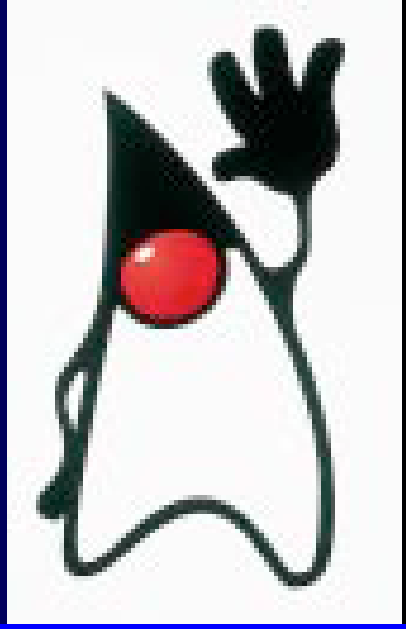


Lập trình Java

Phạm Quang Dũng
BM KHMT - Khoa CNTT - Trường ĐHNN I

MỞ ĐẦU

- Mục đích của khóa học
- Tổ chức của khóa học
- Tài liệu tham khảo
- Phần mềm cần thiết
- Thiết lập môi trường làm việc

Mục đích của khóa học

- Khi hoàn thành khóa học, bạn sẽ hiểu:
  - Cách tạo, biên dịch, và chạy các chương trình Java
  - Các kiểu dữ liệu cơ sở
  - Luồng điều khiển Java - Java control flow
  - Phương thức - Methods
  - Mảng - Arrays
  - Lập trình hướng đối tượng (Object-oriented programming)
  - Các lớp Java lõi (Core Java classes: swing, exception, internationalization, multithreading, multimedia, I/O, networking, Java Collections Framework)

Mục đích của khóa học (tiếp)

- Bạn sẽ có thể:
  - Viết các chương trình đơn giản sử dụng các kiểu dữ liệu cơ sở, các cấu trúc điều khiển, các phương thức và mảng.
  - Tạo và dùng các phương thức
  - Phát triển một giao diện GUI và các Java applets
  - Viết các dự án thú vị
  - Thiết lập một nền tảng chắc chắn trên tư tưởng Java

Tổ chức của khóa học

- Phần I: Cơ bản về lập trình Java
  - Chương 1: Giới thiệu về Java
  - Chương 2: Các toán tử và các kiểu dữ liệu cơ sở
  - Chương 3: Các cấu trúc điều khiển
  - Chương 4: Phương thức - Methods
  - Chương 5: Mảng - Arrays

Tổ chức của khóa học (tiếp)

- Phần II: Lập trình hướng đối tượng (Object-Oriented Programming)
  - Chương 6: Đối tượng và lớp (Objects and Classes)
  - Chương 7: Strings
  - Chương 8: Class Inheritance and Interfaces
  - Chương 9: Object-Oriented Software Development

Tổ chức của khóa học (tiếp)

- Phần III: Lập trình giao diện đồ họa (GUI Programming)
  - Chương 10: Bắt đầu với lập trình GUI
  - Chương 11: Tạo giao diện người dùng
  - Chương 12: Các Applet và GUI nâng cao

Tổ chức của khóa học (tiếp)

- Phần IV: Phát triển các dự án toàn diện [Optional]
  - Chương 13: Xử lý ngoại lệ - Exception Handling
  - Chương 14: Quốc tế hóa - Internationalization
  - Chương 15: Đa luồng - Multithreading
  - Chương 16: Multimedia
  - Chương 17: Input and Output
  - Chương 18: Networking
  - Chương 19: Java Data Structures

Tài liệu tham khảo

1. Introduction to Java Programming, 5th edition
   - Y. Daniel Liang, NXB Prentice Hall, 2004
2. Giáo trình lý thuyết và bài tập Java,
   - Nguyễn Tiến Dũng, NXB Giáo dục, 1999
3. Programming in Java (slides)
4. The Java Language Specification, 3th edition (pdf)
5. Java for students (slides)
6. The Java Tutorial (java.sun.com), ……

☞ Hãy tìm tại website: www.hau1.edu.vn/it/pqdung

Từ vựng

- JRE, Java Runtime Environment
  - Phần mềm cho phép bạn chạy các chương trình Java trên máy tính.
- JDK, Java Development Kit; còn gọi là
- SDK, System Development Kit
  - Phần mềm cho phép bạn tạo và chạy các chương trình Java trên máy tính.
- IDE, Integrated Development Environment
  - Công cụ giúp viết và chạy các chương trình dễ dàng hơn.

Phần mềm cần thiết

- Java SDK 5 (gồm cả JRE)
  - http://java.sun.com/j2se/1.5.0/download.html
  - [Optional] Download the Java documentation
- JCreator 3.0
- JBuilder 2005 Foundation [Optional]
  - http://www.borland.com/downloads/download_jbuilder.html

☞ Tất cả phần mềm trên là free và có thể download tại:

- http://www.hau1.edu.vn/it/pqdung/download

JCreator

- JCreator là một IDE. Nó bao gồm:
  - một trình soạn thảo (editor), để viết chương trình
  - một chương trình gỡ rối (debugger), giúp tìm các lỗi
  - một khung nhìn (viewer), để xem các phần của chương trình
  - một cách thức dễ dàng để chạy các chương trình Java và xem tài liệu

Cấu hình máy tính tối thiểu

- 500 MHz Pentium or better
- 256 MB RAM
- 300 MB Hard disk - JDK 1.5
- 10 MB HD - JCreator
- 240 MB HD - JBuilder 2005 Foundation [Optional]

Thiết lập môi trường làm việc

- Sau khi cài đặt Java SDK 1.5.0 vào thư mục C:\Program Files\Java\jdk1.5.0 tại cửa sổ dòng lệnh Windows (cmd.exe) lần lượt chạy 2 dòng lệnh:
- set path=C:\Program Files\Java\jdk1.5.0\bin ↵
- set classpath=. ↵

thiết lập biến đường dẫn để có thể gọi các chương trình chạy được của Java từ bất kỳ thư mục nào

thiết lập thư mục chứa các lớp người dùng là thư mục hiện tại

LẬP TRÌNH JAVA

Chương 1:
Giới thiệu về Java

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung của chương 1

- Java là gì?
- Lịch sử hình thành và phát triển
- Các đặc điểm của Java
- Bắt đầu với lập trình Java
  - Tạo, biên dịch và chạy một ứng dụng Java

Java là gì?

- Java là một ngôn ngữ lập trình (*programming language):* một ngôn ngữ mà bạn có thể học cách viết và máy tính có thể hiểu được
- Java hiện đang là một ngôn ngữ rất phổ biến
- Java là một ngôn ngữ mạnh và có tầm bao quát rộng
  - nhưng nó ***không đơn giản***!
- Được so sánh với C++, Java rất "tao nhã" (elegant)

Lịch sử

- 1990, James Gosling và Sun Microsystems
- Tên ban đầu: Oak (cây sồi)
- Java, 20/05/1995, Sun World
- HotJava
  - Trình duyệt Web hỗ trợ Java đầu tiên
- JDK Evolutions
- J2SE, J2ME, and J2EE

Các đặc điểm của Java

- Java is simple
- Java is object-oriented
- Java is distributed
- Java is interpreted
- Java is robust
- Java is secure
- Java is architecture-neutral
- Java is portable
- Java's performance
- Java is multithreaded
- Java is dynamic

- đơn giản
- hướng đối tượng
- phân tán
- thông dịch
- mạnh mẽ
- bảo mật
- kiến trúc trung tính
- khả chuyển
- hiệu quả cao
- đa tuyến
- linh động

Các phiên bản JDK (Java Devenlopment Kit)

- Java 1
  - JDK 1.02 (1995)
  - JDK 1.1 (1996)
- Java 2
  - SDK v 1.2 (JDK 1.2, 1998)
  - SDK v 1.3 (JDK 1.3, 2000)
  - SDK v 1.4 (JDK 1.4, 2002)
  - SDK v 5.0 (JDK 5.0, 2004)

JDK Editions

- Java Standard Edition (J2SE)
  - J2SE có thể được dùng để phát triển các ứng dụng hoặc các applet độc lập phía client (client-side).
- Java Enterprise Edition (J2EE)
  - J2EE có thể được dùng để phát triển các ứng dụng phía server (server-side) như các Java servlet và Java ServerPages.
- Java Micro Edition (J2ME).
  - J2ME có thể được sử dụng để phát triển các ứng dụng cho các thiết bị di động như ĐTDĐ.

Bài giảng sử dụng J2SE để giới thiệu lập trình Java.

Java IDE Tools

- JCreator
- Forte by Sun MicroSystems
- Borland JBuilder
- Microsoft Visual J++
- WebGain Café
- IBM Visual Age for Java

Bắt đầu với lập trình Java

- Một chương trình Java đơn giản
- Biên dịch chương trình
- Chạy chương trình

Một chương trình Java đơn giản

Ví dụ 1.1

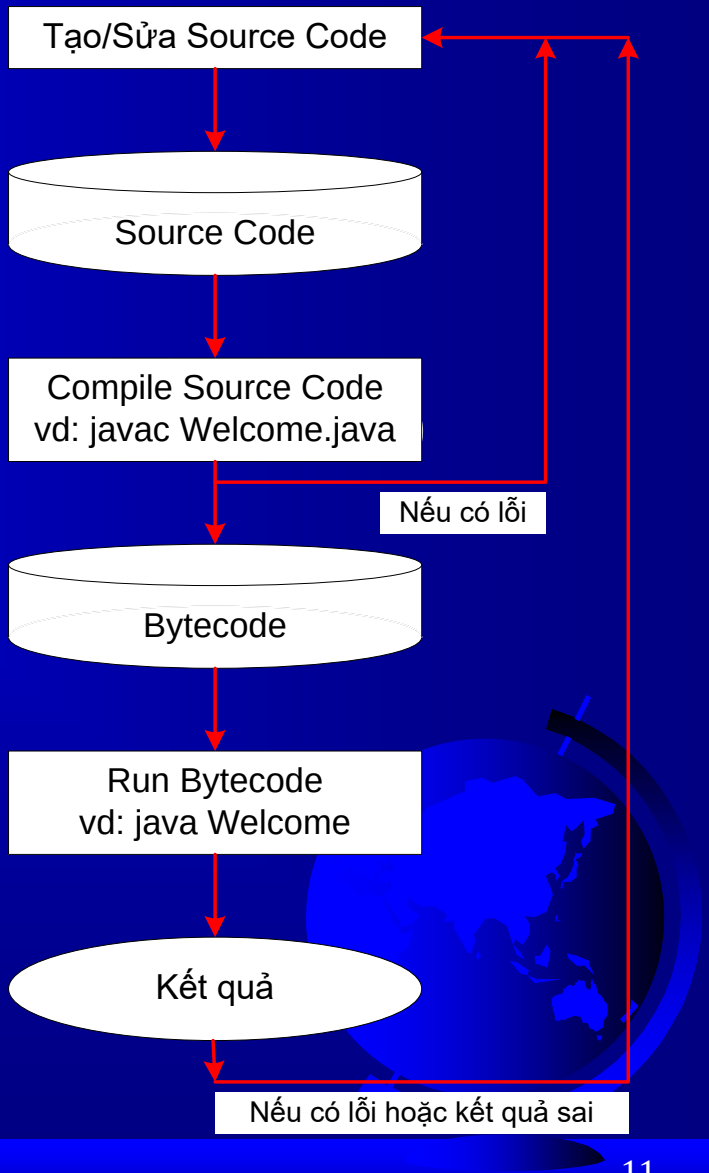
```
// Chương trình in dòng: Welcome to Java!
package ch01;
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Source

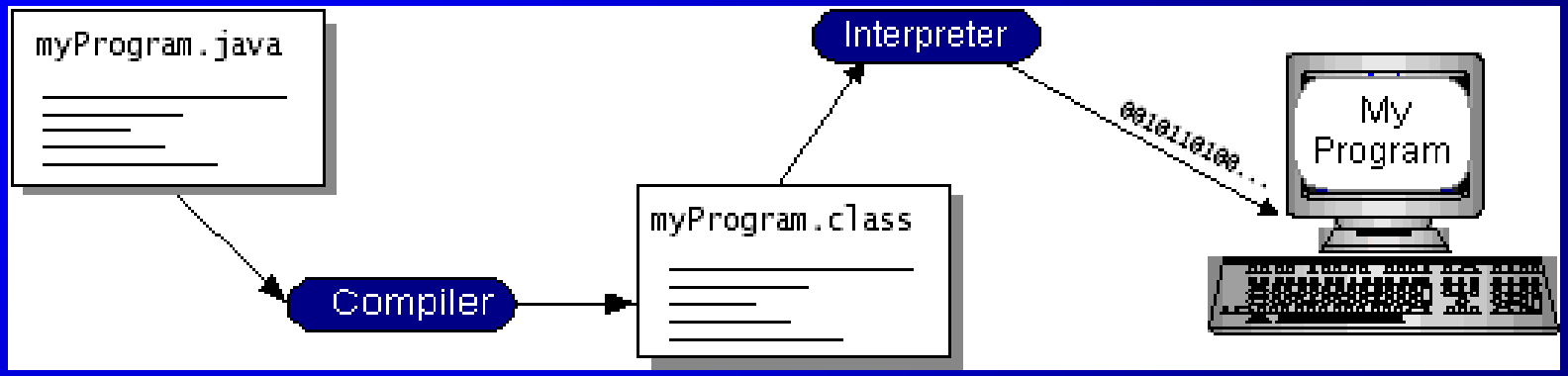
Run

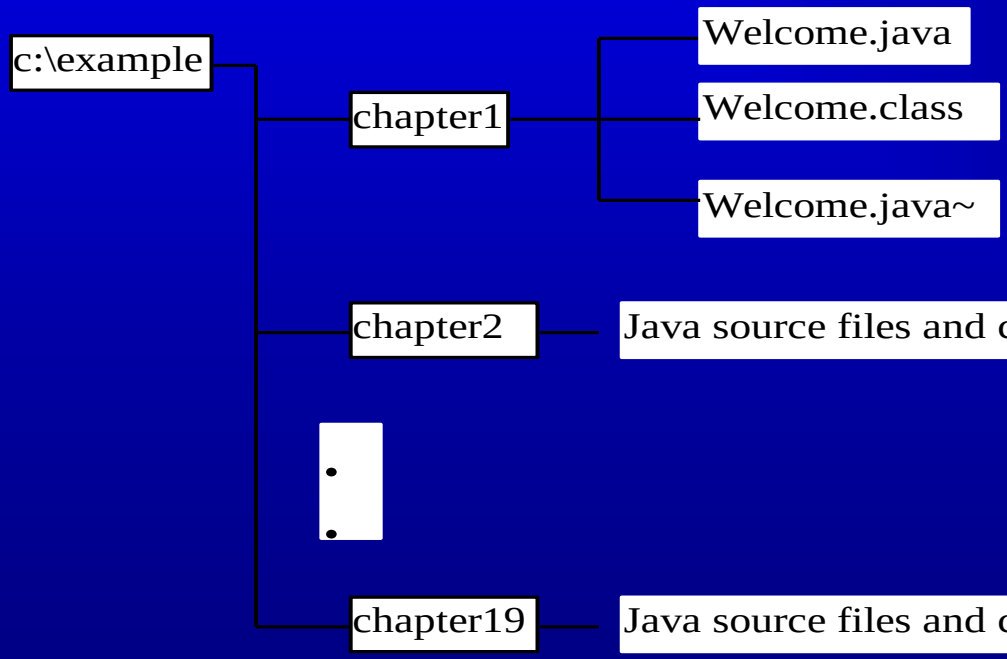
Tạo, biên dịch, chạy chương trình

- Tạo:
  - Soạn thảo chương trình (Notepade, Wordpad…)
  - Ghi tệp tên Welcome.java vào thư mục C:\javapro
- Biên dịch:
  - Trên cửa sổ lệnh (cmd.exe)
  - `cd\ ↵`
  - `cd javapro ↵`
  - `javac Welcome.java ↵`
- Chạy:
  - `java Welcome ↵`



Biên dịch và chạy một chương trình





Cần xác định các tệp được lưu trong thư mục nào

Các thành phần của một chương trình Java

- Chú giải - Comments
- Đóng gói - Package
- Từ khóa - Reserved words
- Từ bổ nghĩa - Modifiers
- Câu lệnh - Statements
- Khối - Blocks
- Lớp - Classes
- Phương thức - Methods
- Phương thức chính - The main method

Comments

- Trong Java, các chú giải có thể được đặt :
  - sau 2 dấu gạch chéo // trên 1 dòng
  - giữa dấu mở /* và đóng */ trên 1 hoặc nhiều dòng
- Khi trình biên dịch gặp:
  - //, nó bỏ qua tất cả các ký tự sau // trên dòng đó
  - /*, nó quét tìm đến */ tiếp sau và bỏ qua mọi ký tự nằm giữa /* và */.

Package

Dòng thứ hai trong chương trình (package ch01;) xác định một tên gói, ch01, cho class Welcome. Forte biên dịch source code trong tệp Welcome.java, tạo ra tệp Welcome.class, và lưu Welcome.class trong thư mục ch01.

Reserved Words

- *Reserved words* hay *keywords* là những từ có nghĩa xác định đối với trình biên dịch và không thể sử dụng cho các mục đích khác trong chương trình.
- VD: khi trình biên dịch gặp từ class, nó hiểu rằng từ ngay sau class là tên của class.
- Các từ khóa khác trong ví dụ 1.1 là public, static, và void. Chúng sẽ được giới thiệu ở phần sau.

Modifiers (Từ bổ nghĩa)

- Java sử dụng một số từ khóa gọi là ***modifiers*** để xác định các thuộc tính của dữ liệu, các phương thức, lớp, và chúng có thể được sử dụng như thế nào.
- Các ví dụ từ bổ nghĩa là public, static, private, final, abstract, và protected.
- Một dữ liệu, phương thức, hoặc lớp public thì có thể truy nhập được bởi chương trình khác. Một dữ liệu hay phương thức private thì không thể.
- Modifiers sẽ được thảo luận ở Chương 6, "Objects and Classes."

Statements

- Một câu lệnh (*statement)* đại diện cho một hành động hoặc một chuỗi các hành động.
- Câu lệnh System.out.println("Welcome to Java!") trong chương trình ví dụ 1.1 là một câu lệnh hiển thị lời chào "Welcome to Java!".
- Mọi câu lệnh trong Java kết thúc bởi một dấu chấm phẩy (;).

Blocks

- Một cặp dấu ngoặc nhọn trong một chương trình hình thành một khối nhóm các thành phần của một chương trình.
- Vai trò tương tự cặp từ khóa Begin …end; trong Pascal

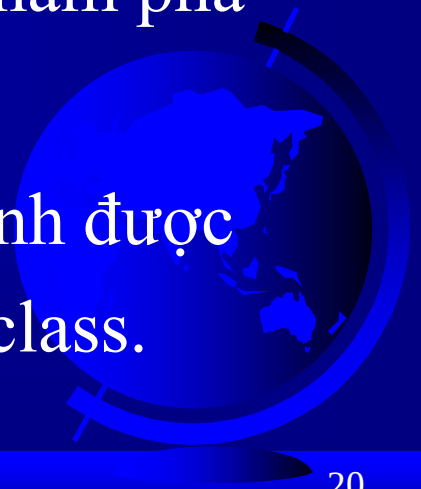
```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Class block

Method block

Classes

- *Class* (*lớp)* là thiết yếu trong xây dựng cấu trúc Java. Một class là một khuôn mẫu hay bản thiết kế cho các đối tượng.
- Để lập trình trong Java, bạn phải hiểu các class và có thể viết, sử dụng chúng.
- Những bí ẩn của class sẽ tiếp tục được khám phá dần xuyên suốt khóa học.
- Bây giờ bạn chỉ cần hiểu một chương trình được xác định bằng cách sử dụng một hay nhiều class.



Methods

- System.out.println là gì? Đó là một ***method*** *(phương thức)*: một tập các câu lệnh thực hiện một chuỗi các thao tác để hiển thị một thông tin trên màn hình.
- Nó thậm chí có thể được sử dụng mà không cần hiểu đầy đủ chi tiết nó làm việc như thế nào.
- Nó được sử dụng bằng cách gọi một câu lệnh với tham số chuỗi ký tự (string) được bao bởi cặp dấu nháy kép. Trong trường hợp này, tham số là "Welcome to Java!"
- Bạn có thể gọi phương thức println với các tham số khác nhau để in ra những message khác nhau.

main Method

- main method cung cấp sự kiểm soát luồng chương trình. Trình biên dịch Java thực hiện ứng dụng bằng cách gọi đến main method.
- Mọi chương trình Java phải có main method, nó là điểm khởi đầu khi thực hiện chương trình.
- Dạng thức của main method:

```
public static void main(String[] args) {
  // Statements;
}
```

Hiển thị văn bản trong Message Dialog Box

- bạn có thể sử dụng phương thức showMessageDialog trong lớp JOptionPane. JOptionPane là một trong nhiều lớp được định nghĩa trước trong hệ thống Java để có thể tái sử dụng.

Source

Phương thức showMessageDialog

```
JOptionPane.showMessageDialog(null,
  "Welcome to Java!",
     "Example 1.2 Output",
  JOptionPane.INFORMATION_MESSAGE));
```

LẬP TRÌNH JAVA

Chương 2:
Các toán tử và các kiểu dữ liệu cơ bản

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung của chương 2

- Giới thiệu lập trình qua một ví dụ
- Các tên, biến và hằng
- Các kiểu dữ liệu cơ sở
  - byte, short, int, long, float, double, char, boolean
- Biểu thức
- Các toán tử Operators, Precedence, Associativity, Operand Evaluation Order: ++, --, *, /, %, +=, -=, *=, /=, %=, ^, &, |, +, -,
- Nhận dữ liệu vào từ các Input Dialog Boxes
- Case Studies (Computing Mortgage, and Computing Changes)
- Style and Documentation Guidelines
- Syntax Errors, Runtime Errors, and Logic Errors

Giới thiệu lập trình qua 1 ví dụ

Ví dụ 2.1: Tính diện tích hình tròn

ComputeArea

Run

Tên - Identifiers

- Một tên là một chuỗi các ký tự gồm các chữ, số, dấu gạch dưới (_), và dấu dollar ($).
- Một tên phải bắt đầu bởi một chữ, dấu gạch dưới (_), hoặc dấu dollar ($). Nó không thể bắt đầu bởi một số.
- Một tên không thể là một từ khóa.
- Một tên không thể là true, false, hoặc null.
- Một tên có thể có độ dài bất kỳ.

Biến - Variables

```
// Tinh dien tich thu nhat
bankinh = 1.0;
dientich = bankinh*bankinh*3.14159;
System.out.println("Dien tich bang " +
dientich + " voi ban kinh la " + bankinh);
```

Khai báo biến

Dạng thức: `datatype variableName;`

Ví dụ:

```
int x;          // Khai báo x là một
                // biến nguyên (integer);
double bankinh
char a;
```

Lệnh gán và biểu thức gán

Dạng thức: variable = expression;

Ví dụ:

```
x = 1;                  // Gán 1 cho x;

bankinh = 1.0;          // Gán 1.0 cho bankinh;

a = 'A';                // Gán 'A' cho a;

x = x + 1;

dttg = Math.sqrt(p*(p-a)*(p-b)*(p-c)) ;
```

Khai báo và khởi tạo trong 1 lệnh

- `int i = 1, j = 5;`
- `double d = 1.4;`
- `float pi = 3.1416;`

Các câu lệnh trên có đúng không?

Hằng - Constants

- Dạng thức:

```
final datatype CONSTANTNAME = VALUE;
```

- Ví dụ:

```
final double PI = 3.14159;
final int SIZE = 3;
```

Các kiểu dữ liệu số

`byte`	8 bits
`short`	16 bits
`int`	32 bits
`long`	64 bits
`float`	32 bits
`double`	64 bits

Toán tử - Operators

+ - * / %

int i1 = 5/2 ; ⇒ kết quả là số nguyên i1 = 2

float i2 = 5.0/2 ; ⇒ kết quả là số thực i2 = 2.5

byte i3 = 5 % 2; ⇒ i3 = 1 (số dư của phép chia)

CHÚ Ý

- Các phép tính với số dấu chấm động được lấy xấp xỉ vì chúng được lưu trữ không hoàn toàn chính xác. Ví dụ:

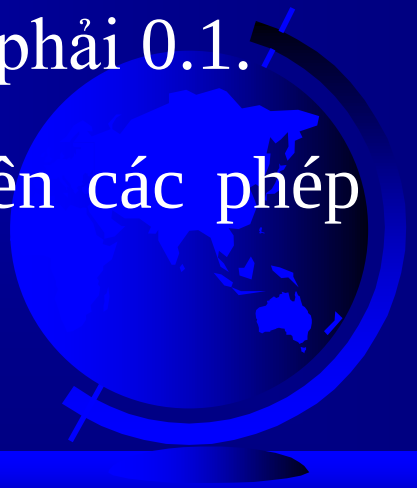
System.out.println(1 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);

hiển thị 0.5000000000000001, không phải 0.5

System.out.println(1.0 - 0.9);

hiển thị 0.09999999999999998, không phải 0.1.

- Các số nguyên được lưu trữ chính xác nên các phép tính với chúng cho kết quả chính xác.



Biểu thức toán học

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

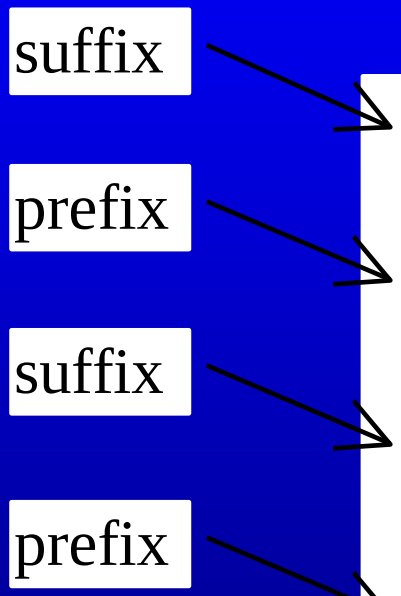
được chuyển thành công thức Java như sau:

(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)

Các toán tử gán tắt

Operator	*Example*	*Equivalent*
`+=`	`i+=8`	`i = i+8`
`-=`	`f-=8.0`	`f = f-8.0`
`*=`	`i*=8`	`i = i*8`
`/=`	`i/=8`	`i = i/8`
`%=`	`i%=8`	`i = i%8`

Các toán tử tăng và giảm




```
x++; // Same as x = x + 1;

++x; // Same as x = x + 1;

x--; // Same as x = x - 1;

--x; // Same as x = x - 1;
```

Các toán tử tăng và giảm (tiếp)

```
int i=10;
int newNum = 10*i++;
```

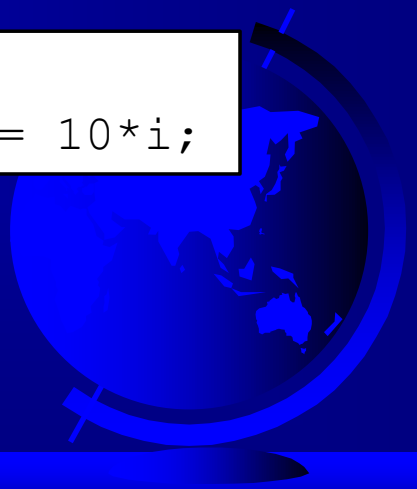
Equivalent to

```
int newNum = 10*i;
i = i + 1;
```

```
int i=10;
int newNum = 10*(++i);
```

Equivalent to

```
i = i + 1;
int newNum = 10*i;
```



Các toán tử tăng và giảm (tiếp)

- Sử dụng các toán tử tăng và giảm giúp các biểu thức ngắn gọn hơn, nhưng cũng làm cho chúng phức tạp và khó đọc hơn.
- Nên tránh sử dụng các toán tử này trong những biểu thức làm thay đổi nhiều biến hoặc sử dụng cùng một biến nhiều lần như sau: int k = ++i + i.

Biểu thức gán và Câu lệnh gán

- Trước Java 2, tất cả các biểu thức có thể được sử dụng như câu lệnh. Kể từ Java 2, chỉ những loại biểu thức sau có thể là câu lệnh:
- variable op= expression; // Với op là +, -, *, /, %
- ++variable;
- variable++;
- --variable;
- variable--;

Chuyển đổi dữ liệu kiểu số (Ép kiểu)

Xét các câu lệnh sau đây:

```
byte i = 100;
long k = i*3+4;
double d = i*3.1+k/2;

int x = k;  //(sai, int < long)
long k = x; //(đúng, long > int)
```

Luật chuyển

- Khi thực hiện một phép tính nhị phân chứa 2 toán hạng khác kiểu, Java tự động chuyển kiểu toán hạng theo luật sau:

1. Nếu một toán hạng kiểu double, toán hạng khác được chuyển đổi thành kiểu double.
2. Nếu không thì, nếu một toán hạng kiểu float, toán hạng khác được chuyển đổi thành kiểu float.
3. Nếu không thì, nếu một toán hạng kiểu long, toán hạng khác được chuyển đổi thành kiểu long.
4. Nếu không thì, cả hai toán hạng được chuyển đổi thành kiểu int.

Mức ưu tiên Ép kiểu

- `double`
- `float`
- `long`
- `int`
- `short`
- `byte`

Ép kiểu mở rộng và thu hẹp

Ép kiểu mở rộng

```
double d = 3; (mở rộng kiểu)
```

Ép kiểu thu hẹp

```
int i = (int)3.0; (thu hẹp kiểu)
```

Có sai không? int x = 5/2.0;

Kiểu dữ liệu ký tự

```
char letter = 'A'; (ASCII)
char numChar = '4'; (ASCII)
char letter = '\u0041'; (Unicode)
char numChar = '\u0034'; (Unicode)
```

4 chữ số hệ 16

Với các ký tự đặc biệt:

```
char tab = '\t';
```

Các ký tự đặc biệt

Description	*Escape Sequence*	*Unicode*
Backspace	`\b`	`\u0008`
Tab	`\t`	`\u0009`
Linefeed	`\n`	`\u000a`
Carriage return	`\r`	`\u000d`
Backslash	`\\`	`\u005C`
Single Quote	`\'`	`\u0027`
Double Quote	`\"`	`\u0022`

Appendix B: ASCII Character Set

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.1 ASCII Character Set in the Decimal Index

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

ASCII Character Set, cont.

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.2 ASCII Character Set in the Hexadecimal Index

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

Ép kiểu giữa kiểu ký tự và kiểu số

```
int i = 'a'; // tương tự int i = (int)'a';

char c = 97; // tương tự char c = (char)97;
```

Kiểu **boolean** và các toán tử

```
boolean a1 = true;
boolean a2 = false;

boolean b = (1 > 2);

boolean b2 = (1 == 2);
```

- Kết quả của phép so sánh là một giá trị logic Boolean: true hoặc false

Các toán tử so sánh

Operator	*Name*
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Các toán tử Boolean

Operator	*Name*	*Example*
!	not	`!b`
&&	and	`(1<x) && (x<100)`
||	or	`a1 || a2`
^	exclusive or	`a1 ^ a2`

Bảng chân lý của toán tử !

p	!p	Example
true	false	!(1 > 2) là true, vì (1 > 2) là false.
false	true	!(1 > 0) là false, vì (1 > 0) là true.

Bảng chân lý của toán tử &&

p1	p2	p1 && p2
F	F	F
F	T	F
T	F	F
T	T	T

Ví dụ:

(3 > 2) && (5 >= 5) là true, vì cả (3 > 2) và (5 >= 5) đều là true.

(3 > 2) && (5 > 5) là false, vì (5 > 5) là false.

Bảng chân lý của toán tử ||

p1	p2	p1 || p2
F	F	F
F	T	T
T	F	T
T	T	T

Ví dụ:

(2 > 3) || (5 > 5) là false, vì cả (2 > 3) và (5 > 5) đều là false.

(3 > 2) || (5 > 5) là true, vì (3 > 2) là true.

Bảng chân lý của toán tử ^

p1	p2	p1 ^ p2
F	F	F
F	T	T
T	F	T
T	T	F

Ví dụ:

(2 > 3) ^ (5 > 1) là true, vì (2 > 3) là false và (5 > 1) là true.

(3 > 2) ^ (5 > 1) là false, vì cả (3 > 2) và (5 > 1) đều là true.

Ví dụ

- System.out.println("Is " + num + " divisible by 2 and 3? " + ((num % 2 == 0) && (num % 3 == 0)));
- System.out.println("Is " + num + " divisible by 2 or 3? " + ((num % 2 == 0) || (num % 3 == 0)));
- System.out.println("Is " + num + " divisible by 2 or 3, but not both? " + ((num % 2 == 0) ^ (num % 3 == 0)));

Source Code

Run

Xác định năm nhuận?

- Một năm là năm nhuận nếu nó chia hết cho 4, nhưng không chia hết cho 100 hoặc nó chia hết cho 400. Source code xác định năm nhuận như sau:

```
boolean NamNhuan = ((nam % 4 == 0) &&
(nam % 100 != 0)) || (nam % 400 == 0);
```

Các toán tử & và |

&&: toán tử AND có điều kiện

&: toán tử AND không có điều kiện

||: toán tử OR có điều kiện

|: toán tử OR không có điều kiện

bt1 && bt2

(1 < x) && (x < 100)

(1 < x) & (x < 100)

Các toán tử & và | (tiếp)

Nếu x bằng 1, x bằng bao nhiêu sau khi thực hiện biểu thức?

```
(x > 1) & (x++ < 10);
```

Nếu x bằng 1, x bằng bao nhiêu sau khi thực hiện biểu thức?

```
(1 > x) && ( 1 > x++);
```

Và (1 == x) | (10 > x++);?

```
(1 == x) || (10 > x++);?
```

Thứ tự ưu tiên các toán hạng

Biểu thức sau được tính như thế nào?

```
3 + 4 * 4  >  5 * (4 + 3) - ++i
```

- Tất nhiên phải ưu tiên trong ngoặc trước, ngoài ngoặc sau
- Chỉ dùng ngoặc tròn
- Nhiều tầng ngoặc thì thứ tự ưu tiên ngoặc từ trong ra ngoài

Thứ tự ưu tiên các toán tử (tiếp)

1. var++, var--
2. +, - (dấu dương, âm), ++var,--var
3. (type) Casting (ép kiểu)
4. ! (Not)
5. *, /, % (nhân, chia thường, chia lấy phần dư)
6. +, - (cộng, trừ)
7. <, <=, >, >= (so sánh)
8. ==, !=; (đẳng thức)
9. & (AND không có điều kiện)
10. ^ (Exclusive OR)
11. | (OR không có điều kiện)
12. && (AND có điều kiện)
13. || (OR có điều kiện)
14. =, +=, -=, *=, /=, %= (toán tử gán)

Sự kết hợp toán tử - Operator Associativity

- Khi tính toán với 2 toán hạng có cùng mức ưu tiên, sự kết hợp toán tử sẽ xác định thứ tự các phép tính. Tất cả các toán tử nhị phân, ngoại trừ toán tử gán, là kết hợp trái (*left-associative*).

  a – b + c – d là tương đương với ((a – b) + c) – d

- Các toán tử gán là kết hợp phải. Do đó biểu thức

  a = b += c = 5 tương đương với a = (b += (c = 5))

Luật tính biểu thức

- Luật 1: Tính bất kỳ biểu thức con nào có thể tính được từ trái sang phải.
- Luật 2: Các toán hạng được áp dụng theo thứ tự ưu tiên của chúng.
- Luật 3: Luật kết hợp áp dụng cho 2 toán hạng cạnh nhau có cùng mức ưu tiên.

Ví dụ

- 3 + 4 * 4 > 5 * (4 + 3) - 1
- 3 + 4 * 4 > 5 * 7 - 1
- 3 + 16 > 5 * 7 - 1
- 3 + 16 > 35 - 1
- 19 > 35 – 1
- 19 > 34
- false

(1) Trong ngoặc trước

(2) Nhân

(3) Nhân

(4) Cộng

(5) Trừ

(6) Lớn hơn

Thứ tự tính toán toán hạng

Các quy tắc ưu tiên và kết hợp xác định thứ tự của các toán tử, nhưng không xác định thứ tự tính toán của các toán hạng nhị phân. Trong Java, các toán hạng được tính từ trái sang phải.

Toán hạng bên trái của một toán tử nhị phân được tính trước bất kỳ phần nào của toán hạng bên phải.

Luật này có quyền ưu tiên hơn các luật đã nêu.

Thứ tự tính toán toán hạng (tiếp)

- Khi các toán hạng có hiệu ứng lề (*side effects*), thứ tự tính toán của các toán hạng rất cần quan tâm.
- Ví dụ, *x* sẽ bằng 1 trong đoạn lệnh sau, vì a được tính bằng 0 trước khi ++*a* tăng nó lên thành 1.

```
int a = 0;
int x = a + (++a); // 0 + 1
```

- Nhưng *x* sẽ bằng 2 trong đoạn lệnh sau, vì ++*a* tăng nó lên thành 1, rồi cộng với chính nó.

```
int a = 0;
int x = ++a + a;    // 1 + 1
```

Ví dụ

- 3 + 4 * 4 > 5 * (4 + 3) - 1

(1) Biểu thức con đầu tiên có thể được tính từ bên trái

- 3 + 16 > 5 * (4+3) - 1

(2) Cộng

- 3 + 16 > 5 * (4+3) - 1

(3) Cộng trong ngoặc

- 19 > 5 * 7 - 1

(4) Nhân

- 19 > 35 – 1

(5) Trừ

- 19 > 34

(6) Lớn hơn

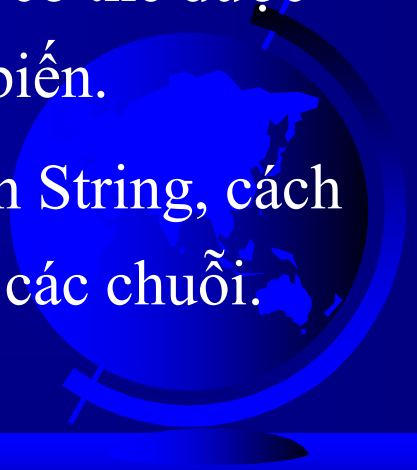
- false

Kiểu chuỗi ký tự

- Kiểu char chỉ biểu diễn 1 ký tự. Để biểu diễn một chuỗi ký tự, sử dụng kiểu dữ liệu String. Ví dụ:

```
String message = "Welcome to Java";
```

- String là một lớp được định nghĩa trước trong thư viện Java giống như System class và JOptionPane class.
- Kiểu String không phải là kiểu cơ sở mà là một kiểu tham chiếu (*reference type*). Bất kỳ lớp Java nào cũng có thể được sử dụng như một kiểu tham chiếu thay cho một biến.
- Hiện tại, bạn chỉ cần hiểu cách khai báo một biến String, cách gán một chuỗi ký tự cho một biến, và cách ghép các chuỗi.



Ghép chuỗi

- String message = "Welcome " + "to " + "Java";

  // ⇒ message = "Welcome to Java"

- String s = "Chuong" + 2;

  // ⇒ s trở thành Chuong2

- String s1 = "Hello" + 'B';

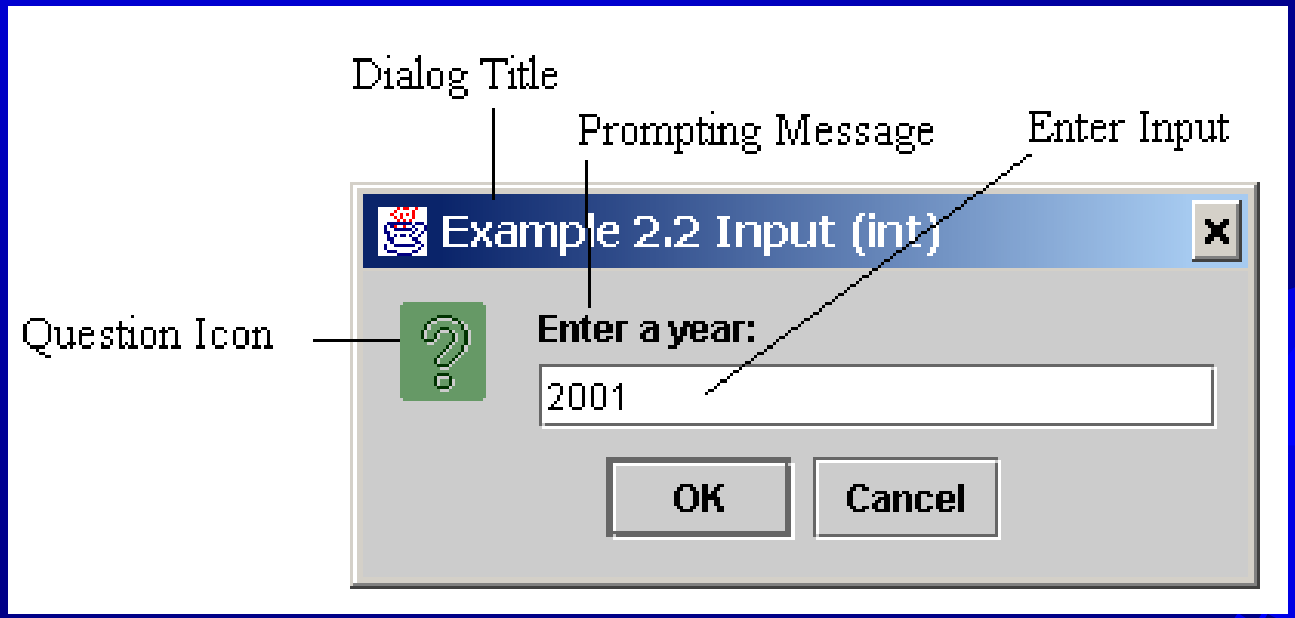
  // s1 trở thành HelloB

Nhận dữ liệu từ Input Dialog Box

String string = JOptionPane.showInputDialog(

null, “Prompt Message”, “Dialog Title”,

JOptionPane.QUESTION_MESSAGE));



Chuyển đổi chuỗi ký tự thành số nguyên

- Dữ liệu trả về từ input dialog box là một chuỗi ký tự. Nếu bạn nhập vào một giá trị số 123, nó trả về chuỗi "123". Để nhận được dữ liệu là một số, bạn phải chuyển đổi.

- Để chuyển đổi một chuỗi ký tự thành một giá trị int, bạn có thể sử dụng phương thức tĩnh parseInt trong lớp Integer như sau:

```
int intValue = Integer.parseInt(intString);
```

trong đó intString là một chuỗi số nguyên như "123".

Chuyển đổi chuỗi ký tự thành số thực

Để chuyển đổi một chuỗi ký tự thành một giá trị double, bạn có thể sử dụng phương thức tĩnh parseDouble trong lớp Double như sau:

```
double doubleValue = Double.parseDouble(doubleString);
```

trong đó doubleString là một chuỗi số thực như “123.45”.

Ví dụ: Nhập dữ liệu từ Dialog Boxes

Chương trình cho phép người sử dụng nhập vào một năm và kiểm tra đó có phải năm nhuận hay không. Sau đó nhập vào một số thực và kiểm tra có phải số dương hay không.

Năm nhuận là năm chia hết cho 4 nhưng không chia hết cho 100, hoặc là năm chia hết cho 400.

InputDialogDemo

Ví dụ: Tính tiền trả của khoản cho vay

Chương trình cho người sử dụng nhập vào lãi suất, số năm và số tiền cho vay rồi tính tiền trả hàng tháng và tổng số tiền phải trả.

$$\frac{loanAmount \times monthlyInterestRate}{1 - \frac{1}{(1 + monthlyInterestRate)^{numOfYears \times 12}}}$$

ComputeLoan

BT 1: Hiển thị thời gian hiện tại

Gọi phương thức System.currentTimeMillis() để lấy tổng số ms tính từ 0h GMT ngày 01/01/1970

- Tính giờ, phút, giây hiện tại ở Việt Nam và hiển thị dạng **h:m:s**

Nhập dữ liệu từ Command Prompt

- Sử dụng MyInput.java
- hoặc sử dụng lớp Scanner (JDK 1.5)

```
import java.util.*;
public class readint{
  static Scanner s=new Scanner(System.in);
  public static void main(String[] abc){
    System.out.print("Doc vao mot so nguyen: ");
    int a=readInt();
    System.out.println("So nguyen la: " + a);
  }
  public static int readInt(){
    return s.nextInt();
  }
}
```

Programming Style

- Chú thích
- Quy ước đặt tên
- Thụt đầu dòng và khoảng cách dòng
- Khối

Chú thích

Đặt một chú thích đầu chương trình để giải thích chương trình làm việc gì, các đặc điểm của CT, các cấu trúc dữ liệu mà CT hỗ trợ và các kỹ thuật đặc biệt mà CT sử dụng.

Đặt trong chú thích tên và mô tả rõ ràng về bạn ở đầu chương trình.

Đặt chú thích thích hợp giải thích các lớp, các đoạn lệnh…

Quy ước đặt tên

- Chọn các tên mô tả và có ý nghĩa.
- Tên biến và phương thức:
  - Sử dụng chữ thường. Nếu tên có chứa một vài từ, hãy viết liền nhau, sử dụng chữ thường ở từ thứ nhất và viết hoa ký tự đầu tiên của các từ tiếp theo.
  - Ví dụ, các biến `radius` và `area`, phương thức `computeArea`.

Quy ước đặt tên (tiếp)

- Tên lớp:
  - Viết hoa ký tự đầu tiên của mỗi từ trong tên. Ví dụ `ComputeArea`.
- Tên hằng:
  - Viết hoa tất cả các ký tự. Ví dụ hằng `PI`.

Thụt đầu dòng và khoảng cách dòng

- Thụt đầu dòng
  - Thụt vào 2 khoảng trống.
  - Cả 2 phía của mỗi toán tử nên có 1 khoảng trống
  - boolean b = 3 + 4 * 4 > 5 * (4 + 3) - ++i;

- Khoảng cách dòng
  - Sử dụng dòng trống để ngăn cách các đoạn code.

Block Styles

Sử dụng end-of-line style cho các dấu ngoặc nhọn.

Next-line style

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

End-of-line style

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

Programming Errors

- Syntax Errors
  - Do trình biên dịch phát hiện
- Runtime Errors
  - Gây ra chương trình bị bỏ qua
- Logic Errors
  - Tạo ra kết quả sai

Compilation Errors

```
public  class  ShowSyntaxErrors
  {
   public        static       void
  main(String[] args) {
     i = 30;
     System.out.println(i+4);
   }
}
```

Runtime Errors

```
public class ShowRuntimeErrors
  {
   public          static          void
  main(String[] args) {
     int i = 1 / 0;
   }
}
```

Logic Errors

```
public class ShowLogicErrors {
  public static void main(String[]
  args) {
    // Cong so1 voi so2
    int so1 = 3;
   int so2 = 5;
    so2 += so1 + so2;
  System.out.println("so2 bang " +
  so2);
   }
}
```

LẬP TRÌNH JAVA

Chương 3:
Các cấu trúc điều khiển

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 3

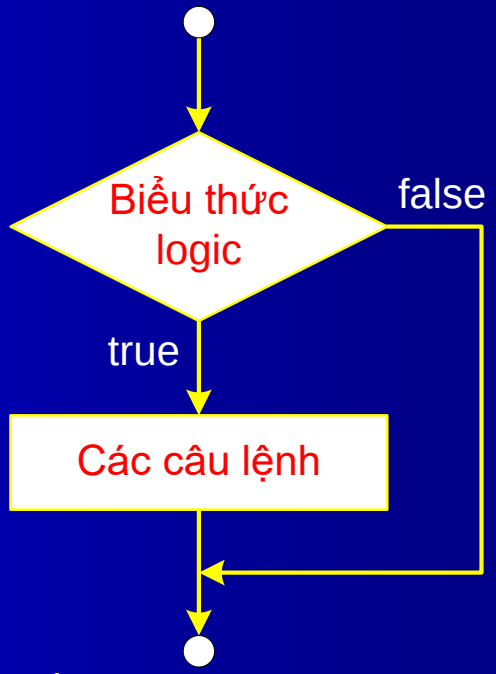
- Các cấu trúc lựa chọn:
  - Sử dụng `if` và `if...else`
  - Cấu trúc `if` lồng nhau
  - Sử dụng câu lệnh `switch`
  - Toán tử điều kiện
- Các cấu trúc lặp
  - Lặp: `while`, `do-while`, `for`
  - Lặp lồng nhau
  - Sử dụng `break` và `continue`

Các lệnh lựa chọn

- Lệnh `if`
- Lệnh `switch`
- Toán tử điều kiện

Lệnh if

```
if (Biểu_thức_logic) {
   các_câu_lệnh;
}
```



Ví dụ:

```
if ((i > 0) && (i < 10)) {
   System.out.println("i la mot " +
      "so nguyen nam giua 0 va 10");
}
```

Thận trọng

Lỗi phổ biến: thêm một dấu chấm phẩy ở cuối mệnh đề if.

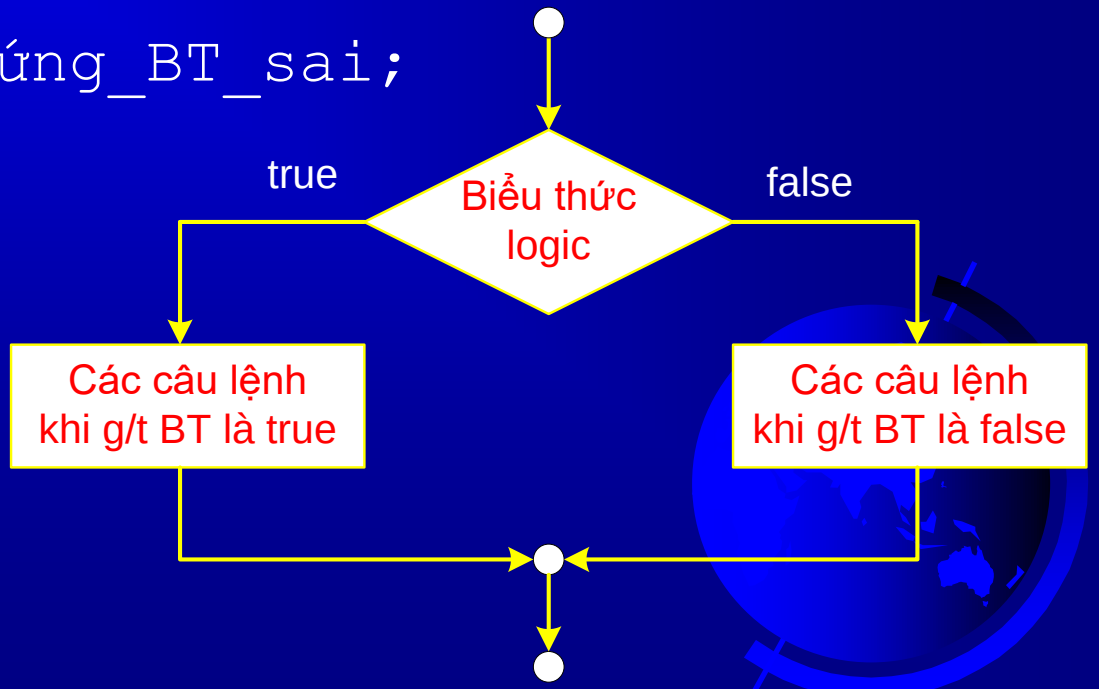
```
if (radius >= 0);
{
  area = radius*radius*PI;
  System.out.println(
    "The area for the circle of radius " +
    radius + " is " + area);
}
```

Wrong

Lỗi này rất khó tìm, vì nó không phải là lỗi biên dịch hay lỗi chạy chương trình, nó là một lỗi logic.

Lệnh if...else

```
if (Biểu_thức_logic) {
   Các_câu_lệnh_ứng_BT_đúng;
}
else {
   Các_câu_lệnh_ứng_BT_sai;
}
```



Ví dụ if...else

```
if (bankinh >= 0) {
   dientich = bankinh*bankinh*PI;

   System.out.println("Dien tich hinh
   tron co ban kinh " + bankinh +
      " la " + dientich);
}
else {
   System.out.println("Du lieu khong
   hop le!");
}
```

Nhiều lệnh if luân phiên

```
if (score >= 90)
  grade = 'A';
else
  if (score >= 80)
    grade = 'B';
  else
    if (score >= 70)
      grade = 'C';
    else
      if (score >= 60)
        grade = 'D';
      else
        grade = 'F';
```

```
if (score >= 90)
  grade = 'A';
else if (score >= 80)
  grade = 'B';
else if (score >= 70)
  grade = 'C';
else if (score >= 60)
  grade = 'D';
else
  grade = 'F';
```

Chú ý 1

Mệnh đề else gắn với mệnh đề if gần nhất trong cùng một khối.

Ví dụ, đoạn lệnh sau:

```
int i = 1; int j = 2; int k = 3;
if (i > j)
  if (i > k)
    System.out.println("A");
else
  System.out.println("B");
```

là tương đương với:

```
int i = 1; int j = 2; int k = 3;
if (i > j)
  if (i > k)
    System.out.println("A");
  else
    System.out.println("B");
```

Chú ý 2

Đoạn lệnh trước sẽ không in ra gì cả. Để bắt mệnh đề else gắn với mệnh đề if đầu tiên, bạn phải thêm một cặp ngoặc nhọn:

```
int i = 1;
int j = 2;
int k = 3;
  if (i > j) {
    if (i > k)
      System.out.println("A");
  }
  else
    System.out.println("B");
```

Đoạn lệnh trên sẽ in ra ký tự B.

Chú ý 3

```
if (n % 2 == 0)
  iseven = true;
else
  iseven = false;
```

tương đương

```
boolean iseven
  = (n % 2 == 0)
```

```
if (n == true)
system.out.prinln
  ("So chan");
```

tương đương

```
if (n)
system.out.prinln
  ("So chan");
```

Lệnh switch

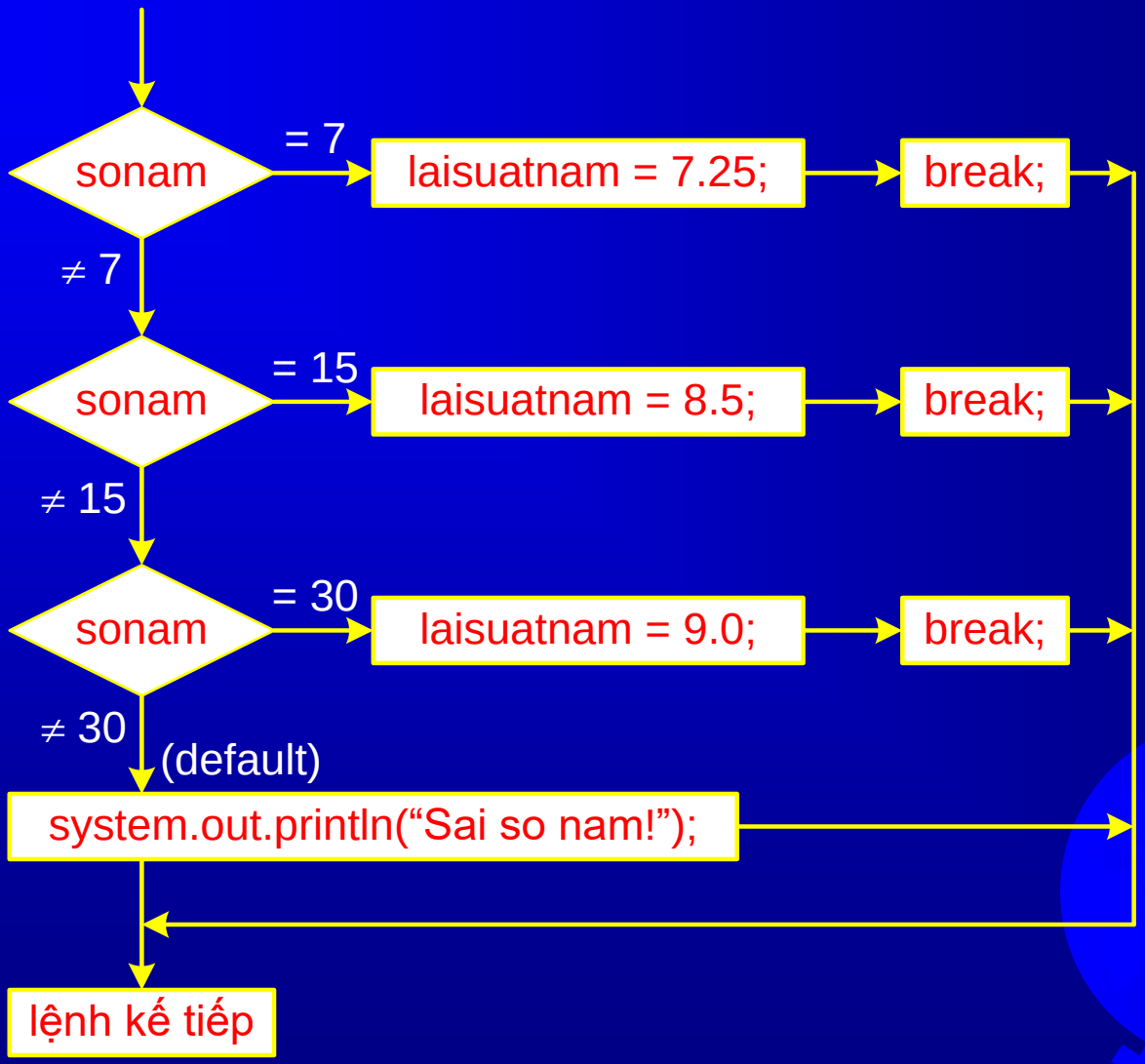
```
switch (bt_switch) {
  case gtri1: lenh(s)1;
              break;
  case gtri2: lenh(s)2;
              break;
        ……
  case gtriN: lenh(s)N;
              break;
  default: lenh(s)-khi-
           default;
}
```

```
switch (sonam) {
   case 7:  laisuatnam = 7.25;
            break;
   case 15: laisuatnam = 8.50;
            break;
   case 30: laisuatnam = 9.0;
            break;
   default: System.out.println
            ("Sai so nam, nhap
            7, 15, hoac 30");
}
```

VdSwitch

Run

Lưu đồ lệnh `switch`



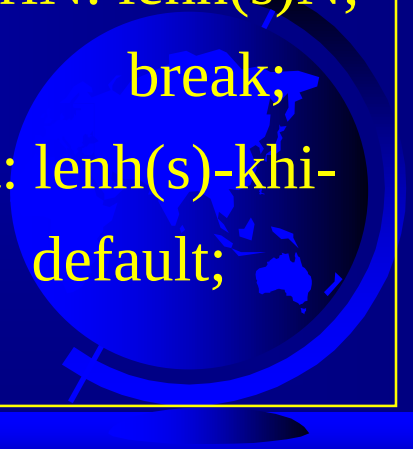
Quy tắc lệnh `switch`

- Biểu thức switch phải sinh ra một giá trị kiểu char, byte, short, hoặc int, và phải luôn được bao trong cặp dấu ngoặc tròn.

- gtri1, ..., gtriN phải có cùng kiểu dữ liệu với giá trị của biểu thức switch.

- Từ khóa break là tùy chọn, nhưng nên được sử dụng cuối mỗi trường hợp để thoát khỏi phần còn lại của lệnh switch. Nếu không có lệnh break, lệnh case tiếp theo sẽ được thực hiện.

```
switch (bt_switch) {
  case gtri1: lenh(s)1;
              break;
  case gtri2: lenh(s)2;
              break;
        ……
  case gtriN: lenh(s)N;
              break;
  default: lenh(s)-khi-
           default;
}
```



Quy tắc lệnh `switch` (tiếp)

- Trường hợp default là tùy chọn, có thể sử dụng để thực hiện các lệnh khi không có trường hợp nào ở trên là đúng.

- Thứ tự của các trường hợp (gồm cả trường hợp default) là không quan trọng. Tuy nhiên, phong cách lập trình tốt là nên theo một trình tự logic của các trường hợp và đặt trường hợp default cuối cùng.

Lưu ý

- Dừng quên dùng lệnh break khi cần thiết. ví dụ đoạn mã sau luôn hiển thị "Sai so nam!" bất chấp sonam là bao nhiêu. Giả sử sonam bằng 15. Lệnh laisuatnam = 8.50 được thực hiện, tiếp theo là lệnh laisuatnam = 9.0, và cuối cùng là lệnh System.out.println("Sai so nam!").

```
switch (sonam) {
  case 7:  laisuatnam = 7.25;
  case 15: laisuatnam = 8.50;
  case 30: laisuatnam = 9.0;
  default: System.out.println("Sai so
nam!");
}
```

Toán tử điều kiện

```
(BT_logic) ? bt1 : bt2 ;
```

- Ví dụ 1:

```
if (x > 0) y = 1
else y = -1;
```

- tương đương với:

```
y = (x > 0) ? 1 : -1;
```

Toán tử điều kiện

Ví dụ 2:

```
System.out.println(
   (so % 2 == 0)? so + "la so chan" :
   so + "la so le");
```

☞ tương đương với:

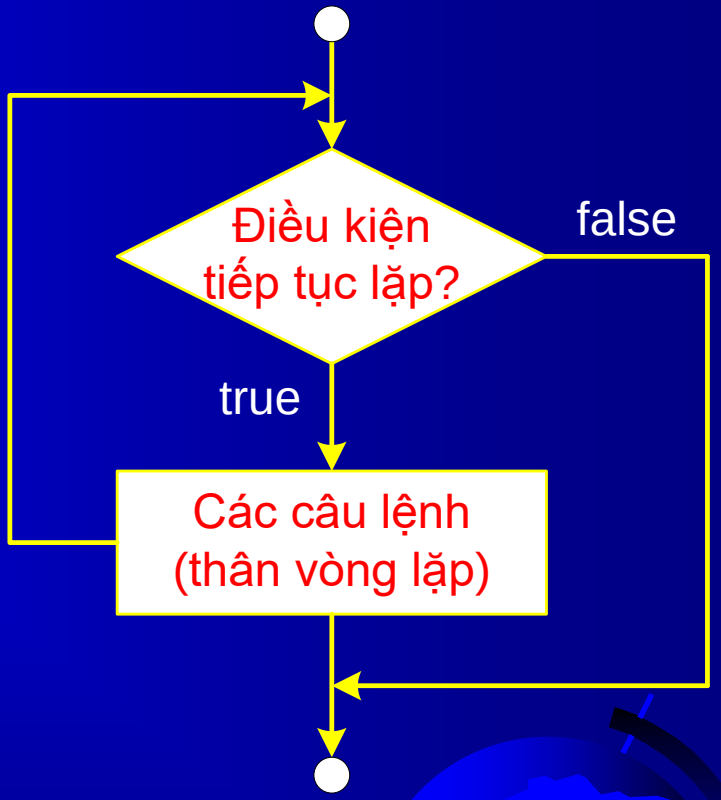
```
if (so % 2 == 0)
   System.out.println(so+"la so chan");
else
   System.out.println(so + "la so le");
```

Các lệnh lặp

- Lệnh lặp `while`
- Lệnh lặp `do-while`
- Lệnh lặp `for`
- `break` và `continue`

Lệnh lặp `while`

```
while (đk_tiếp_tục_lặp) {
  // thân_vòng_lặp;
  các_câu_lệnh;
}
```



Ví dụ:

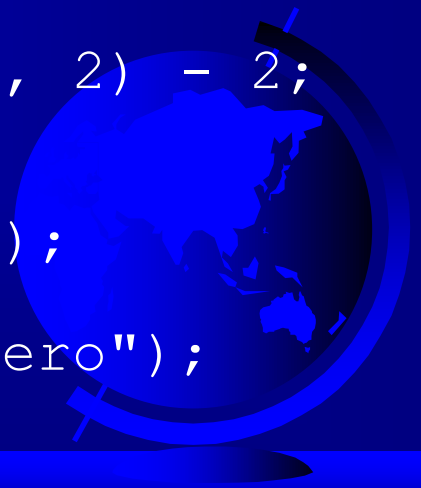
```
int i = 0;
while (i < 100) {
  System.out.println("Welcome to Java!");
  i++;
}
```

TestWhile

Lưu ý

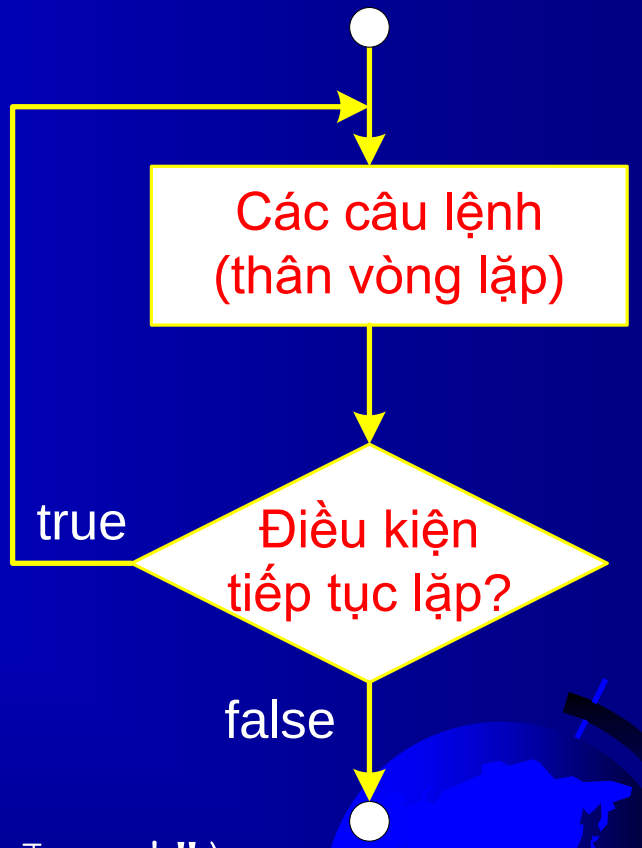
- Đừng sử dụng giá trị dấu chấm động để kiểm tra đẳng thức trong một điều khiển lặp. Vì giá trị dấu chấm động là gần đúng, sử dụng chúng có thể dẫn đến bộ đếm thiếu chính xác và kết quả sai. Ví dụ sau nên sử dụng giá trị int cho biến data. Nếu data có kiểu thực thì data != 0 có thể là true dù data bằng 0.

```
// data should be zero
double data = Math.pow(Math.sqrt(2), 2) - 2;

if (data == 0)
  System.out.println("data is zero");
else
  System.out.println("data is not zero");
```



Lệnh lặp do-while

```
do {
   // thân_vòng_lặp;
   các_câu_lệnh;
} while (đk_tiếp_tục_lặp);
```



Ví dụ:

```
int i = 0;
do {
   System.out.println("Welcome to Java!");
   i++;
} while (i < 100)
```

TestDoWhile

Lệnh lặp `for`

```
for (khởi_tạo; đk_tiếp_tục_lặp; cviệc_sau_mỗi_lần_lặp)
{
   // thân vòng lặp;
   các_câu_lệnh;
}
```

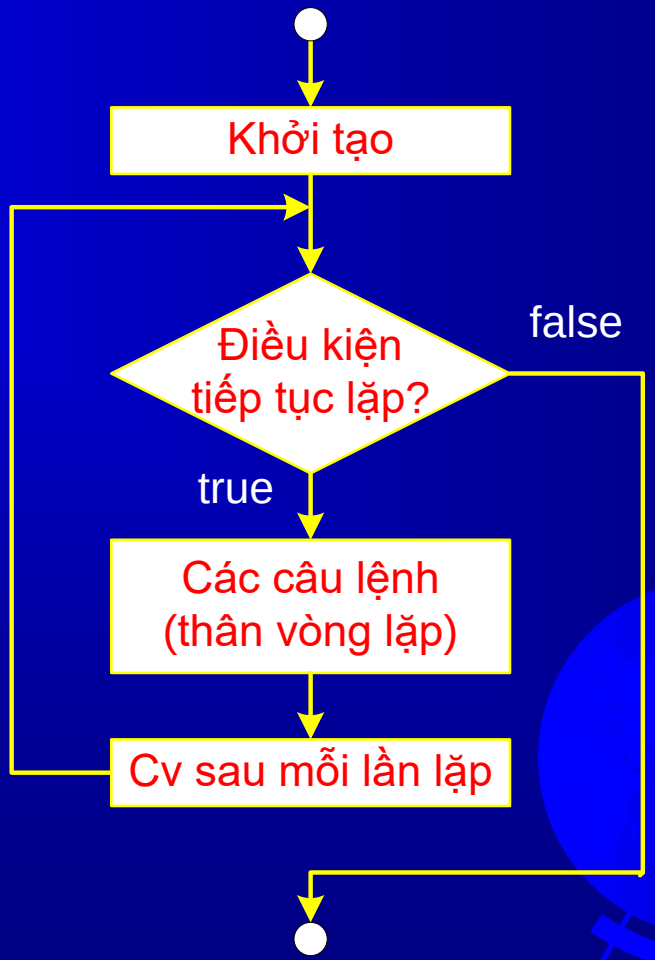
```
int i = 0;
while (i < 100) {
  System.out.println("Welcome to Java! ” + i);
  i++;
}
```

Example:

```
int i;
for (i = 0; i < 100; i++) {
  System.out.println("Welcome to Java! ” + i);
}
```

Lưu đồ lệnh lặp `for`

```
for (khởi_tạo; đk_tiếp_tục_lặp; cviệc_sau_mỗi_lần_lặp)
{
    // thân vòng lặp;
    các_câu_lệnh;
}
```



Lưu ý

- Các trường hợp sau đây là đúng:

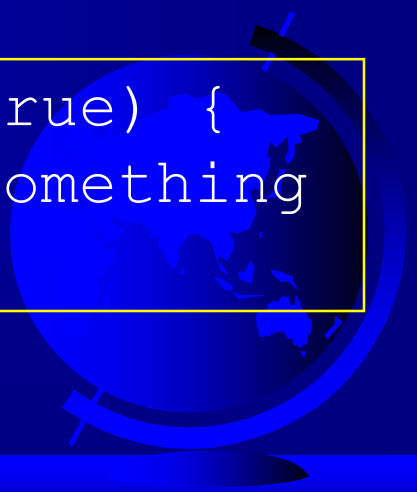
```
for (int i = 1; i < 100;
   System.out.println(i++));
```

```
for (int i = 0, j = 0; (i + j < 10); i++, j++) {
// Do something
}
```

```
for ( ; ; ) {
  // Do something
}
```

tương đương

```
while (true) {
  // Do something
}
```



Các ví dụ lệnh lặp `for`

- Ví dụ: Sử dụng các lệnh lặp for

TestSum

- Ví dụ: Sử dụng các lệnh for lồng nhau

TestMulTable

Cách sử dụng lệnh lặp?

- Ba lệnh lặp while, do, và for là tương đương nhau trong nhiều trường hợp; nghĩa là bạn có thể viết một vòng lặp bằng một dạng bất kỳ trong 3 dạng trên.
- Lệnh lặp for có thể sử dụng khi biết trước số lần lặp, ví dụ khi bạn muốn in ra một thông báo 100 lần.
- Lệnh lặp while có thể sử dụng khi không biết trước số lần lặp, như trong trường hợp đọc vào các số đến khi gặp số 0.
- Lệnh lặp do-while có thể sử dụng thay lệnh while khi thân vòng lặp phải được thực hiện trước khi kiểm tra điều kiện tiếp tục lặp.

Lưu ý

```
for (int i=0; i<10; i++);
{
  System.out.println("i is " + i);
}
```

Wrong

****

```
int i=0;
while (i<10);
{
  System.out.println("i is " + i);
  i++;
}
```

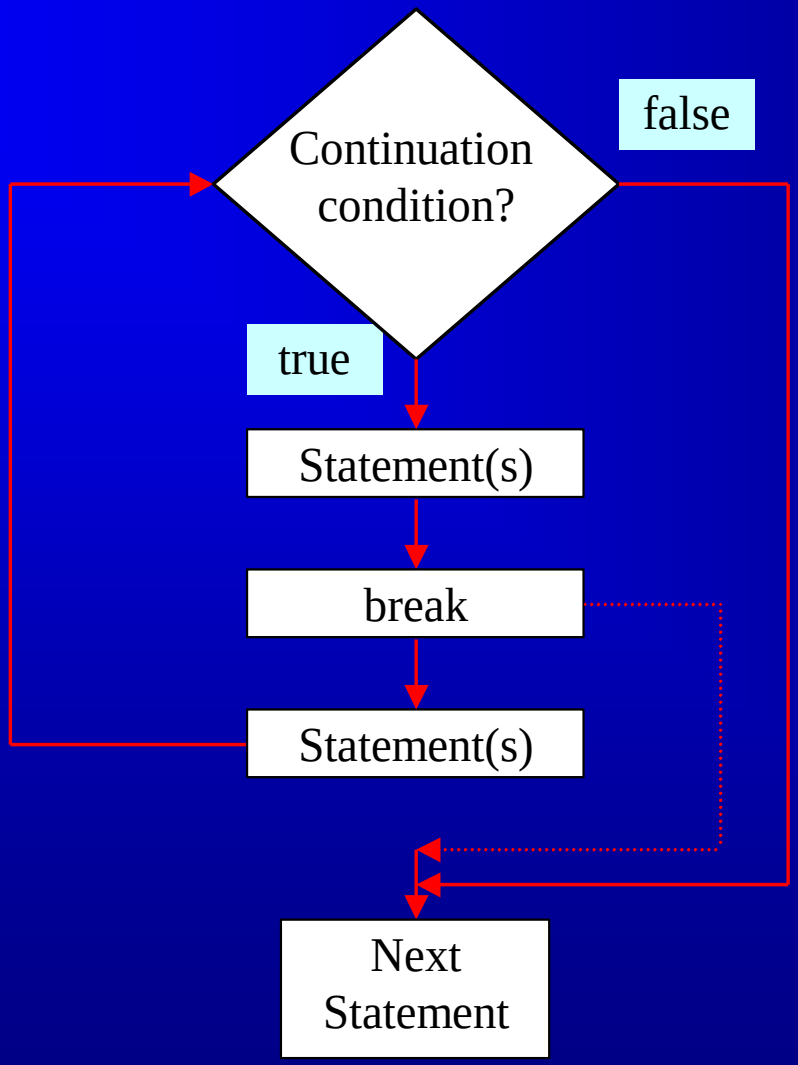
Wrong

****

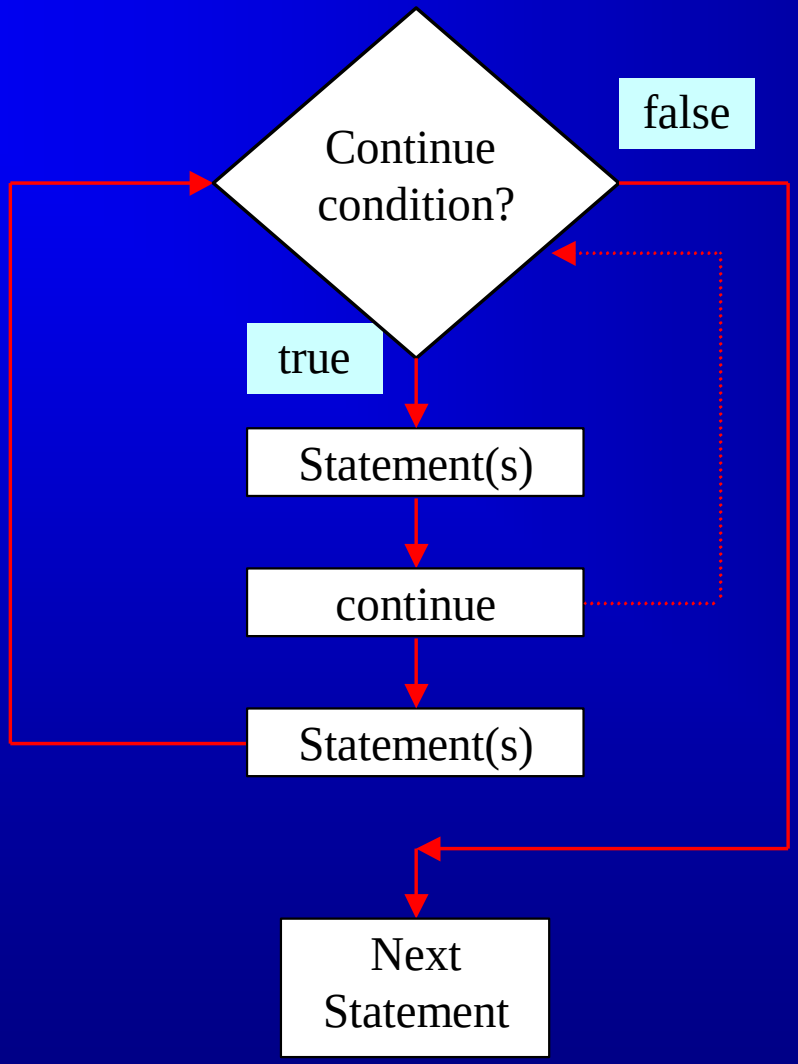
```
int i=0;
do {
  System.out.println("i is " + i);
  i++;
} while (i<10);
```

Correct

Từ khóa `break`



Từ khóa `continue`



Sử dụng `break` và `continue`

- Ví dụ: TestBreak.java

TestBreak

- Ví dụ: TestContinue.java

TestContinue

Ví dụ: Hiển thị ước số chung lớn nhất

GreatestCommonDivisor

Ví dụ: Tìm lượng tiền bán hàng

Bạn vừa mới bắt đầu công việc bán hàng trong một cửa hàng. Thu nhập của bạn bao gồm một lương cơ bản $5,000/năm và tiền hoa hồng được tính như sau:

Sales Amount	**Commission Rate**
$0.01–$5,000	8 %
$5,000.01–$10,000	10 %
>= $10,000.01	12 %

Mục đích của bạn là kiếm được $30,000/năm. Viết chương trình tìm lượng tiền bán hàng nhỏ nhất bạn phải tạo ra để giúp bạn đạt được mục đích.

FindSalesAmount

Ví dụ: Hiển thị kim tự tháp số

Sử dụng các lệnh lặp lồng nhau để in ra màn hình:

```
    1
  2 1 2
3 2 1 2 3
4 3 2 1 2 3 4
5 4 3 2 1 2 3 4 5
```

Mỗi dòng gồm 3 phần:

- Các ký tự trống đầu dòng,
- Các số đầu tiên, ví dụ như 3 2 1 trên dòng 3,
- Các số cuối cùng, ví dụ như 2 3 trên dòng 3.

PrintPyramid

Ví dụ: Hiển thị các số nguyên tố

Ví dụ này hiển thị 50 số nguyên tố đầu tiên trên 5 dòng, mỗi dòng 10 số.

PrimeNumber

LẬP TRÌNH JAVA

Chương 4: Phương thức (Methods)

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

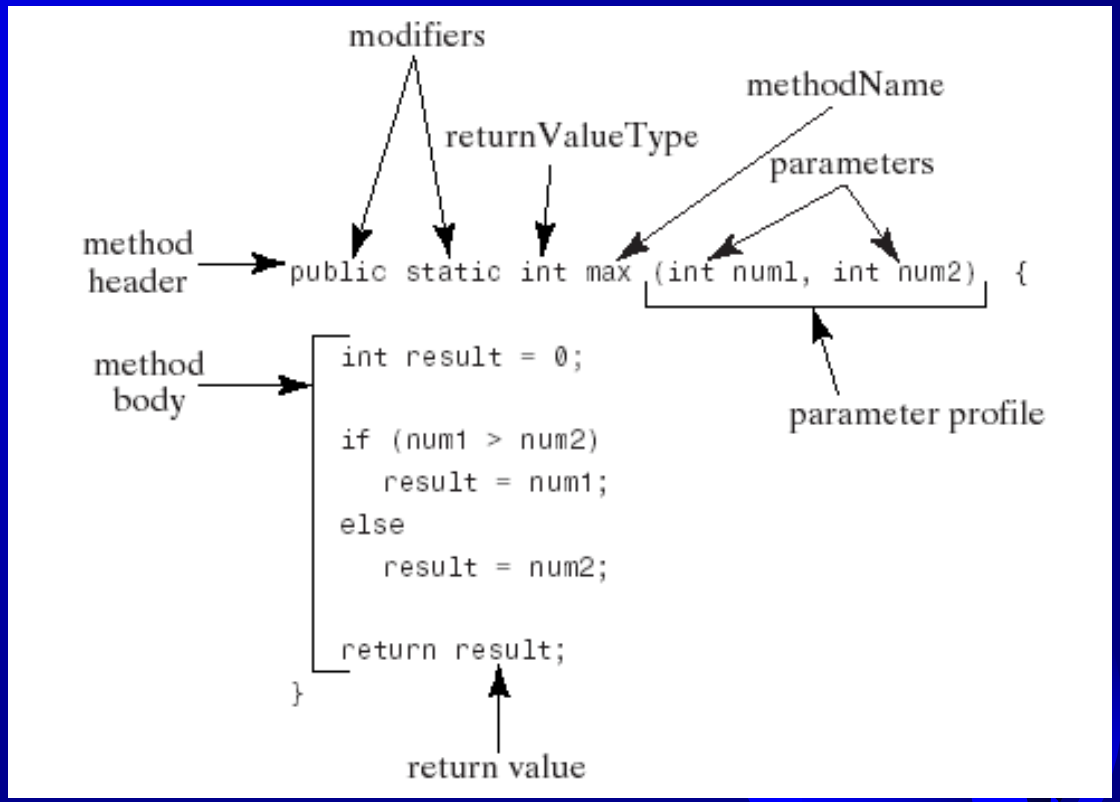
Nội dung chương 4

- Giới thiệu phương thức
  - Lợi ích, Khai báo, Cách gọi
- Truyền tham số
  - Truyền giá trị
- Overloading Methods
  - Lời gọi nhập nhằng
- Phạm vi của biến cục bộ
- Phương thức trừu tượng
- Lớp `Math`
- Case Studies
- Đệ quy

Giới thiệu phương thức (method)

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

Cấu trúc của phương thức:



Giới thiệu phương thức (tiếp)

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

Giới thiệu phương thức (tiếp)

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

Khai báo phương thức

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

Cách gọi phương thức

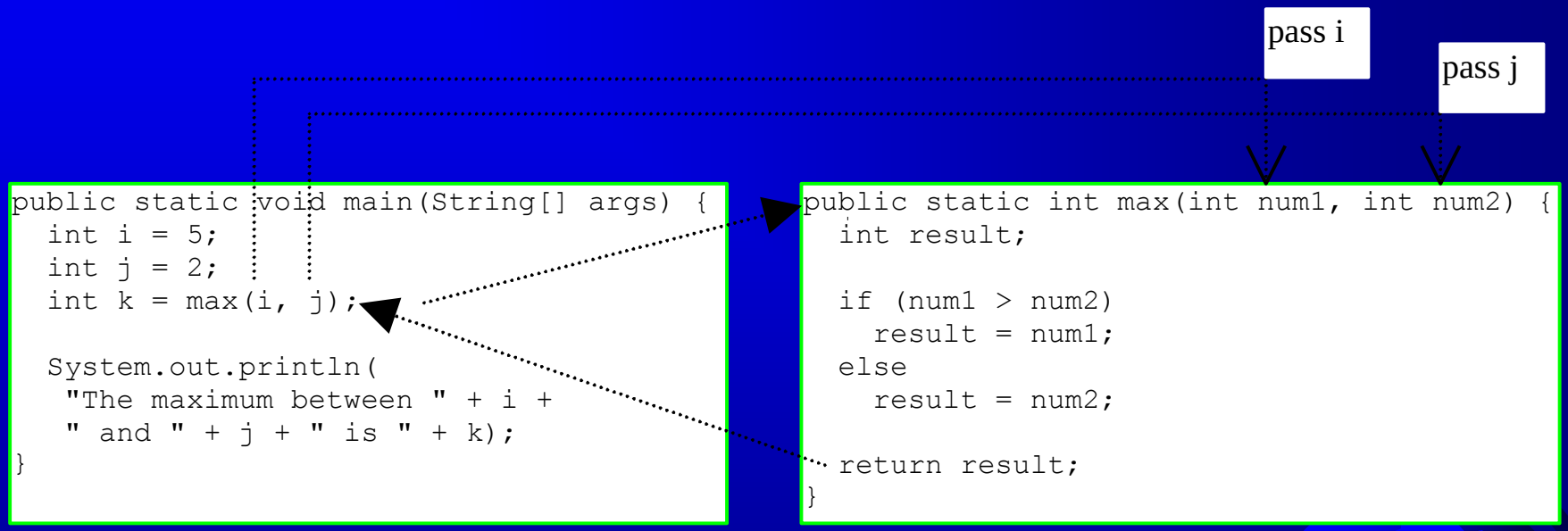
Ví dụ 4.1: Phương thức `max`

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

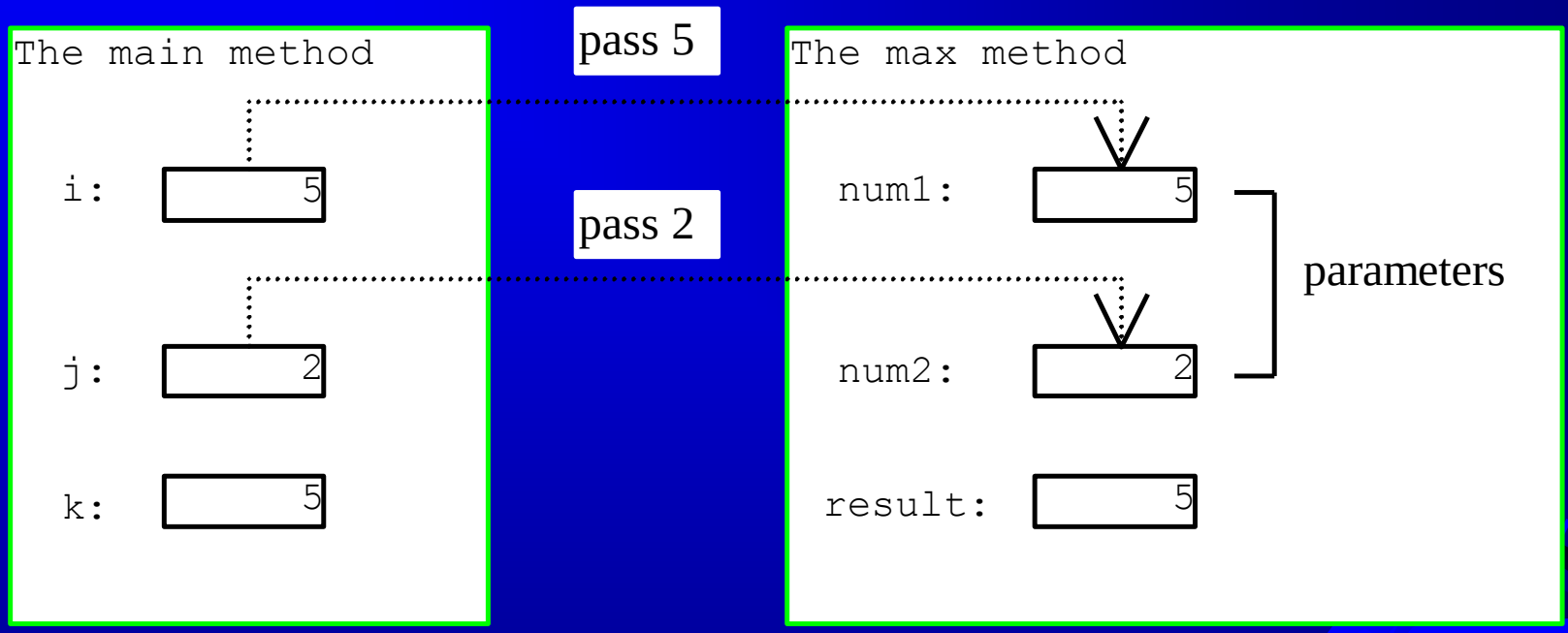
TestMax

Run

Cách gọi phương thức (tiếp)



Cách gọi phương thức (tiếp)



Lưu ý

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

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

- Để sửa lỗi này, xóa `if (n<0)` trong đoạn mã trên.

Truyền tham số

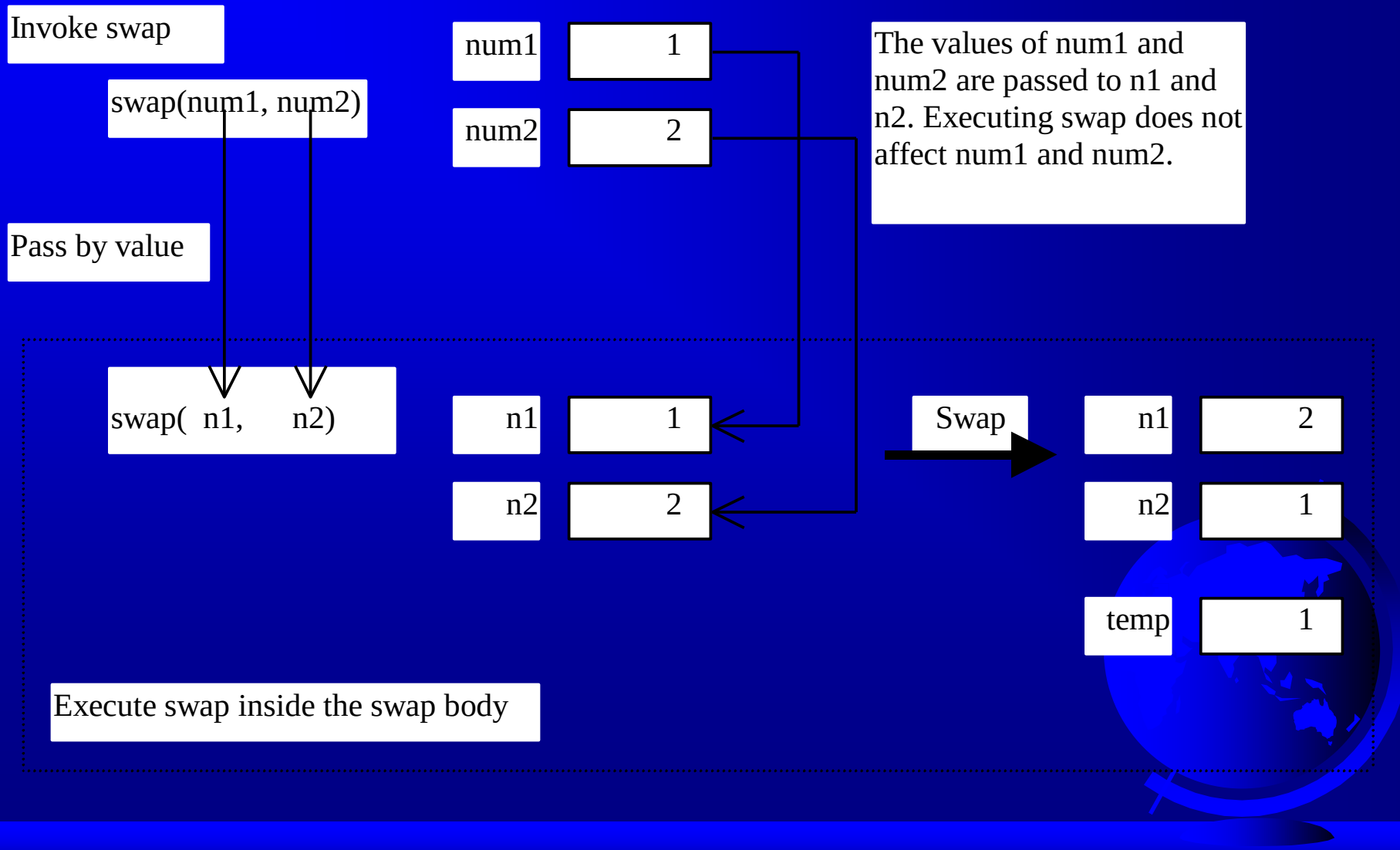
```
public static void nPrintln(String message, int n) {
  for (int i = 0; i < n; i++)
    System.out.println(message);
}
```

Truyền tham trị

Ví dụ 4.2: Truyền tham trị

TestPassByValue

Truyền tham trị (tiếp)



Overloading Methods

Ví dụ 4.3: Overloading method `max`

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

TestMethodOverloading

Gọi mập mờ

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

Gọi mập mờ

```
public class AmbiguousOverloading {
  public static void main(String[] args) {
    System.out.println(max(1, 2));
  }

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

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

Phạm vi của các biến cục bộ

- Biến cục bộ (local variable): biến được khai báo trong một phương thức.
- Phạm vi: phần chương trình mà biến có thể được tham chiếu.
- Phạm vi của một biến cục bộ bắt đầu từ khi khai báo đến cuối block chứa biến đó. Một biến cục bộ phải được khai báo trước khi sử dụng.

Phạm vi của các biến cục bộ (tiếp)

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

Phạm vi của các biến cục bộ (tiếp)

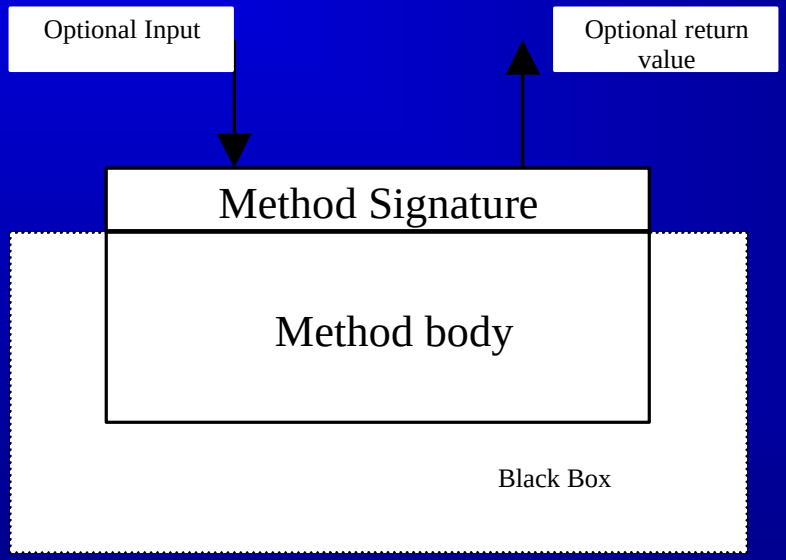
```
// Fine with no errors
public static void correctMethod() {
  int x = 1;
  int y = 1;
  // i is declared
  for (int i = 1; i < 10; i++) {
    x += i;
  }
  // i is declared again
  for (int i = 1; i < 10; i++) {
    y += i;
  }
}
```

Phạm vi của các biến cục bộ (tiếp)

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

Phương thức trừu tượng - Method Abstraction

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



Lợi ích của phương thức

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

The `Math` Class

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

Các phương thức lượng giác

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

Ví dụ

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

Các phương thức số mũ

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

Các phương thức làm tròn

- `double ceil(double x)`

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

- `double floor(double x)`

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

- `double rint(double x)`

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

- `int round(float x)`

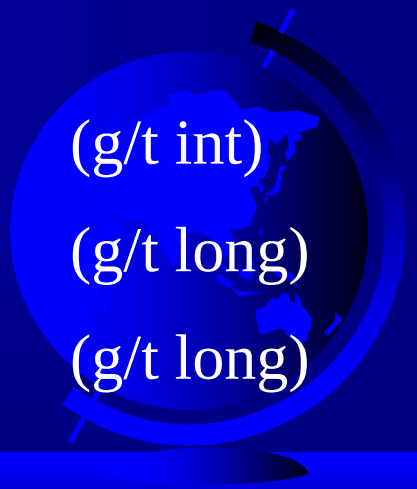
  Trả về `(int)Math.floor(x+0.5)`

- `long round(double x)`

  Trả về `(long)Math.floor(x+0.5)`

Ví dụ

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



min, max, abs, và random

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

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

- `abs(a)`

  Trả về giá trị tuyệt đối của a.

- `random()`

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

Ví dụ

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

Ví dụ: Tính trung bình và độ lệch chuẩn

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

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

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

ComputeMeanDeviation

Ví dụ: Sinh ký tự ngẫu nhiên

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

RandomCharacter

TestRandomCharacter

Appendix B: ASCII Character Set

TABLE B.1 ASCII Character Set in the Decimal Index

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

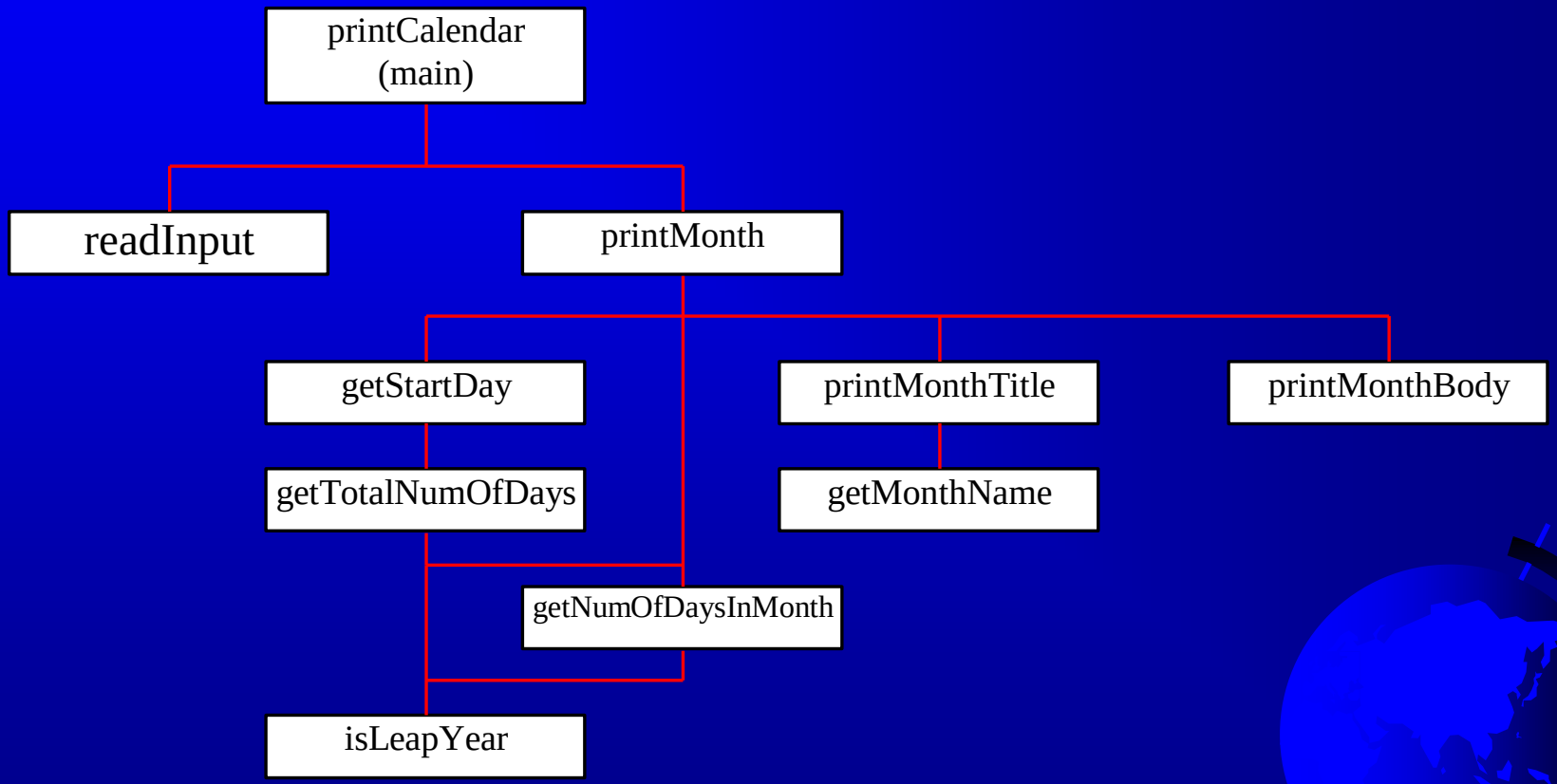
Case Study

Ví dụ: Hiển thị lịch

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

PrintCalendar

Design Diagram



Đệ quy

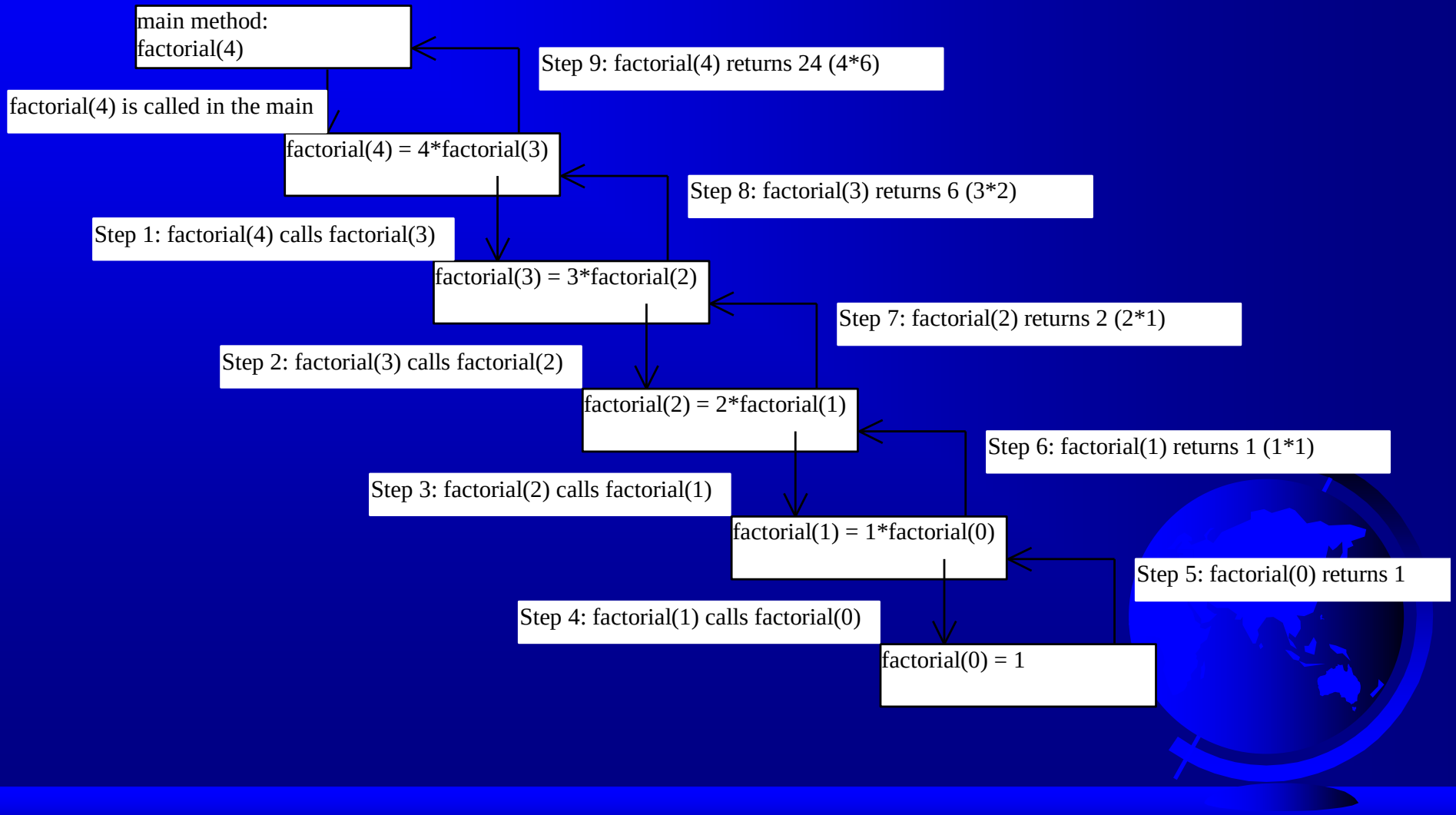
Ví dụ: Tính giai thừa

factorial(0) = 1;

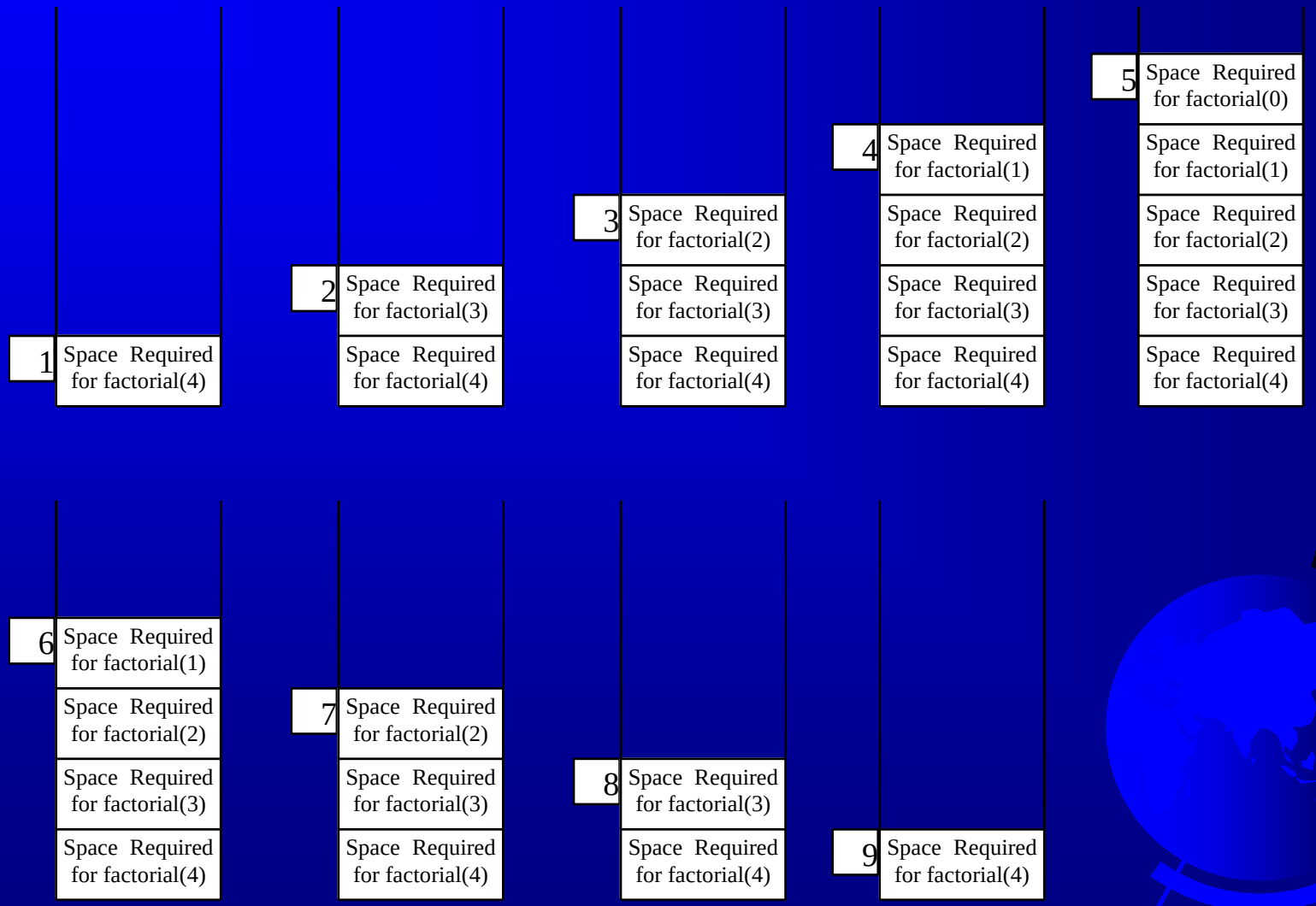
factorial(n) = n*factorial(n-1);

ComputeFactorial

Tính giai thừa (tiếp)



Tính giai thừa (tiếp)



Dãy số Fibonacci

Ví dụ: Tìm dãy số Finonacci

0 1 1 2 3 5 8 13 21 34 55 89…

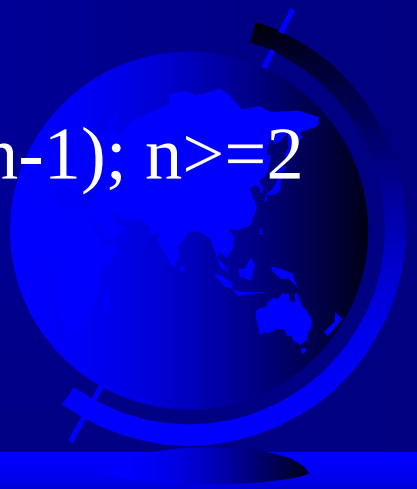
f0 f1

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

fib(0) = 0;

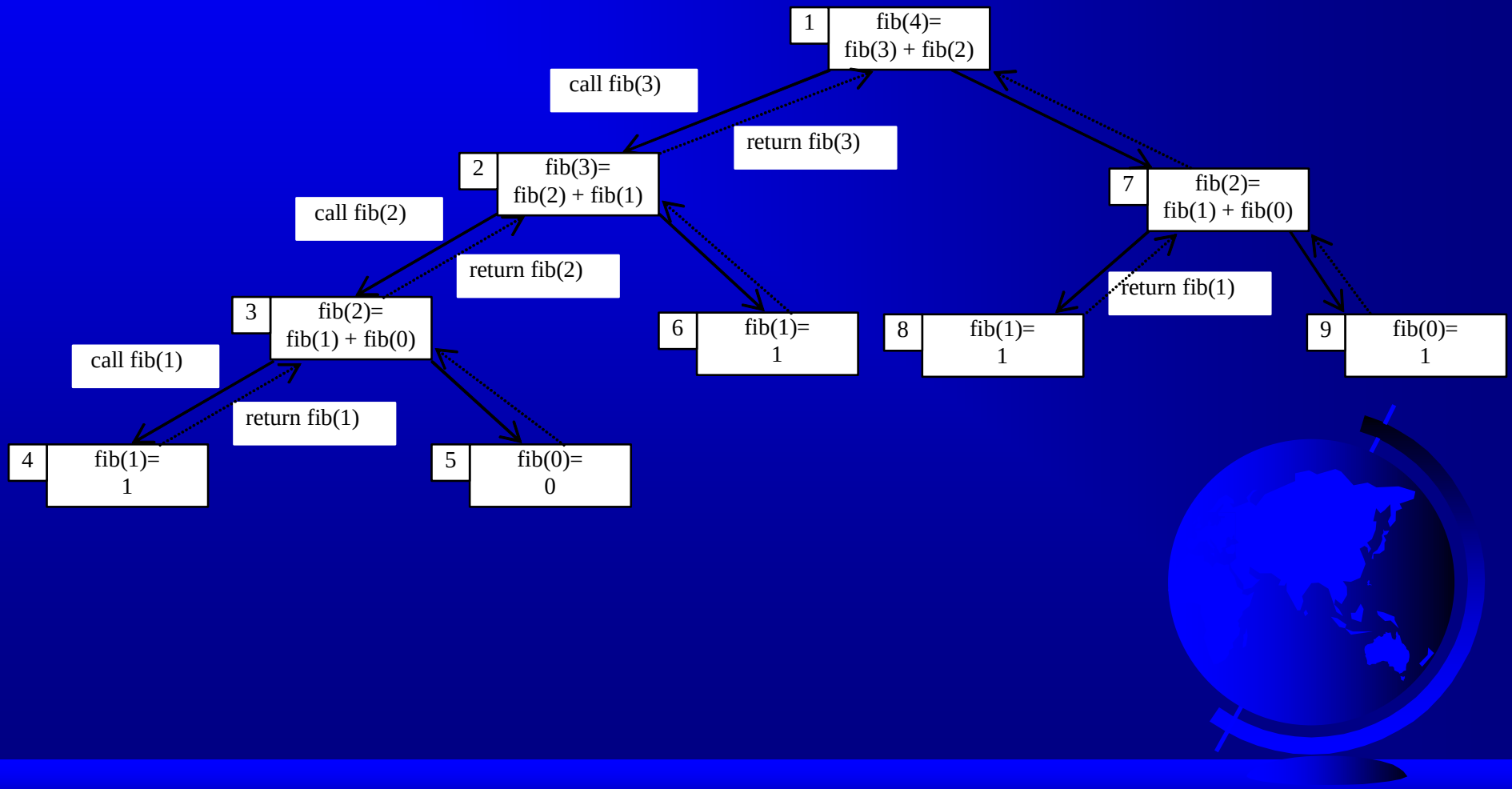
fib(1) = 1;

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



ComputeFibonacci

Dãy số Fibonacci (tiếp)

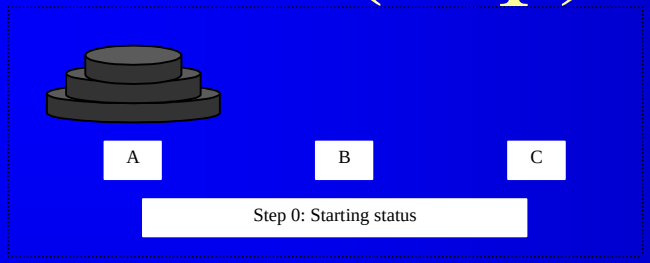


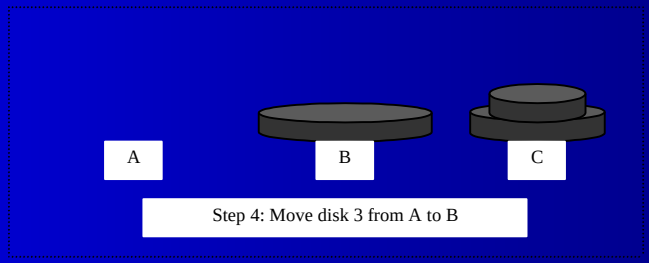
Tháp Hanoi

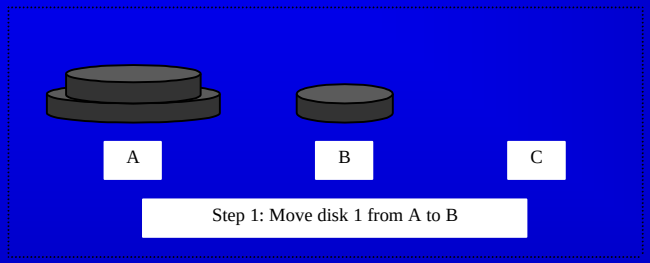
Ví dụ: Giải bài toán Tháp Hanoi

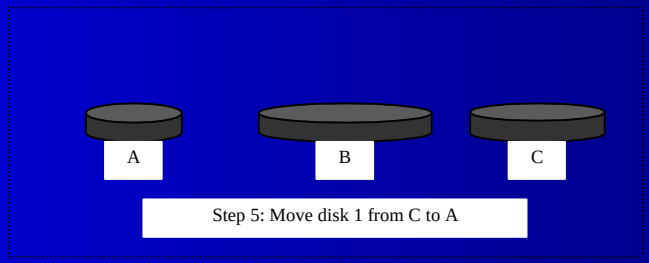
TowersOfHanoi

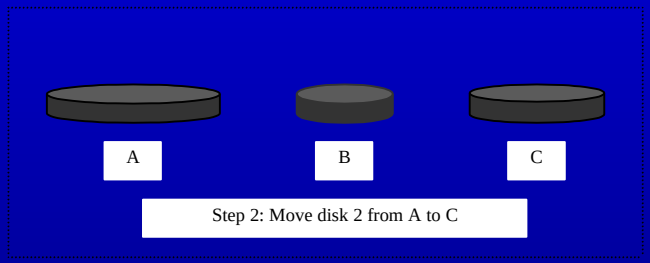
Tháp Hanoi (tiếp)

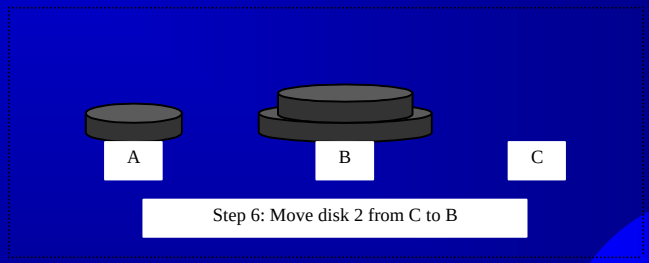


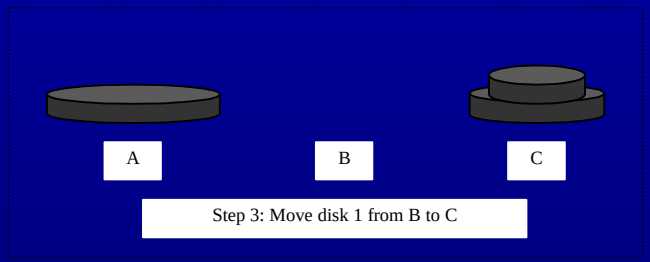


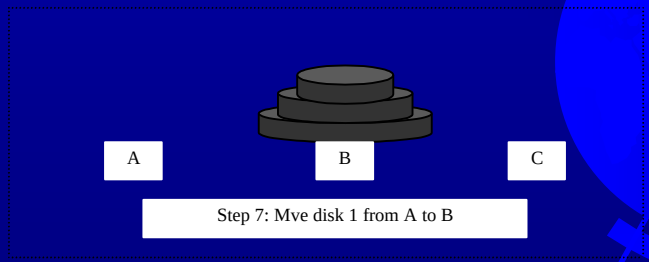












LẬP TRÌNH JAVA

Chương 5: Mảng (Arrays)

Phạm Quang Dũng

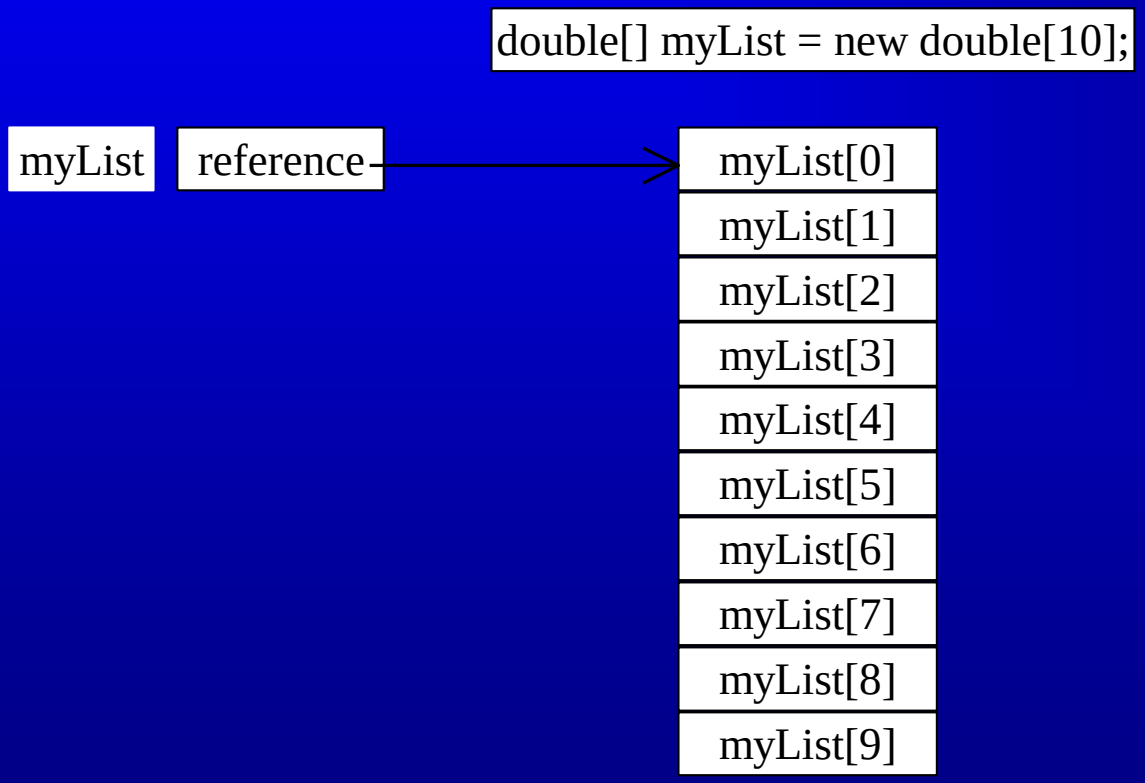
BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 5

- Giới thiệu mảng
- Khai báo biến mảng, tạo mảng (create), khởi tạo mảng (initialize)
- Truyền mảng cho phương thức
- Copy mảng
- Mảng nhiều chiều
- Các phương thức tìm kiếm và sắp xếp

Giới thiệu mảng

Mảng là một cấu trúc dữ liệu biểu diễn một tập các dữ liệu cùng kiểu.



Một mảng 10 phần tử kiểu `double`

Khai báo biến mảng

- `datatype[] arrayname;`

  Ví dụ:

  `double[] myList;`

- `datatype arrayname[];`

  Ví dụ:

  `double myList[];`

Tạo mảng

```
arrayName = new datatype[arraySize];
```

Ví dụ:

```
myList = new double[10];
```

`myList[0]` tham chiếu phần tử đầu tiên của mảng.

`myList[9]` tham chiếu phần tử cuối cùng của mảng.

Khai báo và tạo mảng trong một bước

- ```
  datatype[] arrayname = new
     datatype[arraySize];
  ```

```
double[] myList = new double[10];
```

- ```
  datatype arrayname[] = new
    datatype[arraySize];
  ```

```
double myList[] = new double[10];
```

Độ dài mảng

- Mỗi khi mảng được tạo, kích thước của nó được ấn định, không thể thay đổi. Bạn có thể lấy kích thước mảng bằng cách gọi:

  arrayVariable.length

Ví dụ:

myList.length trả về giá trị 10

Khởi tạo mảng

- Sử dụng vòng lặp:

```
for (int i = 0; i < myList.length; i++)
  myList[i] = i;
```

- Khai báo, tạo, khởi tạo trong một lệnh:

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

- Lưu ý: chỉ trong 1 lệnh, nhiều hơn 1 lệnh là sai:

```
double[] myList;
myList = {1.9, 2.9, 3.4, 3.5};
```

Khai báo, tạo, khởi tạo

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

Câu lệnh trên tương đương với các câu lệnh sau:

```
double[] myList = new double[4];
myList[0] = 1.9;
myList[1] = 2.9;
myList[2] = 3.4;
myList[3] = 3.5;
```

Ví dụ 5.1

- Mục đính: Chương trình nhận 6 số từ bàn phím, tìm số lớn nhất và đếm số lần xuất hiện của giá trị đó.

  Giả sử bạn nhập vào 3, 5, 2, 5, 5, và 5, số lớn nhất là 5 và số lần xuất hiện là 4.

TestArray

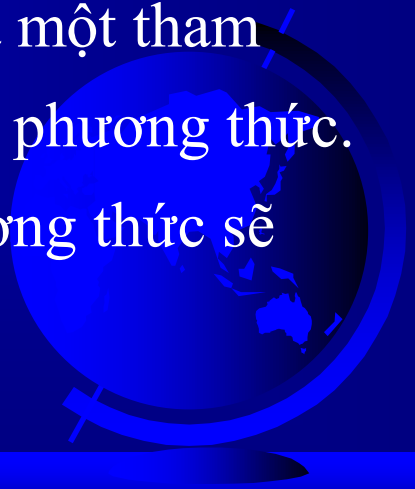
Ví dụ 5.2: Xếp loại

- Mục tiêu: đọc vào điểm của SV (int) từ bàn phím, lấy điểm cao nhất (best), rồi xếp loại theo quy tắc sau:
  - Loại A: điểm >= best–10;
  - Loại B: điểm >= best–20;
  - Loại C: điểm >= best–30;
  - Loại D: điểm >= best–40;
  - Trái lại là loại F.

AssignGrade

Truyền mảng cho phương thức

- Java sử dụng *truyền tham trị* để truyền các tham số cho phương thức. Có nhiều sự khác nhau quan trọng khi truyền tham trị của biến có kiểu dữ liệu cơ sở và biến mảng.
- Với tham số kiểu dữ liệu cơ sở, giá trị thực được truyền. Thay đổi giá trị của tham số cục bộ trong phương thức không làm thay đổi giá trị của biến bên ngoài phương thức.
- Với tham số kiểu mảng, giá trị của tham số chứa một tham chiếu tới mảng; tham chiếu này được truyền cho phương thức. Bất kỳ sự thay đổi nào xuất hiện trong thân phương thức sẽ làm thay đổi mảng gốc được truyền.

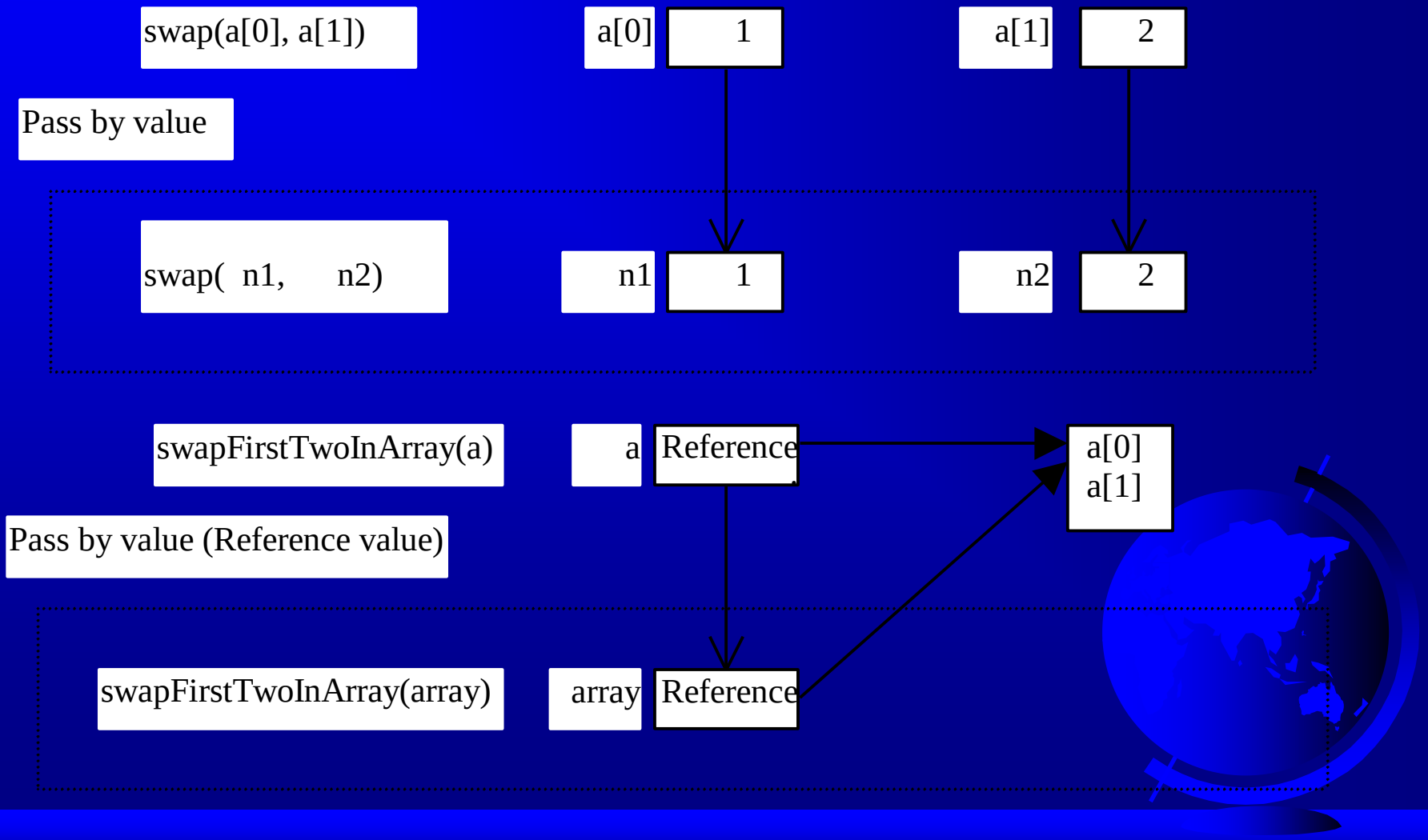


Ví dụ 5.3
Truyền tham số kiểu mảng

- Mục tiêu: Minh họa sự khác nhau giữa truyền tham số kiểu dữ liệu cơ sở và kiểu mảng.

TestPassArray

Ví dụ 5.3 (tiếp)


swap(a[0], a[1])
a[0]
1
a[1]
2
Pass by value
swap(n1, n2)
n1
1
n2
2
swapFirstTwoInArray(a)
a
Reference
a[0]
a[1]
Pass by value (Reference value)
swapFirstTwoInArray(array)
array
Reference

Ví dụ 5.4: Sử dụng mảng tính độ lệch

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

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

Deviation

Ví dụ 5.5
Đếm tần số xuất hiện của các ký tự

- Sinh ngẫu nhiên 100 ký tự chữ thường và gán cho 1 mảng ký tự.
- Đếm tần số xuất hiện của mỗi ký tự trong mảng.
- Tìm trung bình (mean) và độ lệch chuẩn (standard deviation) của các lần đếm.

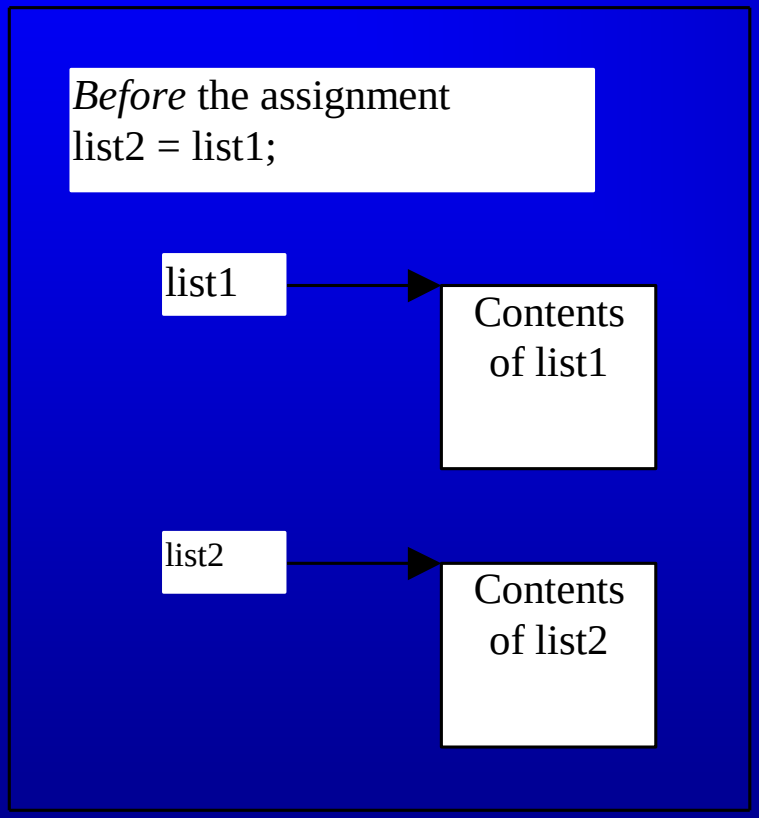
CountLettersInArray

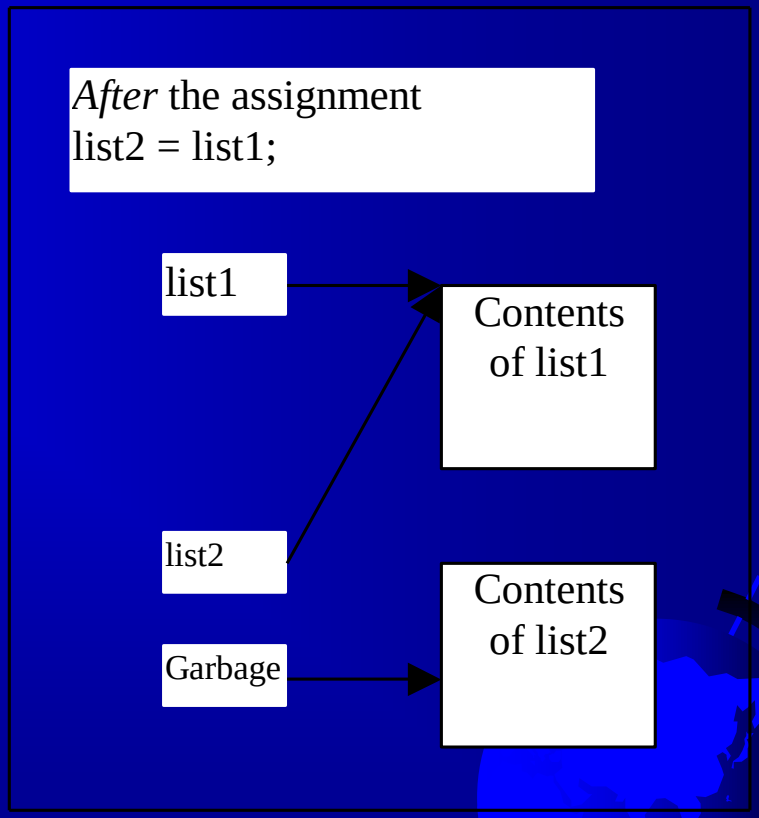
Ví dụ 5.6: Copy mảng

Trong ví dụ này, bạn sẽ thấy rằng một phép gán đơn giản không thể copy mảng. Chương trình chỉ đơn giản tạo 2 mảng và định copy từ mảng này sang mảng kia sử dụng một câu lệnh gán.

TestCopyArray

Copy mảng


Before the assignment
list2 = list1;
list1
Contents of list1
list2
Contents of list2


After the assignment
list2 = list1;
list1
Contents of list1
list2
Garbage
Contents of list2

Copy mảng

Sử dụng một vòng lặp:

```
int[] sourceArray = {2, 3, 1, 5, 10};
int[] targetArray = new int[sourceArray.length];

for (int i = 0; i < sourceArrays.length; i++)
    targetArray[i] = sourceArray[i];
```

Tiện ích arraycopy

```
arraycopy(sourceArray, src_pos,
  targetArray, tar_pos, length);
```

Ví dụ:

```
System.arraycopy(sourceArray, 0, targetArray,
  0, sourceArray.length);
```

Mảng nhiều chiều

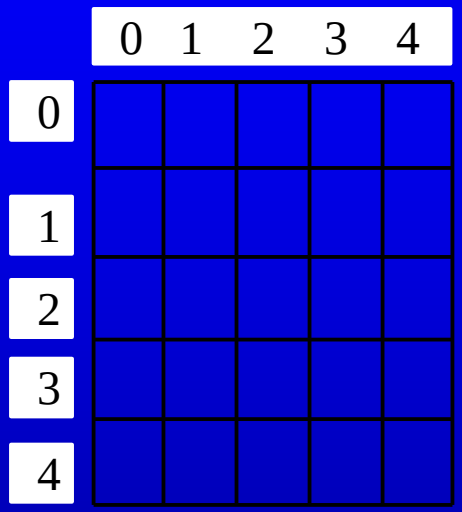
Khai báo và tạo biến mảng nhiều chiều:

```
int[][] matrix = new int[10][10];
```

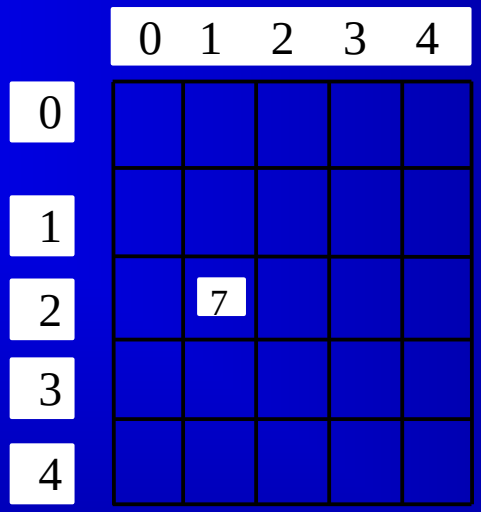
hoặc:

```
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i=0; i<matrix.length; i++)
  for (int j=0; j<matrix[i].length; j++)
  {
    matrix[i][j] = (int)(Math.random()*1000);
  }

double[][] x;
```

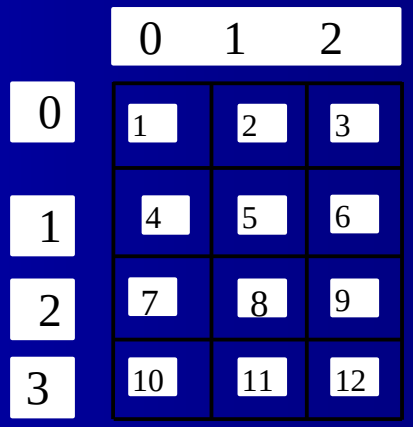
Minh họa mảng nhiều chiều




```
matrix = new int[5][5];
```




```
matrix[2][1] = 7;
```




```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Khai báo, tạo, khởi tạo trong 1 lệnh

Ví dụ khai báo, tạo và khởi tạo một mảng 2 chiều:

```
int[][] array = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
};
```

Câu lệnh trên tương đương với các câu lệnh sau:

```
int[][] array = new int[4][3];
array[0][0] = 1;  array[0][1] = 2;  array[0][2] =
   3;
array[1][0] = 4;  array[1][1] = 5;  array[1][2] =
   6;
array[2][0] = 7;  array[2][1] = 8;  array[2][2] =
   9;
```

Độ dài của mảng nhiều chiều

```
int[][] array = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
};

array.length
array[0].length
array[1].length
array[2].length
```

Mảng gồ ghề

- Mỗi hàng của một mảng 2 chiều là một mảng đơn. Vì vậy, mỗi hàng có thể có độ dài khác nhau. Một mảng như vậy được gọi là *mảng gồ ghề (ragged array)*. Ví dụ:

```
int[][] matrix = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```

Ví dụ 5.7: Cộng và nhân 2 ma trận

- Mục tiêu: Sử dụng mảng 2 chiều để tạo 2 ma trận A, B rồi tính ma trận tổng A+B và ma trận tích A*B.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & a_{14}+b_{14} & a_{15}+b_{15} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & a_{24}+b_{24} & a_{25}+b_{25} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} & a_{34}+b_{34} & a_{35}+b_{35} \\ a_{41}+b_{41} & a_{42}+b_{42} & a_{43}+b_{43} & a_{44}+b_{44} & a_{45}+b_{45} \\ a_{51}+b_{51} & a_{52}+b_{52} & a_{53}+b_{53} & a_{54}+b_{54} & a_{55}+b_{55} \end{pmatrix}$$

TestMatrixOperation

Ví dụ 5.7 (tiếp): Cộng và nhân 2 ma trận

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} \end{pmatrix}$$

$$c_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + a_{i3} \times b_{3j} + a_{i4} \times b_{4j} + a_{i5} \times b_{5j}$$

Ví dụ 5.8: Chấm điểm Multiple-Choice Test

- Mục tiêu: viết chương trình tính điểm bài thi trắc nghiệm dạng multiple-choice.

	Students' Answers to the Questions:									
	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

Key to the Questions:										
	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

GradeExam

Ví dụ 5.9: Tính tổng điểm

- Mục tiêu: viết một chương trình tính tổng điểm cho sinh viên trong 1 lớp. Giả sử điểm được lưu trong một mảng 3 chiều có tên scores. Chiều thứ nhất ứng với một sinh viên, chiều thứ hai ứng với một kỳ thi, chiều thứ ba ứng với các điểm của kỳ thi. Giả sử có 7 sinh viên, 5 kỳ thi, và mỗi kỳ thi có 2 điểm: điểm trắc nghiệm và điểm lập trình. Do đó scores[i][j][0] biểu diễn điểm trắc nghiệm trong kỳ thi j của sinh viên i. Chương trình sẽ hiển thị tổng điểm của mỗi sinh viên.

TotalScore

Tìm kiếm trong mảng

- Tìm kiếm là quá trình tìm một phần tử xác định trong mảng, ví dụ tìm điểm nào đó trong danh sách điểm.
- Giống như sắp xếp, tìm kiếm là một tác vụ thường xuyên trong lập trình máy tính.
- Có nhiều giải thuật và cấu trúc dữ liệu để tìm kiếm. Chương này sẽ thảo luận 2 phương pháp phổ biến là: tìm kiếm tuyến tính (*linear search)* và tìm kiếm nhị phân (*binary search).*

Tìm kiếm tuyến tính

- Phương pháp tìm kiếm tuyến tính so sánh phần tử khóa key với mỗi phần tử trong mảng list[].
- Việc tìm kiếm sẽ kết thúc khi tìm thấy một phần tử mảng bằng key hoặc khi duyệt hết mảng mà không tìm thấy.
- Nếu tìm thấy, tìm kiếm tuyến tính sẽ trả về chỉ số của phần tử bằng key.
- Nếu không tìm thấy, kết quả bằng -1.

Ví dụ 5.10: Tìm kiếm tuyến tính

- Mục tiêu: Tạo ngẫu nhiên 10 phần tử mảng kiểu `int` rồi hiển thị. Nhận key từ user rồi thực hiện tìm kiếm tuyến tính.

LinearSearch

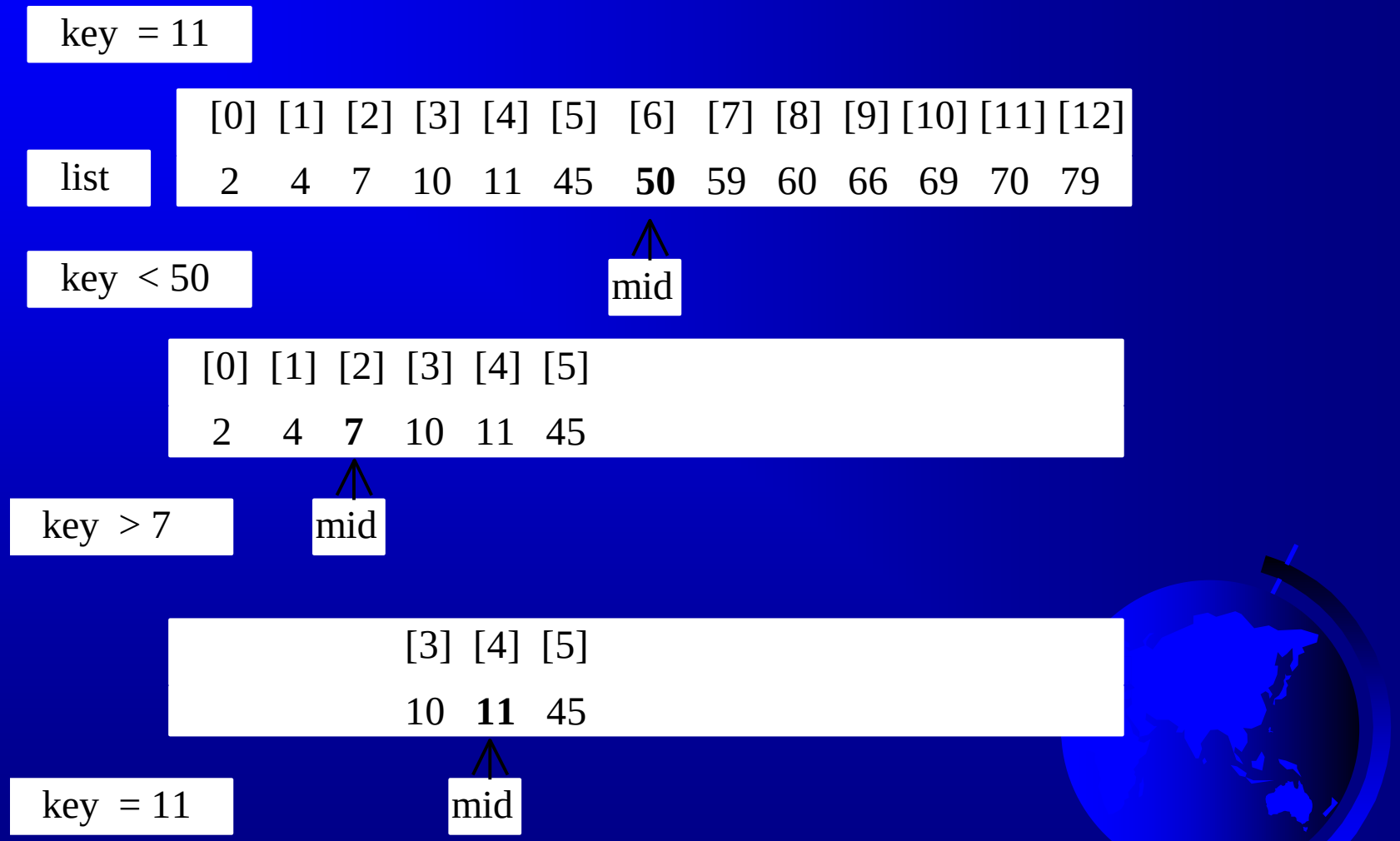
Tìm kiếm nhị phân - Binary Search

- Để thực hiện được tìm kiếm nhị phân, các phần tử mảng phải được sắp xếp theo thứ tự. Không làm mất tính tổng quát, giả sử mảng được sắp xếp tăng dần.
- vd: 2 4 7 10 11 45 50 59 60 66 69 70 79
- Trước tiên so sánh key với phần tử nằm giữa mảng. Có thể xảy ra 3 trường hợp sau:

Tìm kiếm nhị phân (tiếp)

- Nếu key bằng phần tử giữa, việc tìm kiếm kết thúc vì tìm thấy.
- Nếu key nhỏ hơn phần tử giữa, bạn chỉ cần tìm key trong nửa đầu của mảng theo phương pháp nhị phân .
- Nếu key lớn hơn phần tử giữa, bạn chỉ cần tìm key trong nửa cuối của mảng theo phương pháp nhị phân.

Binary Search, cont.



Ví dụ 5.11: Tìm kiếm nhị phân

- Mục tiêu: Tạo mảng 10 phần tử kiểu int rồi hiển thị. Nhận key từ user rồi thực hiện tìm kiếm nhị phân.

BinarySearch

Ví dụ 5.12: Sắp xếp mảng

- Mục tiêu: Viết chương trình sử dụng phương pháp *Sắp xếp chọn* (`selectionSort`) để sắp xếp dãy số thực.

int[] myList = {2, 9, 5, 4, 8, 1, 6}; // chưa sắp xếp

Sắp xếp tăng dần: 1, 2, 4, 5, 6, 8, 9

SelectionSort

Ví dụ 5.12 (tiếp): Sắp xếp chọn

int[] myList = {2, 9, 5, 4, 8, 1, 6}; // chưa sắp xếp

Tìm phần tử lớn nhất trong myList và đổi chỗ với phần tử cuối cùng của myList.

2, *9*, 5, 4, 8, 1, *6* => 2, *6*, 5, 4, 8, 1, *9* *(size = 7)*

2, 6, 5, 4, *8*, *1* => 2, 6, 5, 4, *1*, *8* *(size = 6)*

2, *6*, 5, 4, *1* => 2, *1*, 5, 4, *6* *(size = 5)*

2, 1, *5*, *4* => 2, 1, *4*, *5*

2, 1, *4* => 2, 1, *4*,

2, 1 => 1, 2

Kết quả: 1, 2, 4, 5, 6, 8, 9

Bài tập 5.5: Sắp xếp nổi bọt - Bubble Sort

```
int[] myList = {2, 9, 5, 4, 8, 1, 6};
```

Lần 1: 2, 5, 4, 8, 1, 6, 9
Lần 2: 2, 4, 5, 1, 6, 8, 9
Lần 3: 2, 4, 1, 5, 6, 8, 9
Lần 4: 2, 1, 4, 5, 6, 8, 9
Lần 5: 1, 2, 4, 5, 6, 8, 9
Lần 6: 1, 2, 4, 5, 6, 8, 9

LẬP TRÌNH JAVA

Chương 6: Đối tượng và lớp Object & Class

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 6

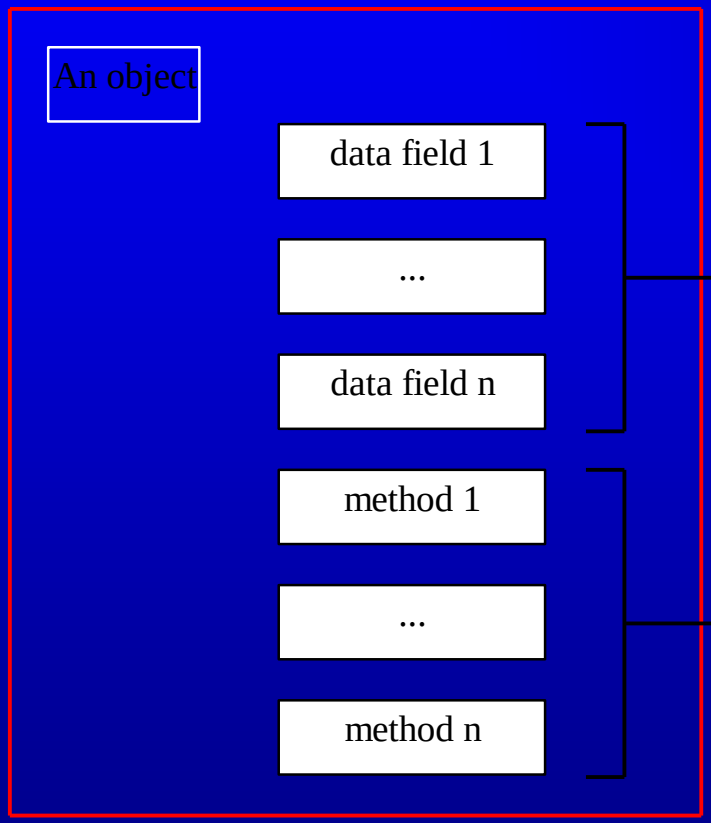
- Khái niệm lập trình hướng đối tượng (OOP)
- Tạo các đối tượng và các biến tham chiếu đối tượng
  - Sự khác nhau giữa dữ liệu kiểu cơ sở và kiểu đối tượng
  - Tự động tập hợp dữ liệu không hợp lệ
- Constructors
- Từ bổ nghĩa (`public`, `private` và `static`)
- Phương thức, biến Class và Instance
- Tầm tác dụng của biến
- Sử dụng từ khóa `this`
- Case Studies (lớp `Mortgage` và lớp `Count`)

Đối tượng (Objects) và Lớp (Class)

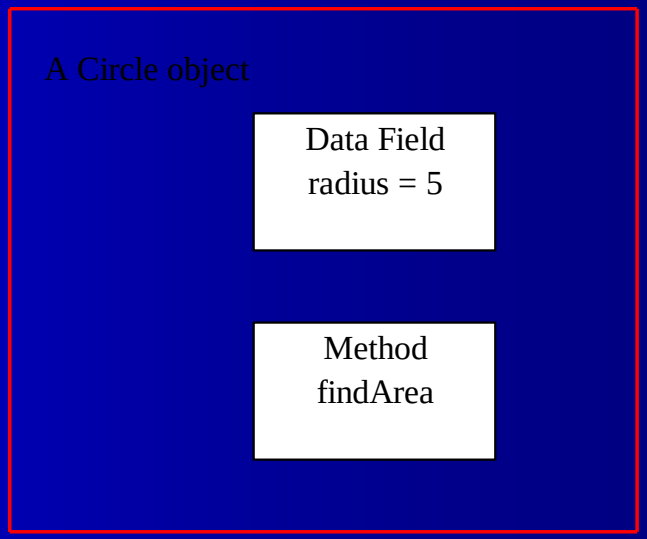
- Đối tượng đại diện cho một thực thể trong thế giới thật
- vd: 1 ô tô, 1 con người, 1 hình tròn, 1 khoản tiền
- Mỗi đối tượng có một `identity, state`, và các `behavior` duy nhất:
- `State` = tập các `data field` (`properties`)
- `Behavior` = tập các `method`
- `State` xác định đối tượng, `behavior` xác định đối tượng làm cái gì.

Khái niệm lập trình hướng đối tượng

OOP - Lập trình sử dụng các đối tượng



Một đối tượng tổng quát



Đối tượng Circle

Khai báo lớp

```
class Circle {
  double radius = 1.0;

  Circle(){
  }

  Circle(double newRadius){
    radius = newRadius;
  }

  double findArea(){
    return radius * radius * 3.14159;
  }
}
```

Data field

Contructors

Method

Khai báo biến tham chiếu đối tượng

```
ClassName objectReference;
```

Ví dụ:

```
Circle myCircle;
```

`myCircle` là 1 instance của lớp `Circle`.

Tạo đối tượng

```
objectReference = new ClassName();
```

Ví dụ:

```
myCircle = new Circle();
```

Tham chiếu đối tượng sẽ được gán cho biến `myCircle`.

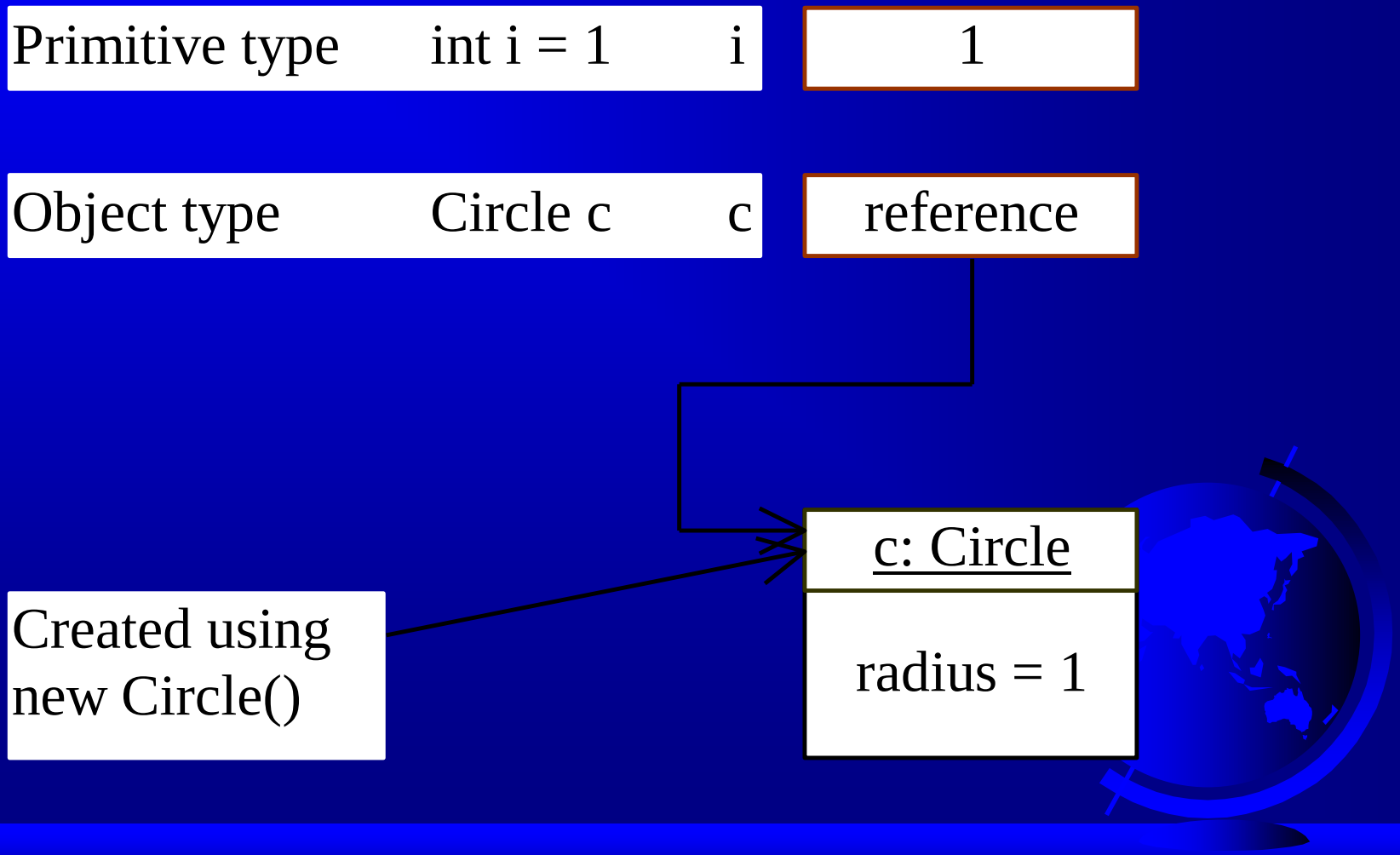
Khai báo/Tạo đối tượng trong 1 lệnh

```
ClassName objectReference = new ClassName();
```

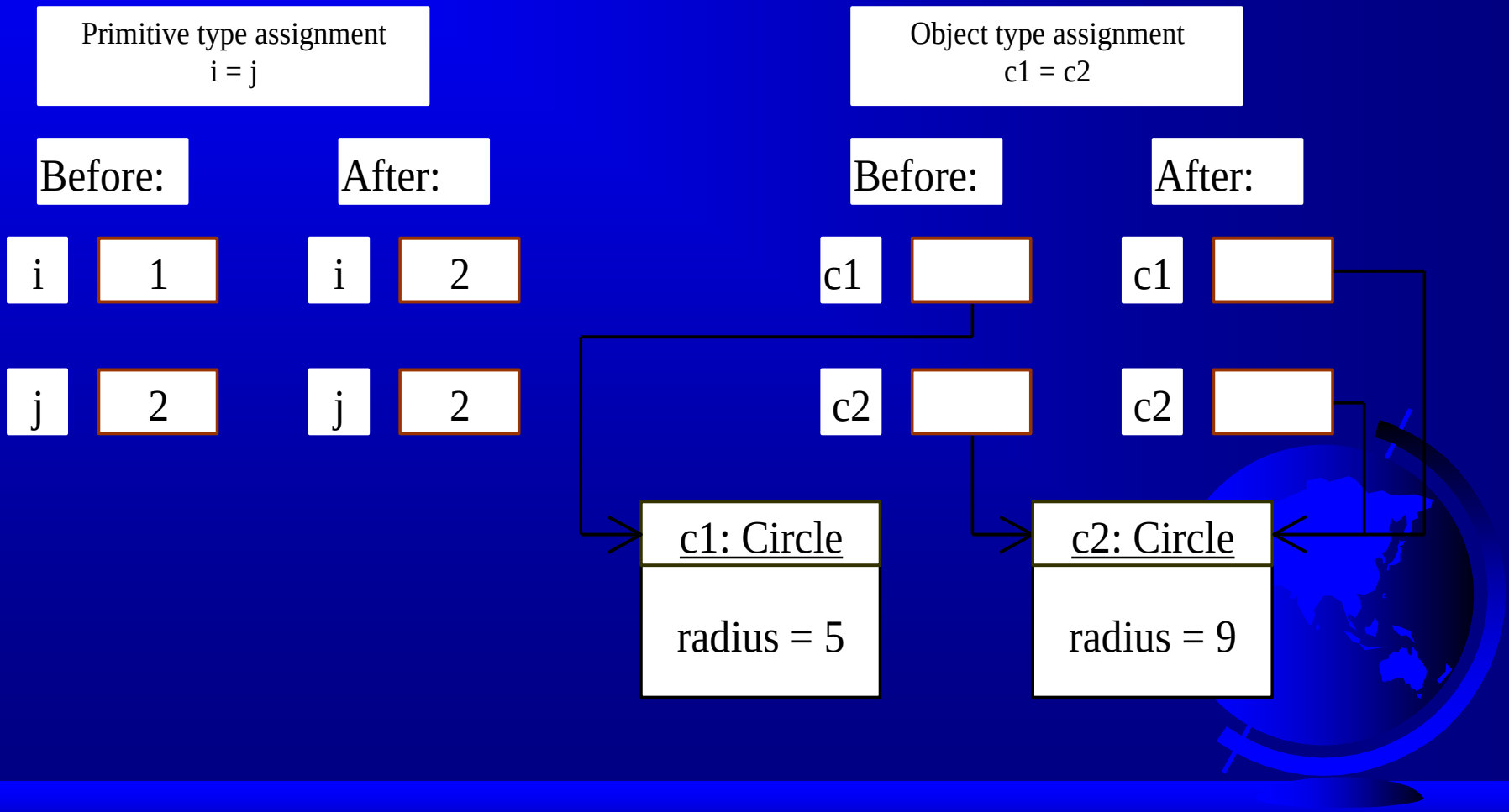
Ví dụ:

```
Circle myCircle = new Circle();
```

Sự khác nhau giữa biến kiểu dữ liệu cơ sở và biến kiểu đối tượng



Copy biến kiểu dữ liệu cơ sở và biến kiểu đối tượng



Tập hợp dữ liệu không sử dụng

- Theo hình trước, sau lệnh gán c1 = c2, c1 trỏ tới cùng một đối tượng được tham chiếu bởi c2. Đối tượng trước đó được tham chiếu bởi c1 trở nên vô dụng, được gọi là garbage. Garbage được tự động tập hợp lại bởi JVM.
- Lời khuyên: Nếu bạn không cần sử dụng một đối tượng nào đó, bạn nên gán biến tham chiếu đối tượng đó trỏ tới null. Java VM sẽ tự động tập hợp bộ nhớ nếu đối tượng không được tham chiếu bởi bất kỳ biến nào.

Truy nhập đối tượng

- Tham chiếu dữ liệu của đối tượng:

```
objectReference.data
```

vd: `myCircle.radius`

- Gọi phương thức của đối tượng:

```
objectReference.method
```

vd: `myCircle.findArea()`

Constructor

```
Circle(double r) {
   radius = r;
}

Circle() {
   radius = 1.0;
}

myCircle = new Circle(5.0);
```

Constructor là một dạng đặc biệt của phương thức, được gọi để xây dựng đối tượng.

Constructor (tiếp)

- Một constructor không có tham số được gọi là *default constructor*.
- Các Constructor phải có cùng tên với class của nó.
- Các Constructor không có kiểu dữ liệu trả về, kể cả kiểu void.
- Các Constructor được gọi sử dụng toán tử `new` khi tạo một đối tượng. Nó đóng vai trò khởi tạo đối tượng.

Ví dụ 6.1: Sử dụng đối tượng

- Mục tiêu: Minh họa việc tạo đối tượng, truy nhập dữ liệu, sử dụng phương thức.

Circle

TestCircle

Ví dụ 6.2:
Sử dụng các lớp từ thư viện Java

- Mục tiêu: Minh họa việc sử dụng các lớp từ thư viện Java. Sử dụng lớp JFrame trong gói javax.swing để tạo 2 frame; sử dụng các phương thức trong lớp JFrame để thiết lập tiêu đề, kích thước, vị trí của các frame và hiển thị chúng.

TestFrame

Ví dụ 6.3: Sử dụng Constructor

- Mục tiêu: Minh họa vai trò của các constructor và sử dụng chúng để tạo các đối tượng.

SimpleCircle

Từ bổ nghĩa và các pp truy nhập

Mặc định: các lớp, biến hoặc dữ liệu có thể được truy nhập bởi bất kỳ lớp nào trong cùng gói (package).

- `public`

  Lớp, dữ liệu, phương thức có thể được truy nhập bởi tất cả các lớp trong bất kỳ gói nào.

- `private`

  Dữ liệu hoặc phương thức chỉ có thể được truy nhập bởi lớp khai báo.

Các phương thức `get` và `set` được sử dụng để đọc và thay đổi các thuộc tính private

Ví dụ 6.4: Sử dụng từ bổ nghĩa `private` và các phương thức truy nhập

Trong ví dụ này, radius sử dụng dữ liệu private, các phương thức truy nhập `getRadius` và `setRadius` được cung cấp để lấy và thay đổi radius.

TestCircle

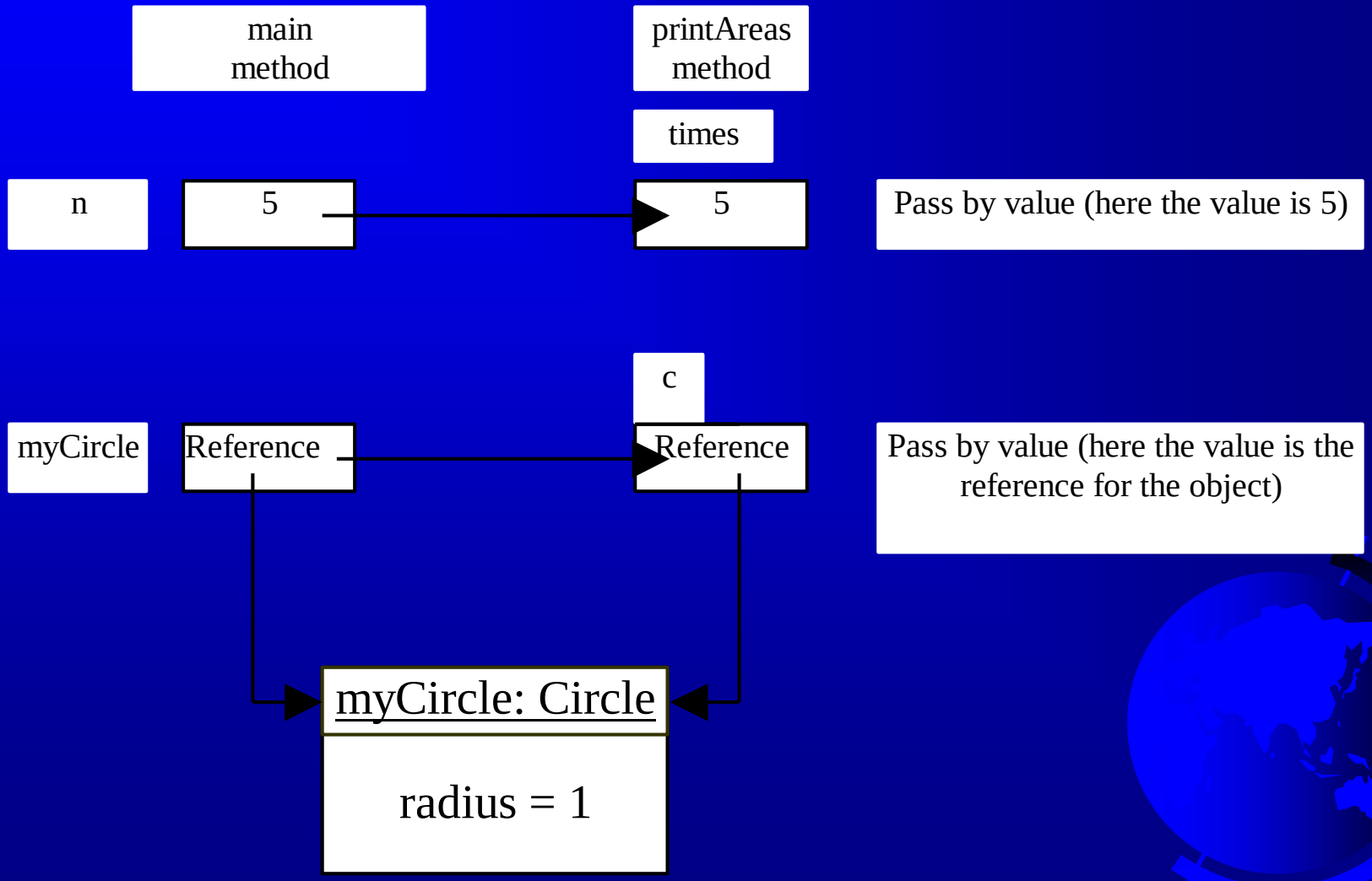
Truyền đối tượng cho phương thức

- Truyền tham trị (giá trị là tham chiếu tới đối tượng)

Ví dụ 6.5: Truyền tham số là đối tượng

TestPassObject

Truyền đối tượng cho phương thức (tiếp)

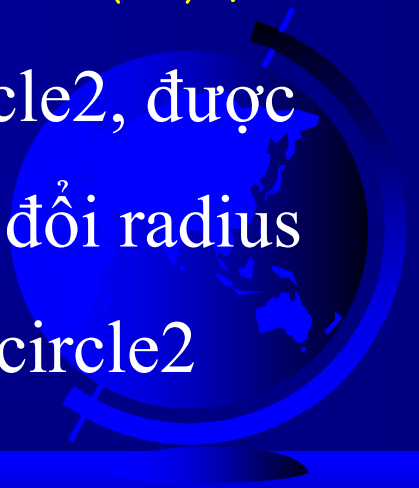


Biến Instance

- Biến instance (vd: biến radius) thuộc một instance xác định, nó không được chia sẻ giữa các đối tượng trong cùng 1 class.

```
Circle circle1 = new Circle();
Circle circle2 = new Circle(5);
```

radius của circle1 độc lập với radius của circle2, được chứa ở những vùng nhớ khác nhau. Sự thay đổi radius của circle1 không ảnh hưởng tới radius của circle2



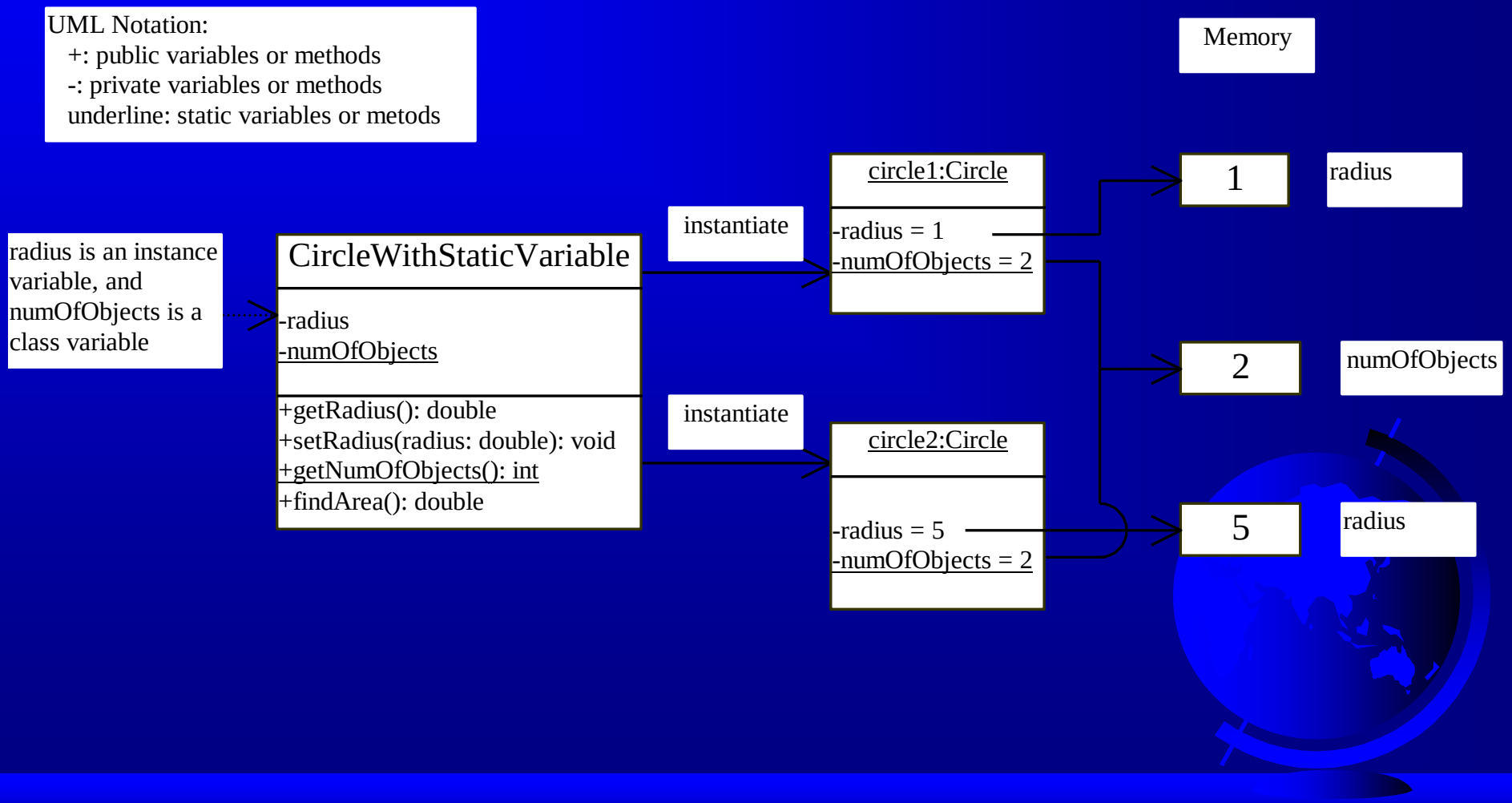
Các biến, hằng, phương thức static

- Để các biến, hằng, phương thức trong class được chia sẻ cho tất cả các instance, sử dụng từ bổ nghĩa static trong khai báo.

- Biến static chứa giá trị trong vùng nhớ chung.

```
private static int numOfObj;
public static int getNumOfObj() {
     return numOfObj;
}
public final static double PI=3.1416;
```

Các biến, hằng, phương thức static (tiếp)



Ví dụ 6.6: Sử dụng biến, phương thức Instance và Static

Mục tiêu: Minh họa vai trò, cách sử dụng của các biến instance và static. Biến static numOfObjects để theo dõi số đối tượng Circle được tạo.

CircleWithStaticVariable

TestCircleWithStaticVariable

Phạm vi các biến

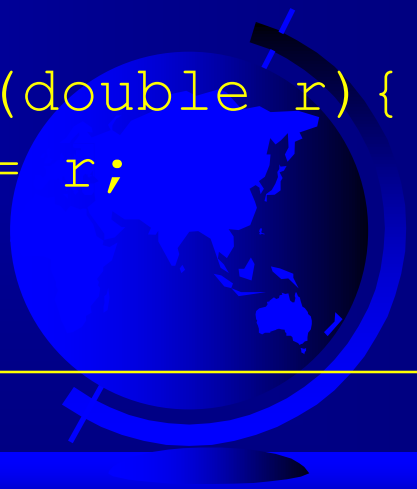
- Phạm vi của các biến static và instance là toàn bộ lớp. Chúng có thể được khai báo bất kỳ nơi nào trong lớp.
- Phạm vi của biến cục bộ bắt đầu từ khi khai báo đến hết block chứa biến đó. Một biến cục bộ phải được khai báo trước khi sử dụng.

Từ khóa `this`

- Dùng `this` để:
  - thay thế cho đối tượng hiện tại.
  - gọi các constructor khác của đối tượng.

```
class Foo {
  int i = 5;

  void setI(int i){
     this.i = i
   }
}
```

```
public class Circle{
  private double rd;

  public Circle(double r){
        this.rd = r;
    }
}
```



Mảng đối tượng

- Khai báo và tạo mảng các đối tượng:

```
Circle[] circleArray = new Circle[10];
```

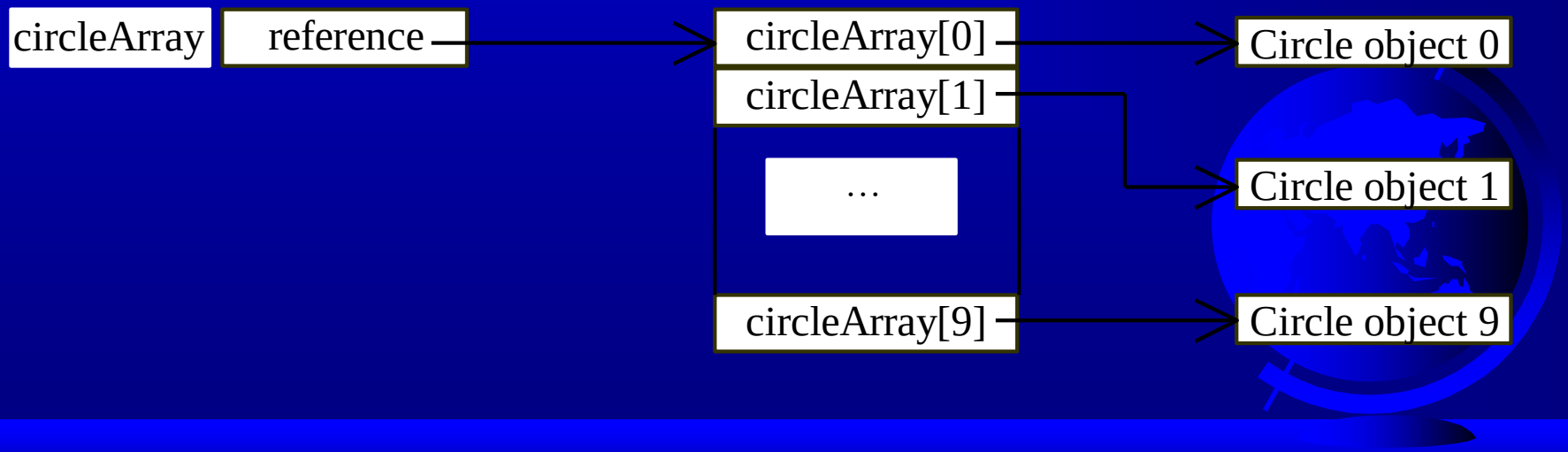
- Khởi tạo:

```
for(int i=0; i<circleArray.length; i++){
  circleArray[i] = new Circle();
}
```

Mảng đối tượng (tiếp)

```
Circle[] circleArray = new Circle[10];
```

Một mảng các đối tượng thực chất là *mảng các biến tham chiếu*. Do đó gọi circleArray[1].findArea() sẽ gọi 2 mức tham chiếu: circleArray tham chiếu toàn bộ mảng, circleArray[1] tham chiếu tới một đối tượng Circle.



Mảng đối tượng (tiếp)

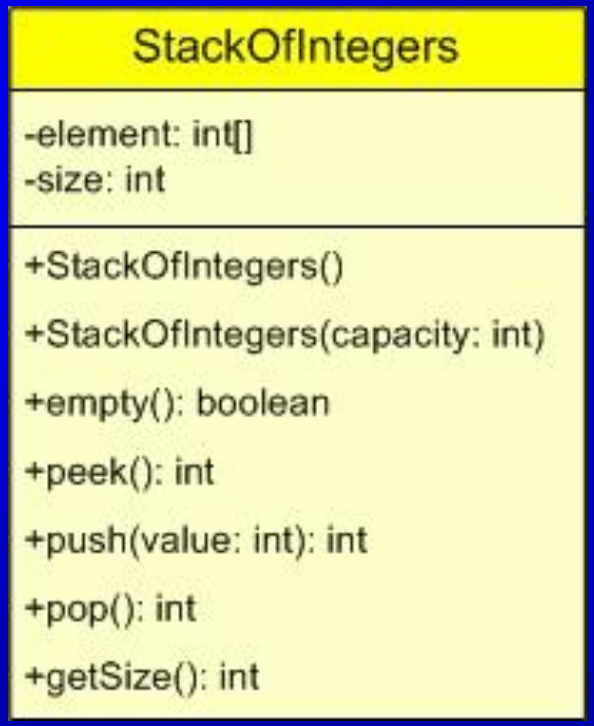
Ví dụ 6.7: Tính tổng diện tích của các hình tròn

TotalArea

Sự trừu tượng của lớp
Class Abstraction

- ✓ Sự thực hiện của lớp tách riêng với sự sử dụng nó.
- ✓ Người tạo lớp cung cấp sự miêu tả lớp để người sử dụng biết cách sử dụng.
- ✓ Người sử dụng không cần biết lớp được thực hiện như thế nào.

Ví dụ 6.8: Stack Of Intergers



StackOfIntegers

TestStackOfIntegers

Các lớp Java API và Core Java

- `java.lang`

  Chứa các lớp Java lõi (core Java class), gồm lớp số (numeric class), chuỗi ký tự, đối tượng. Gói này được import hoàn toàn vào tất cả các CT Java.

- `java.awt`

  Chứa các lớp đồ họa.

- `java.applet`

  Chứa các lớp hỗ trợ applet.

Các lớp Java API và Core Java (tiếp)

- `java.io`

  Chứa các lớp cho các luồng vào-ra và các file.

- `java.util`

  Chứa nhiều tiện ích, ví dụ date.

- `java.net`

  Chứa các lớp hỗ trợ giao tiếp mạng.

Các lớp Java API và Core Java (tiếp)

- `java.awt.image`

  Chứa các lớp giúp quản lý các ảnh bitmap.

- `java.awt.peer`

  Thực hiện Platform-specific GUI.

- Các lớp khác:

  `java.sql`

  `java.rmi`

LẬP TRÌNH JAVA

Chương 7: String

Phạm Quang Dũng
BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 7

Xử lý chuỗi ký tự sử dụng các lớp String,StringBuffer, và StringTokenizer:

- Sử dụng lớp String để xử lý các chuỗi cố định.
- Sử dụng các phương thức static trong lớp Character.
- Sử dụng lớp StringBuffer để xử lý các chuỗi linh động.
- Sử dụng lớp StringTokenizer để trích chuỗi con.
- Sử dụng các đối số dòng lệnh (command-line arguments).

Lớp `String`

- Xây dựng chuỗi:
  - `String message = "Welcome to Java!"`
  - `String message = new String("Welcome to Java!");`
  - `String s = new String();`
- Lấy độ dài chuỗi và lấy các ký tự cụ thể trong chuỗi.
- Ghép chuỗi (concat)
- Chuỗi con (substring(index), substring(start, end))
- So sánh (equals, compareTo)
- String Conversions
- Tìm 1 ký tự hoặc chuỗi con trong 1 chuỗi
- Chuyển đổi giữa Strings và Arrays
- Chuyển đổi các ký tự và các g/t số thành chuỗi

String

+String()
+String(value: String)
+String(value: char[])
+charAt(index: int): char
+compareTo(anotherString: String): int
+compareToIgnoreCase(anotherString: String): int
+concat(anotherString: String): String
+endsWithSuffixe(suffix: String): boolean
+equals(anotherString: String): boolean
+equalsIgnoreCase(anotherString: String): boolean
+indexOf(ch: int): int
+indexOf(ch: int, fromIndex: int): int
+indexOf(str: String): int
+indexOf(str: String, fromIndex: int): int
+intern(): String
+regionMatches(toffset: int, other: String, offset: int, len: int): boolean
+length(): int
+replace(oldChar: char, newChar: char): String
+startsWith(prefix: String): boolean
+subString(beginIndex: int): String
+subString(beginIndex: int, endIndex: int): String
+toCharArray(): char[]
+toLowerCase(): String
+toString(): String
+toUpperCase(): String
+trim(): String
+copyValueOf(data: char[]): String
+valueOf(c: char): String
+valueOf(data: char[]): String
+valueOf(d: double): String
+valueOf(f: float): String
+valueOf(i: int): String
+valueOf(l: long): String

Xây dựng chuỗi

```
String newString = new String(stringLiteral);
```

Vd:

```
String message = new String("Welcome to Java!");
```

- Vì chuỗi được sử dụng thường xuyên, Java cung cấp lệnh ngắn để tạo chuỗi:

```
String message = "Welcome to Java!";
```

- Cũng có thể tạo chuỗi từ mảng các ký tự:

```
char[] charArr = {'G','o','o','d',' ','d','a','y'};
String message = new String(charArr);
```

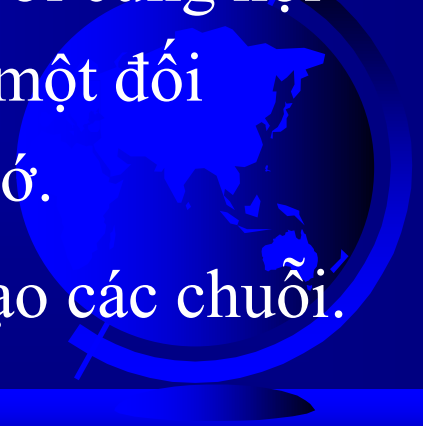
Các chuỗi là không thể thay đổi

- CHÚ Ý: Một đối tượng String là không thể thay đổi nội dung. Xét ví dụ sau:

```
String s = "Java";
s = "HTML";
```

Vd trên: đối tượng String lưu chuỗi "Java" vẫn tồn tại nhưng không được tham chiếu bởi biến nào cả.

- Nếu sử dụng lệnh tắt tạo 2 đối tượng String với cùng nội dung, JVM lưu 2 đối tượng đó vào trong cùng một đối tượng để cải thiện hiệu năng và tiết kiệm bộ nhớ.
- Do đó, người ta thường sử dụng lệnh tắt để tạo các chuỗi.



Các chuỗi là không thể thay đổi (tiếp)

- CHÚ Ý: Một chuỗi được tạo bởi lệnh tắt được coi là một *canonical string* (chuỗi hợp quy tắc tiêu chuẩn).
- Bạn có thể sử dụng phương thức `intern` của lớp String để trả về một canonical string, tương tự như string được tạo bởi lệnh tắt.

Ví dụ

```
String s = "Welcome to Java!";
String s1 = new String("Welcome to Java!");
String s2 = s1.intern();
System.out.println("s1 == s is " + (s1 == s));
System.out.println("s2 == s is " + (s2 == s));
System.out.println("s1 == s2 is " +(s1 == s2));
```

Hiển thị:

```
s1 == s is false
s2 == s is true
s1 == s2 is false
```

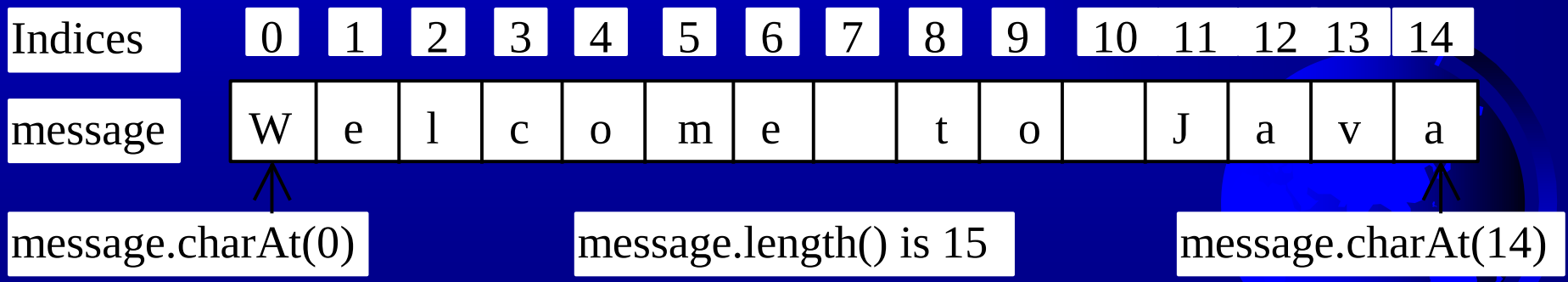
Lấy độ dài chuỗi

Sử dụng phương thức `length()` :

```
message = "Welcome";
message.length() (returns 7)
```

Lấy các ký tự trong chuỗi

- Không sử dụng `message[0]`
- Mà dùng `message.charAt(index)`
- Chỉ số (index) bắt đầu từ `0`



Ghép chuỗi

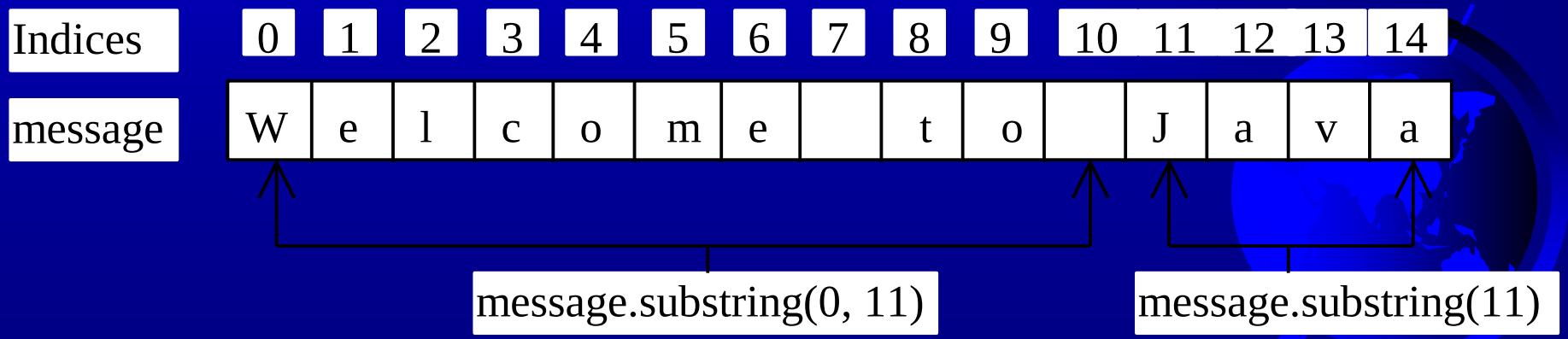
```
String s3 = s1.concat(s2);
```

String s3 = s1 + s2;

Trích chuỗi con

`String` là một lớp không thể thay đổi; giá trị của nó không bị thay đổi một cách riêng lẻ.

```
String s1 = "Welcome to Java";
String s2 = s1.substring(0, 11) + "HTML";
```



So sánh chuỗi

- equals

```
String s1 = "Welcome to Java!";
String s2 = new String("Welcome to Java!";

if (s1.equals(s2)){
  // s1 and s2 have the same contents
}

if (s1 == s2) {
  // s1 and s2 have the same reference
 }
```

Chú ý: `s1.equals(s2)` là True khi và chỉ khi `s1.intern() == s2.intern()`.

So sánh chuỗi (tiếp)

- compareTo(Object object)

```
String s1 = "Welcome";
String s2 = "welcome";

if (s1.compareTo(s2) > 0) {
  // s1 is greater than s2
}
else if (s1.compareTo(s2) == 0) {
  // s1 and s2 have the same reference
}
else
  // s1 is less than s2
```

String Conversions

Nội dung của một chuỗi không thể thay đổi mỗi khi chuỗi được tạo. Nhưng bạn có thể convert một chuỗi thành một chuỗi mới bằng cách sử dụng các phương thức sau:

- `"Welcome".toLowerCase()`
- `"Welcome".toUpperCase()`
- `"  Welcome ".trim()`
- `"Welcome".replace('e','A')`
- `"Welcome".replaceFirst('e','A')`
- `"Welcome".replaceAll('e','A')`

Tìm ký tự và chuỗi con

Sử dụng các phương thức sau:

```
public int indexOf(int ch)
public int lastIndexOf(int ch)
public int indexOf(int ch, int fromIndex)
public int lastIndexOf(int ch, int endIndex)
public int indexOf(String str)
public int lastIndexOf(String str)
public int indexOf(String ch, int fromIndex)
public int lastIndexOf(String str, int endIndex)
```

Ví dụ

```
"Welcome to Java!".indexOf('W') returns 0.
"Welcome to Java!".indexOf('x') returns -1.
"Welcome to Java!".indexOf('o', 5) returns 9.
"Welcome to Java!".indexOf("come") returns 3.
"Welcome to Java!".indexOf("Java", 5) returns
11.
"Welcome to Java!".indexOf("java", 5) returns
-1.
```

Chuyển đổi ký tự và số thành chuỗi

- `char[] chars = "Java".toCharArray();`
- `String str = new String(new char[]`
  `{'J','a','v','a'});`
- Lớp String cung cấp một số phương thức static valueOf để chuyển đổi một ký tự, mảng ký tự, và các giá trị số thành chuỗi.
- Những phương thức này có cùng tên valueOf với tham số khác nhau có kiểu char, char[], double, long, int, và float.

```
String str = String.valueOf(5.44);
String str = String.valueOf(new char[]
                    {'J','a','v','a'});
```

Ví dụ 7.1: Tìm chuỗi đối xứng

- Mục tiêu: Kiểm tra xem một chuỗi có đối xứng hay không.

CheckPalindrome

Lớp Character

Character
+Character(value: char) +charValue(): char +compareTo(anotherCharacter: Character): int +equals(anotherCharacter: Character): boolean +isDigit(ch: char): boolean +isLetter(ch: char): boolean +isLetterOrDigit(ch: char): boolean +isLowerCase(ch: char): boolean +isUpperCase(ch: char): boolean +toLowerCase(ch: char): char +toUpperCase(ch: char): char

Ví dụ

```
Character charObj = new Character('b');
charObj.compareTo(new Character('a')) returns
1
charObj.compareTo(new Character('b')) returns
0
charObj.compareTo(new Character('c')) returns
-1
charObj.compareTo(new Character('d') returns
-2
charObj.equals(new Character('b')) returns
```

Ví dụ 7.2: Đếm mỗi ký tự trong chuỗi

Mục tiêu: đếm tần số xuất hiện của mỗi ký tự trong một chuỗi. Giả sử không phân biệt ký tự hoa, thường.

CountEachLetter

Lớp `StringBuffer`

- Lớp `StringBuffer` là một lựa chọn khác của lớp `String`. Nói chung, một string buffer có thể được sử dụng thay thế cho một string.

- `StringBuffer` linh hoạt hơn `String`. Bạn có thể add, insert, hoặc append nội dung mới vào một string buffer. Với một string thì không thể thay đổi nội dung nó được tạo.

StringBuffer

+append(data: char[]): StringBuffer
+append(data: char[], offset: int, len: int): StringBuffer
+append(v: *aPrimitiveType*): StringBuffer
+append(str: String): StringBuffer
+capacity(): int
+charAt(index: int): char
+delete(startIndex: int, endIndex: int): StringBuffer
+deleteCharAt(int index): StringBuffer
+insert(index: int, data: char[], offset: int, len: int): StringBuffer
+insert(offset: int, data: char[]): StringBuffer
+insert(offset: int, b: *aPrimitiveType*): StringBuffer
+insert(offset: int, str: String): StringBuffer
+length(): int
+replace(int startIndex, int endIndex, String str): StringBuffer
+reverse(): StringBuffer
+setCharAt(index: int, ch: char): void
+setLength(newLength: int): void
+substring(start: int): StringBuffer
+substring(start: int, end: int): StringBuffer

`StringBuffer` Constructors

- `public StringBuffer()`

  Xây dựng một string buffer rỗng có dung lượng = 16.

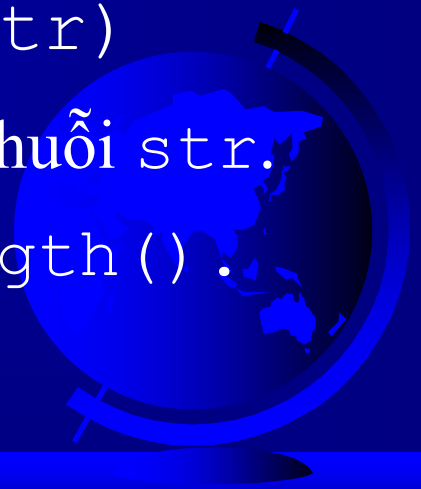
- `public StringBuffer(int length)`

  Xây dựng một string buffer rỗng có dung lượng = length.

- `public StringBuffer(String str)`

  Xây dựng một string buffer với nội dung là chuỗi `str`.

  Dung lượng khởi tạo bằng `16 + str.length()`.



Nối thêm nội dung vào một String Buffer

```
StringBuffer strBuf = new StringBuffer();
strBuf.append("Welcome");
strBuf.append(' ');
strBuf.append("to");
strBuf.append(' ');
strBuf.append("Java");
```

Sửa đổi String Buffer

Giả sử `strBuf` chứa chuỗi "Welcome to Java!" trước mỗi lời gọi phương thức sau:

```
strBuf.insert(11,"HTML and ");

strBuf.delete(8,11);

strBuf.deleteCharAt(8);

strBuf.reverse();

str.replace(11,15,"HTML");

strBuf.setCharAt(0,'w');
```

Ví dụ 7.3: Kiểm tra chuỗi đối xứng, bỏ qua các ký tự khác chữ và số

PalindromeIgnoreNonAlphanumeric

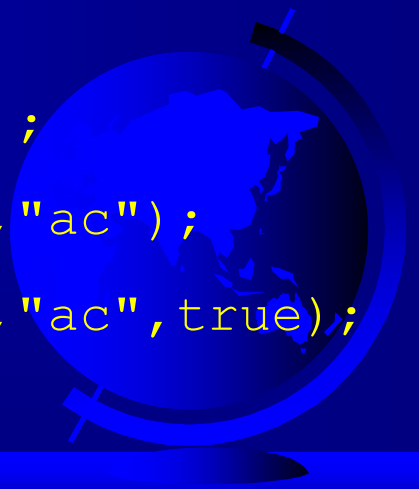
Các Constructor của lớp `StringTokenizer`

Chuỗi có thể được bẻ thành các mảnh gọi là token bởi các delimiter.

- `StringTokenizer(String s, String delim, boolean returnTokens)`
- `StringTokenizer(String s, String delim)`
- `StringTokenizer(String s)`

Vd: `String s = "Java is cool."`

```
StringTokenizer tkz = new StringTokenizer(s);
StringTokenizer tkz = new StringTokenizer(s,"ac");
StringTokenizer tkz = new StringTokenizer(s,"ac",true);
```

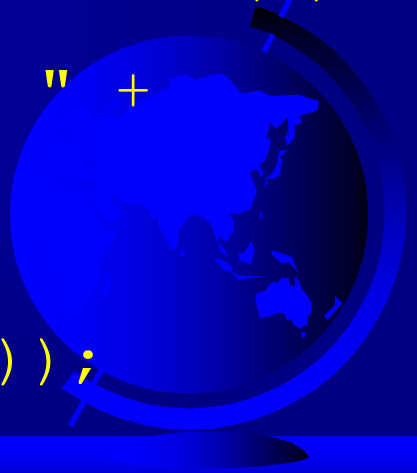


Các phương thức của lớp StringTokenizer

- boolean hasMoreTokens()
- String nextToken()
- String nextToken(String delim)

Vd: String s = "Java is cool."

```
StringTokenizer tkz = new StringTokenizer(s);
System.out.println("Tong so token = " +
  tkz.countTokens());
while (tkz.hasMoreTokens())
  System.out.println(tkz.nextToken());
```



Ví dụ 7.4
Testing `StringTokenizer`

- Mục tiêu: Sử dụng một string tokenizer, lấy các từ trong chuỗi rồi hiển thị chúng.

TestStringTokenizer

Lớp Scanner

- Có từ JDK 1.5 (gói java.util.Scanner)
- Có thể dùng lớp Scanner để:
  - ấn định 1 từ làm Delimiter
  - nhập dữ liệu thuộc các kiểu khác nhau
- Ấn định 1 từ làm Delimiter:

```
String s = "Java is fun! Java is cool!";
Scanner scr = new Scanner(s);
scr.useDelimiter("Java");
while (scr.hasNext())
  System.out.prinln(scr.next());
```

Lớp Scanner (tiếp)

- Nhập dữ liệu:
  - Tạo đối tượng Scanner để đọc dữ liệu vào từ System.in

```
Scanner scanner = new Scanner(System.in)
```

  - Sử dụng các phương thức `next()`, `nextByte()`, `nextShort()`, `nextInt()`, `nextLong()`, `nextFloat()`, `nextDouble()`, hoặc `nextBoolean()` để lấy một chuỗi, byte, short, int, long, float, double, hoặc boolean.
  - Nên tạo 1 lớp (vd MyInput) gồm các phương thức đọc dữ liệu kiểu string và các kiểu cơ bản, đặt cùng thư mục CT.

MyInputUsingScanner

Các tham số dòng lệnh

- Có thể truyền chuỗi cho phương thức main từ dòng lệnh khi chạy chương trình.
- Khi phương thức main được gọi, trình biên dịch Java tạo 1 mảng chứa các tham số dòng lệnh và truyền tham chiếu mảng cho args.

```
class TestMain {
   public static void main(String[] args) {
   ...
   }
}
```

```
java TestMain arg0 arg1 arg2 ... argn
```

Xử lý các tham số dòng lệnh

- Các chuỗi được truyền cho chương trình chính được chứa trong args - là một mảng các chuỗi. Các phần tử: `args[0], args[1], ..., args[n]`, tương ứng với `arg0, arg1, ..., argn` trong dòng lệnh.
- `args.length` là số chuỗi được truyền.

Ví dụ 7.6:
Sử dụng các tham số dòng lệnh

- Mục tiêu: Viết chương trình thực hiện các phép toán 2 ngôi trên số nguyên. Chương trình nhận vào 3 tham số: 1 toán tử và 2 số nguyên.

Calculator

Run

java Calculator 10 + 5

java Calculator 10 - 5

java Calculator 10 / 5

java Calculator 10 "*" 5

LẬP TRÌNH JAVA

Chương 8: Kế thừa và đa hình thái
Inheritance & Polymorphism

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 8

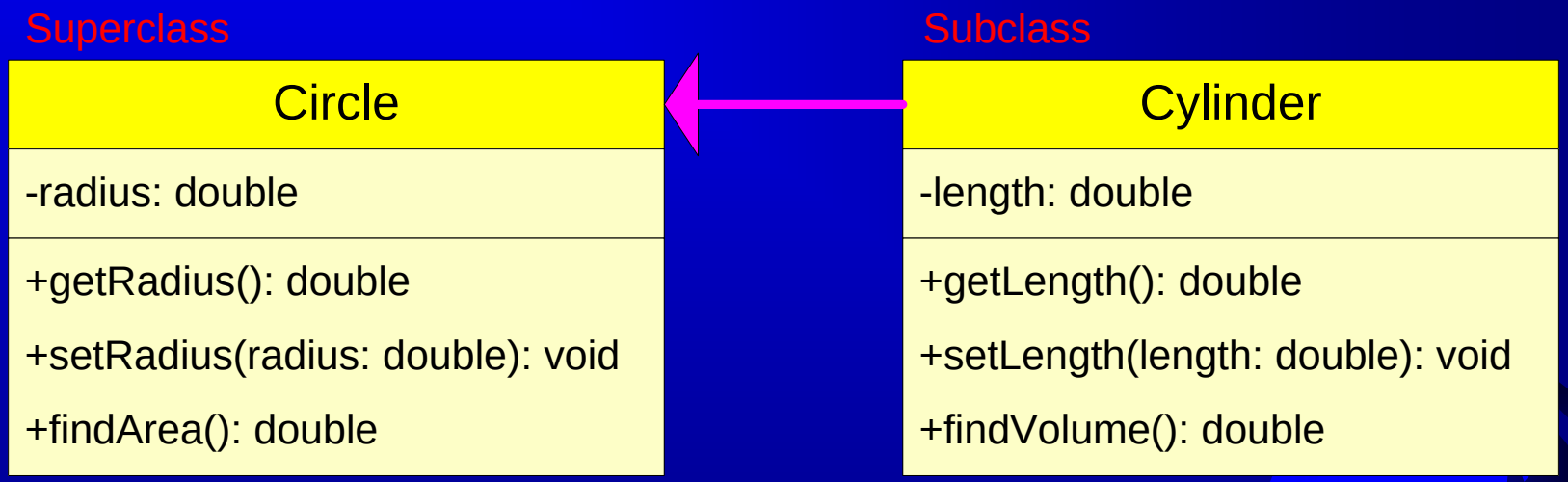
- Phát triển 1 subclass từ 1 superclass thông qua kế thừa
- Dùng từ khóa super gọi các constructor và phương thức của superclass
- Chồng phương thức trong subclass
- Lớp Object
- Đa hình thái, nối kết động, lập trình dùng chung
- Ép kiểu đối tượng và toán tử `instanceof`
- Dữ liệu và phương thức `protected`
- Các `Abstract class` và `Interface`

Superclass và Subclass

- Lập trình hướng đối tượng cho phép bạn phát triển những lớp mới từ các lớp đã tồn tại.
- Vd: lớp C1 được phát triển từ lớp C2:
  - C1: subclass, extended class, derived class
  - C2: superclass, parent class, base class
- Subclass thừa kế từ superclass các trường dữ liệu và phương thức *có thể truy nhập được*, và cũng có thể thêm vào các trường dữ liệu và phương thức mới.

Superclass và Subclass (tiếp)

- Thực tế, subclass thường được mở rộng để chứa nhiều thông tin chi tiết và nhiều chức năng hơn so với superclass của nó.



Circle | Cylinder1 | TestCylinder1

Sử dụng từ khóa `super`

- `super` được dùng để thay cho superclass, tương tự như `this` thay cho đối tượng được gọi.
- Dùng `super` để:
  - gọi 1 contructor của superclass
  - gọi 1 phương thức của superclass

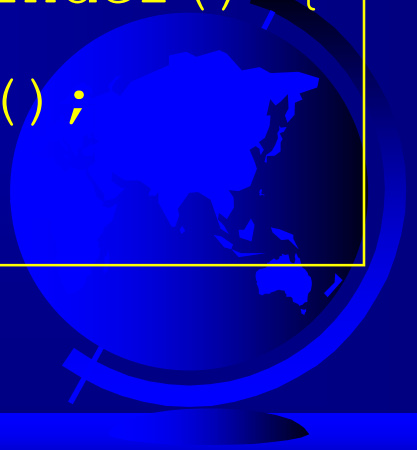
Gọi Superclass Contructor

- `super()`, hoặc `super(tham_số)`
- Lệnh trên phải được *đặt tại dòng đầu tiên* của subclass constructor và là cách duy nhất để gọi 1 superclass constructor.

```
public Cylinder() {
}
```

==

```
public Cylinder() {
    super();
}
```



Gọi Superclass Method

- `super.method(tham_số)`

Vd:

```
double findVolume() {
      return super.findArea() * length;
}
```

Overriding Method

- Đôi khi subclass cần phải thay đổi sự thực hiện của phương thức trong superclass → Chồng phương thức.
- Vd: phương thức findArea của lớp Circle tính diện tích hình tròn. Phương thức này nên được chồng trong lớp Cylinder để tính diện tích bề mặt hình trụ.

Cylinder

Chồng phương thức (tiếp)

- Để chồng, phương thức xác định trong subclass phải có *cùng signature* và *cùng kiểu dữ liệu trả về* với phương thức trong superclass.
- Review: Các Overloading method có cùng tên, nhưng phải khác signature.
- Một phương thức chỉ có thể được chồng chỉ khi nó có thể truy nhập được → không thể chồng 1 private method.
- Một static method có thể được kế thừa, nhưng không thể được chồng.

Lớp `Object`

- Mọi lớp trong Java được thừa kế từ lớp `java.lang.Object`
- Nếu không có sự kế thừa nào được xác định khi một lớp được tạo thì superclass của nó chính là lớp Object.
- 3 phương thức của lớp Object thường được sử dụng:
  - `public boolean equals(Object obj)`
  - `public int hashCode()`
  - `public String toString()`

Phương thức `equals`

```
object1.equals(object2);
```

- Sự thực hiện ngầm định:

```
public boolean equals(Object obj) {
      return (this == obj);
}
```

- Được chồng trong các subclass để kiểm tra 2 đối tượng riêng biệt có cùng nội dung hay không.
- Vd: str1.equals(str2) trong lớp String
- Lưu ý khi viết phương thức chồng:

Dùng `equals(Object obj)`, not `equals(Circle obj)`

Phương thức `toString`

- Gọi `obj.toString()` trả về chuỗi biểu diễn cho đối tượng `obj`, ngầm định là `classname@hashcode`

```
Cylinder myCyl = new Cylinder(5.0,2.0);
System.out.println(myCyl.toString());
```

→ Cylinder@15037e5

- Thường chồng phương thức `toString` để trả về một chuỗi dễ hiểu biểu diễn đối tượng. Vd:

```
public String toString() {
   return "Cylinder length = " + length +
                "  radius = " + getRadius();
}
```

Đa hình thái - Polymorphism

- Xét ví dụ:

TestPolymorphirm

→ Có thể gọi m với bất kỳ đối tượng nào, vd:
`new GraduateStudent()`, `new Student()`, `new Person()`, và `new Object()`

- Đa hình thái: Một đối tượng của subclass có thể được sử dụng bởi bất kỳ mã lệnh nào được thiết kế để làm việc với một đối tượng của superclass.

Nối kết động - Dynamic binding

- Ở vd trước, mỗi lớp có sự thực hiện phương thức `toString` riêng. Việc thực hiện phương thức nào sẽ được JVM xác định khi chạy chương trình → Nối kết động.
- obj là instance của các lớp c_1, c_2, …, c_n. Khi obj gọi phương thức p, JVM tìm sự thực hiện p trong các lớp theo thứ tự c_1, c_2, …, c_n đến khi tìm thấy.

Lập trình dùng chung

- Đa hình thái cho phép các phương thức được sử dụng chung cho một dải rộng các tham số đối tượng → generic programming.
- Trong lập trình hướng đối tượng, nên lập trình theo cách dùng chung: khai báo 1 biến có kiểu superclass, nó sẽ có thể chấp nhận một giá trị của bất kỳ kiểu subclass nào.
- Tuy nhiên muốn vậy cần ép kiểu đối tượng.

Ép kiểu đối tượng

```
m(new Student());
```

thực hiện gán đối tượng new Student() cho một tham số kiểu Object, tương đương với 2 lệnh:

```
Object obj = new Student();     // ép kiểu ngầm
m(obj);
```

upcasting

➢ Muốn ấn định obj (kiểu Object) là một đối tượng Student:

```
Student std = (Student) obj; //ép kiểu rõ ràng
```

downcasting

not `Student std = obj;`

Toán tử `instanceof`

Để ép kiểu đối tượng thành công, trước đó cần chắc chắn rằng đối tượng cần ép là 1 instance của đối tượng kia.

→ dùng toán tử `instanceof`

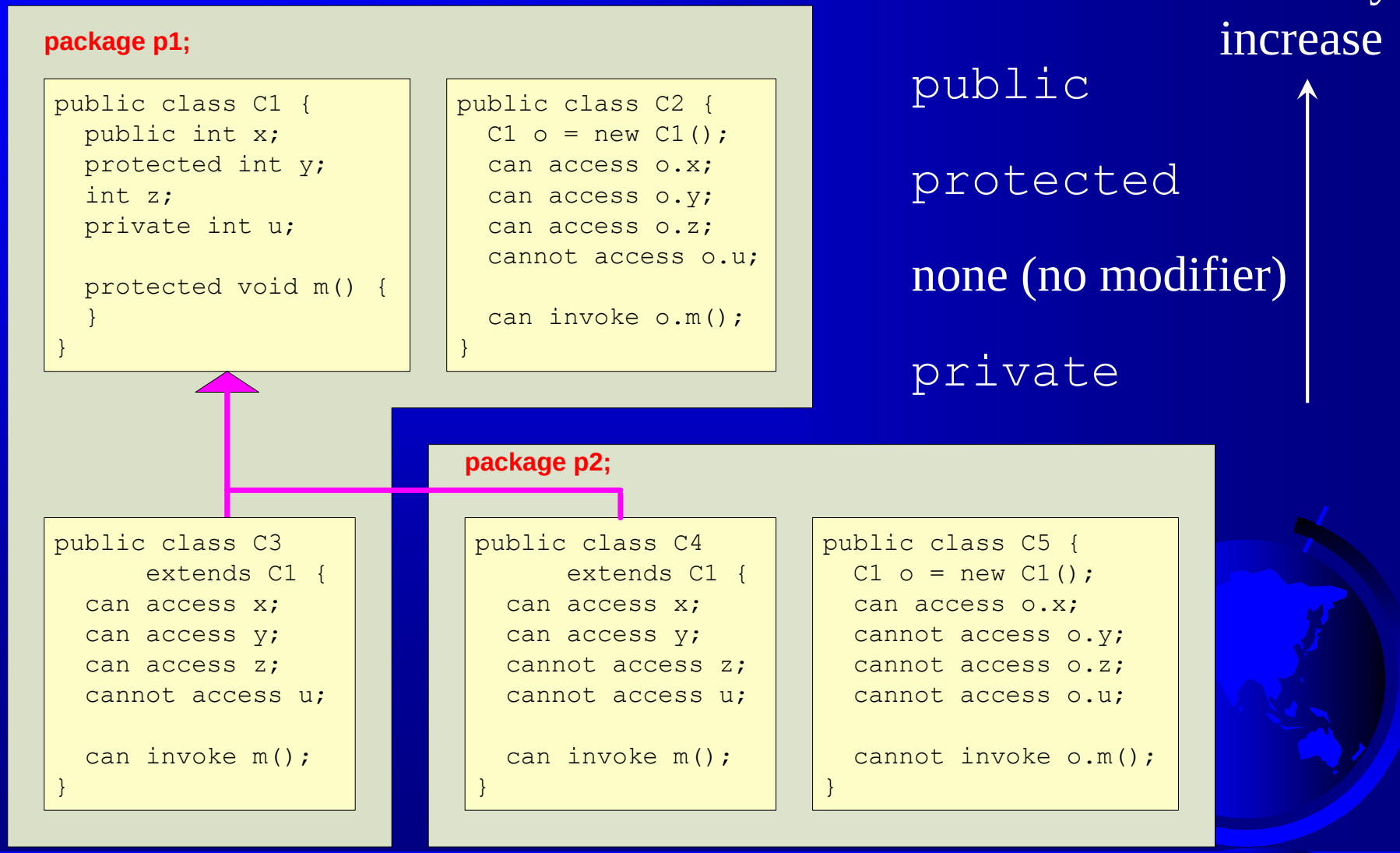
```
/** Giả sử myObj được khai  báo kiểu Object */
/** Thực hiện ép kiểu nếu myObj là 1 instance của Cylinder */
if (myObj instanceof Cylinder) {
   Cylinder myCyl = (Cylinder)myObj;
   System.out.println("The tich hinh
            tru la " + myCyl.findVolume());
     …
}
```

Ví dụ: Minh họa đa hình thái và ép kiểu

Mục tiêu: viết 1 chương trình tạo 2 đối tượng: 1 circle và 1 cylinder. Sau đó gọi phương thức displayObject để hiển thị diện tích nếu đối tượng là 1 circle, hiển thị diện tích và thể tích nếu đối tượng là 1 cylinder.

TestPolymorphirmCasting

Visibility Modifiers



`private` và `protected` modifiers

- Sử dụng `private` để ẩn hoàn toàn các thành phần của class (dữ liệu, phương thức), chúng sẽ *không thể được truy nhập trực tiếp từ bên ngoài lớp*.
- Sử dụng `protected` để cho phép các thành phần của class *được truy nhập bởi các subclass trong bất kỳ package nào*, *hoặc các class trong cùng package*.
- 2 từ khóa trên *chỉ có thể sử dụng cho các thành phần của class*, không thể sử dụng cho class.

`public` và default modifiers

- Sử dụng default modifier (no modifier) thì các thành phần của class *được truy nhập từ bất kỳ lớp nào trong cùng package*, nhưng không thể từ package khác.
- Sử dụng `public` cho phép các thành phần của class có thể *được truy nhập từ bất kỳ lớp nào*.
- `public` và default modifier có thể được sử dụng cho các thành phần của class, cũng như sử dụng cho chính class.

Chú ý

- Các ký hiệu -, #, + được sử dụng để biểu diễn tương ứng các modifier `private`, `protected`, và `public`.
- Khi chồng phương thức:
  - Nếu phương thức trong superclass là `protected` thì có thể thay đổi phương thức chồng trong subclass thành `public`.
  - Nhưng nếu phương thức trong superclass là `public` thì phương thức chồng trong subclass bắt buộc cũng phải là `public`.

Abstract classes

- Đôi khi một superclass quá trừu tượng đến mức nó không thể có instance. Nó được gọi là một abstract class.
- Các abstract class có dữ liệu và phương thức tương tự như các class khác.
- Không thể dùng toán tử new để tạo các instance của abstract class.
- Ví dụ header của 1 abstract class:

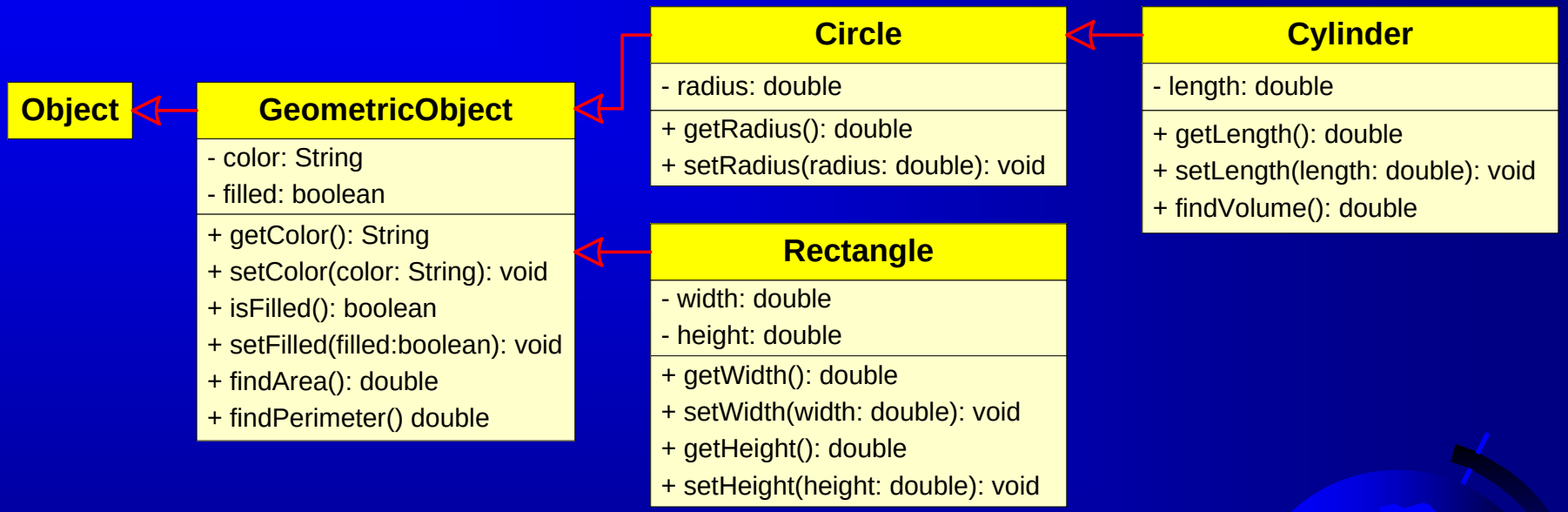
```
public abstract class GeometricObject
```

Abstract methods

- Là các phương thức chỉ có header, không có sự thực hiện (vì quá trừu tượng).
- Sự thực hiện của nó được cung cấp bởi các subclass thông qua các phương thức chồng.
- Một class có chứa abstract method phải là abstract class.
- Ví dụ khai báo 1 abstract method:

```
public abstract double findArea();
```

Ví dụ


Object
GeometricObject
- color: String
- filled: boolean
+ getColor(): String
+ setColor(color: String): void
+ isFilled(): boolean
+ setFilled(filled:boolean): void
+ findArea(): double
+ findPerimeter() double
Circle
- radius: double
+ getRadius(): double
+ setRadius(radius: double): void
Rectangle
- width: double
- height: double
+ getWidth(): double
+ setWidth(width: double): void
+ getHeight(): double
+ setHeight(height: double): void
Cylinder
- length: double
+ getLength(): double
+ setLength(length: double): void
+ findVolume(): double

Interface

- Mỗi lớp Java có thể kế thừa trực tiếp từ một superclass (với từ khóa extends)
  → đơn kế thừa.
- Đôi khi ta cần nhận được một subclass từ một vài class → đa kế thừa.
- Có thể sử dụng các interface.

Interface

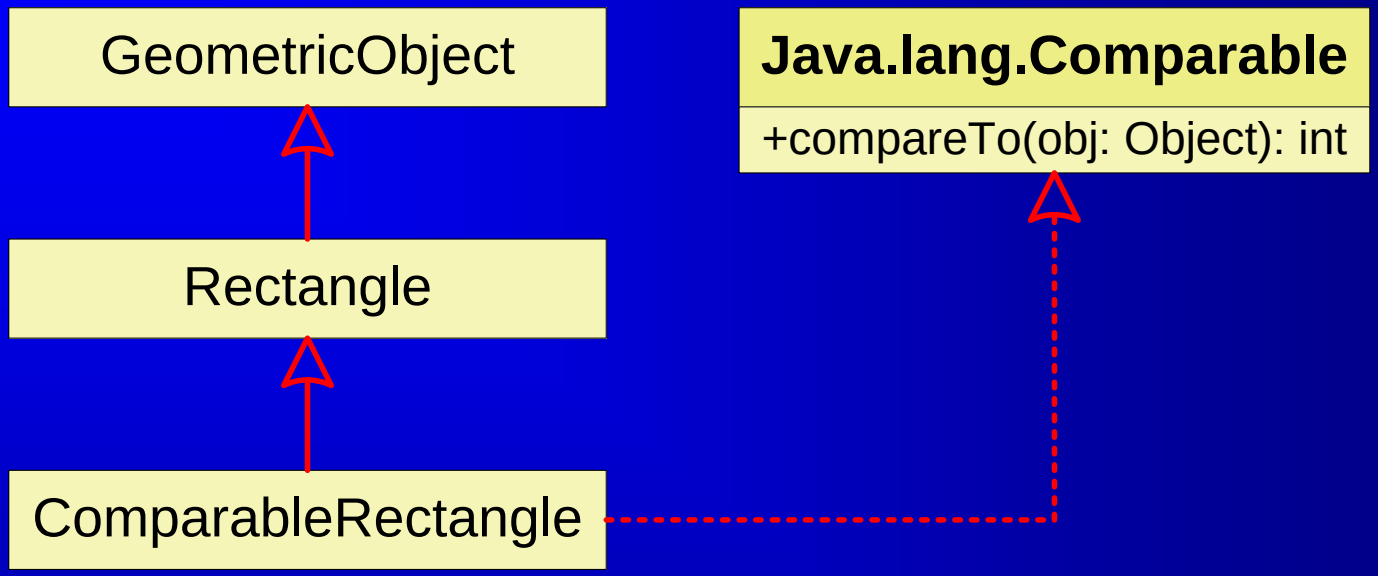
- Interface là một cấu trúc giống như class mà chỉ chứa các hằng và các phương thức trừu tượng.

```
modifier interface InterfaceName {
    /** khai báo hằng */
    /** method header */
}
```

VD: // Interface so sánh 2 đối tượng, được xác định trong java.lang

```
package java.lang;
public interface Comparable {
        public int compareTo(Object obj);
}
```

Thực hiện Interface



ComparableObject extends Rectange and implements Comparable

ComparableRectangle

Max

TestMaxObject

Interface vs. Abstract class

- Trong interface, dữ liệu phải là hằng, abstract class có thể có cả dữ liệu biến.
- Trong interface, phương thức chỉ có header mà ko có thực hiện, abstract class có thể có phương thức đầy đủ.
- Trong interface, tất cả dữ liệu là public final static và phương thức là public abstract

```
public interface T1{
 public static final int k=1;
 public abstract void p();
}
```

===

```
public interface T1{
      int k=1;
      void p();
}
```

Interface vs. Abstract class

- Có thể truy nhập các hằng trong interface sử dụng cú pháp InterfaceName.CONSTANT_NAME.
- Có thể sử dụng interface để thực hiện đa thừa kế với class và các subinterface.

```
public class NewClass extends BaseClass
    implements Interface1, …, InterfaceN {
    …
 }
public interface NewInterface extends Interface1, …, InterfaceN
      {
              // các hằng và phương thức trừu tượng
      }
```

LẬP TRÌNH JAVA

Chương 9: Bắt đầu với lập trình GUI

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 9

- Sơ đồ phân cấp lớp GUI
- Frames
  - Tạo frame, centering frames, adding components to frames
- Layout Managers
  - FlowLayout, GridLayout, BorderLayout
- Drawing on Panels
  - The paintComponent method
- Using Colors, Fonts, and Font Metrics
- Drawing Geometric Figures
  - Lines, Rectangles, Ovals, Arcs, and Polygons
- Event-Driven Programming
  - Event Source, Listener, Listener Interface

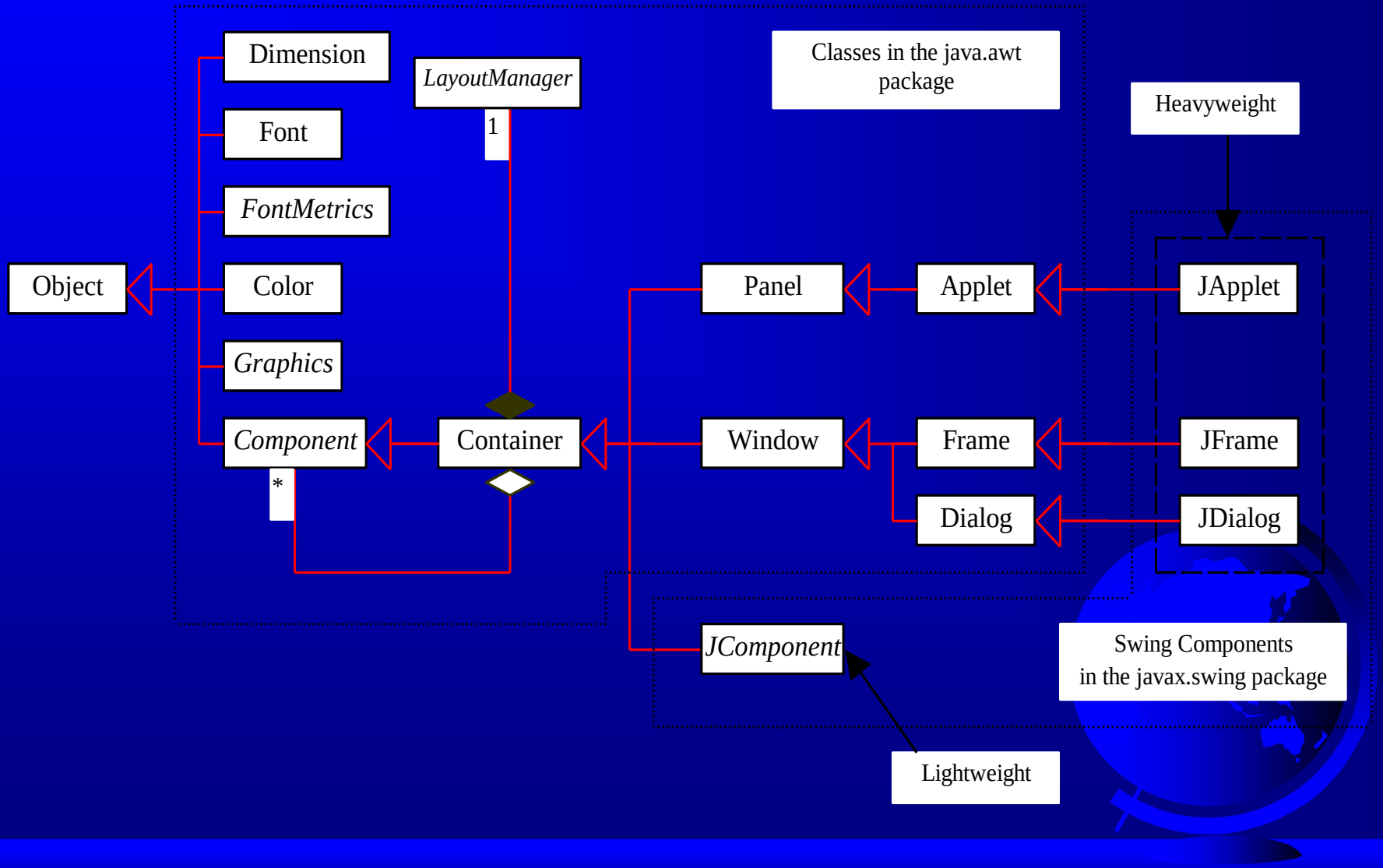
Các thành phần GUI

- Các đối tượng GUI: button, label, text field, check box, radio button, combo box, …
- Mỗi loại đối tượng được xác định trong 1 lớp: JButton, JLabel, JTextField, JCheckBox, JRadioButton, JComboBox, …
- Mỗi lớp thành phần GUI có một số constructor để tạo các đối tượng thành phần GUI.

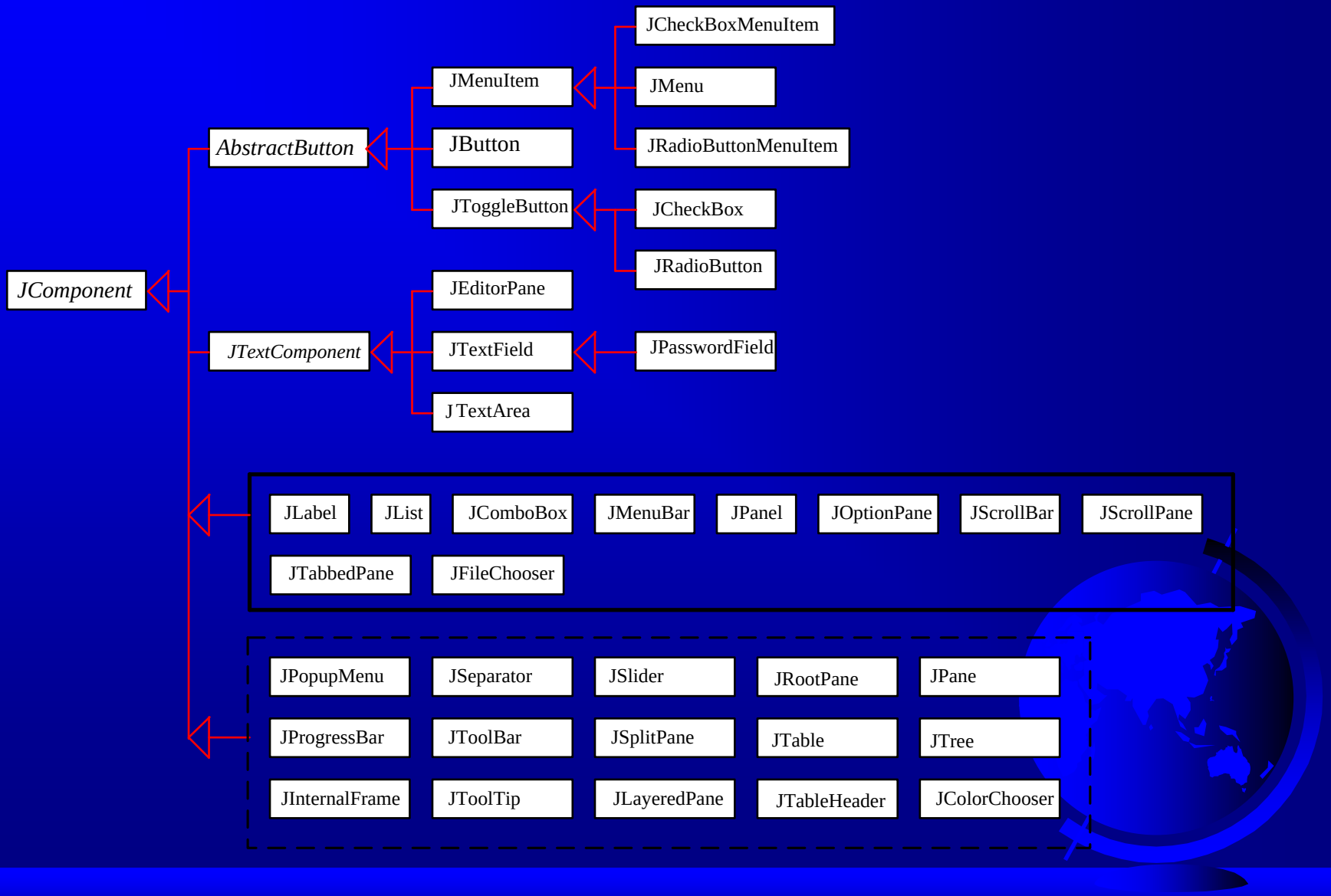
Swing vs. AWT

- AWT: Abstract Windows Toolkit:
  - Java 1
  - Được gắn với platform xác định
  - Thích hợp với việc phát triển các ứng dụng GUI đơn giản.
- Swing components:
  - Java 2
  - Không gắn với platform cố định
  - Mạnh, đa năng, linh hoạt

Sơ đồ phần cấp lớp GUI (Swing)



JComponent


JCheckBoxMenuItem
JMenuItem
JMenu
AbstractButton
JButton
JRadioButtonMenuItem
JToggleButton
JCheckBox
JRadioButton
JComponent
JEditorPane
JTextComponent
JTextField
JPasswordField
JTextArea
JLabel
JList
JComboBox
JMenuBar
JPanel
JOptionPane
JScrollBar
JScrollPane
JTabbedPane
JFileChooser
JPopupMenu
JSeparator
JSlider
JRootPane
JPane
JProgressBar
JToolBar
JSplitPane
JTable
JTree
JInternalFrame
JToolTip
JLayeredPane
JTableHeader
JColorChooser

Các lớp GUI: nhóm container

- Được dùng để chứa các thành phần khác.
- Các lớp container (Swing):
  - Container
  - JFrame
  - JDialog
  - JApplet
  - JPanel

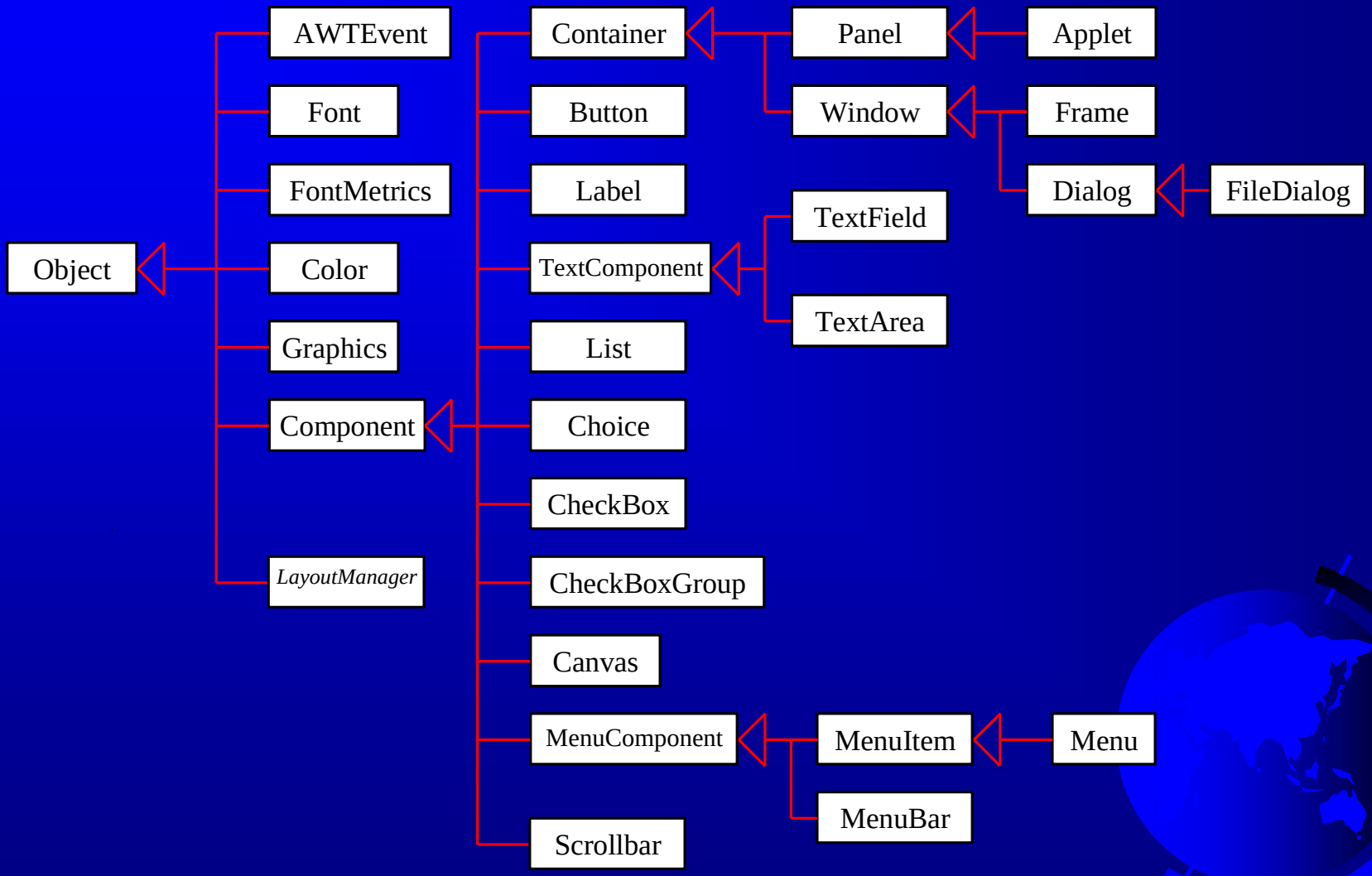
Các lớp GUI: nhóm component

- Gồm các subclass của lớp JComponent.
- Các lớp GUI component (Swing):
  - JButton
  - JLabel
  - JTextField
  - JTextArea
  - JComboBox
  - JList
  - JRadioButton
  - JMenu
  - …

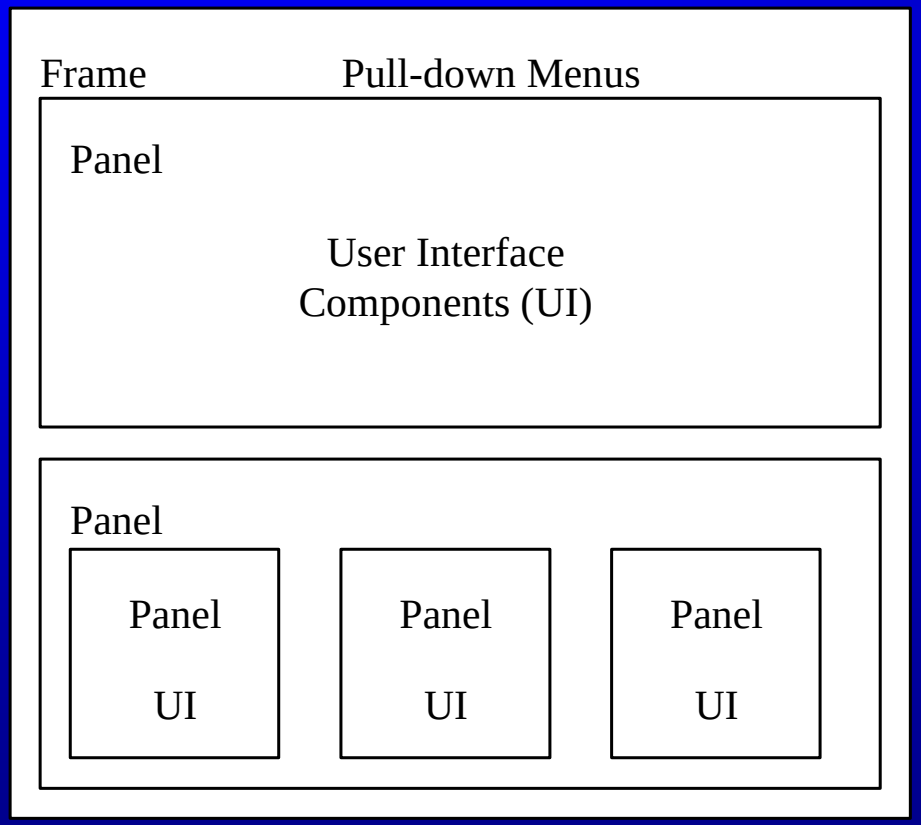
Các lớp GUI: nhóm helper

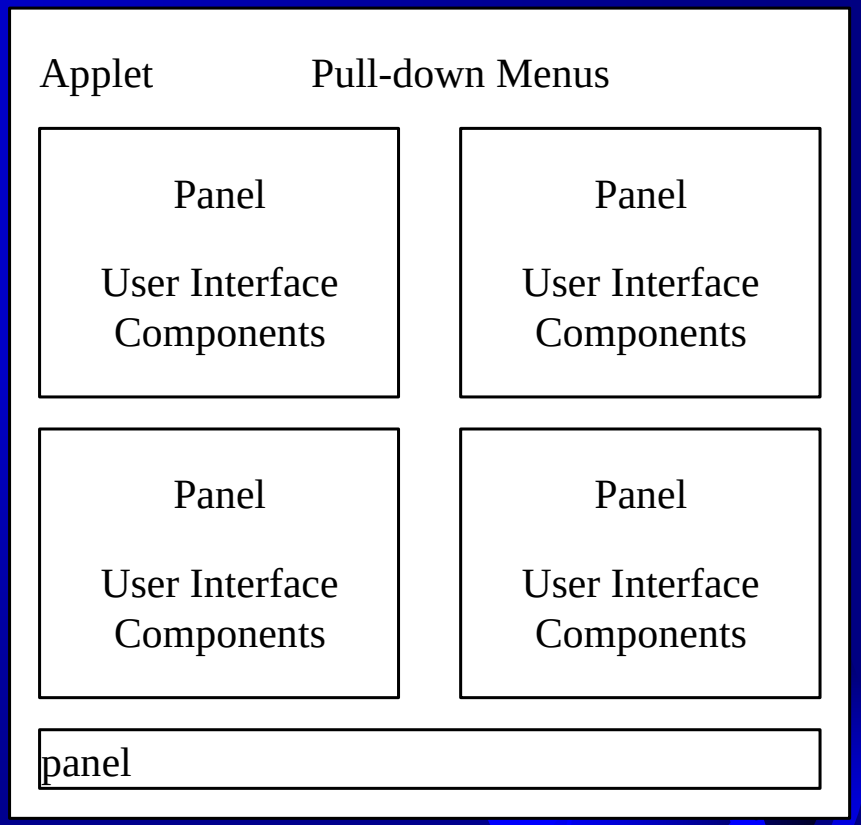
- Được các component và container dùng để vẽ và đặt các đối tượng.
- Các lớp helper (Swing):
  - Graphics
  - Color
  - Font
  - FontMetrics
  - Dimension
  - LayoutManager

AWT (Optional)



Các thành phần giao diện người sử dụng





Frames

- Frame là một cửa sổ không chứa trong cửa sổ khác.
- Frame là nền tảng để chứa các thành phần GUI khác trong các ứng dụng Java GUI.
- Trong các chương trình Swing GUI, sử dụng lớp JFrame để tạo các cửa sổ.

Tạo Frame

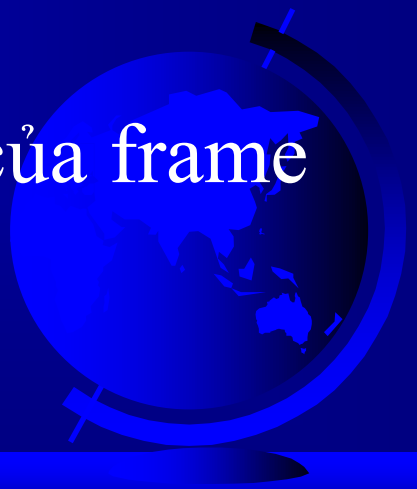
```
import javax.swing.*;
public class MyFrame {
  public static void main(String[] args) {
    JFrame frame = new JFrame("Test Frame");
    frame.setSize(400, 300);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
  }
}
```

Chú ý: Chạy chương trình cần JDK 1.3 hoặc cao hơn

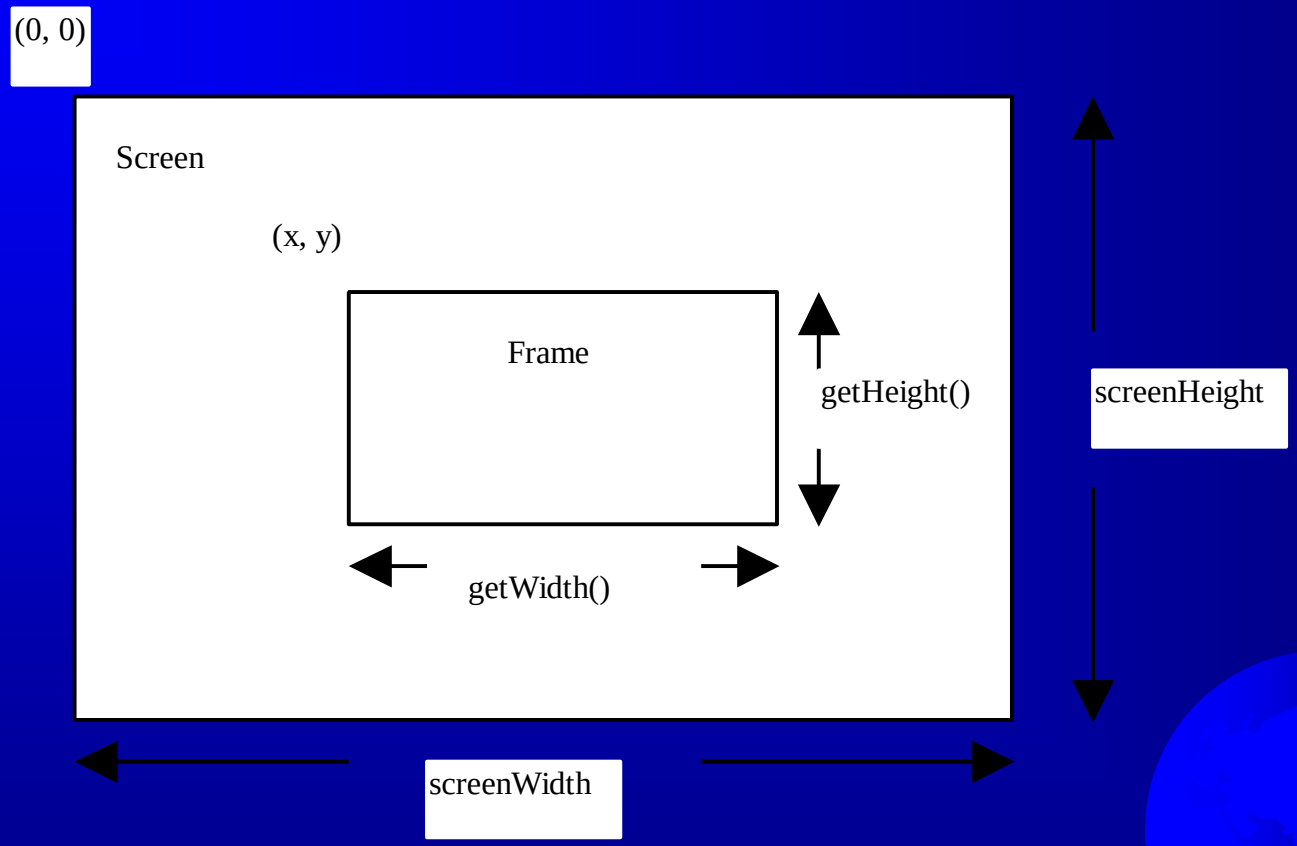
MyFrame

Căn giữa Frame

- Mặc định, frame được hiển thị ở góc trên bên trái của màn hình.
- Để hiển thị frame ở một vị trí xác định, sử dụng phương thức `setLocation(x, y)` trong lớp `JFrame`.
- Phương thức này đặt góc trái trên của frame tại vị trí có tọa độ (x, y).



Căn giữa Frame (tiếp)



CenterFrame

Đưa các thành phần vào trong Frame

```
// Dua nut bam vao trong frame
frame.getContentPane().add(
   new JButton("OK"));
```

MyFrameWithComponents

LƯU Ý

Content pane là một lớp con của Container. Câu lệnh ở slide trước tương đương với 2 câu lệnh sau:

```
Container container = frame.getContentPane();
container.add(new JButton("OK"));
```

Content pane được sinh ra khi một đối tượng JFrame được tạo. Đối tượng JFrame sử dụng content pane để chứa các thành phần trong frame.

Layout Managers

- Các layout manager của Java cung cấp cơ chế để tự động ánh xạ các thành phần GUI của bạn trên tất cả các hệ thống cửa sổ.
- Các thành phần GUI được đặt trong các container. Mỗi container có một layout manager để sắp xếp các thành phần đó.

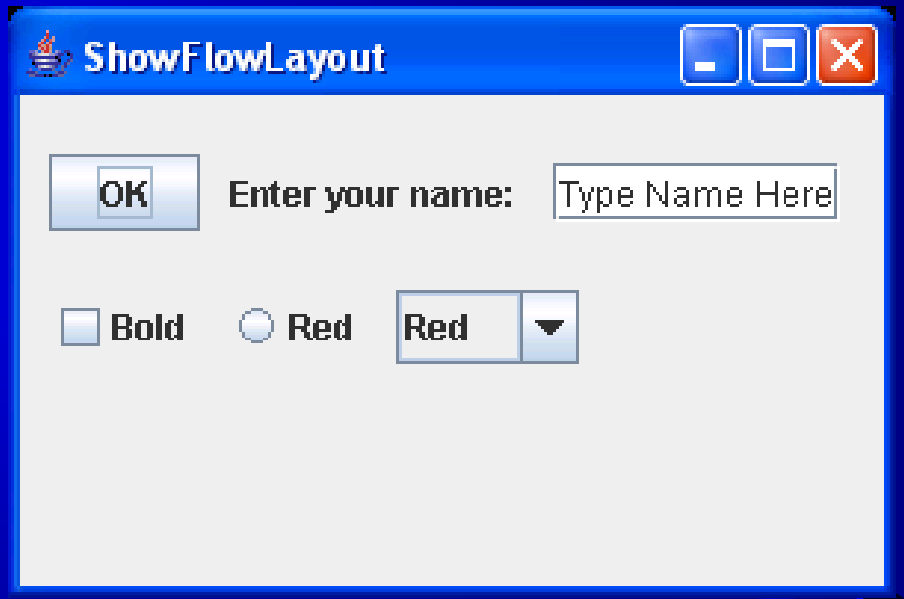
Thiết lập Layout Manager

```
LayoutManager layMan = new XLayout();
container.setLayout(layMan);
```

- XLayout:
  - FlowLayout
  - GridLayout
  - BorderLayout
  - CardLayout
  - GridBagLayout

Ví dụ 9.1: `FlowLayout` Manager

Các thành phần được sắp xếp trong container từ trái sang phải, từ trên xuống dưới theo thứ tự chúng được đưa vào.



ShowFlowLayout

`FlowLayout` Constructors

- `public FlowLayout(int align, int hGap, int vGap)`

  Xây dựng một `FlowLayout` mới có cách sắp hàng (alignment), khoảng trống ngang (horizontal gap), khoảng trống dọc (vertical gap) xác định. Các khoảng trống giữa các thành phần được tính bằng pixel.

- `public FlowLayout(int alignment)`

  Xây dựng một `FlowLayout` mới có alignment xác định, khoảng trống ngang và dọc đều có mặc định bằng 5 pixel.

- `public FlowLayout()`

  Xây dựng một `FlowLayout` mới có cách sắp hàng mặc định căn giữa và khoảng trống ngang và dọc mặc định bằng 5 pixel.

Ví dụ 9.2: `GridLayout` Manager

`GridLayout` manager sắp xếp các thành phần trong một lưới (ma trận) với số hàng và số cột được xác định bởi constructor. Các thành phần được đặt trong lưới từ trên xuống dưới, từ trái sang phải.

ShowGridLayout

`GridLayout` Constructors

- `public GridLayout(int rows,int columns)`

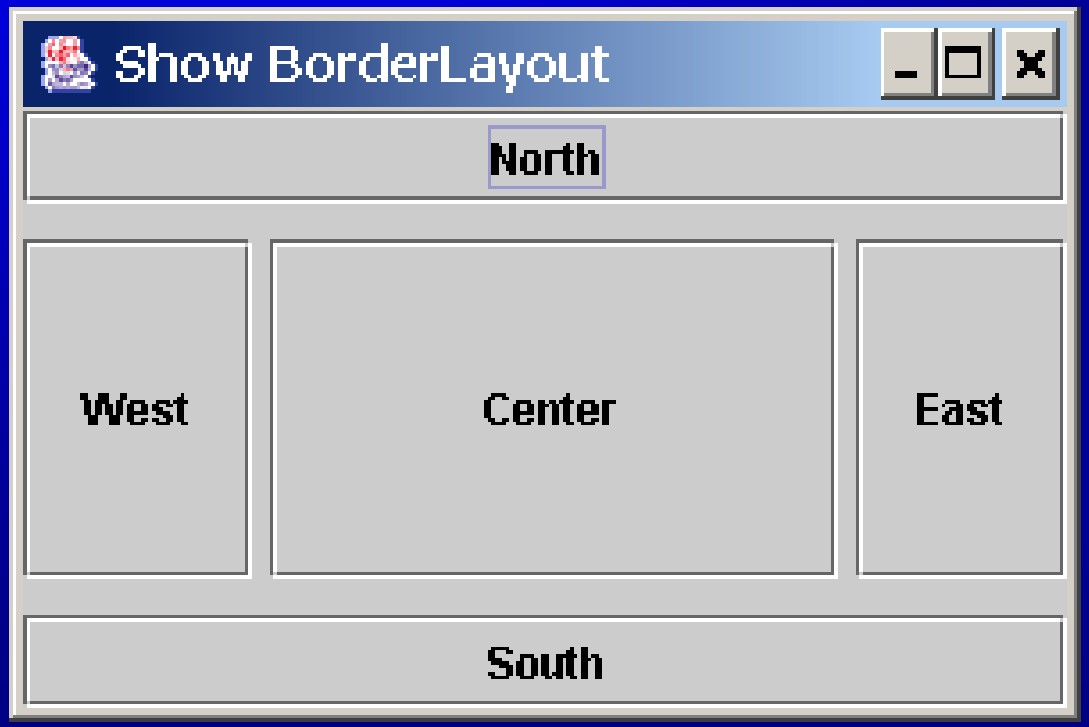
  Xây dựng một `GridLayout` mới có số hàng và số cột xác định.

- `public GridLayout(int rows, int columns, int hGap, int vGap)`

  Xây dựng một `GridLayout` mới có số hàng và số cột xác định, và khoảng trống ngang và dọc giữa các thành phần được xác định.

Ví dụ 9.3: `BorderLayout` Manager

- `BorderLayout` manager chia container thành 5 khu vực: East, South, West, North, và Center.



Ví dụ 9.3 (tiếp)

- Các thành phần được đưa vào BorderLayout bằng phương thức add:

```
add(Component, constraint)
```

- constraint:
  - BorderLayout.EAST,
  - BorderLayout.SOUTH,
  - BorderLayout.WEST,
  - BorderLayout.NORTH,
  - BorderLayout.CENTER.

ShowBorderLayout

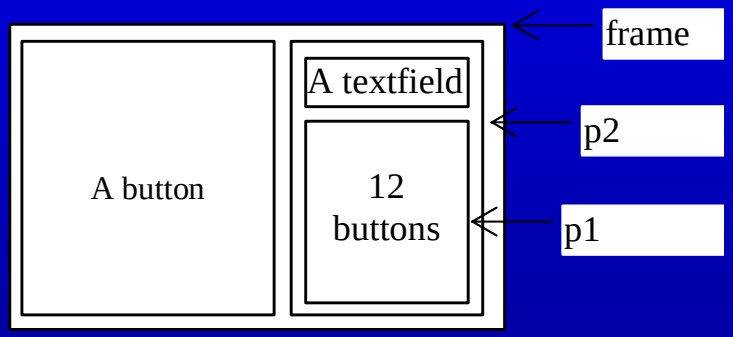
Sử dụng Panel làm Container

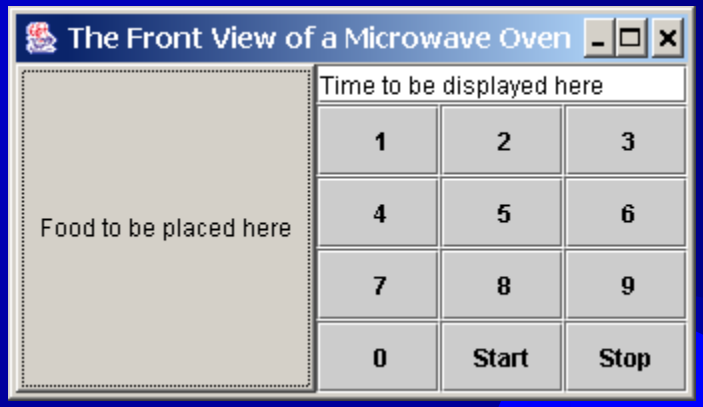
- Các panel đóng vai trò như các container nhỏ để nhóm các thành phần GUI.
- Bạn nên đặt các thành phần GUI trong các panel và đặt các panel trong một frame, hoặc cũng có thể đặt panel trong panel.

```
JPanel p = new JPanel();
p.add(new JButton("OK");
frame.getContentPanel().add(p);
```

Ví dụ 9.4: Panel

Chương trình tạo một giao diện cho lò vi sóng, sử dụng các panel để tổ chức các thành phần.





TestPanels

Vẽ trên Panel

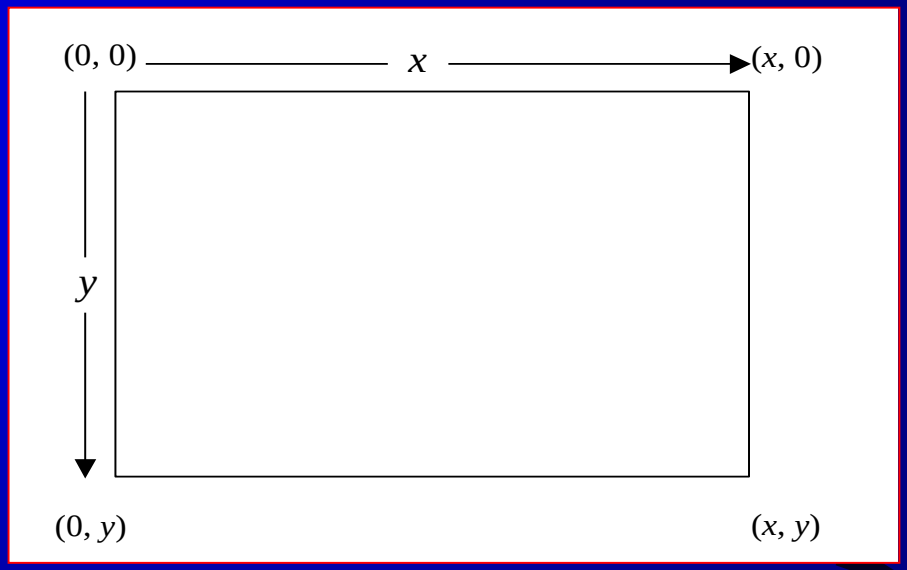
- `JPanel` còn có thể được sử dụng để vẽ đồ họa, văn bản và cho phép tương tác với người sử dụng.

- Để vẽ trên panel:
  - Tạo một lớp subclass của `JPanel`
  - Chồng phương thức `paintComponent`.
  - Sau đó có thể hiển thị các chuỗi ký tự, vẽ các khối hình học và hiển thị ảnh trên panel.

Vẽ trên Panel (tiếp)

```
public class DrawMessage extends JPanel {
  /** Main method */
  public static void main(String[] args) {
    JFrame frame = new JFrame("DrawMessage");
    frame.getContentPane().add(new
DrawMessage());

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);
    frame.setSize(300, 200);
    frame.setVisible(true);
  }

  /** Paint the message */
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawString("Welcome to Java!", 40, 40);
```

Vẽ trên Panel (tiếp)



LƯU Ý

- Lớp Graphics là một lớp trừu tượng để hiển thị hình vẽ và ảnh trên màn hình trên các platform khác nhau.
- Lớp Graphics gói gọn các chi tiết platform và cho phép bạn vẽ các thứ theo cách giống nhau không liên quan đến các platform cụ thể.
- Lời gọi `super.paintComponent(g)` là cần thiết để đảm bảo rằng vùng hiển thị được xóa sạch trước khi hiển thị một bản vẽ.

LƯU Ý

Để vẽ các hình, thông thường bạn tạo một lớp con của JPanel và chồng phương thức paintComponent để "nói" cho hệ thống phải vẽ như thế nào. Thực tế bạn có thể vẽ các thứ trên bất kỳ thành phần GUI nào.

Lớp `Color`

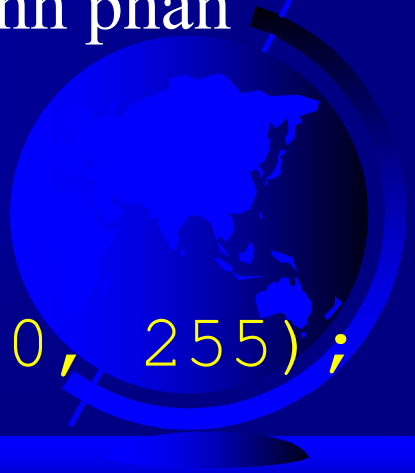
☞ Bạn có thể thiết lập màu cho các thành phần GUI bằng cách sử dung lớp java.awt.Color. Các màu được tạo từ 3 màu cơ bản là red, green, blue; mỗi màu đó được biểu diễn bởi một giá trị byte (0-255) miêu tả cường độ. Đây được gọi là hệ màu RGB (*RGB model*).

```
Color c = new Color(r, g, b);
```

`r`, `g`, `b` xác định một màu được tạo bởi các thành phần tương ứng red, green, blue.

Ví dụ:

```
Color c = new Color(228, 100, 255);
```



Thiết lập màu

- Bạn có thể sử dụng các phương thức sau để thiết lập màu background và foreground của các thành phần:

```
setBackground(Color c)

setForeground(Color c)
```

Ví dụ:

```
JButton jbtOK = new JButton();
jbtOK.setBackground(Color.yellow);
jbtOK.setForeground(new Color(255,0,0));
```

Lớp Font

```
Font myFont = Font(name, style, size);
```

Ví dụ:

```
Font font1 = new Font("SansSerif", Font.BOLD, 16);
Font font2 = new Font("Serif",
                      Font.BOLD+Font.ITALIC, 12);
```

Tìm tất cả tên Font khả dụng

```
import java.awt.GraphicsEnvironment;

public class testAllFonts {
  public static void main(String[] args) {
  GraphicsEnvironment e =

  GraphicsEnvironment.getLocalGraphicsEnviron
  ment();
  String[] fontnames =
  e.getAvailableFontFamilyNames();
     for (int i
                              names.length;
  i++)
```

testAllFonts

Thiết lập Font

```
public void paint(Graphics g) {
  Font myFont = new Font("Times", Font.BOLD, 18);
  g.setFont(myFont);
  g.drawString("Welcome to Java", 20, 40);

  //set a new font
  g.setFont(new Font("Courier",Font.BOLD+Font.ITALIC,16));
  g.drawString("Welcome to Java", 20, 70);
}
```

testSetFonts

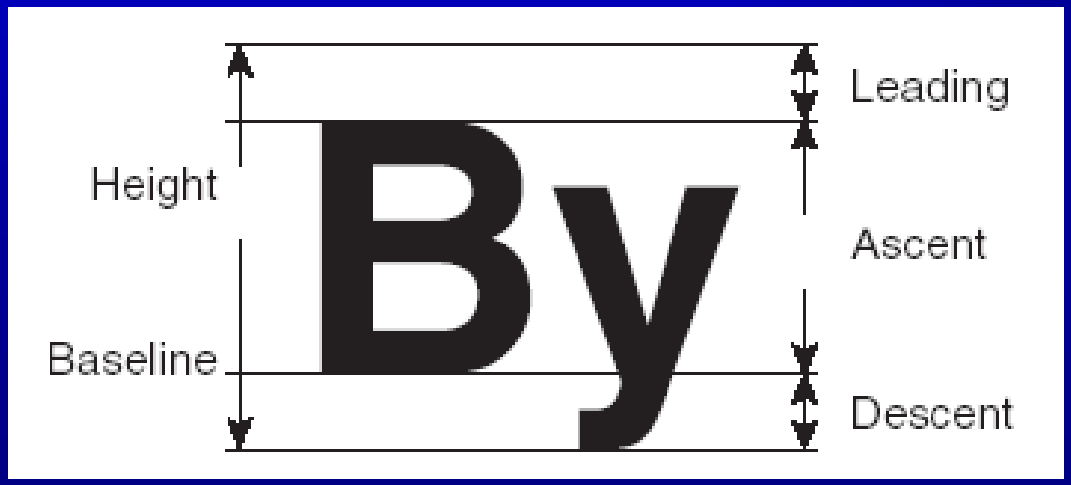
Lớp FontMetrics

- Bạn có thể hiển thị một chuỗi ký tự tại vị trí bất kỳ trong panel bằng cách sử dụng lớp FontMetrics.
- Để nhận đối tượng FontMetrics cho một font xác định, sử dụng phương thức getFontMetrics:

```
public void paint(Graphics g) {
   g.getFontMetrics(Font f);  // hoặc
   g.getFontMetrics();
}
```

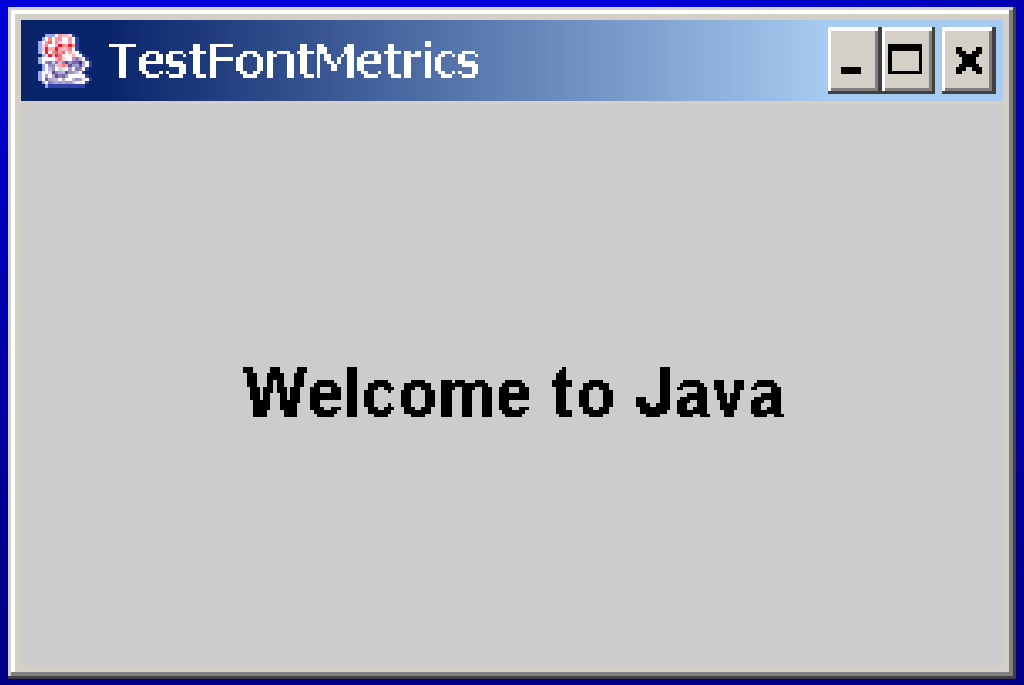
Các phương thức lấy thuộc tính chuỗi của lớp `FontMetrics`

- `public int getAscent()`
- `public int getDescent()`
- `public int getLeading()`
- `public int getHeight()`
- `public int stringWidth(String str)`

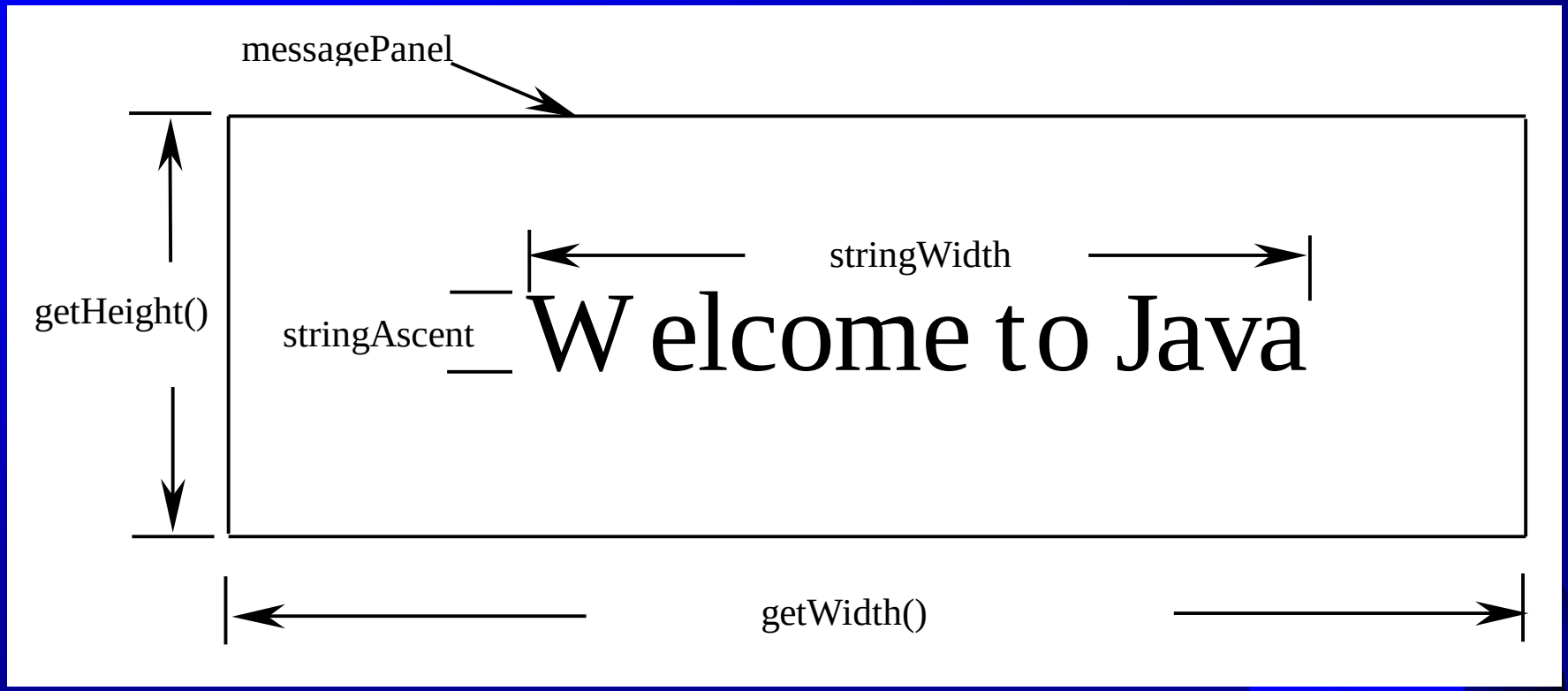


Ví dụ 9.5: Sử dụng `FontMetrics`

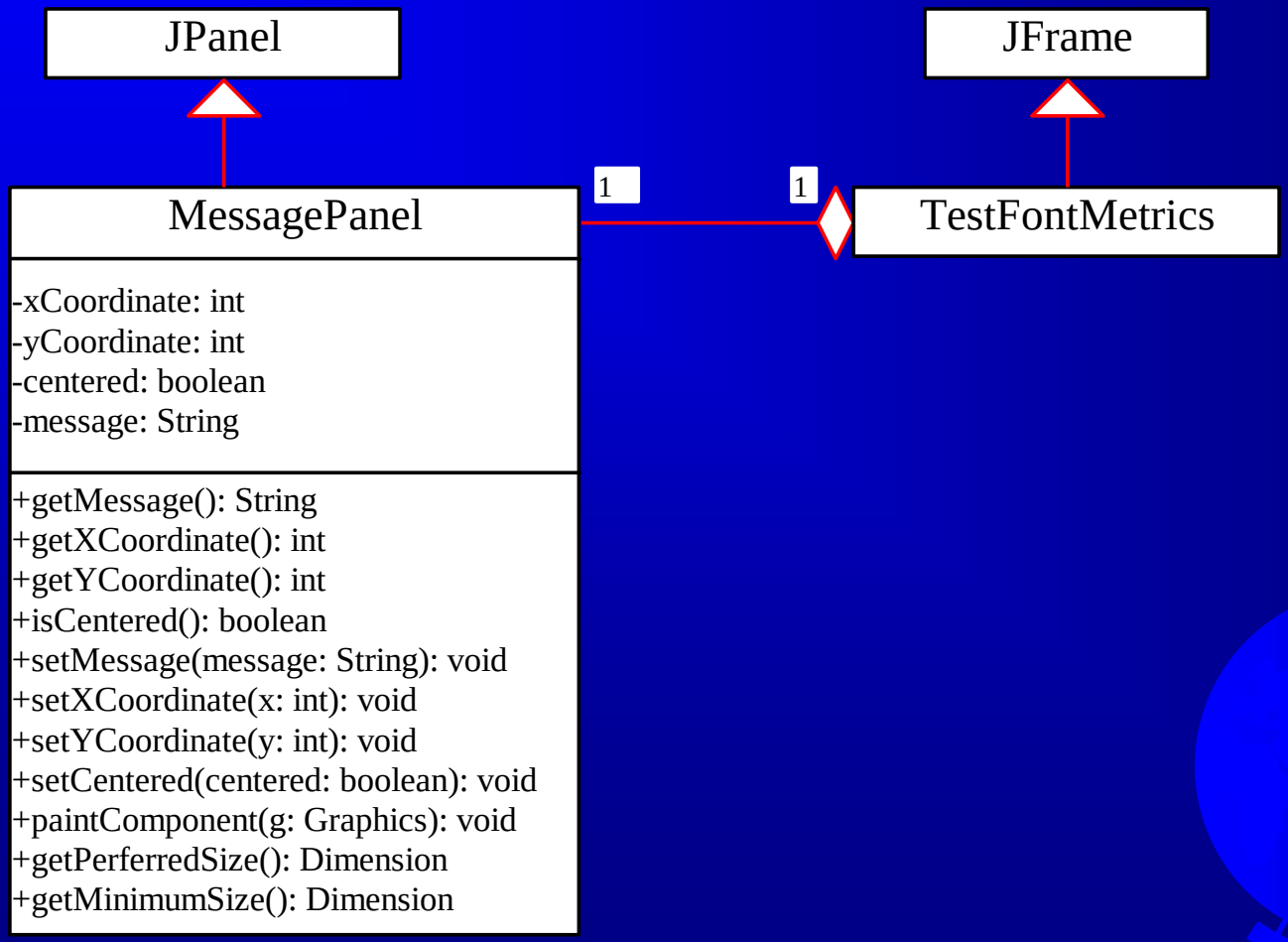
- Mục tiêu: Hiển thị "Welcome to Java" căn giữa trong frame.



CenterMessage


messagePanel
getHeight()
stringAscent
stringWidth
Welcome to Java
getWidth()

Ví dụ 9.5 (tiếp)

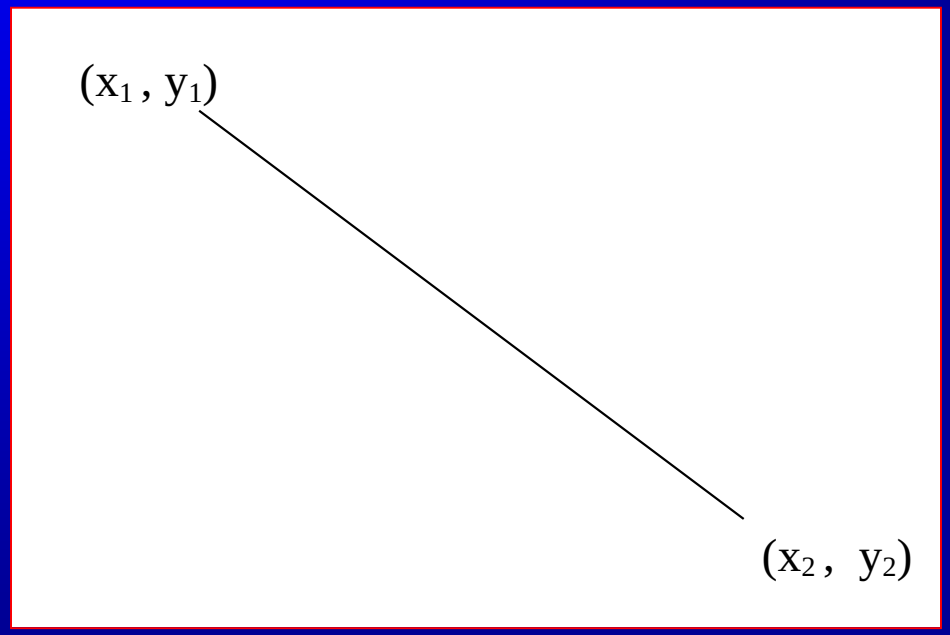

JPanel
JFrame
MessagePanel
1
1
TestFontMetrics
-xCoordinate: int
-yCoordinate: int
-centered: boolean
-message: String
+getMessage(): String
+getXCoordinate(): int
+getYCoordinate(): int
+isCentered(): boolean
+setMessage(message: String): void
+setXCoordinate(x: int): void
+setYCoordinate(y: int): void
+setCentered(centered: boolean): void
+paintComponent(g: Graphics): void
+getPerferredSize(): Dimension
+getMinimumSize(): Dimension

Vẽ các hình hình học trên Panel

- Vẽ đường thẳng
- Vẽ hình chữ nhật
- Vẽ hình bầu dục
- Vẽ cung tròn
- Vẽ đa giác

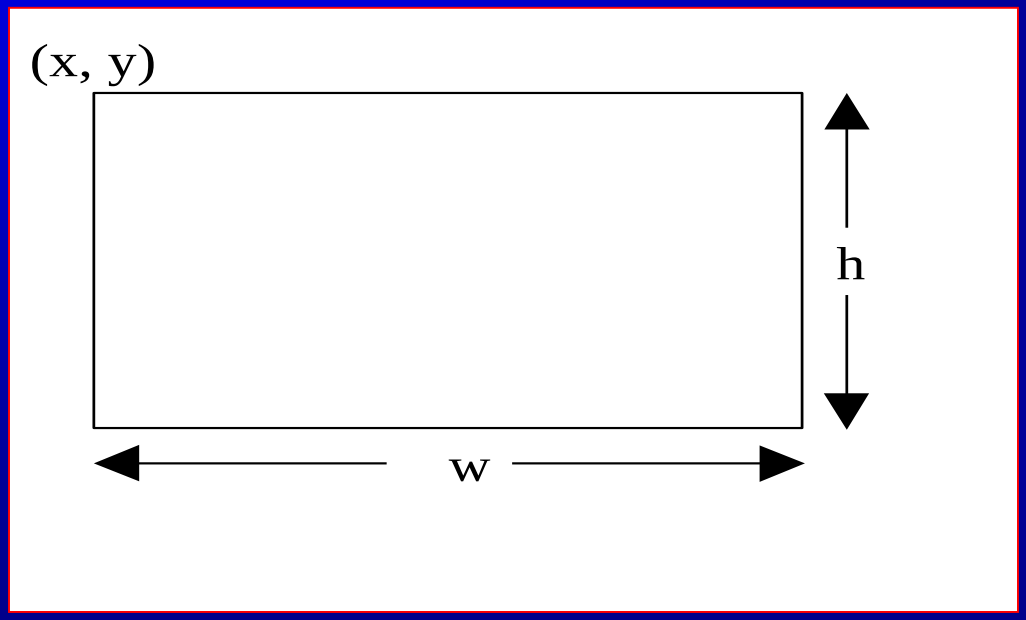
Vẽ đường thẳng

```
drawLine(x1, y1, x2, y2);
```



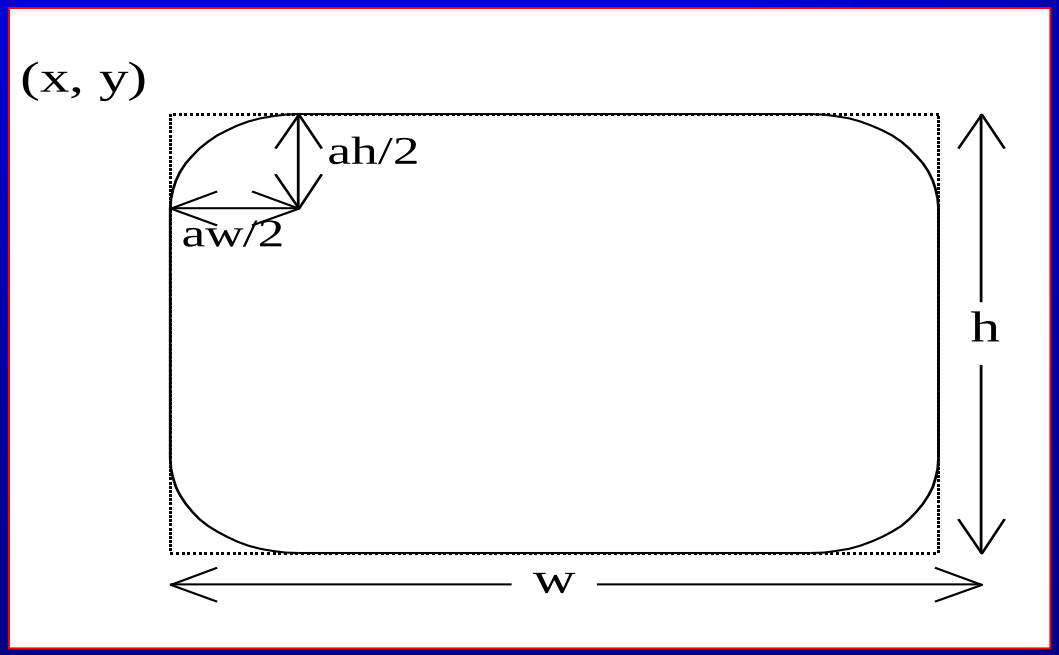
Vẽ hình chữ nhật

- `drawRect(x, y, w, h);`
- `fillRect(x, y, w, h);`



Vẽ hình chữ nhật góc tròn

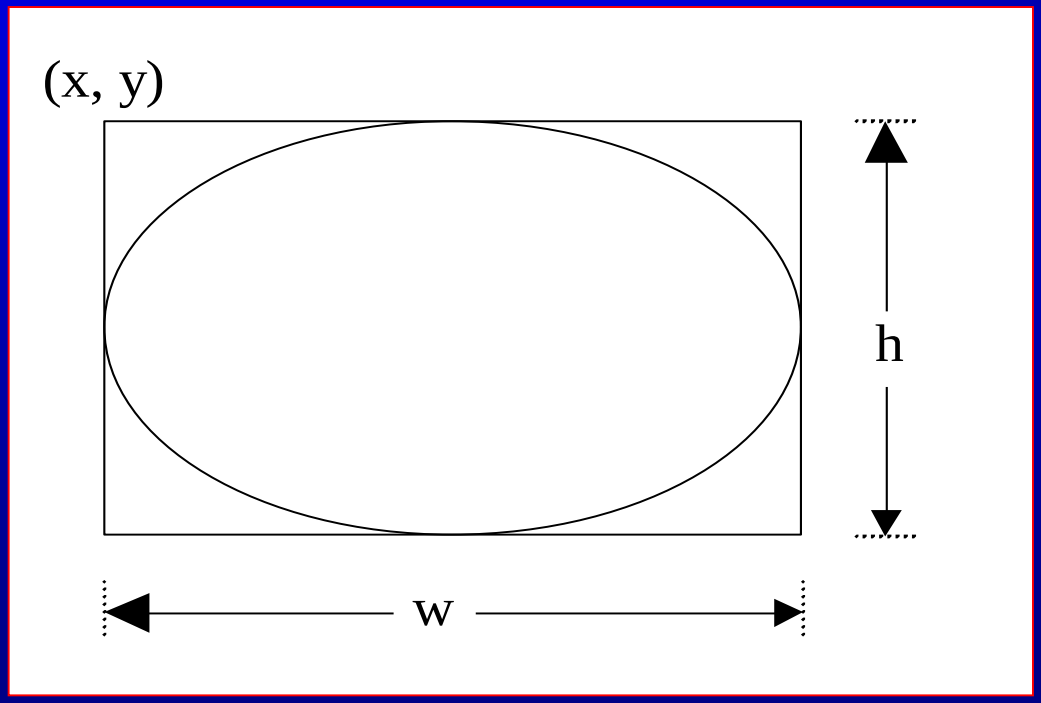
- `drawRoundRect(x, y, w, h, aw, ah);`
- `fillRoundRect(x, y, w, h, aw, ah);`



DrawRectangles

Vẽ hình bầu dục

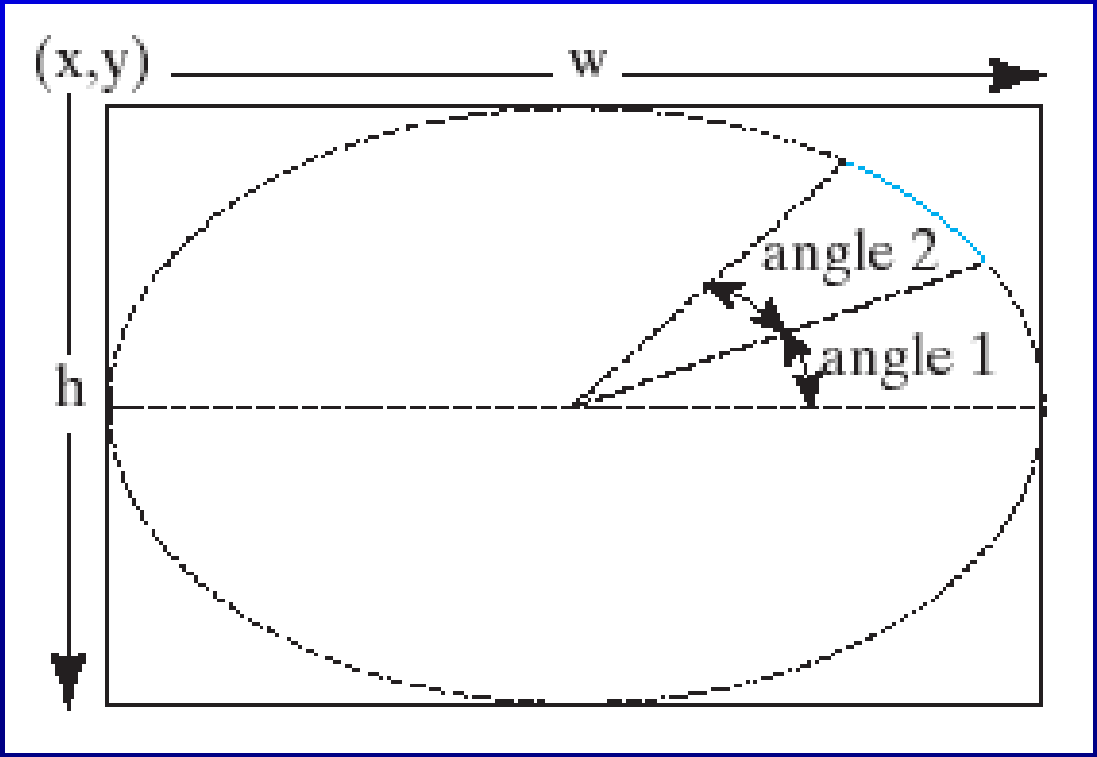
- `drawOval(x, y, w, h);`
- `fillOval(x, y, w, h);`



DrawOvals

Vẽ cung tròn

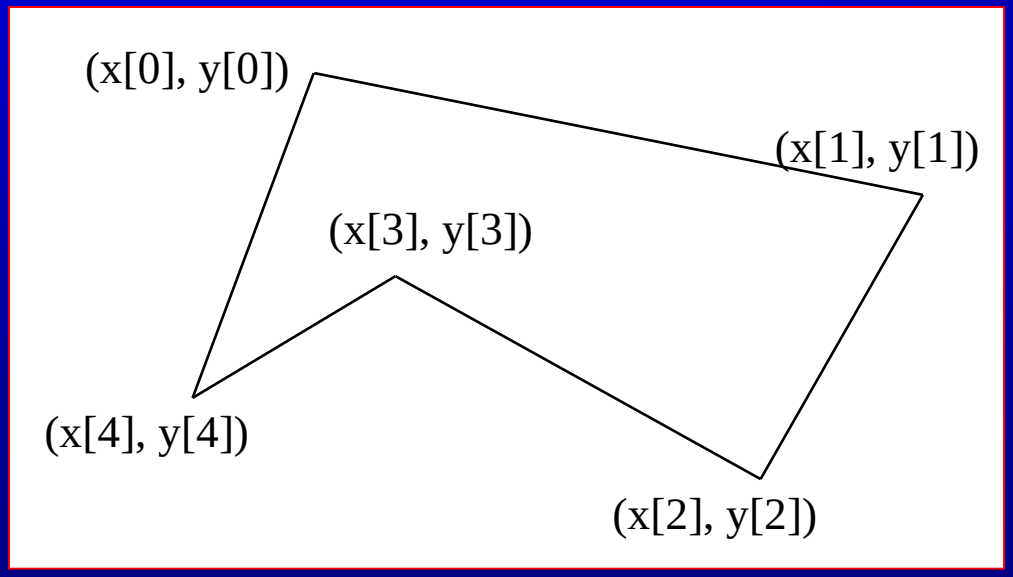
- `drawArc(x, y, w, h, angle1, angle2);`
- `fillArc(x, y, w, h, angle1, angle2);`



DrawArgs

Vẽ đa giác

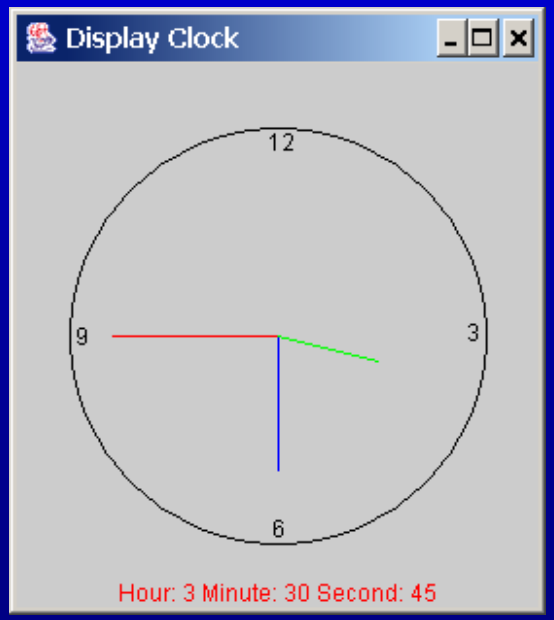
```
int[] x = {40, 70, 60, 45, 20};
int[] y = {20, 40, 80, 45, 60};
g.drawPolygon(x, y, x.length);
g.fillPolygon(x, y, x.length);
```



DrawPolygon

Ví dụ 9.6: Vẽ chiếc đồng hồ

- Mục tiêu: Sử dụng các phương thức vẽ và lượng giác để vẽ một chiếc đồng hồ hiển thị giờ, phút, giây hiện tại trong một frame.



DrawClock

DisplayClock

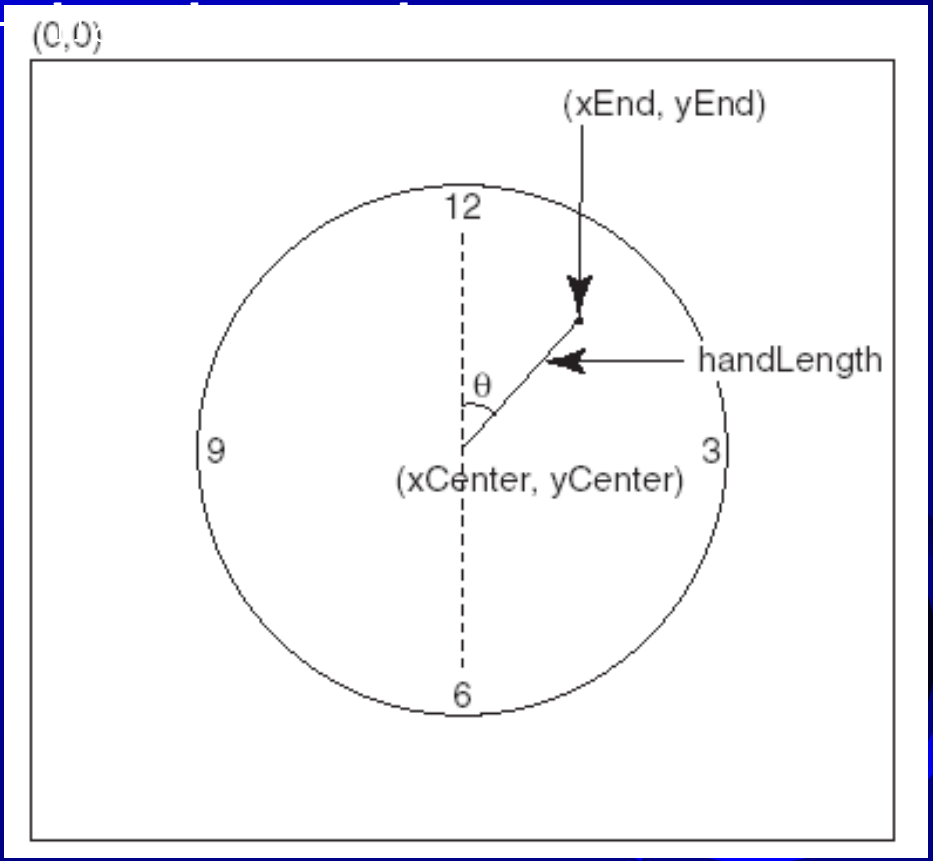
Run

Ví dụ 9.6 (tiếp)

```
xEnd = xCenter + handLength x
sin(θ)

yEnd = yCenter -
cos(θ)
```

Vì 1 phút có 60 giây, góc của kim giây là:

```
second x (2π/60)
```

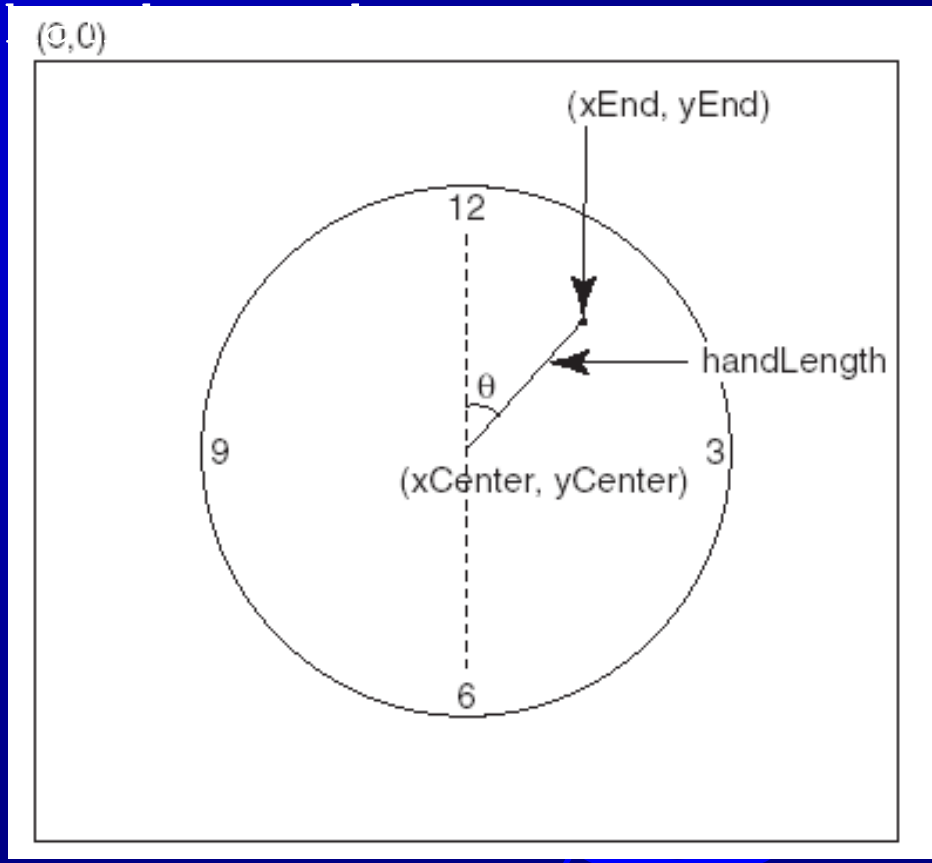


Ví dụ 9.6 (tiếp)

```
xEnd = xCenter + handLength ×
sin(θ)

yEnd = yCenter -
cos(θ)
```

Vị trí của kim phút được xác định bởi phút và giây theo công thức minute + second/60. Ví dụ, nếu thời gian là 3 phút 30 giây. Tổng số phút là 3.5. Vì 1 giờ có 60 phút, góc của kim phút là:

```
(minute + second/60) ×
(2π/60)
```

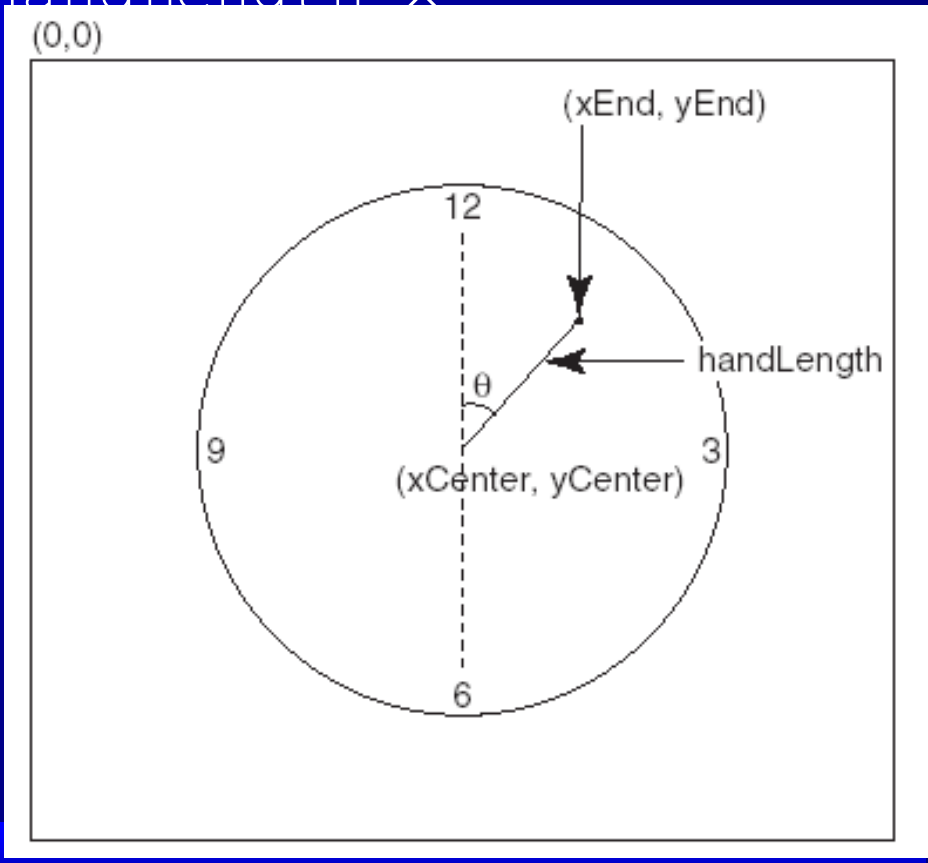


Ví dụ 9.6 (tiếp)

```
xEnd = xCenter + handLength x
sin(θ)

yEnd = yCenter - handLength x
cos(θ)
```

Vì hình tròn được chia thành 12 giờ, góc của kim giờ là:

```
(hour + minute/60 +
second/(60 x 60))) x
(2π/12)
```



Lập trình hướng sự kiện

- *Lập trình hướng thủ tục (Procedural programming)* chương trình được thực hiện theo thứ tự thủ tục.

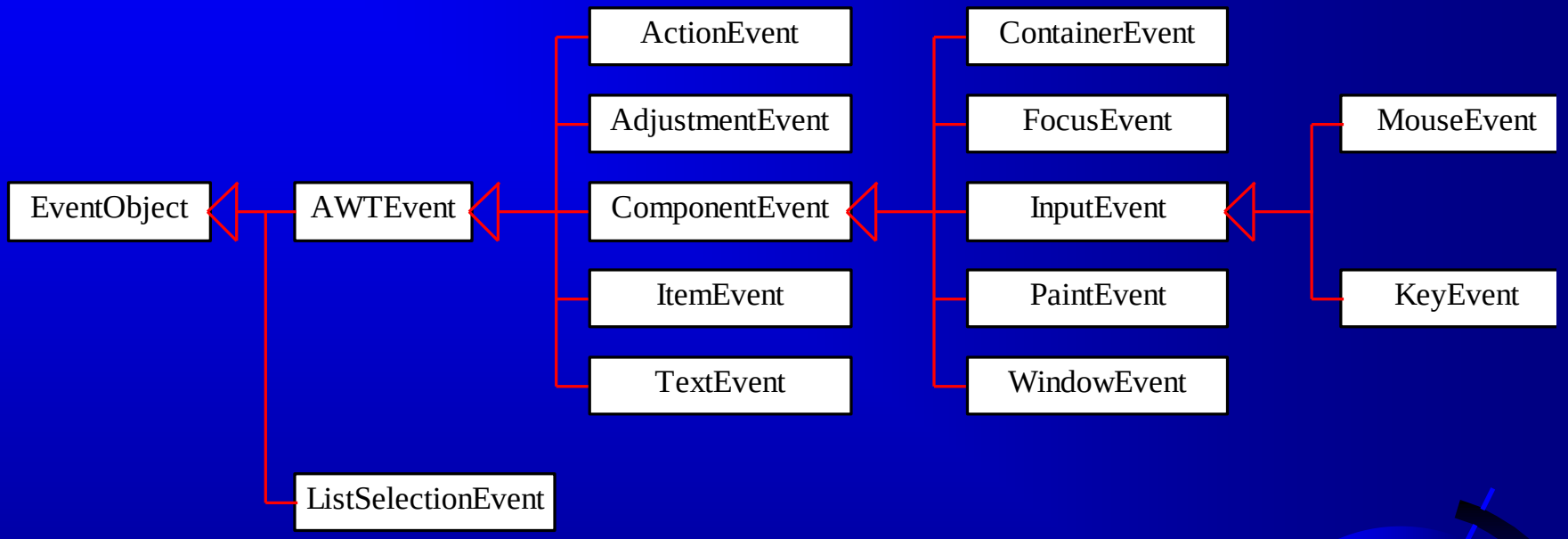
- Trong *lập trình hướng sự kiện (event-driven programming)*, mã lệnh được thực hiện vào lúc kích hoạt sự kiện.

Sự kiện

- Một *sự kiện* (*event*) có thể được định nghĩa là một loại tín hiệu báo cho chương trình có điều gì đó đã xảy ra..

- Sự kiện được sinh ra bởi các hành động của người sử dụng (ví dụ: di chuột, kích phím chuột, ấn phím) hoặc bởi HĐH (vd: timer).

Các lớp sự kiện

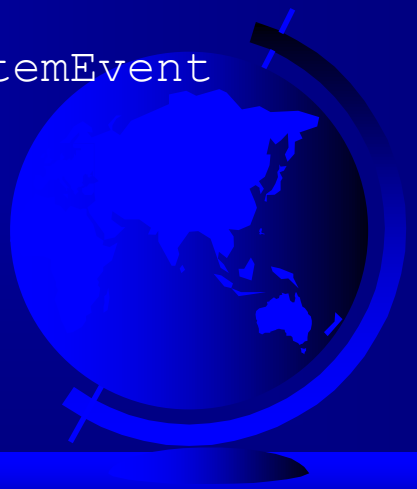


Các lớp sự kiện trên nằm trong gói `java.awt.event`

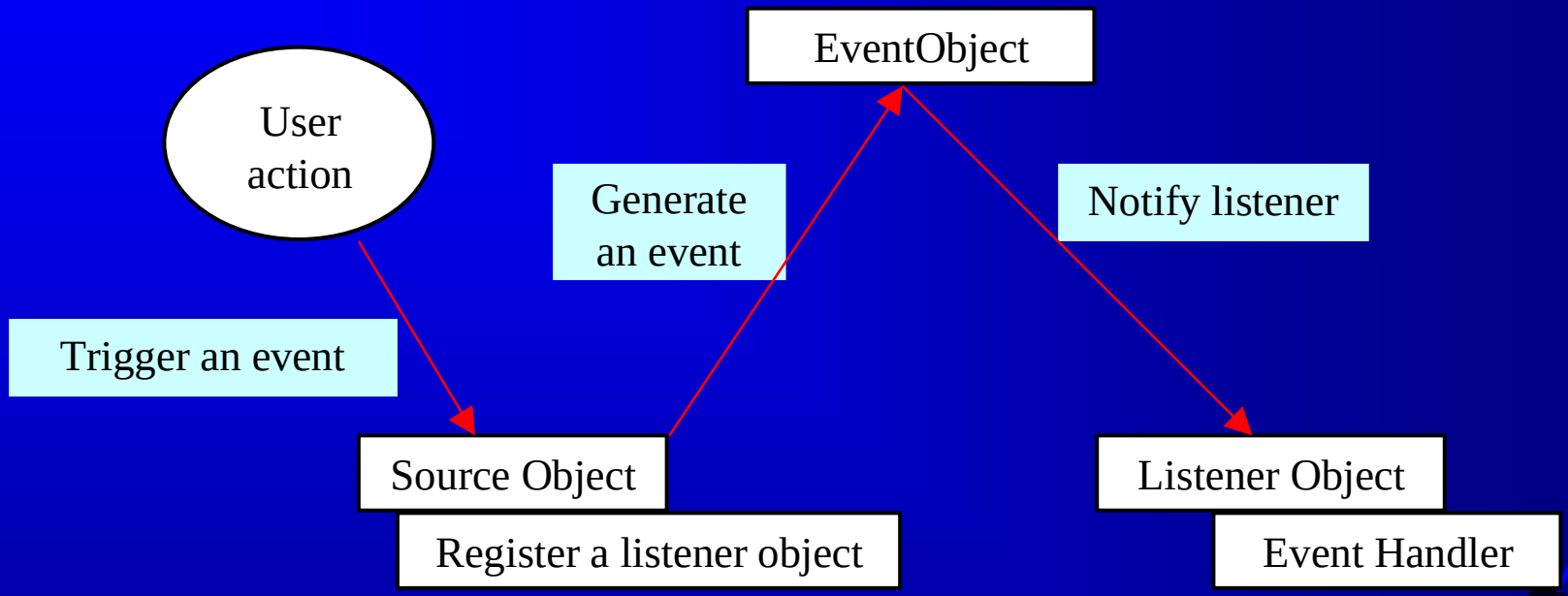
riêng ListSelectionEvent ở trong gói `javax.swing.event`

Selected User Actions

User Action	Source Object	Event Type Generated
Clicked on a button	`JButton`	`ActionEvent`
Changed text	`JTextComponent`	`TextEvent`
Double-clicked on a list item	`JList`	`ActionEvent`
Selected or deselected an item with a single click	`JList`	`ItemEvent`
Selected or deselected an item	`JComboBox`	`ItemEvent`



Mô hình ủy quyền



Selected Event Handlers

Event Class	Listener Interface	Listener Methods (Handlers)
`ActionEvent`	`ActionListener`	`actionPerformed(ActionEvent)`
`ItemEvent`	`ItemListener`	`itemStateChanged(ItemEvent)`
`WindowEvent`	`WindowListener`	`windowClosing(WindowEvent)`
		`windowOpened(WindowEvent)`
		`windowIconified(WindowEvent)`
		`windowDeiconified(WindowEvent)`
		`windowClosed(WindowEvent)`
		`windowActivated(WindowEvent)`
		`windowDeactivated(WindowEvent)`
`ContainerEvent`	`ContainerListener`	`componentAdded(ContainerEvent)`
		`componentRemoved(ContainerEvent)`

Ví dụ 9.7:
Xử lý sự kiện hành động đơn giản

- Mục tiêu: Hiển thị 2 nút bấm OK và Cancel trong cửa sổ. Khi kích chuột vào một nút, một message được hiển thị để chỉ ra nút nào đã được kích.

TestActionEvent

Ví dụ 9.8: Xử lý sự kiện cửa sổ

- Mục tiêu: Minh họa việc xử lý sự kiện cửa sổ. Bất kỳ lớp con nào của lớp Window có thể tạo ra các sự kiện cửa sổ sau: đã mở, đang đóng, đã đóng, đã được kích hoạt, đã mất kích hoạt, đã được thu nhỏ thành biểu tượng, phóng to trở lại cửa sổ. Chương trình này tạo một frame, lắng nghe các sự kiện cửa sổ, và hiển thị thông báo chỉ ra sự kiện đang xuất hiện.

TestWindowEvent

Ví dụ 9.9:
Nhiều Listener cho một nguồn đơn

- Mục tiêu: Sửa đổi ví dụ 9.7 để thêm vào mỗi nút bấm một listener mới. Hai nút OK và Cancel sử dụng lớp frame như listner. Khi một nút được bấm, cả 2 listener đáp ứng lại sự kiện hành động.

TestMultipleListener

LẬP TRÌNH JAVA

Chương 10: Tạo giao diện người sử dụng

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 10

- JButton
- JCheckBox
- JRadioButton
- JLabel
- JTextField
- JTextArea
- JComboBox
- JList
- JScrollBar
- Borders
- Dialogs
- Menus
- Multiple Windows
- Scroll Pane
- Tabbed Pane

JButton

- *Button* là một thành phần gây ra một sự kiện hành động khi được kích chuột.
- Các constructor của JButton:

```
JButton()

JButton(String text)

JButton(String text, Icon icon)

JButton(Icon icon)
```

Các thuộc tính `JButton`

- `text`
- `icon`
- `mnemonic`
- `horizontalAlignment`
- `verticalAlignment`
- `horizontalTextPosition`
- `verticalTextPosition`

Using Buttons

ButtonDemo

Đáp ứng các sự kiện **JButton**

```
public void actionPerformed(ActionEvent e)
{
  // Get the button label
  String actionCommand = e.getActionCommand();

  // Make sure the event source is Left button
  if (e.getSource() instanceof JButton)
    // Make sure it is the right button
    if ("Left".equals(actionCommand))
     System.out.println ("Button pressed!");
}
```

JCheckBox

- *Check box* là một thành phần cho phép người dùng bật hay tắt một lựa chọn, giống như 1 công tắc đèn.
- Các constructor:

```
JCheckBox()
JCheckBox(String text)
JCheckBox(String text, boolean selected)
JCheckBox(Icon icon)
JCheckBox(String text, Icon icon)
JCheckBox(String text,Icon icon,boolean selected)
```

Các thuộc tính **JCheckBox**

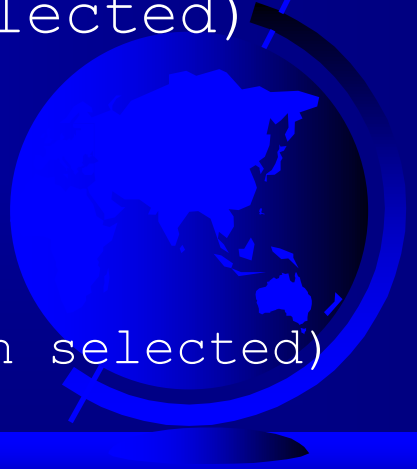
- `JCheckBox` có tất cả các thuộc tính trong `JButton`. Ngoài ta, `JCheckBox` có thuộc tính: `selected`
- Using Check Box:

CheckBoxDemo

JRadioButton

- Các *Radio button* là sự biến đổi của các check box. Chúng thường được sử dụng trong một nhóm khi mà chỉ có 1 button được chọn tại một thời điểm.
- Các constructor:

```
JRadioButton()
JRadioButton(String text)
JRadioButton(String text, boolean selected)
JRadioButton(Icon icon)
JRadioButton(String text, Icon icon)
JRadioButton(String text, Icon icon, boolean selected)
```



Các thuộc tính **JRadioButton**

`JRadioButton` có tất cả các thuộc tính trong `JButton`. Ngoài ta, `JRadioButton` có thuộc tính:

`selected`

Gộp nhóm các Radio Button

```
ButtonGroup btg = new ButtonGroup();

btg.add(jrb1);

btg.add(jrb2);
```

Using Radio Buttons:

RadioButtonDemo

JLabel

- *Label* dùng để hiển thị một chuỗi văn bản thông thường nhằm mô tả thêm thông tin cho các đối tượng khác.
- Các constructor của JLabel:

```
JLabel()
JLabel(String text)
JLabel(String text,int hAlignment)
JLabel(Icon icon)
JLabel(Icon icon, int hAlignment)
JLabel(String text,Icon icon,int hAlignment)
```

Các thuộc tính JLabel

- text
- icon
- horizontalAlignment
- verticalAlignment

Using Labels

LabelDemo

JTextField

- ***Text field*** là ô nhập dữ liệu dạng văn bản trên 1 dòng.
- Các constructor của JTextField:

`JTextField()`

`JTextField(int columns)`

Tạo một text field trống có số cột xác định.

`JTextField(String text)`

Tạo một text field với văn bản có sẵn.

`JTextField(String text, int columns)`

Tạo một text field với văn bản có sẵn và số cột xác định.

Các thuộc tính **JTextField**

- text
- horizontalAlignment
- editable
- columns

Các phương thức `JTextField`

- `getText()`

  Trả về chuỗi ký tự trong text field.

- `setText(String text)`

  Đặt chuỗi ký tự trong text field.

- `setEditable(boolean editable)`

  Cho phép hoặc vô hiệu hóa soạn thảo trong text field. Mặc định, `editable` là `true`.

- `setColumns(int)`

  Thiết lập số cột trong text field. Chiều dài của text field có thể thay đổi.

TextFieldDemo

JTextArea

- ***TextArea*** là khung cho phép người sử dụng nhập vào nhiều dòng văn bản.
- Các constructor của JTextArea:

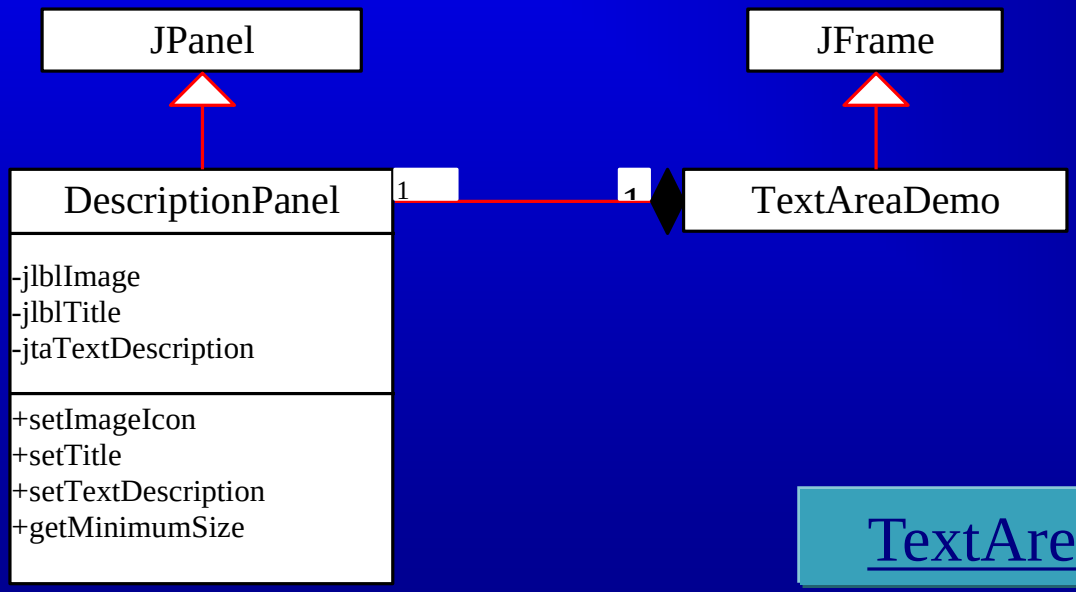
```
JTextArea()

JTextArea(String s)

JTextArea(int rows, int columns)

JTextArea(String s, int rows, int columns)
```

Các thuộc tính **JTextArea**

- text
- editable
- columns
- lineWrap
- wrapStyleWord
- rows
- lineCount
- tabSize

Sử dụng **Text Area**

Chương trình hiển thị 1 ảnh và 1 title trong 1 label, hiển thị văn bản trong text area.



TextAreaDemo

JComboBox

- *Combo box* là danh sách đơn giản các mục chọn. Cơ bản nó thực hiện chức năng giống như 1 list, nhưng chỉ có thể lấy 1 giá trị.
- Các constructor:

`JComboBox()`

tạo 1 combo box rỗng

`JComboBox(Object[] stringItems)`

tạo 1 combo box chứa các phần tử trong dãy

Các phương thức JComboBox

`jcbo.addItem(Object item)`

thêm 1 mục chọn vào `JComboBox jcbo`

`jcbo.getItem()`

`jcbo.getItemAt(int index)`

lấy 1 mục chọn từ `JComboBox jcbo`

`jcbo.removeItemAt(int index)`

loại 1 mục chọn khỏi `JComboBox jcbo`

Using Combo Box:

ComboBoxDemo

Sử dụng `itemStateChanged` Handler

Khi một lựa chọn được check hoặc uncheck, `itemStateChanged()` cho `ItemEvent` và `actionPerformed()` handler cho `ActionEvent` sẽ được gọi.

```
public void itemStateChanged(ItemEvent e){
   // Make sure the source is a combo box
   if (e.getSource() instanceof JComboBox)
      String s = (String)e.getItem();
}
```

JList

- *List* là một thành phần cơ bản thực hiện chức năng giống combo box, nhưng nó cho phép người sử dụng chọn một hoặc nhiều giá trị.
- Các constructor:

`JList()`

tạo 1 list rỗng.

`JList(Object[] stringItems)`

tạo 1 list chứa các phần tử trong dãy.

Các thuộc tính `JList`

- `selectedIndex`
- `selectedIndices`
- `selectedValue`
- `selectedValues`
- `selectionMode`
- `visibleRowCount`

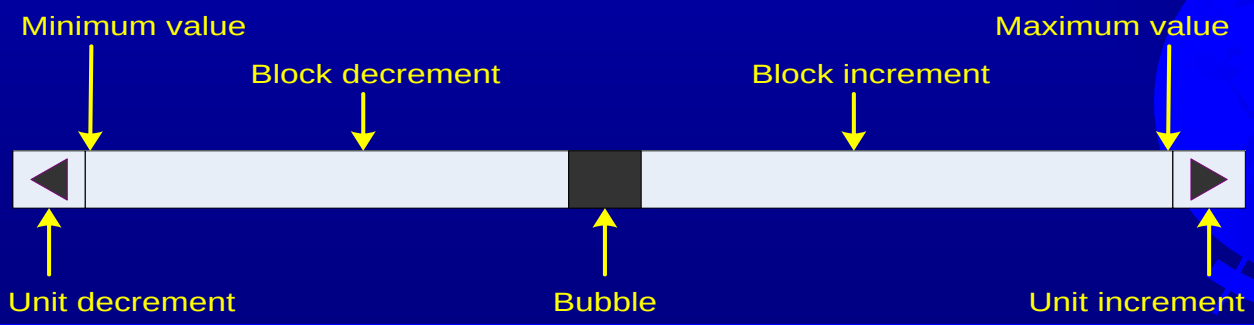
Using Lists:

ListDemo

JScrollBar

- *ScrollBar* là một điều khiển cho phép người sử dụng chọn từ một dải các giá trị.
- Các constructor:

```
JScrollBar()
JScrollBar(int orientation)
JScrollBar(int orientation, int value,
           int extent, int min, int max)
```



Các thuộc tính `JScrollBar`

- `orientation`: 1 - dọc, 0 - ngang
- `maximum, minimum`
- `visibleAmount` (`extent`): độ rộng của phần con chạy
- `value`: giá trị hiện thời của scroll bar
- `blockIncrement`: giá trị được cộng thêm khi kích hoạt vùng tăng.
- `unitIncrement`: giá trị được cộng thêm khi kích hoạt đầu tăng.

Các phương thức **JScrollBar**

- `setBlockIncrement(int increment)`
- `setMaximum(int maximum)`
- `setMinimum`
- `setOrientation(int orientation)`
- `setUnitIncrement(int increment)`
- `setValue(int value)`
- `setVisibleAmount (extent):`
- `getBlockIncrement()`
- `...`

ScrollBarDemo

Borders

- Bạn có thể thiết lập một border trên bất kỳ đối tượng nào của lớp `JComponent`, nhưng thường hữu ích khi thiết lập một titled border trên `JPanel` để nhóm một tập các thành phần giao diện người sử dụng có liên quan.
- Using border:

BorderDemo

Các phương thức tĩnh để tạo Borders

- createTitledBorder(String title)
- createLoweredBevelBorder()
- createRaisedBevelBorder()
- createLineBorder(Color color)
- createLineBorder(Color color, int thickness)
- createEtchedBorder()
- createEtchedBorder(Color highlight,
  Color shadow, boolean selected)
- createEmptyBorder()
- createMatteBorder(int top, int left,
  int bottom, int right, Icon tileIcon)
- createCompoundBorder(Border outsideBorder,
  Border insideBorder)

Dialogs

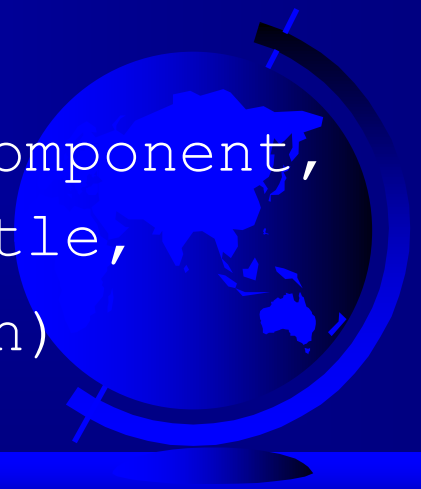
☞ Có thể sử dụng lớp `JOptionPane` để tạo 4 loại dialog chuẩn:

- ***Message Dialog*** hiển thị một message và đợi người sử dụng kích nút OK để đóng hộp thoại.
- ***Confirmation Dialog*** hiển thị câu hỏi và đề nghị người sử dụng trả lời, vd: OK hay Cancel
- ***Input Dialog*** hiển thị câu hỏi và nhận dữ liệu vào từ 1 text field, combo box hoặc list.
- ***Option Dialog*** hiển thị câu hỏi và nhận câu trả lời từ một tập các lựa chọn.

Tạo các Message Dialog

Sử dụng phương thức tĩnh trong lớp `JOptionPane`

```
showMessageDialog(Component parentComponent,
                  Object message)

showMessageDialog(Component parentComponent,
Object message,String title, int messageType)

showMessageDialog(Component parentComponent,
          Object message, String title,
          int messageType, Icon icon)
```



Tạo các Confirmation Dialog

Sử dụng phương thức tĩnh trong lớp `JOptionPane`

```
showConfirmDialog(Component parentComponent,
                  Object message)

showConfirmDialog(Component parentComponent,
Object message, String title, int optionType)

showConfirmDialog(Component parentComponent,
Object message, String title, int optionType,
          [[int messageType], Icon icon])
```

Tạo các Option Dialog

Sử dụng phương thức tĩnh trong lớp `JOptionPane`

```
showOptionDialog(Component parentComponent,
Object message, String title, int optionType,
int messageType, Icon icon, Object[] options,
Object initialValue)
```

Sử dụng các dialog:

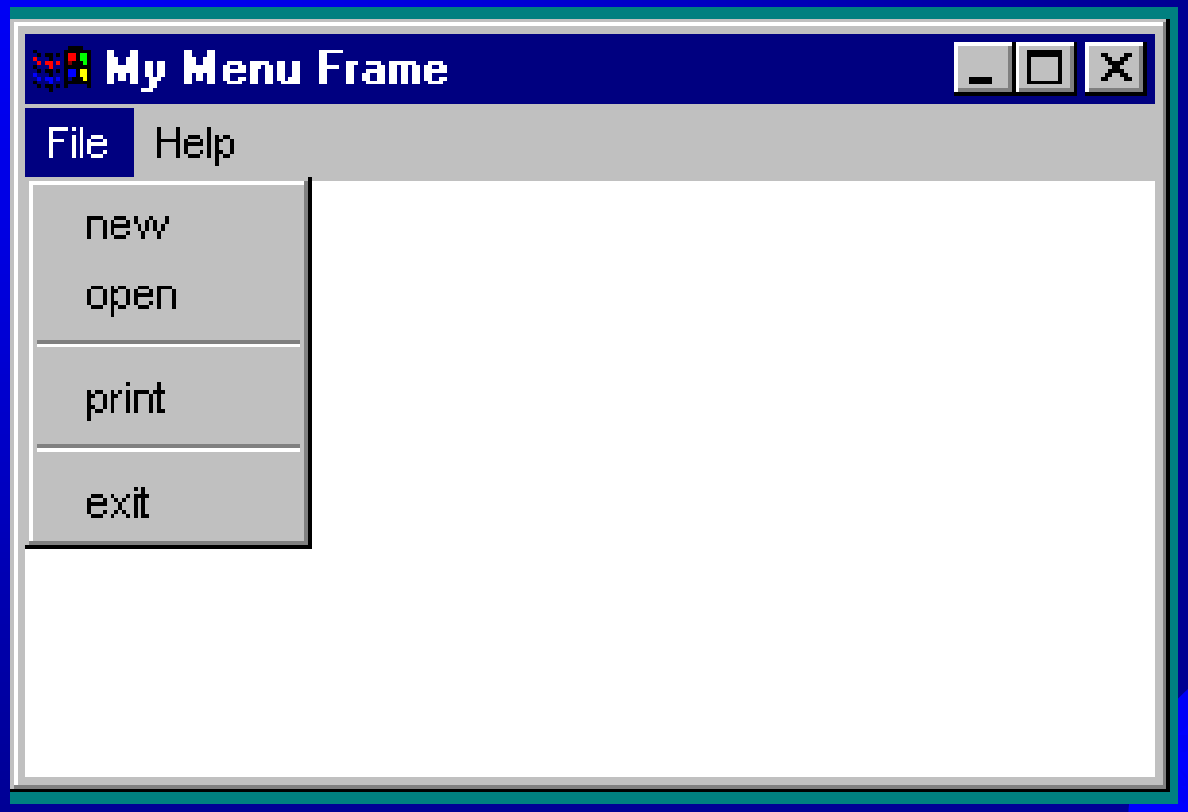
JOptionPaneDemo

Menus

- Java cung cấp một số lớp - `JMenuBar`, `JMenu`, `JMenuItem`, `JCheckBoxMenuItem`, và `JRadioButtonMenuItem` – để thực thi menu trong một frame.

- Một `JFrame` hoặc `JApplet` có thể chứa một *menu bar* trên đó có gắn các *pull-down menu*. Các menu chứa các *menu item* để người dùng lựa chọn (hoặc bật/tắt). Menu bar có thể được xem như một cấu trúc để hỗ trợ các menu.

Menu Demo



Lớp **JMenuBar**

Menu bar chứa các menu; menu bar chỉ có thể được thêm vào 1 frame. Đoạn code sau tạo và thêm một JMenuBar vào 1 frame:

```
JFrame f = new JFrame();
f.setSize(300, 200);
f.setVisible(true);
JMenuBar mb = new JMenuBar();
f.setJMenuBar(mb);
```

Lớp **Menu**

Bạn gắn các menu vào một `JMenuBar`. Đoạn code sau tạo 2 menu `File` và `Help`, và thêm chúng vào `JMenuBar mb`:

```
JMenu fileMenu = new JMenu("File", false);
JMenu helpMenu = new JMenu("Help", true);
mb.add(fileMenu);
mb.add(helpMenu);
```

Lớp **JMenuItem**

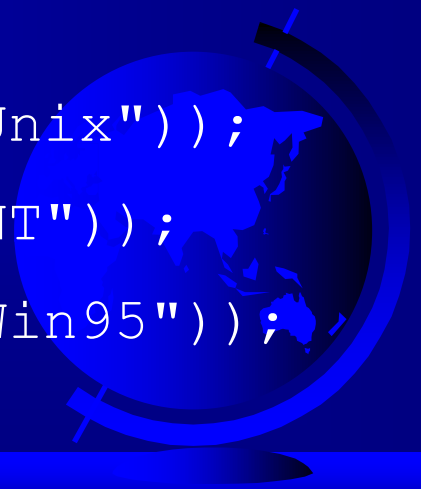
Đoạn code sau thêm các mục chọn (menu item) và các separator trong menu `fileMenu`:

```
fileMenu.add(new JMenuItem("New"));
fileMenu.add(new JMenuItem("Open"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Print"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Exit"));
```

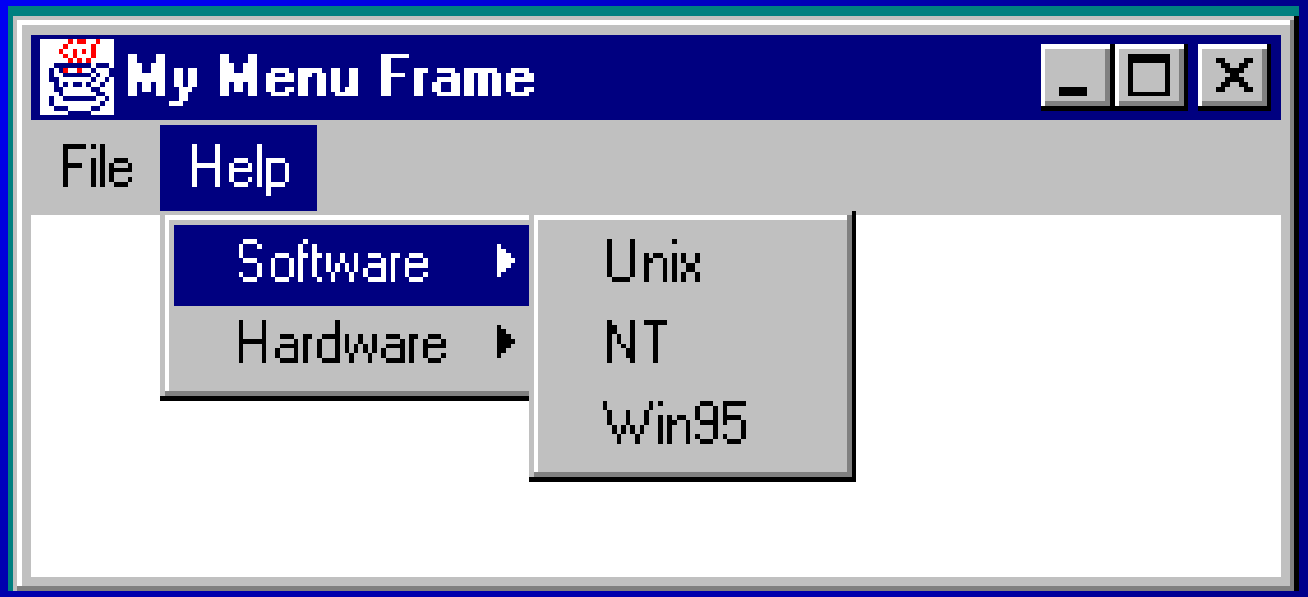
Submenus

Bạn có thể thêm các submenus vào các menu item. Đoạn code sau thêm các submenu “Unix”, “NT”, và “Win95” vào trong mục chọn “Software”.

```
JMenu softwareHelpSubMenu = new JMenu("Software");
JMenu hardwareHelpSubMenu = new JMenu("Hardware");
helpMenu.add(softwareHelpSubMenu);
helpMenu.add(hardwareHelpSubMenu);
softwareHelpSubMenu.add(new JMenuItem("Unix"));
softwareHelpSubMenu.add(new JMenuItem("NT"));
softwareHelpSubMenu.add(new JMenuItem("Win95"));
```



Submenu Demo



Sử dụng Menu

Tạo một giao diện thực hiện các phép toán số học giữa 2 số Number1 và Number2. Giao diện chứa các nhãn và text field cho Number 1, Number 2, và Result.

MenuDemo

Tạo thêm Window - bước 1

Bước 1: Tạo 1 subclass của lớp `JFrame` (được gọi là 1 `SubFrame`) để xác định cửa sổ mới làm việc gì. Ví dụ, tất cả các chương trình ứng dụng GUI mở rộng `JFrame` và là các subclass của `JFrame`.

Tạo thêm Window - bước 2

Bước 2: Tạo 1 instance của `SubFrame` trong ứng dụng hoặc trong applet.

Ví dụ:

```
SubFrame subFrame = new
   SubFrame("SubFrame Title");
```

Tạo thêm Window - bước 3

Bước 3: Tạo 1 `JButton` để kích hoạt `subFrame`.

```
add(new JButton("Activate SubFrame"));
```

Tạo thêm Window - bước 4

Bước 4: chồng phương thức `actionPerformed()` như sau:

```
public actionPerformed(ActionEvent e)
{
   String actionCommand = e.getActionCommand();
   if (e.target instanceof Button)
   {
      if ("Activate
            SubFrame".equals(actionCommand))
      {
         subFrame.setVisible(true);
      }
   }
}
```

Ví dụ: Tạo nhiều Window

Ví dụ tạo 1 main window có 1 text area trong scroll pane, 1 button "Show Histogram". Khi người sử dụng kích vào button, 1 cửa sổ mới xuất hiện để hiển thị biểu đồ cho biểu diễn tần số xuất hiện của các ký tự trong text area.

MultipleWindowsDemo

Histogram

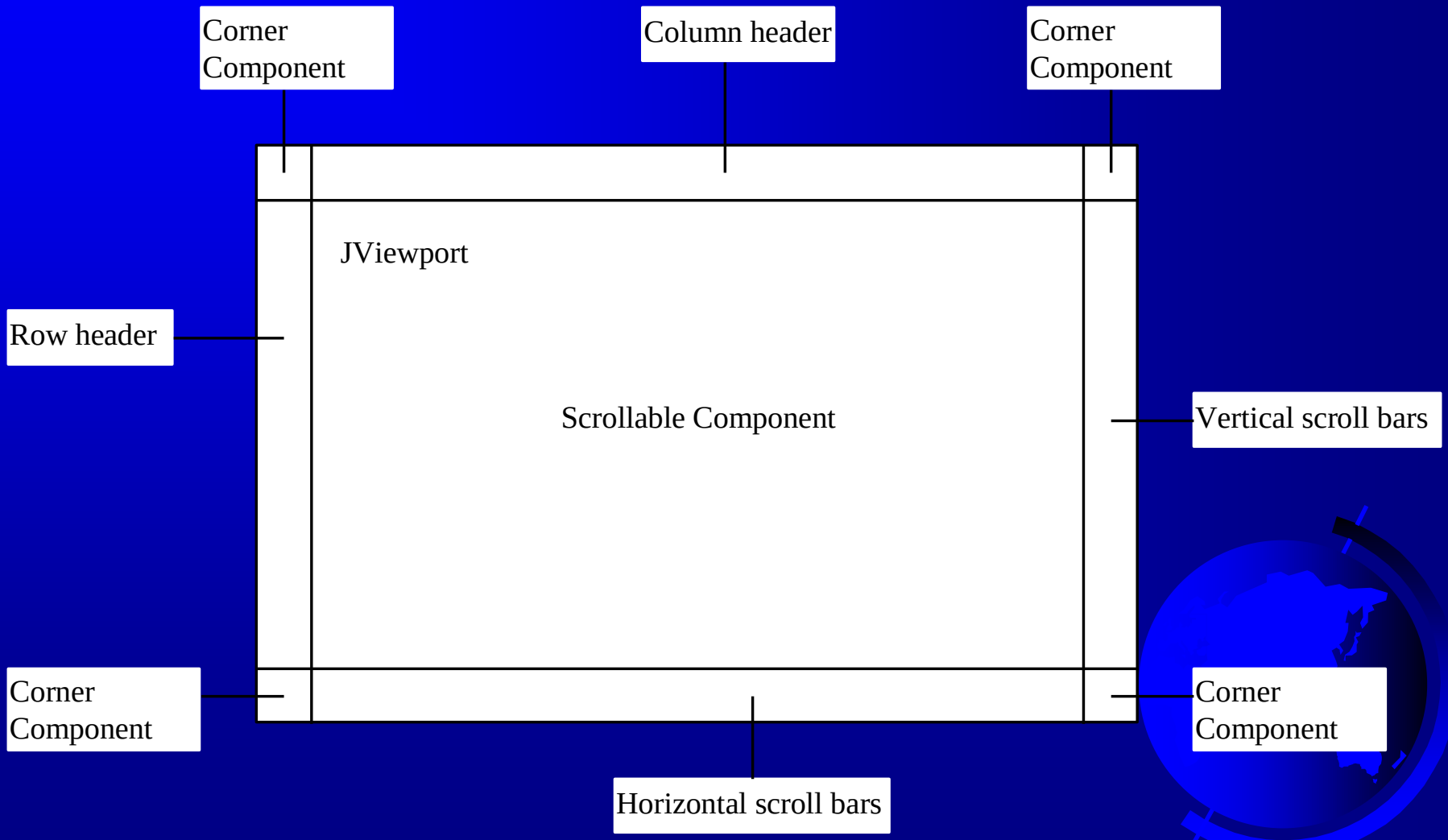
JScrollPane

Scroll pane là một thành phần tự động hỗ trợ cuộn cửa sổ mà không cần lập trình.

Using Scroll Pane:

ScrollMap

Cấu trúc Scroll Pane



JTabbedPane

Tabbed pane cung cấp một tập các tab loại trừ lẫn nhau để truy nhập nhiều thành phần.

Using Tabbed Pane:

FigurePanel

LẬP TRÌNH JAVA

Chương 11: Applet, Ảnh, Âm thanh

Phạm Quang Dũng

BM KHMT - Khoa CNTT - Trường ĐHNN I

Nội dung chương 11

- Applet:
  - khái niệm
  - các phương thức
  - nhúng applet vào trang web
  - xem applet: bằng trình duyệt, bằng appletviewer
  - truyền tham số cho applet
  - làm cho applet chạy như 1 ứng dụng
- Hiển thị ảnh
- Chơi file âm thanh

Applets

- Một applet là một Panel cho phép tương tác với một chương trình Java.
- Một applet thường được nhúng vào trong một trang Web và có thể chạy từ một trình duyệt.
- Bạn cần đoạn mã HTML đặc biệt trong trang Web để "nói" cho trình duyệt về applet.
- Vì lý do bảo mật, các applet chạy trong 1 sandbox: chúng không có quyền truy nhập đến hệ thống file trên các máy client.

Applet Support

- Hầu hết các trình duyệt ngày nay có hỗ trợ Java 1.4 nếu chúng có plugin thích hợp.
- Internet Explorer 5.5 đã được cập nhật, nhưng Netscape thì chưa.
- Sự hỗ trợ tốt nhất không phải là trình duyệt mà là chương trình độc lập appletviewer
- Nói chung bạn nên cố gắng viết các applet có thể chạy với mọi trình duyệt.

Một applet là gì

- Bạn viết 1 applet bằng cách mở rộng lớp Applet
- Applet chỉ là 1 lớp giống các lớp khác, nếu muốn bạn có thể sử dụng chúng trong các chương trình.
- Khi bạn viết 1 applet, bạn chỉ đang viết một phần của chương trình.
- Trình duyệt cung cấp main method

Phả hệ của Applet

```
java.lang.Object
   |
   +----java.awt.Component
          |
          +----java.awt.Container
                 |
                 +----java.awt.Panel
                        |
                        +----java.applet.Applet
```

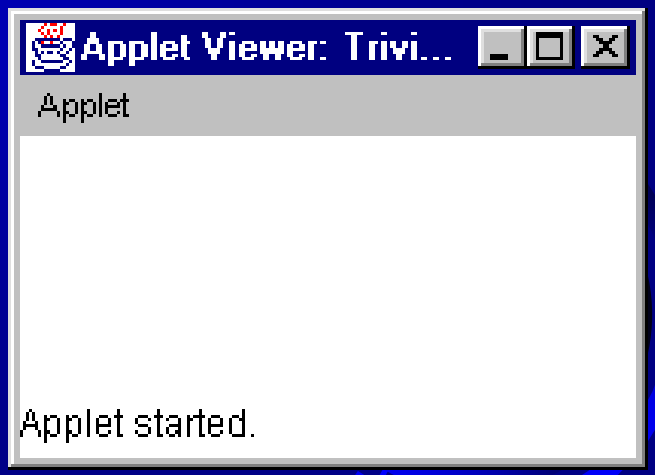
Applet đơn giản nhất có thể

TrivialApplet.java

```
import java.applet.Applet;
public class TrivialApplet extends Applet { }
```

TrivialApplet.html

```
<applet>
    code="TrivialApplet.class”
    width=150 height=100>
</applet>
```



Applet có ý nghĩa đơn giản nhất

```
import java.awt.*;
import java.applet.Applet;

public class HelloWorld extends Applet {
    public void paint( Graphics g ) {
        g.drawString( "Hello World!", 30, 30 );
    }
}
```

Các phương thức Applet

public void init ()
public void start ()
public void stop ()
public void destroy ()
public void paint (Graphics)

Also:

public void repaint()
public void update (Graphics)
public void showStatus(String)
public String getParameter(String)

Tại sao một applet chạy được

- Bạn viết 1 applet bằng cách mở rộng lớp Applet.
- Applet xác định các phương thức init( ), start( ), stop( ), paint(Graphics), destroy( )
- Các phương thức trên không thực hiện việc gì cả. Chúng là các stub (gốc).
- Bạn khiến applet làm gì đó bằng cách chồng các phương thức trên.

public void init ( )

- Đây là phương thức đầu tiên để thực hiện.
- Nó là nơi lý tưởng để khởi tạo các biến.
- Nó là nơi tốt nhất để xác định các thành phần GUI (buttons, text fields, scrollbars, etc.), sắp đặt và thêm các listener vào chúng.
- Hầu hết các applet bạn viết sẽ có một phương thức init( ).

public void start ( )

- Không phải luôn cần đến.
- Được gọi sau init( )
- Được gọi khi trang được tải hoặc khởi động lại.
- Thường được sử dụng chung với stop( )
- start() và stop( ) thường được sử dụng khi Applet đang tính toán mất nhiều thời gian và bạn không muốn tiếp tục để còn chuyển đến trang khác.

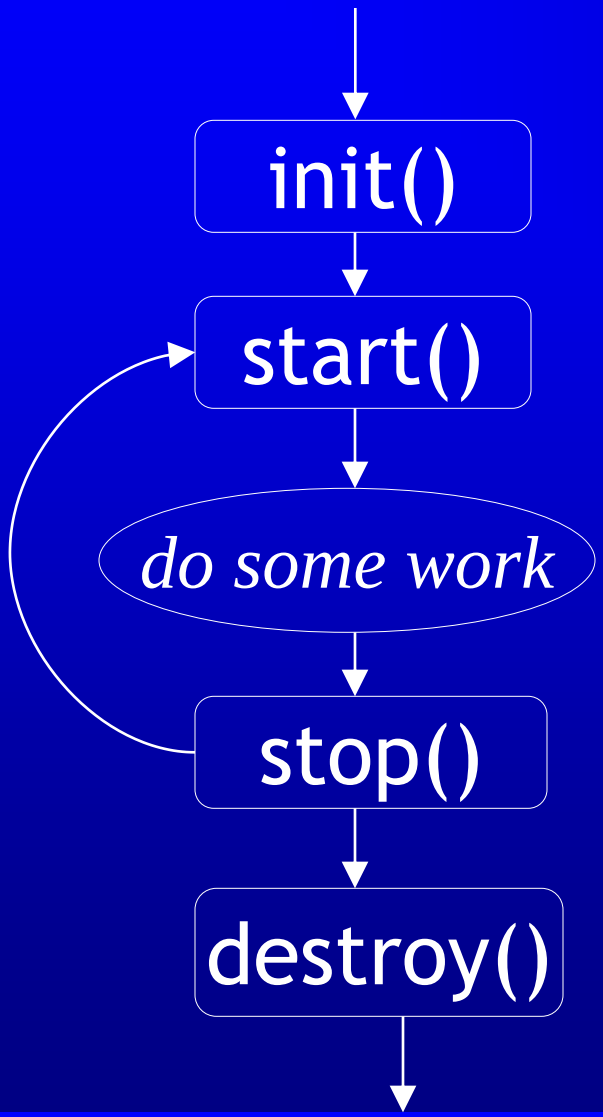
public void stop( )

- Không phải luôn cần đến.
- Được gọi khi trình duyệt bỏ lại trang web
- Được gọi ngay trước destroy( )
- Dùng stop( ) nếu applet đang tính toán nặng nhọc mà bạn không muốn tiếp tục, để trình duyệt đến trang khác.
- Thường được sử dụng chung với start()

public void destroy( )

- Hiếm khi cần đến
- Được gọi sau stop( )
- Sử dụng để giải phóng các tài nguyên hệ thống một cách rõ ràng.
- Các tài nguyên hệ thống thường được giải phóng một cách tự động.

Trình tự các phương thức được gọi



- init và destroy chỉ được gọi đúng 1 lần.
- start và stop được gọi mỗi khi trình duyệt đưa ra và rời khỏi trang web.
- *do some work* là đoạn mã được gọi bởi các *listener*
- paint được gọi khi applet cần được vẽ lại.

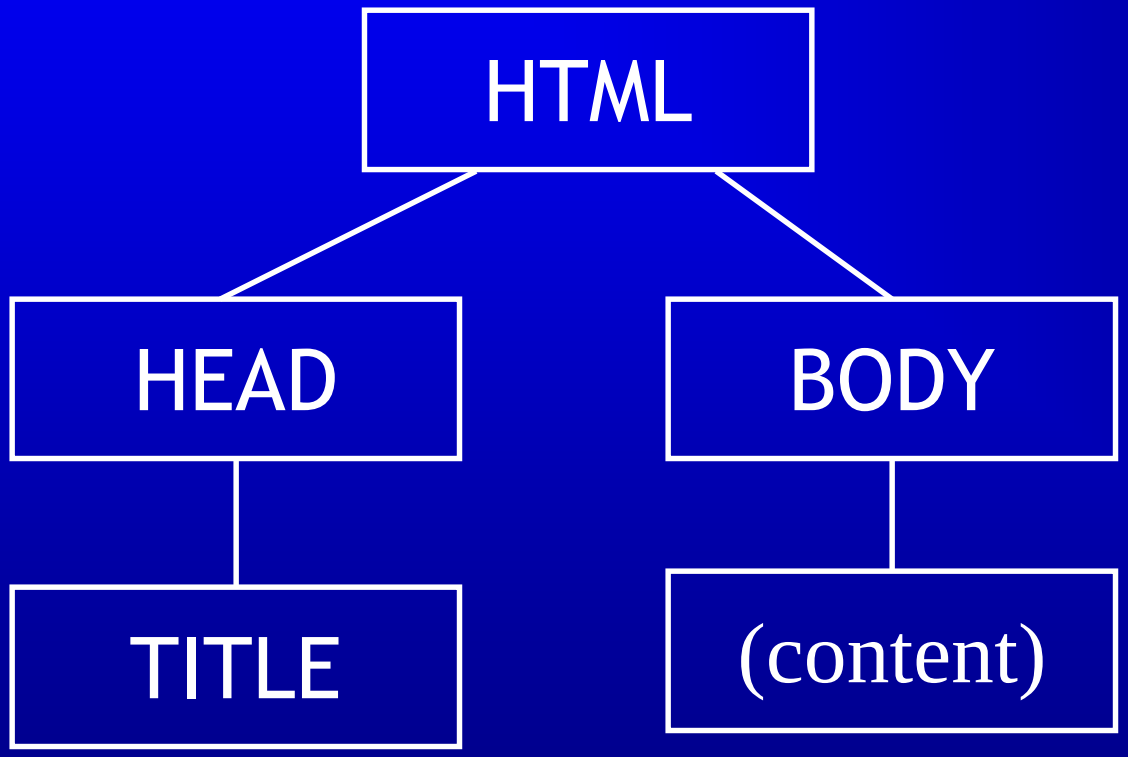
Các phương thức Applet hữu dụng khác

- System.out.println(String s)
  - Chỉ làm việc trong appletviewer, không trong các trình duyệt.
  - Tự động mở một output window.
- showStatus(String) hiển thị chuỗi String trong dòng trạng thái của applet.
  - Mỗi lần gọi ghi đè lên lời gọi trước đó.
  - Bạn phải cho thời gian để đọc dòng trạng thái!

Applets are not magic!

- Bất kỳ cái gì bạn có thể thực hiện trong 1 applet, bạn đều có thể thực hiện trong 1 ứng dụng.
- Bạn có thể thực hiện một số việc trong 1 ứng dụng, nhưng không thể thực hiện được trong 1applet.
- Nếu bạn muốn truy nhập các file từ 1applet, nó phải là 1 “trusted” applet.
- Các trusted applet không nằm trong phạm vi của khóa học này.

Cấu trúc của một trang HTML



- Hầu hết các HTML tag là các container.
- Một container từ <tag> đến </tag>

HTML

```
<html>
  <head>
    <title> Hi World Applet </title>
  </head>

  <body>
    <applet code="HiWorld.class”
     width=300 height=200>
        <param name="arraysize" value="10">
    </applet>
  </body>
</html>
```

Xem Applet bằng appletviewer

- Bạn có thể test hay view các applet trong cửa sổ DOS prompt bằng cách gọi tiện ích appletviewer

WelcomeApplet

Run

Xem Applet từ trình duyệt Web

- Bạn cũng có thể test hay view các applet trong một trình duyệt Web

LoanApplet

Run

Truyền String cho Applet

Để truyền tham số cho applet, tham số phải được khai báo bằng thẻ <param> (không có end tag) đặt trong thẻ <applet>

<param name=parametername value=stringvalue>

Vd:

<applet>

.....

<param name = Message value = "Welcome to Java">

<param name = X value = 20>

<param name = Y value = 30>

</applet>

DisplayMessage

Run

Làm cho Applet chạy như 1 ứng dụng

- Nói chung, 1 applet có thể được convert thành 1 ứng dụng mà không mất chức năng.
- Một ứng dụng được convert thành 1 applet chỉ cần nó không vi phạm sự giới hạn bảo mật áp đặt cho applet.
- Bạn có thể thực thi phương thức main trong 1 applet để làm cho nó có thể chạy như 1 ứng dụng:
  - Khi chạy chương trình như 1 applet, phương thức main bị bỏ qua.
  - Khi chạy chương trình như 1 ứng dụng, phương thức main được gọi.

Làm cho Applet chạy như 1 ứng dụng

```
public static void main(String[] args) {
JFrame frame = new JFrame("Running a
                  program as applet and frame");
TestApplet applet = new TestApplet();
frame.getContentPane().add(applet,
                           BorderLayout.CENTER);
applet.init();
applet.start();
frame.setSize(300,300);
frame.setVisible(true);
}
```

DisplayMessageApp

Run

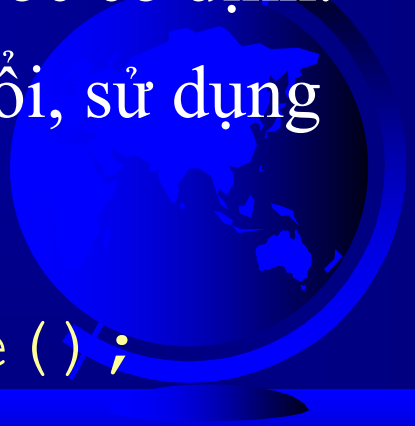
Hiển thị ảnh

- Bạn đã học cách sử dụng lớp `ImageIcon` để tạo biểu tượng từ 1 file ảnh và dùng phương thức `setIcon` hoặc constructor để đặt biểu tượng trong 1 thành phần GUI.

```
ImageIcon imgIcon = new ImageIcon(path);
```

- Biểu tượng ảnh đó hiển thị 1 ảnh kích thước cố định. Để hiển thị ảnh trong 1 kích thước thay đổi, sử dụng lớp `java.awt.Image`

```
Image img = imgIcon.getImage();
```



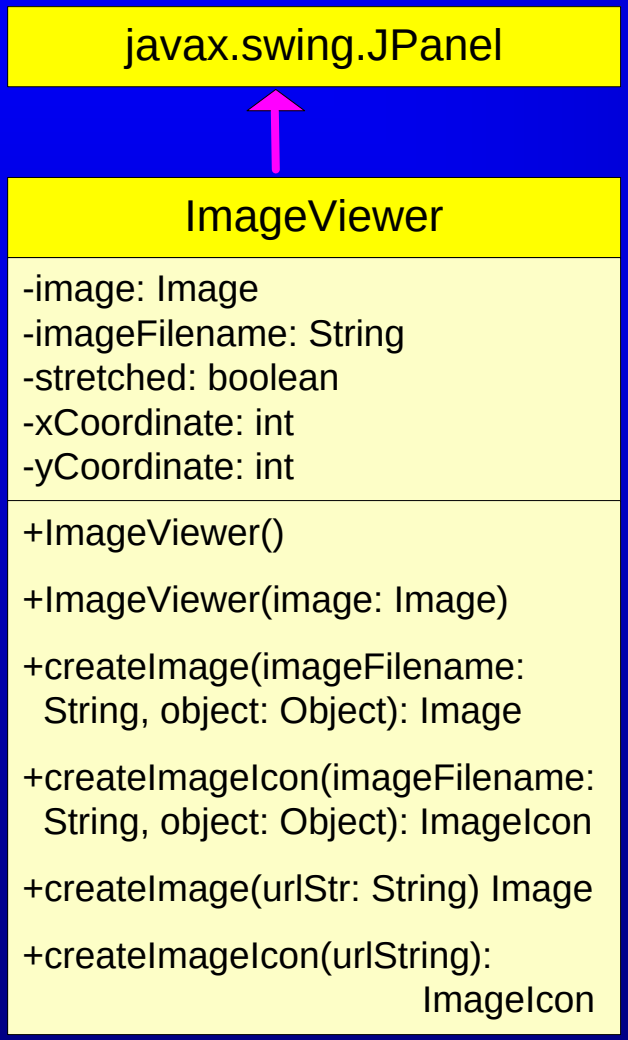
Hiển thị ảnh (tiếp)

- Có thể sử dụng Label làm nơi hiển thị ảnh → đơn giản, thuận tiện, tuy nhiên không tùy biến được nhiều cách hiển thị ảnh.
- Để hiển thị ảnh một cách linh hoạt, sử dụng phương thức `drawImage` của lớp Graphics trên 1 panel

```
drawImage(Image img, int x, int y,
[int width, int height,] [Color
bgcolor,] ImageObserver observer)
```

DisplayImage

Xây dựng lớp ImageViewer



- Mục đích: Xây dựng lớp ImageViewer để có thể tái sử dụng hiển thị ảnh trên 1 panel.

ImageViewer

SixFlags

Image Animation

- Mục đích: Minh họa cách hiển thị một chuỗi các ảnh tạo thành đoạn phim. Sử dụng 1 timer để gây ra sự kiện repaint vẽ ảnh khác trên vùng hiển thị.

ImageAnimation

Run

Playing Audio

- Với Java2, bạn có thể chơi các file âm thanh có định dạng WAV, MIDI, AIFF, AU, RMF.
- Để chơi tệp âm thanh trong 1 applet, đầu tiên cần tạo 1 *audio clip object* cho tệp.

```
AudioClip adClip = Applet.newAudioClip(path);
```

- Các phương thức trong `java.applet.AudioClip`
  - `+ play()`
  - `+ loop()`
  - `+ stop()`

FlagAnthem

Run