%` với `printf()` ?

**Đáp.** Sử dụng `%%`.

**Hỏi.** Điều gì xảy ra nếu chương trình của tôi cố gắng đọc dữ liệu từ đầu vào chuẩn sau khi nó đã hết dữ liệu ?

**Đáp.** Bạn sẽ nhận được lỗi sau

```
java.lang.RuntimeException: Tried to read from empty stdin
```

Hãy dùng `StdIn.isEmpty()` để kiểm tra xem đầu vào chuẩn còn có dữ liệu.

**Hỏi.** Làm thế nào để kết thúc dòng ?

**Đáp.** Các hệ điều hành khác nhau sử dụng các biểu tượng khác nhau. Trên các hệ thống Unix, các ký tự xuống dòng là `'\n'`, trên Windows mỗi dòng được kết thúc bằng một chuỗi ký tự `\r\n`, trên máy Mac mỗi dòng được kết thúc bởi chuỗi `\n\r`. Khi viết một chương trình, bạn nên tránh sử dụng tính năng cụ thể của hệ điều hành. Nếu không chương trình có thể không làm việc như mong đợi trên các hệ thống khác. Hãy sử dụng `System.out.println()` để in dấu xuống dòng, hoặc dùng lệnh sau để lấy chuỗi ký tự xuống dòng

```
String NEWLINE = System.getProperty("line.separator");
```

**Hỏi.** Đoạn mã nào dưới đây hiệu quả hơn ?

```
String s;
while (!StdIn.isEmpty()) {
 s = StdIn.readString();
 ...
}

while (!StdIn.isEmpty()) {
 String s = StdIn.readString();
 ...
}
```

**Đáp.** Hiệu quả giống nhau, đoạn mã thứ hai có phong cách tốt hơn vì nó giới hạn phạm vi của biến `s`.

**Hỏi.** Hệ tọa độ của Java và `StdDraw` khác nhau thế nào ?

**Đáp.** `StdDraw` sắp xếp các trục tọa độ Descartes với `(0, 0)` ở phía dưới bên trái, trong khi Java sử dụng `(0, 0)` phía trên bên trái. `StdDraw` sử dụng tọa độ giá trị thực, trong khi Java sử dụng tọa độ nguyên.

**Hỏi.** Làm thế nào để tạo ra màu sắc khi sử dụng thư viện đồ họa?

**Đáp.** Hãy vào link sau để xem cách sử dụng màu sắc trong Java.

<http://introcs.cs.princeton.edu/java/15inout/colors.html>

**Hỏi.** Khác biệt chính giữa các định dạng đồ họa PNG, JPEG, và PostScript là gì?

**Đáp.** Đồ họa trên hầu hết các trang web có định dạng PNG, GIF, hoặc JPEG. Cả ba định dạng hiển thị ảnh theo dòng (raster-based). PNG và GIF rất lý tưởng cho việc hiển thị ảnh có nhiều các đường thẳng và hình hình học, trong khi JPEG là thích hợp nhất cho máy ảnh số. PostScript là một định dạng dựa trên véc-tơ. Ví dụ, nó đại diện một vòng tròn bằng một đối tượng hình học thay vì một tập hợp hàng ngàn điểm ảnh. Chất lượng không bị suy giảm nếu bạn phóng to hoặc thu nhỏ nó. Vì lý do này, hầu hết các máy in sử dụng PostScript để in tài liệu và đồ họa.

**Hỏi.** Thông báo lỗi `Exception in thread "main" java.lang.NoClassDefFoundError: StdIn` nghĩa là gì?

**Đáp.** Bạn có thể quên đặt `StdIn.java` (`StdIn.class`) trong thư mục làm việc hiện tại của bạn và biên dịch nó.

**Hỏi.** Làm thế nào tôi có thể tạo ra ảnh động GIF ?

**Đáp.** Hiện nay thư viện đồ họa của chúng tôi chưa hỗ trợ lưu định dạng tập tin ảnh động như MPEG, GIF, hoặc AVI. Tuy nhiên, với một trình tiện ích bên ngoài (ví dụ như ImageMagick) việc này không phải là khó khăn, và chúng tôi đã sử dụng nó để tạo ra một số hình ảnh động trên website này. Ý tưởng là sử dụng `StdDraw.save()` lưu lại một chuỗi các file PNG có tên `frame100.png`, `frame101.png`, `frame102.png`, v.v... Giả sử ImageMagick được cài đặt (Unix, Linux, OS X, và Windows đều được hỗ trợ), bạn có thể sử dụng lệnh

```
convert -delay 10 -loop 5 frame*.png duke.gif
```

Lệnh trên nối các ảnh `frame*.png` theo thứ tự từ điển vào tập tin `duke.gif`, mỗi ảnh hiển thị 10 ms, lặp tất cả 5 lần.

## 5.8 Bài tập

1. Viết chương trình `MaxMin.java` đọc các số nguyên từ đầu vào chuẩn (`StdIn`) và in ra giá trị lớn nhất và giá trị nhỏ nhất.
2. Sửa chương trình bài trước để bắt buộc người dùng nhập các số dương (nếu không cần in thông báo để nhập lại).
3. Viết chương trình `Stats.java` nhập số nguyên  $N$  từ dòng lệnh, sau đó đọc  $N$  số thực từ đầu vào chuẩn rồi in ra *giá trị trung bình* và *độ lệch chuẩn* của dãy số đó theo công thức sau

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i, \quad \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}$$

4. Mở rộng chương trình ở bài trước để lọc ra các số cách xa trung bình của dãy hơn 1,5 lần độ lệch chuẩn.
5. Viết chương trình `LongestRun.java` đọc từ đầu vào chuẩn dãy số nguyên và in ra dãy các số giống nhau liên tục dài nhất. Ví dụ dãy 1 2 2 1 5 1 1 7 7 7 7 1 1 cho kết quả 4 7 (Dãy liên tục dài nhất có 4 con số 7).
6. Viết bộ lọc một dãy số nguyên từ đầu vào chuẩn, loại bỏ đi các con số trùng lặp. Ví dụ dãy 1 2 2 1 5 1 1 7 7 7 7 1 1 1 1 1 1 1 1 cho kết quả 1 2 1 5 1 7 1.
7. Viết chương trình nhập số  $N$  từ dòng lệnh và nhập  $N-1$  số nguyên khác nhau trong khoảng  $[1, N]$  từ đầu vào chuẩn và in ra giá trị thiếu.
8. Viết chương trình `GeometricMean.java` nhập các số thực dương từ đầu vào chuẩn và in ra trung bình hình học của các số này

$$\text{trung bình hình học} = \left( \prod_{i=1}^N x_i \right)^{1/N}$$

Viết tiếp chương trình `HarmonicMean.java` tính trung bình điều hòa của  $N$  số thực.

$$\text{trung bình điều hòa} = \frac{1}{N} \left( \sum_{i=1}^N \frac{1}{x_i} \right)$$

9. Chương trình `Dragon.java` in gì với câu lệnh sau

```
echo F F | java Dragon | java Dragon | java Dragon
```

```

1 public class Dragon {
2 public static void main(String[] args) {
3 String dragon = StdIn.readString();
4 String nogard = StdIn.readString();
5 StdOut.print(dragon + "L" + nogard);
6 StdOut.print(" ");
7 StdOut.print(dragon + "R" + nogard);
8 StdOut.println();
9 }
10 }

```

10. Viết chương trình `TenPerLine.java` đọc dãy số nguyên trong khoảng  $[0, 99]$  và in chúng ra mỗi dòng 10 số. Sau đó viết chương trình `RandomIntSeq.java` lấy 2 đối số  $M, N$  từ dòng lệnh và in ra  $N$  số nguyên ngẫu nhiên trong khoảng  $[0, M-1]$ . Kiểm tra chương trình với lệnh

```
java RandomIntSeq 200 100 | java TenPerLine
```

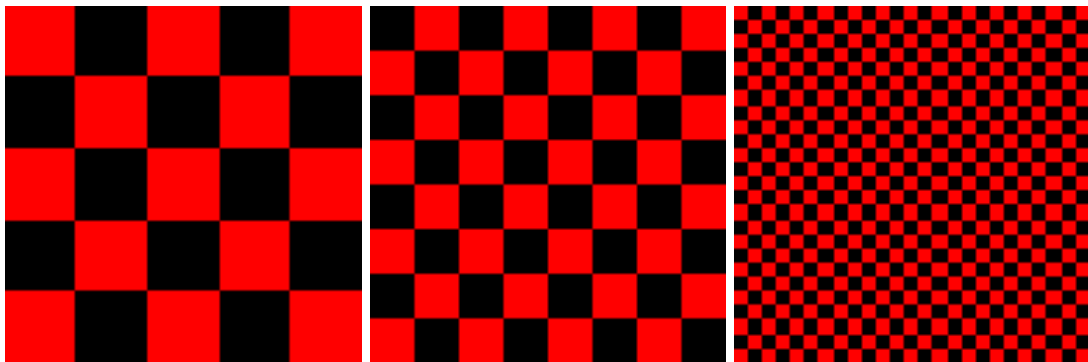
11. Viết chương trình `WordCount.java` đọc đầu vào chuẩn và in ra số từ trong đó. Một từ là một đoạn ký tự không có khoảng trống và được bao quanh bởi khoảng trống.
12. Viết chương trình đọc vào từ đầu vào chuẩn các dòng, mỗi dòng bao gồm 1 cái tên (sinh viên) và 2 số nguyên (điểm môn học), sau đó in ra thông tin đã đọc theo dạng bảng (giống hàng các cột) và thêm thông tin điểm trung bình các môn học của mỗi sinh viên trên dòng tương ứng.
13. Trong các yêu cầu sau đây, yêu cầu nào đòi hỏi ta phải ghi lại toàn bộ các giá trị đã đọc vào (trong mảng), yêu cầu nào chỉ cần một số cố định các biến.
- Tính giá trị lớn nhất, nhỏ nhất.
  - In giá trị lớn thứ  $k$ .
  - In tổng bình phương các số.
  - In trung bình cộng.
  - In số phần trăm số lớn hơn trung bình cộng.
  - In các số đã đọc theo thứ tự tăng dần.
  - In các số theo thứ tự ngẫu nhiên.
14. Viết chương trình in bảng số tiền thanh toán hàng tháng, tiền gốc còn lại, và tiền lãi đã trả cho một khoản vay, lấy ba con số từ dòng lệnh: số năm vay, tiền vay gốc, và lãi suất (xem bài tập 2.24).
15. Viết chương trình `Closest.java` dùng ba đối số  $x, y, z$  từ dòng lệnh, đọc từ đầu vào chuẩn một chuỗi các tọa độ  $(x_i, y_i, z_i)$ , và in ra tọa độ của điểm gần  $(x, y, z)$  nhất. Nhớ rằng bình phương khoảng cách giữa  $(x, y, z)$  và  $(x_i, y_i, z_i)$  là  $(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2$ . Để chương trình hiệu quả, dùng sử dụng `Math.sqrt()` hoặc `Math.pow()`.
16. Viết chương trình `Centroid.java` tính vị trí và trọng tâm của một tập các đối tượng trên mặt phẳng với khối lượng khác nhau. Nhập vào số đối tượng  $N$  từ dòng lệnh và các giá trị  $(x_i, y_i, m_i)$  từ đầu vào chuẩn. Công thức tính như sau

$$m = m1 + m2 + \dots + mN$$

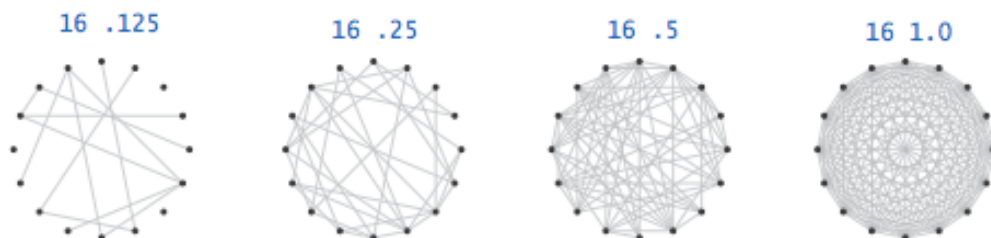
$$x = (m1 \ x1 + \dots + mN \ xN) / m$$

$$y = (m1 \ y1 + \dots + mN \ yN) / m$$

17. Viết chương trình `SignalAnalyzer.java` đọc trong một chuỗi các con số thực giữa -1 và 1 rồi in ra *cường độ trung bình*, *công suất trung bình*, và *số lần đảo chiều*. Cường độ trung bình là trung bình giá trị tuyệt đối của dữ liệu. Công suất trung bình là trung bình của bình phương dữ liệu. Số lần đảo chiều là số lần dữ liệu chuyển đổi giá trị dữ liệu từ một số âm sang một số dương, hoặc ngược lại. Ba số liệu thống kê này được sử dụng rộng rãi trong phân tích tín hiệu số.
18. Viết chương trình `CheckerBoard.java` dùng đối số dòng lệnh `N` và in bàn cờ `N x N` với dùng hình vuông màu đỏ và đen. Hình vuông màu đỏ ở phía dưới bên trái.



19. Viết chương trình lấy từ dòng lệnh một số nguyên `N` và một giá trị thực `p` (giữa 0 và 1), vẽ `N` điểm cách đều nhau trên chu vi của một vòng tròn, sau đó, với xác suất `p`, nối mỗi cặp điểm bằng một đường màu xám.



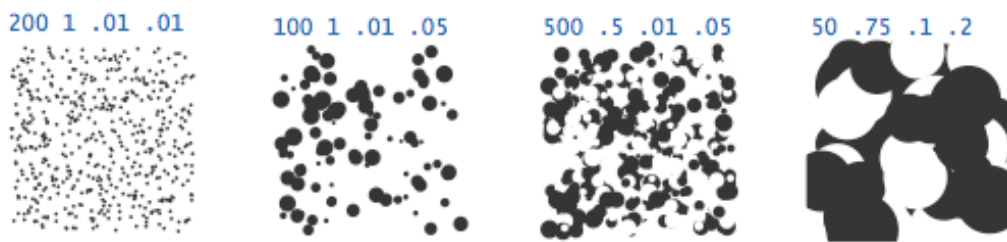
20. Viết chương trình vẽ các hình RÔ, CỎ, BÍCH, TÉP. Để vẽ chất CỎ, hãy vẽ một hình thoi, sau đó đính 2 hình bán nguyệt lên phía trên bên trái và bên phải.
21. Viết chương trình `Rose.java` lấy số `N` từ dòng lệnh và in ra bông hoa có `N` cánh với `N` lẻ và `2N` cánh với `N` chẵn. Gợi ý: sử dụng hệ tọa độ cực  $(r, \theta)$  và vẽ đồ thị hàm số  $r = \sin(N \times \theta)$  với  $\theta$  trong khoảng  $[0, 2\pi]$ .



22. Viết chương trình **Banner.java** lấy chuỗi ký tự **s** từ dòng lệnh và hiển thị biểu ngữ đó chạy trên màn hình, di chuyển từ trái sang phải và cuộn lại khi biểu ngữ vượt quá mép bên phải. Có thể thêm đối số dòng lệnh thứ hai để kiểm soát tốc độ chạy biểu ngữ.
23. Sửa chương trình **PlayThatTune.java** lấy đối số dòng lệnh **A** điều khiển âm lượng (nhân mỗi giá trị mẫu với **A**) và trường độ **B** (nhân trường độ với **B**).
24. Viết chương trình sử dụng **StdDraw** tạo ra các thiết kế dưới đây.



25. Viết chương trình lấy tên tập tin **.wav** và tốc độ **r** từ dòng lệnh, sau đó chơi nhạc với tốc độ **r**. Đầu tiên, hãy dùng **StdAudio.read()** để đọc tập tin vào mảng **a[]**. Nếu **r = 1**, chỉ cần chơi lại **a[]**. Nếu không, tạo một mảng **b[]** có kích thước xấp xỉ **r \* a.length**. Nếu **r < 1**, các giá trị của **b** sẽ lấy mẫu từ **a**. Nếu **r > 1**, các giá trị của **b** sẽ nội suy từ **a**.
26. Viết chương trình **Circles** in ra các hình tròn có kích thước và màu sắc ngẫu nhiên giống như dưới đây dùng 4 đối số dòng lệnh: số vòng tròn **N**, xác suất màu đen **p**, bán kính nhỏ nhất **minRadius**, bán kính lớn nhất **maxRadius**



## 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

