ĐẠI NAM
UNIVERSITY

CHƯƠNG 1
LẬP TRÌNH JAVA CƠ BẢN

ThS. Phạm Văn Tiệp

Nội dung

- Giới thiệu chung về Java
- Các phần tử cơ bản của Java
- Các phương thức vào ra cơ bản
- Câu trúc điều khiên
- Mảng và chuỗi

ĐẠI NAM
UNIVERSITY

GIỚI THIỆU CHUNG VỀ JAVA

Lịch sử phát triển của Java

- Ngôn ngữ lập trình Java được phát triển vào năm 1991 bởiSun Microsystems (nay là Oracle).
- Tiêu chí phát triển: "Write Once, Run Anywhere"
- Ban đầu được sử dụng để xây dựng ứng dụng điều khiển các bộ xử lý bên trong các thiết bị điện tử dân dụng như máy điện thoại cầm tay, lò vi sóng…
- Bắt đầu được sử dụng từ năm 1995 (với tên gọi là “Oak” → cây sồi)

James Gosling

Lịch sử phát triển của Java

Các ưu điểm của Java:

- **Đơn giản**
  - Loại bỏ con trỏ
  - Không có goto, file header
  - Loại bỏ struct và union
- **Hướng đối tượng**
  - Java được thiết kế xoay quanh mô hình hướng đối tượng.
- **Mạnh**
  - Chặt chẽ → Loại bỏ các kiểu dữ liệu dễ gây lỗi
- **Độc lập phần cứng**
  - Viết một lần, chạy nhiều nơi.

Lịch sử phát triển của Java

Các nhược điểm của Java:

- Tốc độ chậm
- Khó sử dụng các đặc tính của hệ nền
- Bộ thông dịch Java phải được cài đặt trên máy đích.

Lịch sử phát triển của Java

- Ngày nay, nhắc đến Java, không còn nhắc đến như một ngôn ngữ mà còn là một công nghệ, một nền tảng phát triển.
  - Java có một cộng đồng phát triển mạnh mẽ
  - Một tập hợp các thư viện với số lượng lớn (từ Sun và các nguồn khác)

Các công nghệ Java

Giới thiệu tóm tắt về 3 công nghệ của Java: **J2SE, J2ME, J2EE**.

- **J2SE**: Java 2 Platform, Standard Edition: Phiên bản desktop của Java cung cấp GUI, threads, networking, XML, Applets, security, database,…

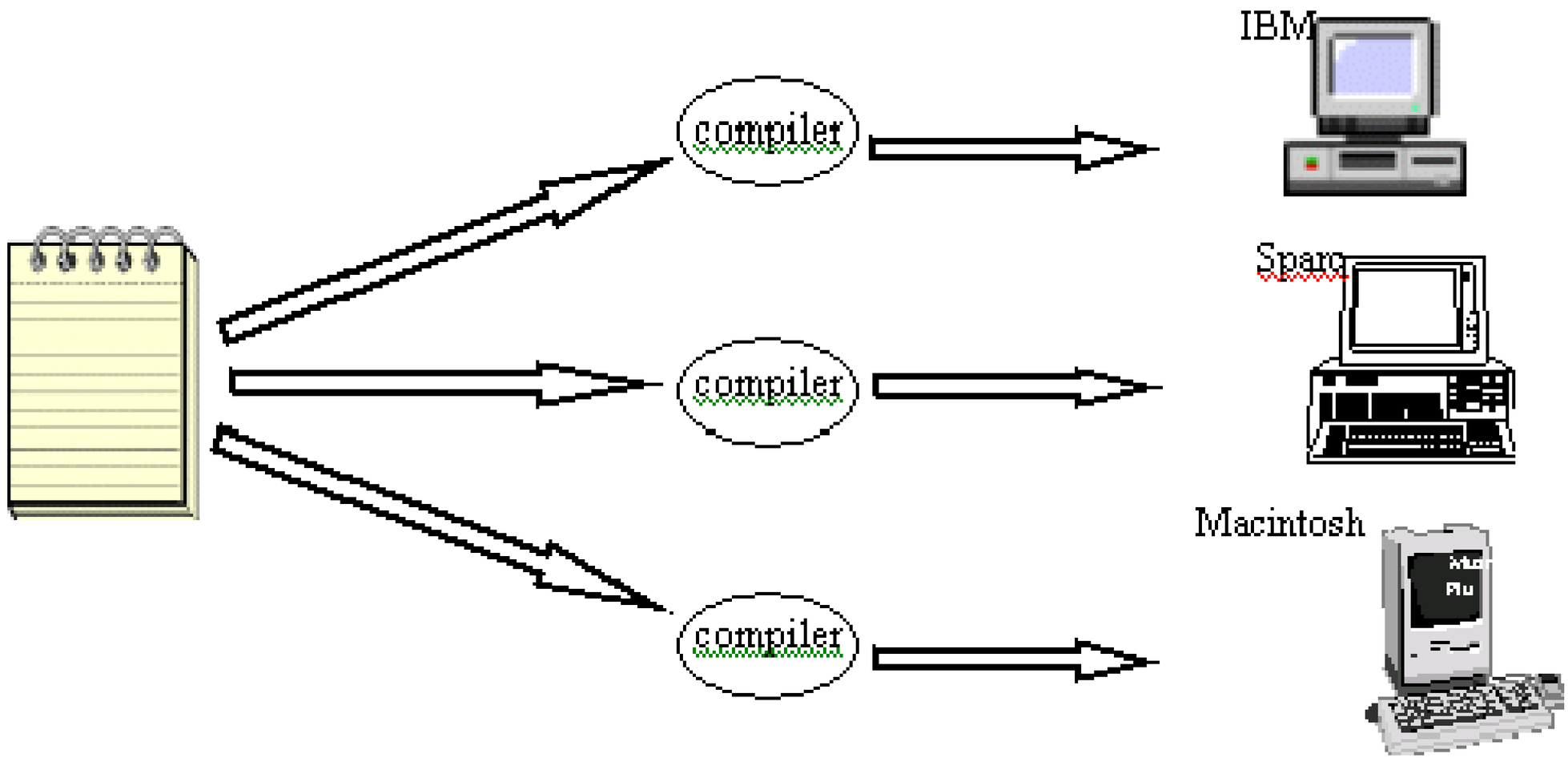
- **J2ME**: Java 2 Platform, Micro Edition: phiên bản dành cho các thiết bị di động, không phải là J2SE thu nhỏ, gồm 2 phần cơ bản là câu hình (configuration) và mô tả (profile).

- **J2EE**: Java 2 Platform, Enterprise Edition: phiên bản cho các ứng dụng lớn, có phân tán, gồm rất nhiều công nghệ được tích hợp như: Java Servlet, JavaServer Pages, JDBC, JMS, Java Webservice,…

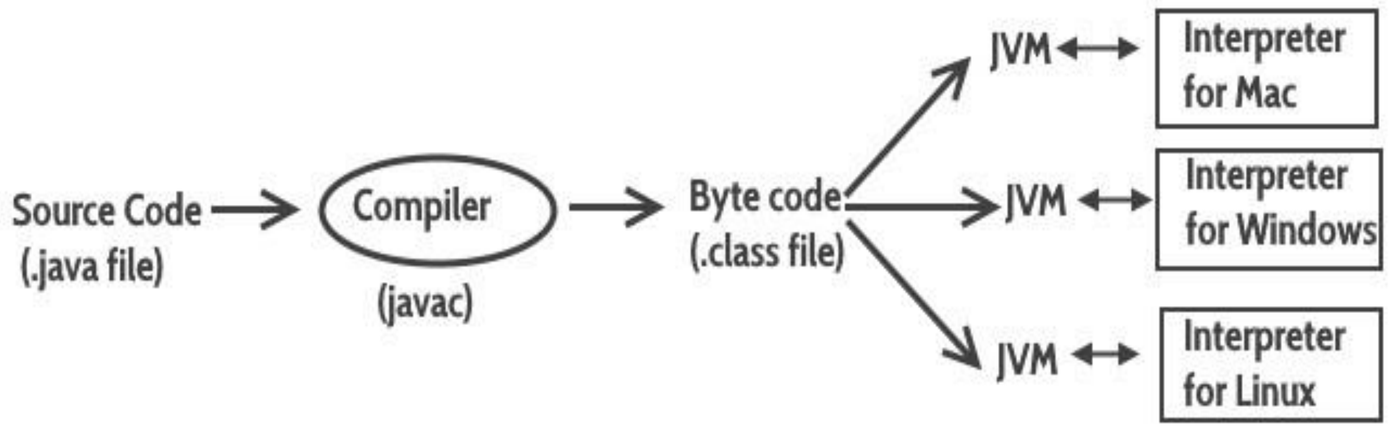
Các công nghệ Java

- **Java là ngôn ngữ vừa biên dịch vừa thông dịch**
  - **Biên dịch**: Mã nguồn được biên dịch bằng công cụ JAVAC để chuyển thành dạng ByteCode
  - **Thông dịch**: Bytecode thực thi trên từng loại máy cụ thể nhờ chương trình thông dịch (năm trong máy ảo Java)
  - Nhằm mục đích viết một lần, chạy nhiều nơi

Kiểu dịch của các trình biên dịch ngôn ngữ cũ



Cơ chế máy ảo Java



- **Trình biên dịch** chuyên mã nguồn thành tập các lệnh không phụ thuộc vào phần cứng cụ thể.
- **Trình thông dịch** trên mỗi máy chuyển tập lệnh này thành chương trình thực thi
- Máy ảo tạo ra một môi trường để thực thi các lệnh bằng cách:
  - Nạp các file .class
  - Quản lý bộ nhớ
  - Dọn "rác"

JDK – JAVA DEVELOPMENT KIT

JDK- Java Development Kit- Bộ công cụ phát triển ứng dụng Java bao gồm 4 thành phần:

Classes, Compiler, Debugger, Java Runtime Environment.

- JDK Alpha và Beta (1995)
- JDK 1.0 (23/1/1996)
- JDK 1.1 (19/2/1997)
- J2SE 1.2 (8/12/1998)
- J2SE 1.3 (8/5/2000)
- J2SE 1.4 (6/5/2002)
- J2SE 5.0 (30/9/2004)
- Java SE 6 (11/12/2006)
- Java SE 7 (28/7/2011)
- Java SE 8 (18/3/2014)
- Java SE 9 (9/2017)
- Java SE 10 (3/2018)
- JDK SE 11 (9/2018)
- JDK SE 12 (3/2019)
- JDK SE 13 (9/2019)
- JDK SE 14 (3/2020)
- JDK SE 15 (9/2020)
- JDK SE 16 (3/2021)
- JDK SE 17 (9/2021)
- JDK SE 18 (3/2022)
- JDK SE 19 (9/2022)
- JDK SE 20 (3/2023)
- **JDK SE 21 (LTS) (9/2023)**
- JDK SE 22 (3/2024)

JDK – JAVA DEVELOPMENT KIT

Bao gồm:

- javac: Chương trình dịch chuyển mã nguồn sang bytecode
- java: Bộ thông dịch, thực thi java application
- appletviewer: Bộ thông dịch, thực thi java applet mà không cần sử dụng trình duyệt như Nestcape, FireFox hay IE, v.v.
- javadoc: Bộ tạo tài liệu dạng HTML từ mã nguồn và chú thích
- jdb: Bộ gỡ lỗi (java debugger)
- javap: Trình dịch ngược bytecode
- jar: Dùng để đóng gói lưu trữ các module viết bằng Java (tạo ra file đuôi .jar), là phương pháp tiện lợi để phân phối những chương trình Java.

Môi trường lập trình

- **Cài Java Development Kit (JDK)**

  https://www.oracle.com/java/technologies/downloads/#jdk21-windows

- **Cài IDE**
  - Notepad / Notepad++ (https://notepad-plus-plus.org)
  - Eclipse (http://www.eclipse.org)
  - NetBeans (http://netbeans.org)
  - IntelliJ IDEA (http://www.jetbrains.com/idea)

Cài đặt môi trường

- **Cài đặt JDK**

https://www.oracle.com/java/technologies/downloads/#jdk21-windows

- **Cài đặt Eclipse**

https://eclipse.org/downloads/

ĐẠI NAM
UNIVERSITY

CÁC PHẦN TỬ CƠ BẢN CỦA JAVA

Biến

- Là phần tử trong chương trình có thể thay đổi giá trị
- **Khai báo:**

  DataType varName1,varName2,...,varNameN;

hoặc

  DataType varName1 = Literal1,...,varNameN = LiteralN;

- Trongđó:
  - varName: là tên biến, đặt theo quy tắc định danh
  - Literal: có thể là một biến khác đã được khai báo trước

Trước khi sử dụng trong biểu thức, biến phải được khởi tạo giá trị.

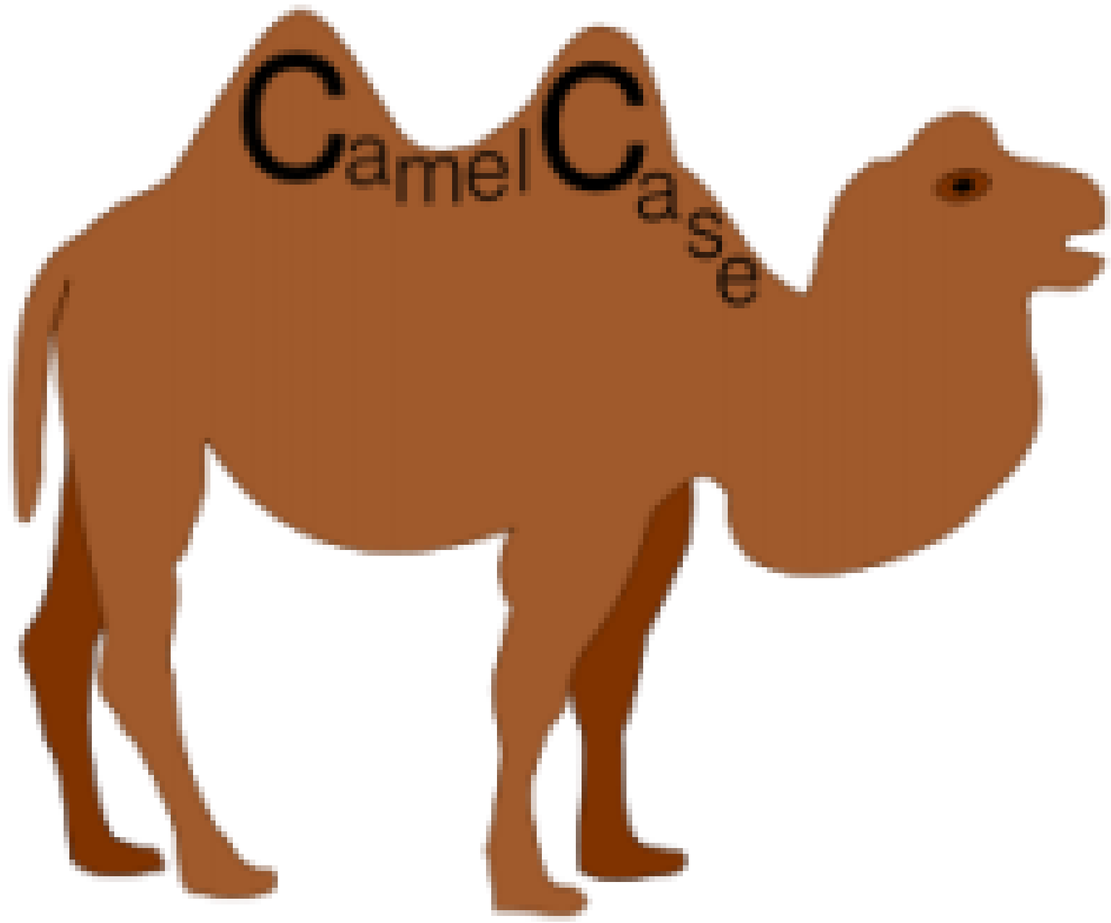
Quy tắc đặt tên biến

- Gồm các ký tự chữ, ký tự số, dấu gạch dưới ‘_’, và dấu ‘$’.
- Bắt đầu bằng ký tự chữ.
- Không được trùng với từ khóa và từ dành riêng của Java.
- Có phân biệt chữ hoa – thường.
- Chỉ gồm một từ đơn và nên viết chữ thường.
- Nếu tên biến gồm nhiều từ, ký tự đầu của từ đầu viết thường, ký tự đầu của mỗi từ kế tiếp viết hoa

Từ khóa và định danh

- Là những từ được Java quy định ý nghĩa và cách sử dụng
- Định danh: xâu ký tự, xác định duy nhất một phần tử trong chương trình
- **Quy định với định danh:**
  - Không đặt trùng với từ khóa
  - Không bắt đầu bằng chữ số
  - Ký tự được phép sử dụng: chữ cái, chữ số, $, _
  - Phân biệt chữ hoa, chữ thường

Quy tắc “con lạc đà”



Quy tắc "con lạc đà"

- **Biến**: bắt đầu bằng chữ thường, viết hoa chữ cái đầu tiên các từ còn lại
- **Hằng**: Toàn bộ bằng chữ hoa
- **Lớp**: viết hoa chữ cái đầu tiên các từ
- **Thuộc tính, phương thức**: bắt đầu bằng chữ thường, viết hoa chữ cái đầu tiên các từ còn lai.
- **Gói**: sử dụng chữ thường, không đặt trùng với từ khóa

Biến	myName, numberOfStudent
Hằng	AX_LINE, USER_PARAMETER
Lớp	HelloWorld, Student
Thuộc tính	studentID, mark
Phương thức	setValue(), getValue(), sortByName()
Gói	sis.subject, sis.student

Packages (Gói)

- **Package** được dùng để đóng gói các lớp trong chương trình lại với nhau thành một khối.
- Gói (packages), là thư viện chứa các lớp, các đối tượng, các hằng,… cung cấp cho việc lập trình: java.awt, java.io, javax.swing,…

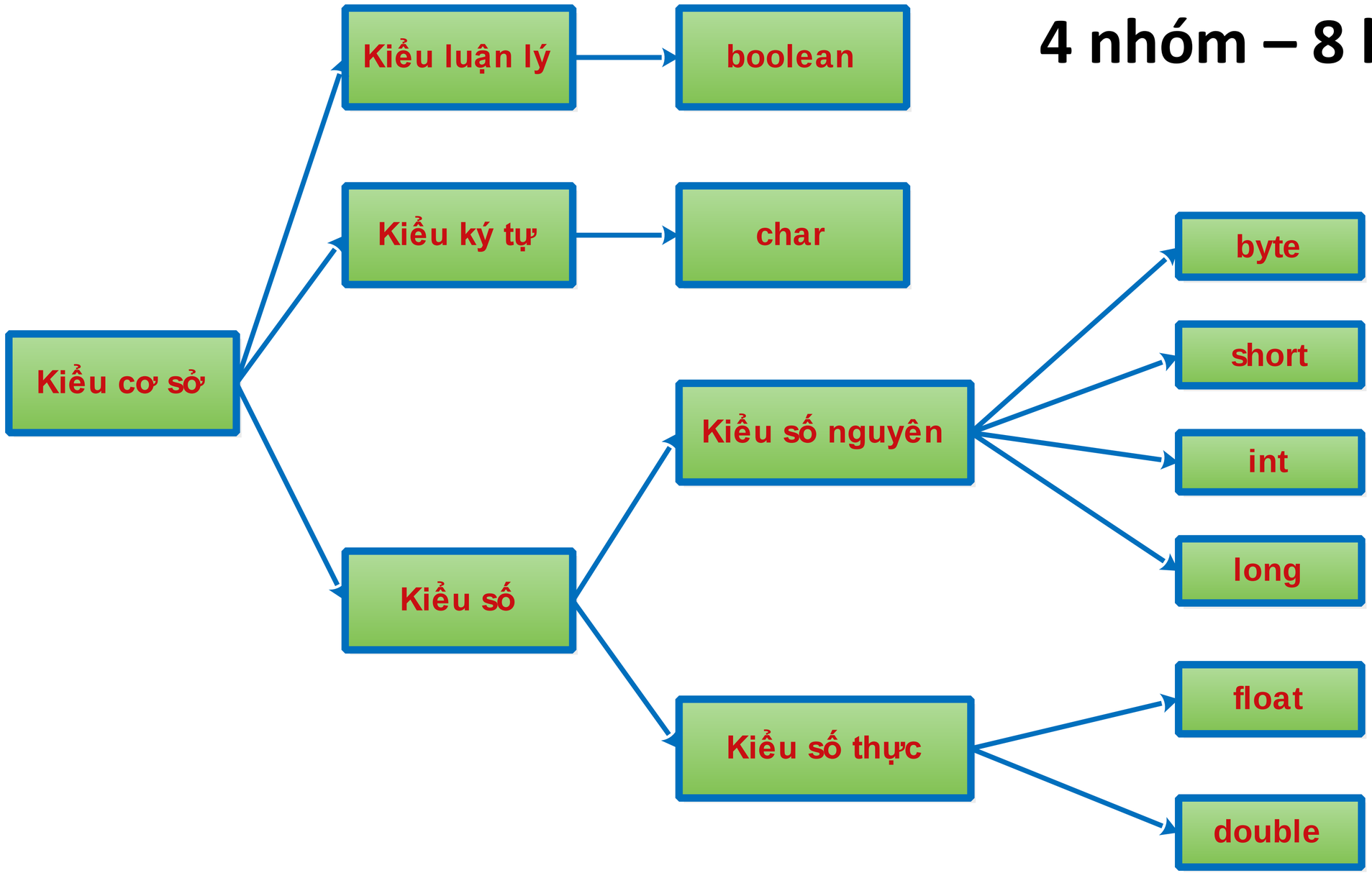
Các kiểu dữ liệu

Trong Java kiểu dữ liệu được chia thành hai loại:

- **Kiểu dữ liệu nguyên thủy (primitive)**
  - Số nguyên (integer)
  - Số thực (float)
  - Ký tự (char)
  - Giá trị logic (boolean)
- **Kiểu dữ liệu tham chiếu (reference)**
  - Mảng (array)
  - Đối tượng (object)

Kiểu dữ liệu cơ sở

4 nhóm – 8 kiểu



Lớp bao (Wrapper class)

Các kiểu dữ liệu nguyên thủy không có các phương thức liên quan đến nó.

- Mỗi kiểu dữ liệu nguyên thủy có một lớp tương ứng gọi là lớp bao (wrapper class)
- Các lớp bao sẽ "gói" dữ liệu nguyên thủy và cung cấp các phương thức thích hợp cho dữ liệu đó.
- Mỗi đối tượng của lớp bao đơn giản là lưu trữ một biến đơn và đưa ra các phương thức để xử lý nó.

Kiểu dữ liệu cơ sở

Kiểu cơ sở	Lớp bao (Wrapper class)
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Wrapper class rất hữu ích, hỗ trợ tối đa cho kiểu cơ sở

Kiểu dữ liệu cơ sở

Kiểu dữ liệu	Kích thước	Giá trị mặc định	Giá trị nhỏ nhất	Giá trị lớn nhất
byte	8	0	-128	127
short	16	0	-32768	32767
int	32	0	-2147483648	2147483647
long	64	0L	-2^{63}	$2^{63} - 1$
float	32	0.0f		
double	64	0.0d		
boolean	Không xác định	false	NA	NA
char	16	\u0000	NA	NA

Toán tử số học

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
-	Phép đổi dấu	Số thực hoặc số nguyên	int a, b; -a; -14
+	Phép toán cộng	Số thực hoặc số nguyên	float x, y; x + y; 5 + 4
-	Phép toán trừ	Số thực hoặc số nguyên	
*	Phép toán nhân	Số thực hoặc số nguyên	
/	Phép toán chia	Số thực hoặc số nguyên	10.0/3.0; (bằng 3.33...) 10/3.0; (bằng 3.33...) 10.0/3.0; (bằng 3.33...)
/	Phép chia lấy phần nguyên	Giữa 2 số nguyên	10/3; (bằng 3)
%	Phép chia lấy phần dư	Giữa 2 số nguyên	10%3; (bằng 1)

Toán tử nhị phân

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
&	Phép VÀ nhị phân	2 số nhị phân	0 & 0; (bằng 0) 101 & 110; (bằng 100)
|	Phép HOẶC nhị phân	2 số nhị phân	0 | 0; (bằng 0) 0 | 1; (bằng 1)
^	Phép XOR nhị phân	2 số nhị phân	101 ^ 110; (bằng 011)
<<	Phép DỊCH TRÁI nhị phân	Số nhị phân	a << n; (bằng a * 2^n)
>>	Phép DỊCH PHẢI nhị phân	Số nhị phân	a << n; (bằng a / 2^n)
~	Phép ĐẢO BIT nhị phân (lấy bù 1)	Số nhị phân	~ 0; (bằng 1) ~ 1; (bằng 0) ~ 110; (bằng 001)

Toán tử quan hệ

Toán tử	Ý nghĩa	Ví dụ
>	So sánh lớn hơn giữa hai số nguyên hoặc số thực	2 > 3; (có giá trị 0) 4 > 1; (có giá trị 1)
>=	So sánh lớn hơn hoặc bằng giữa hai số nguyên hoặc số thực	6>=3; (có giá trị 1)
<	So sánh nhỏ hơn giữa hai số nguyên hoặc số thực	3 < 4; (có giá trị 1)
<=	So sánh nhỏ hơn hoặc bằng giữa hai số nguyên hoặc số thực	2 < = 2; (có giá trị 1)
==	So sánh bằng giữa hai số nguyên hoặc số thực	3==5; (có giá trị 0)
!=	So sánh khác hoặc bằng giữa hai số nguyên hoặc số thực	4!=5; (có giá trị 1) 5!=5; (có giá trị 0)

Toán tử logic

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
&&	Phép VÀ LOGIC. Biểu thức VÀ LOGIC nhận giá trị 1 khi và chỉ khi cả 2 toán hạng đều bằng 1.	Hai biểu thức logic	2 < 4 && 5 > 6; (có giá trị 0)
||	Phép HOẶC LOGIC. Biểu thức HOẶC LOGIC nhận giá trị 0 khi và chỉ khi cả 2 toán hạng đều bằng 0.	Hai biểu thức logic	3 > 2 || 4 < 3; (có giá trị 1)
!	Phép PHỦ ĐỊNH LOGIC một ngôi. Biểu thức PHỦ ĐỊNH LOGIC có giá trị bằng 1 nếu toán hạng bằng 0 và có giá trị bằng 0 nếu toán hạng bằng 1.	Biểu thức logic	! (2 > 5); (có giá trị 1)

Các toán tử khác

- Toán tử rút gọn: +=, −=, *=, /=, ...
- Toán tử tăng 1 đơn vị: ++
- Toán tử giảm 1 đơn vị: −−
- Toán tử điều kiện: ?:

(boolean_expression) ? true_expression : false_expression

  - Nếu boolean_expression đúng, tính giá trị true-expression
  - Nếu boolean_expression sai, tính giá trị false-expression

Hằng số

- Phần tử trong chương trình không thể thay đổi giá trị
- **Cú pháp:**

  final DataType CONSTANT_NAME = Literal;
- **Trong đó:**
  - final: từ khóa
  - DataType: Kiểu dữ liệu
  - CONSTANT_NAME: Tên hằng. Tuân thủ quy tắc định danh
  - Literals: Giá trị hằng

Giá trị hằng (Literals)

- **Boolean**: true, false
- **Số nguyên**:
  - Hệ cơ số 8: Bắt đầu bằng chữ số 0

    Ví dụ: 012 = 001010(2) = 8 + 2 = 10(10)
  - Hệ cơ số 16: Bắt đầu bằng 0x

    Ví dụ: 0x2A = 00101010 = 2x16 + 10 = 42
  - Kiểu dữ liệu long: Kết thúc bằng ký tự L hoặc l

    Ví dụ: 10L

Giá trị hằng (Literals)

- **Số thực**:
  - Mặc định có kiểu double
  - Kiểu float: Kết thúc bằng ký tự F hoặc f
  - Dạng dấu phẩy động: Ký tự e (hoặc E) kèm theo số mũ

    Ví dụ: 1.2E7
- **Ký tự**: Đặt giữa dấu nháy đơn. Ví dụ: ‘a’
- **Xâu ký tự**: Đặt giữa dấu nháy kép

  Ví dụ: “CNTT-DAINAM”

Toán tử gán

Cú pháp:

variable = expression;

- Biến variable và biểu thức expression nên có cùng kiểu dữ liệu
- Trong thường hợp hai vế có kiểu dữ liệu khác nhau:
  - Vế trái có kiểu dữ liệu "rộng" hơn: ép kiểu tự động
  - Ngược lại: không hợp lệ. Nếu vẫn muốn thực hiện phép gán, cần ép kiểu
- Trong có các giá trị khác kiểu, tất cả các giá trị được ép tự động thành kiểu rộng nhất

Toán tử gán

Ví dụ:

```
long a = 1.2;                 //không hợp lệ
long b = (long) 1.2;          //hợp lệ
int m = b/2;                  //không hợp lệ
char ch = ‘a’;                //hợp lệ
int n = ch;                   //hợp lệ
short k = ch;                 //không hợp lệ
short p = (short) ch;         //hợp lệ
float x = 1.2;                //không hợp lệ
float y = 1.2f;               //hợp lệ
```

ĐẠI NAM
UNIVERSITY

CÁC PHƯƠNG THỨC VÀO RA CƠ BẢN

http://dainam.edu.vn

Hiển thị dữ liệu

- Phương thức System.out.println(): Hiển thị dữ liệu và xuống dòng
- Phương thức System.out.print(): Hiển thị dữ liệu
- Phương thức System.out.printf(): Hiển thị dữ liệu có định dạng
- Phương thức System.out.format(): Hiển thị dữ liệu có định dạng
- Có thể dùng toán tử + để nối các dữ liệu khi hiển thị

Định dạng dữ liệu khi hiển thị

- Dạng Boolean: **%b**
- Dạng ký tự: **%c**
- Dạng số nguyên: **%d**
- Dạng số thực: **%f**
- Dạng xâu ký tự: **%s**

Nhập dữ liệu từ bàn phím

- Khá phức tạp vì Java coi dữ liệu nhận được từ bàn phím là luồng vào.
- **Thực hiện**
  - **Đọc dữ liệu vào bộ đệm:**

```
BufferedReader br = new BufferedReader(
        new InputStreamReader(System.in));
```

  - **Chuyển dữ liệu từ bộ đệm vào xâu**

```
String inValue = br.readLine( );
```

  - **Chuyển dữ liệu từ xâu thành giá trị : sử dụng các lớp**
    - Giá trị kiểu int: Integer.parseInt(inValue)
    - Giá trị kiểu long:Long.parseLong(inValue)
    - Giá trị kiểu float: Float.parseFloat(inValue)
    - Giá trị kiểu double: Double.parseDouble(inValue)

Ví dụ

```
/** The Addition class calculates the sum of two numbers */
import java.io.*;
public class Addition {
/** The main method begins execution of Java application
*@param args: input parameter
*/
public static void main (String[] args) throws
    IOException{
String inputData;
BufferedReader br = new BufferedReader(new
    InputStreamReader(System.in));
System.out.print("Enter the 1st number:");
inputData = br.readLine();
int number1 = Integer.parseInt(inputData);
```

Ví dụ (tiếp)

```
    System.out.print(“Enter the 2nd number:”);
    inputData = br.readLine();
    int number2 = Integer.parseInt(inputData);
    int sum = number1 + number2;
    System.out.println(“The sum of two numbers: ” + sum);
  }
}
```

Ví dụ

- Chú thích tạo tài liệu Javadoc:

  /** Comment something */

- Sử dụng các gói thư viện được Java định nghĩa sẵn

  import somepakage

  - java.io: Thư viện xuất nhập dữ liệu

- Bỏ qua các ngoại lệ (lỗi) trong khi thực thi chương trình:

throwssomeException

  - IOException: ngoại lệ xuất hiện khi xuất nhập dữ liệu

Nhập dữ liệu – Lớp Scanner

- Được cung cấp bởi thư viện java.util
- Quét luồng dữ liệu người dùng nhập từ bàn phím và phân tách các giá trị có kiểu dữ liệu nguyên thủy hoặc xâu.
- Rất hữu dụng

```
import java.util.Scanner
...
   Scanner inputData = new Scanner(System.in);
   System.out.print(“Enter the 1st number:”);
   int number1 = inputData.nextInt();
   System.out.print(“Enter the 2nd number:”);
   int number2 = inputData.nextInt();
   int sum = number1 + number2;
   System.out.println(“The sum of two numbers: ” + sum);
```

ĐẠI NAM
UNIVERSITY

CẤU TRÚC RẼ NHÁNH

http://dainam.edu.vn

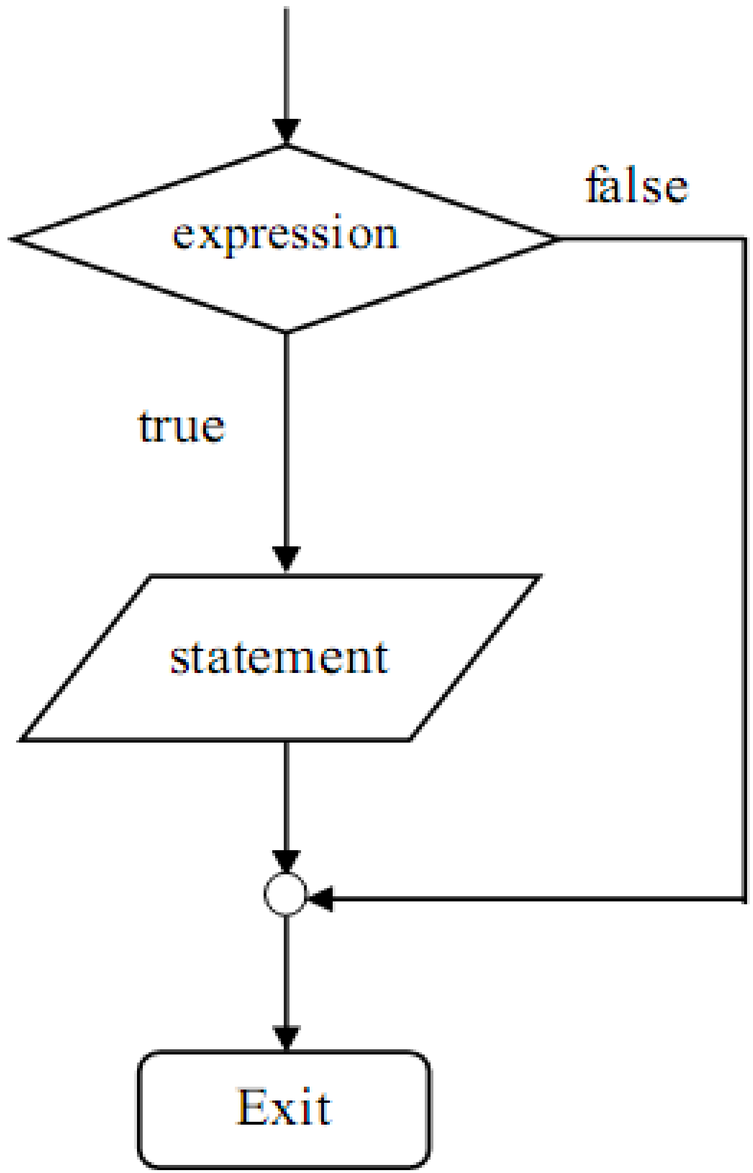
Cấu trúc if, if…else

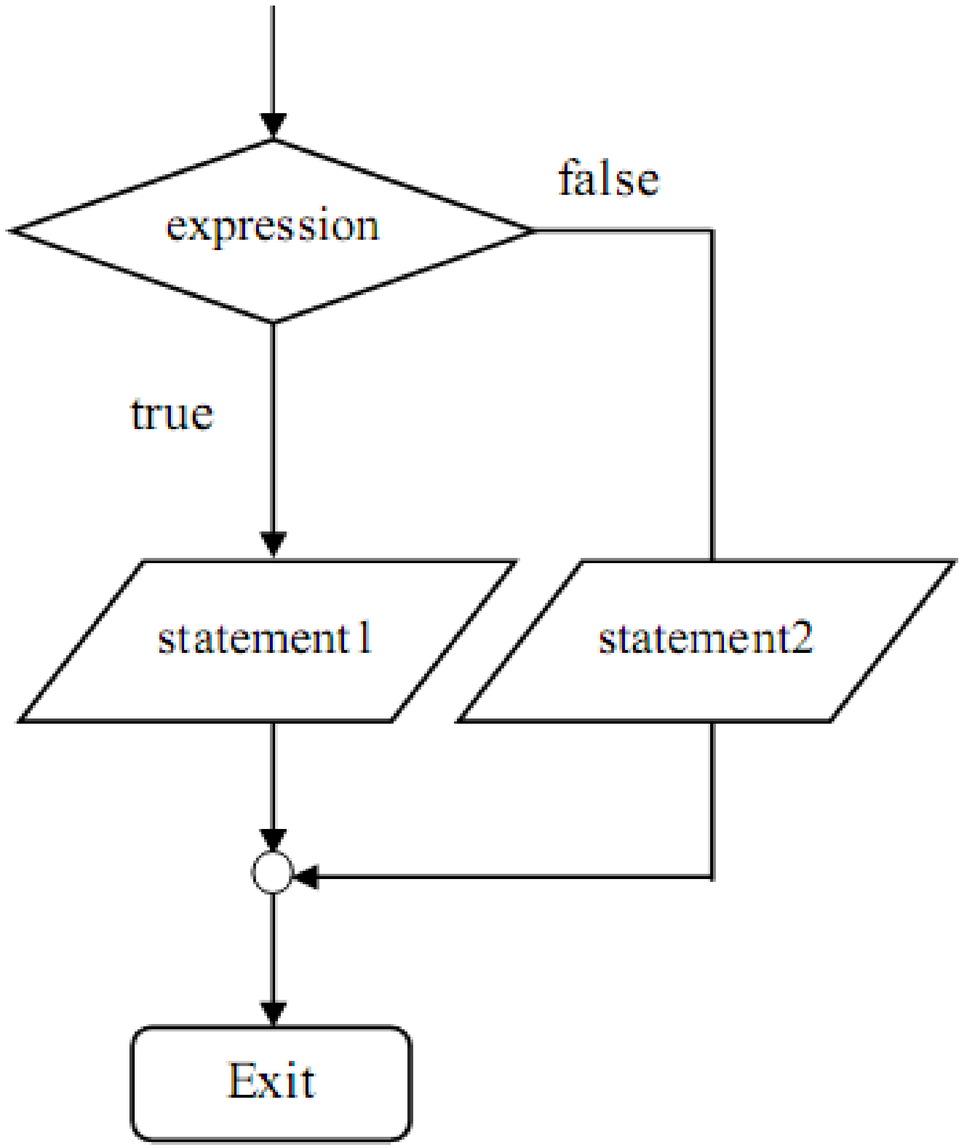
Cú pháp:

```
if (expression)
{
  statement;
}
```

```
if (expression)
{
  statement1;
}
else
{
  statement2;
}
```

Lưu đồ hoạt động





Cấu trúc if, if…else

- **Ví dụ 1**

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

- **Ví dụ 2**

```
if ((y%4 == 0 && y%100!=0) || y%400==0)
    System.out.println ("y la năm nhuận");
else
    System.out.println("y không là năm nhuận");
```

Cấu trúc 3 ngôi

- **Có dạng:**

<Điều kiện> **?** <Biểu thức 1> **:** <Biểu thức 2>

Nếu <Điều kiện> đúng thì <Biểu thức 1> thực hiện, ngược lại <Biểu thức 2> thực hiện

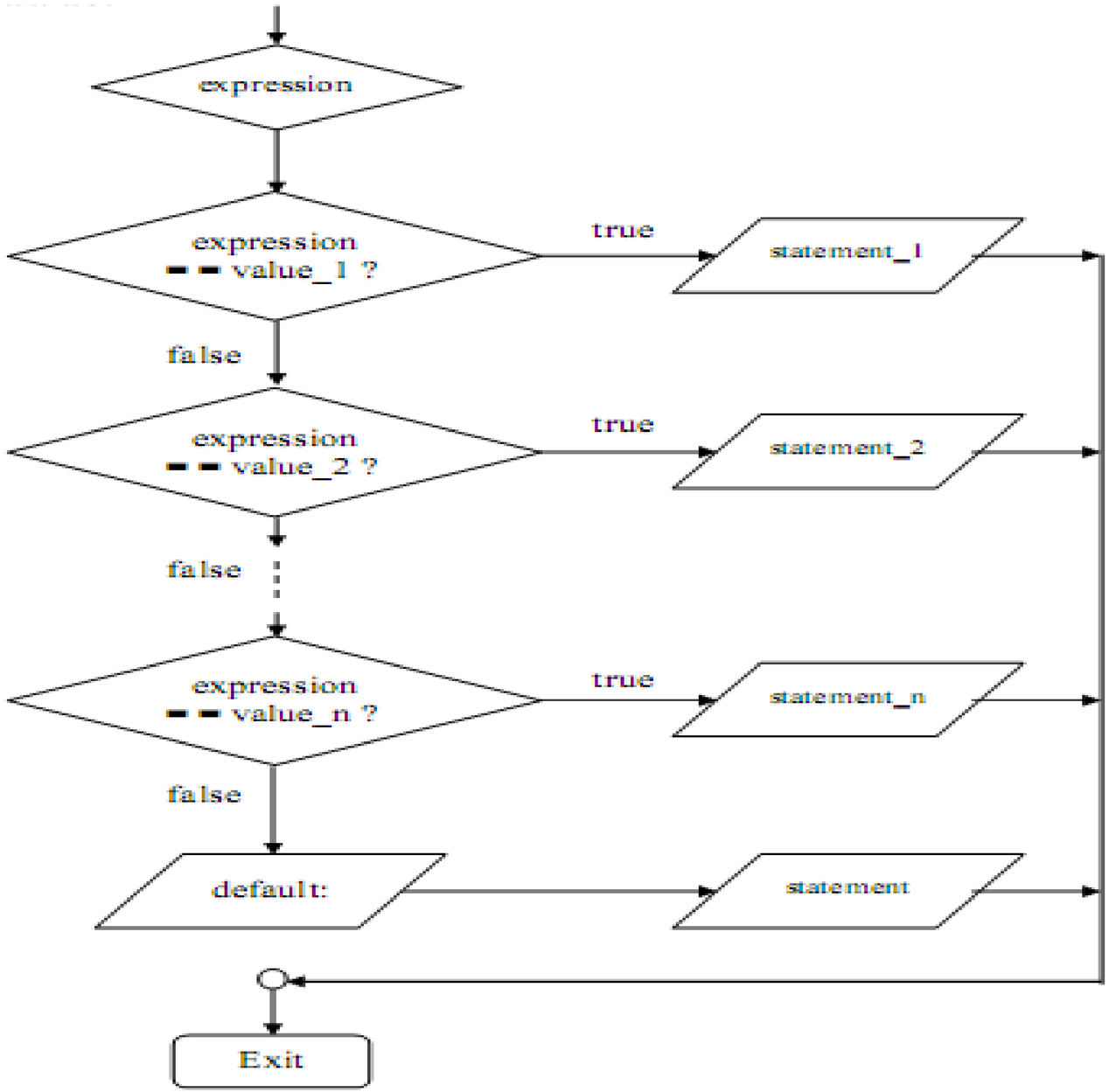
- Là dạng rút gọn của if…else
- Ví dụ

string a = (i % 2 == 0) ? “so chan” : “so le”

Cấu trúc switch

```
switch (biểu thức)
{
  case  n1:
            các câu lệnh 1;
            break ;
  case  n2:
            các câu lệnh 2;
            break ;
  ……
  case  nk:
            các câu lệnh k;
            break ;
  default:
          các câu lệnh;
           break;
}
```

Lưu đồ hoạt động



Ví dụ

```
switch(number){
  case 1:  System.out.print(“One”);
  break;
  case 2:  System.out.print(“Two”);
  break;
  case 3:  System.out.print(“Three”);
  break;
  default:  System.out.print(“I don’t know.”);
}
```

Switch – Nhóm giá trị

```
switch(month){
   case 1:
   case 3:
   case 5:
   case 7:
   case 8:
   case 10:
      case 12: System.out.print(“The month has 31 days”);
      break;
  case 4:
  case 6:
  case 9:
      case 11: System.out.print(“The month has 30 days”);
      break;
  default:  System.out.print(“The month has 28 or 29 days”);
}
```

ĐẠI NAM
UNIVERSITY

CẤU TRÚC LẶP

Cấu trúc while, do … while

Thực hiện lặp đi lặp lại các lệnh trong khối lệnh khi expression còn có giá trị true.

```
while (expression)
{
   statement;
}
```

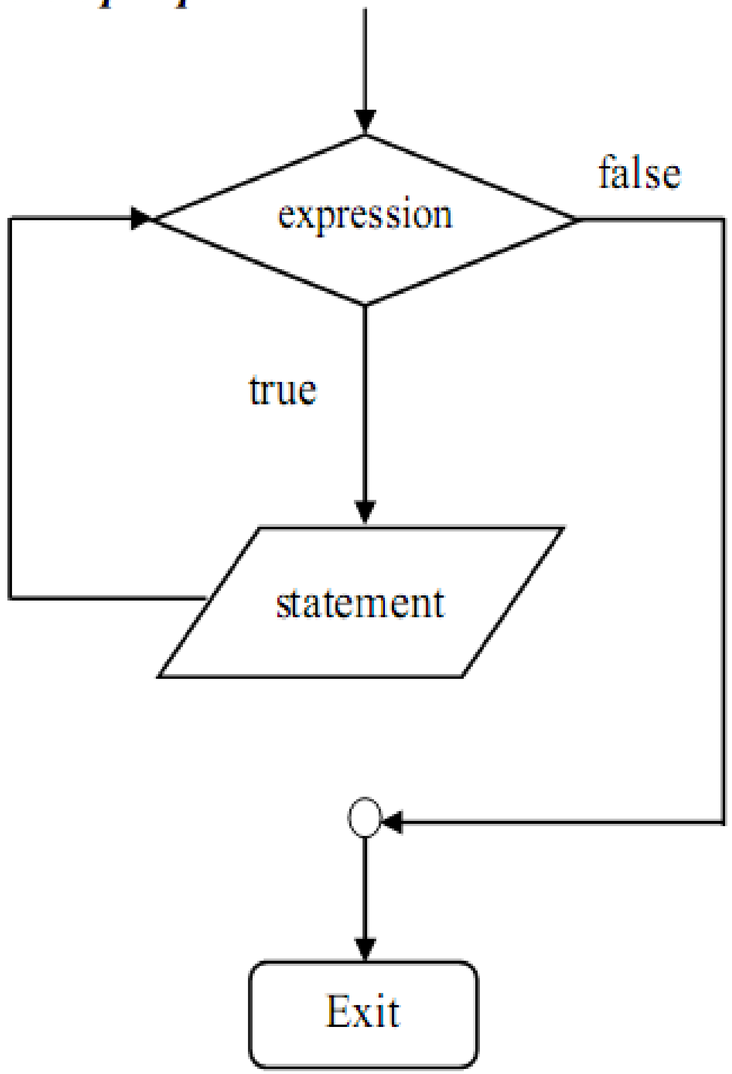
statement có thể không được thực hiện lần nào.

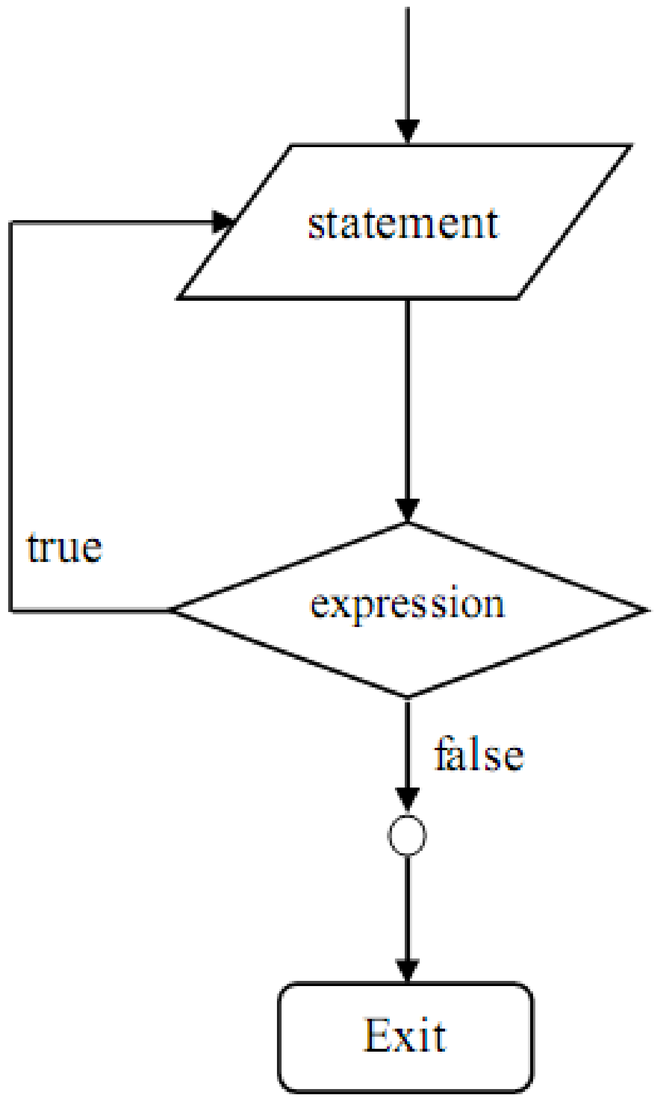
```
do
{
   statement;
}while (expression);
```

statement được thực hiện tối thiểu 1 lần.

Lưu đồ hoạt động

Lưu đồ cú pháp:





Ví dụ

- Sử dụng cấu trúc while

```
int n, i, factorial = 1;
i = 1;
while(i < = n){
   factorial *= i;
   i++;
}
```

- Sử dụng cấu trúc do...while

```
int n, i, factorial = 1;
i = 1;
do{
   factorial *= i;
   i++;
}while(i < = n);
```

Cấu trúc for

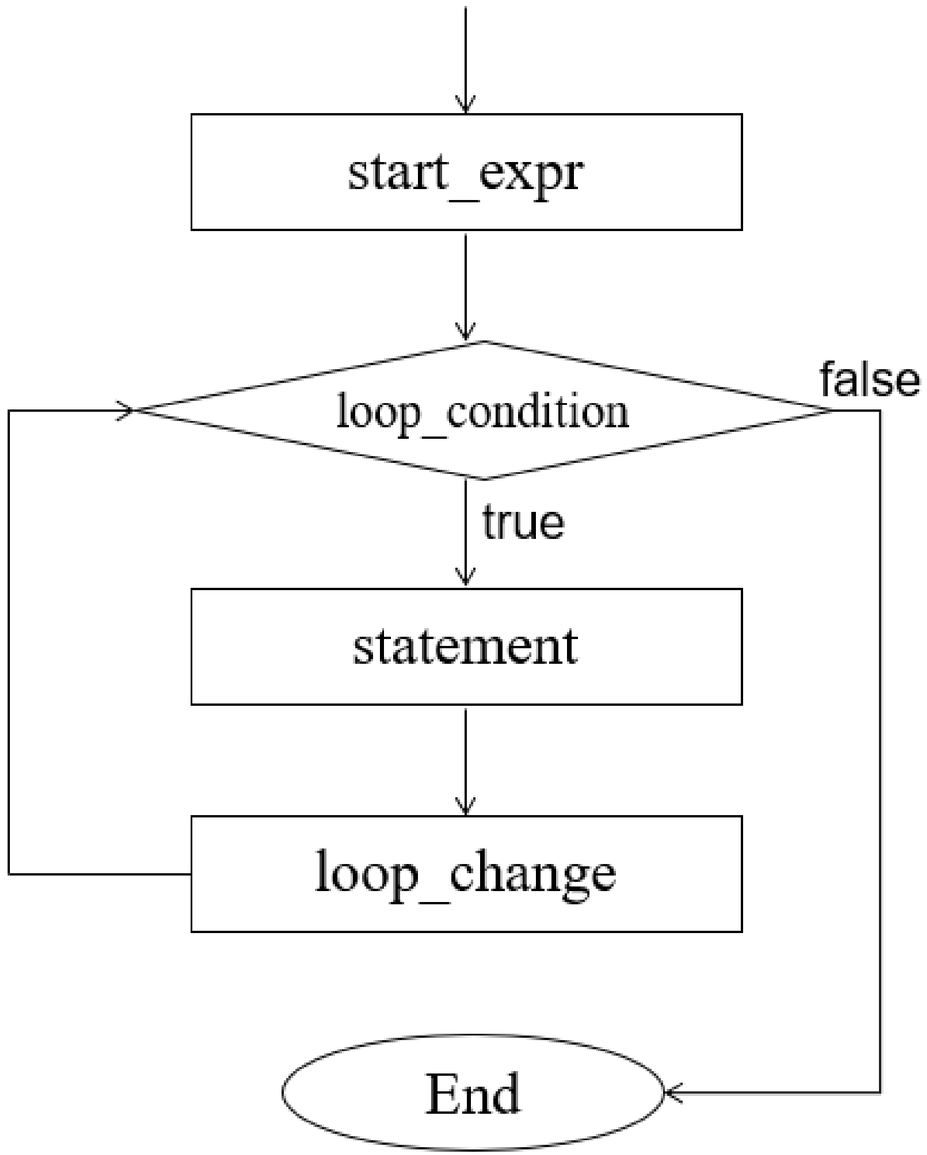
- **Cú pháp:**

```
for(start_expr; loop_condition; loop_change){
   statement;
}
```

- **Trong đó:**
  - start_expr: Biểu thức khởi tạo
  - loop_condition: Biểu thức điều kiện thực hiện vòng lặp
  - loop_change: Biểu thức thay đổi biến
- Ví dụ

```
int n, factorial = 1;
for (int i = 1; i < n; i++)
   factorial *= i;
```

Cấu trúc for


start_expr
loop_condition
false
true
statement
loop_change
End

Cấu trúc foreach

- **Cú pháp:**

```
for(declaration : expression){
   statement;
}
```

- **Trong đó:**
  - **Declaration:** Biến khối được khai báo
  - **Expression:** Expression này có thể là một biến mảng hoặc gọi phương thức mà trả về một mảng.
- Ví dụ:

```
int [] numbers = {10, 20, 30, 40, 50};
for(int x : numbers ){
        System.out.print( x );
        System.out.print(",");
}
```

Các lệnh thay đổi cấu trúc vòng lặp

- **continue**
  - Bỏ qua việc thực hiện các câu lệnh nằm sau lệnh continue trong thân vòng lặp.
  - Chuyển sang thực hiện một vòng lặp mới
- **break**
  - Thoát khỏi vòng lặp ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn.
- **Hai dạng:**
  - Không gán nhãn: Chuyển ra ngoài vòng lặp, thực hiện câu lệnh ngay sau vòng lặp
  - Gán nhãn: Chuyển ra ngoài vòng lặp, thực hiện câu lệnh tiếp theo sau vòng lặp được đánh dấu bởi nhãn

Ví dụ

```
int sum = 0;
outer: for(int i = 0; i < 10; i++){
  inner: for(int j = i; j < 10; j++){
      sum ++;
      if (j == 1) continue;
      if (j == 2) continue outer;
      if (j == 3) break;
      if (j == 4) break outer;
  } // terminate inner
} // terminate outer
System.out.println(“ sum = “ + sum);
```

ĐẠI NAM
UNIVERSITY

CHUỖI KÝ TỰ

String trong Java

Được xây dựng như là một lớp trong Java

- **Khai báo một đối tượng**

  String variable = new String(literalString);

  String varialbe = new String(char[] charArr)

  Hoặc đơn giản hơn:

  String variable = literalString;

- **Trong đó:**

  variable: biến

  literalString: Giá trị hằng xâu ký tự

  charArr: Mảng ký tự

- **Ví dụ:**

  String s = “HELLO”;

Các phương thức của String

- **Vấn đề:**

  Cần có các giá trị để phục vụ cho việc hiển thị và tính toán

- **Ví dụ:**
  - Tính chiều dài của chuỗi s
  - Nối chuỗi s1 vào chuỗi s
  - Lấy một ký tự tại vị trí index trong chuỗi s
  - So sánh hai chuỗi s1 và s2
  - Tìm vị trí xuất hiện đầu tiên của chuỗi s2 trong chuỗi s
- **Giải quyết:**

  Sử dụng các phương thức của String

Các phương thức của String

- char charAt(int index): trả về ký tự có chỉ số index, chỉ số bắt đầu từ 0
- int length(): trả về kích thước của xâu
- String toLowerCase(): chuyển thành chữ thường
- String toUpperCase(): chuyển thành chữ hoa
- int compareTo(String anotherString): so sánh hai xâu theo thứ tự từ điển, ưu tiên chữ thường. Trả về:
  - = 0: 2 xâu giống nhau
  - < 0: xâu nhỏ hơn anotherString
  - > 0: xâu lớn hơn anotherString
- int compareToIgnoreCase(String str): so sánh không kể chữ hoa, chữ thường

Các phương thức của String (tiếp)

- boolean equals (Object object): so sánh với một đối tượng bất kỳ, trong Java, một đối tượng bất kỳ đều có thể chuyển thành String
- boolean equalsIgnoreCase(String anotherString): so sánh với một đối tượng bất kỳ, không phân biệt chữ hoa chữ thường
- char[] toCharArray: chuyển xâu thành mảng ký tự
- String concat(String str): ghép nội dung str vào cuối xâu
- int indexOf(int ch): trả về chỉ số của ký tự đầu tiên có giá trị bằng ch
- int indexOf(String str): trả về vị trí của str trong xâu
- int lastIndexOf(String str)
- String substring(int index)

Các phương thức của String (tiếp)

- String replace (char oldChar, char newChar): trả về xâu mà tất cả ký tự oldChar trong xâu thành newChar
- String replace (String oldString, String newString)
- String replaceFirst (String oldString, String newString)
- String[] split (String regex): chia xâu thành các xâu con bởi xâu regex
- String trim(): trả về xâu được loại bỏ dấu cách ở đầu và cuối

Các phương thức của String (tiếp)

- static String copyValueOf(char[] data): trả về xâu có chứa các ký tự của mảng data
- static String copyValueOf(char[] data, int offset, int count): trả về xâu có chứa count ký tự từ chỉ số offset của mảng data
- static String format(): trả về xâu hiển thị giá trị của đối tượng bất kỳ theo định dạng
- static String valueOf(): trả về xâu chứa nội dung của một đối tượng nào đó

StringBuffer

String là kiểu bất biến:

- Đối tượng không thay đổi giá trị sau khi được tạo ra. Các chuỗi của lớp String được thiết kế để không thay đổi giá trị.
- Khi các chuỗi được ghép nối với nhau một đối tượng mới được tạo ra để lưu trữ kết quả → Ghép nối chuỗi thông thường rất tốn kém về bộ nhớ.

```
String s = "";
s += "Hello";
s += " ";
s += "World";
s += "!";
System.out.println(s);
```

StringBuffer

- Trong trường hợp phải làm việc với các chuỗi biến đổi (chuỗi dạng mutable) → Sử dụng StringBuffer
- Một chuỗi mà có thể sửa đổi được xem như là chuỗi dạng mutable. Các lớp StringBuffer và StringBuilder được sử dụng để tạo các chuỗi dạng mutable.
- Phương thức khởi tạo:
  - StringBuffer(): tạo chuỗi dung lượng mặc định 16
  - StringBuffer(String str)
  - StringBuffer(int capacity)

Các phương thức của StringBuffer

- append(String s): được sử dụng để phụ thêm (append) chuỗi đã cho với chuỗi này.
- insert(int offset, String s): được sử dụng để chèn chuỗi đã cho với chuỗi này tại vị trí đã cho.
- replace(int startIndex, int endIndex, String str): được sử dụng để thay thế chuỗi từ chỉ mục ban đầu startIndex và chỉ mục kết thúc endIndex đã cho.
- delete(int startIndex, int endIndex): được sử dụng để xóa chuỗi từ chỉ mục startIndex và endIndex đã cho.
- reverse(): được sử dụng để đảo ngược chuỗi.
- capacity(): được sử dụng để trả về dung lượng capacity hiện tại.

Các phương thức của StringBuffer

- ensureCapacity(int minimumCapacity): được sử dụng để bảo đảm rằng capacity ít nhất bằng với minimum đã cho.
- charAt(int index): được sử dụng để trả về ký tự tại vị trí đã cho.
- length(): được sử dụng để trả về độ dài của chuỗi (chẳng hạn như tổng số ký tự).
- substring(int beginIndex): được sử dụng để trả về chuỗi con từ chỉ mục bắt đầu beginIndex đã cho.
- substring(int beginIndex, int endIndex): được sử dụng để trả về chuỗi con từ beginIndex đến endIndex đã cho.

Ví dụ

```
StringBuffer buffer = new StringBuffer(15);
buffer.append("This is ") ;
buffer.append("String") ;
buffer.insert(7," a") ;
buffer.append('.');
System.out.println(buffer.length()); // 17
System.out.println(buffer.capacity()); // 32
String output = buffer.toString();
System.out.println(output); // "This is a String."
```

StringBuilder

- String là không thể thay đổi (immutable), và không cho phép có class con.
- StringBuilder được sử dụng để tạo chuỗi có thể thay đổi (mutable)
- Phương thức khởi tạo:
  - StringBuilder(): tạo chuỗi dung lượng mặc định 16
  - StringBuilder(String str)
  - StringBuilder(int capacity)

Các phương thức của StringBuilder

- append(String s): được sử dụng để nối thêm các chuỗi được chỉ định với chuỗi này.
- insert(int offset, String s): được sử dụng để chèn chuỗi chỉ định với chuỗi này tại vị trí quy định.
- replace(int startIndex, int endIndex, String str): được sử dụng để thay thế chuỗi từ vị trị startIndex đến endIndex bằng chuỗi str.
- delete(int startIndex, int endIndex): được sử dụng để xóa chuỗi từ vị trí startIndex đến endIndex.
- reverse(): được sử dụng để đảo ngược chuỗi.
- capacity(): được sử dụng để trả về dung lượng hiện tại.

Các phương thức của StringBuilder

- ensureCapacity(int minimumCapacity): được sử dụng để đảm bảo dung lượng ít nhất bằng mức tối thiểu nhất định.
- charAt(int index): được sử dụng trả về ký tự tại vị trí quy định.
- length(): được sử dụng trả về chiều dài của chuỗi nghĩa là tổng số ký tự.
- substring(int beginIndex): được sử dụng trả về chuỗi con bắt đầu từ vị trí được chỉ định.
- substring(int beginIndex, int endIndex): được sử dụng trả về chuỗi con với vị trí bắt đầu và vị trí kết thúc được chỉ định.

StringTokenizer

- Phân tách xâu ký tự thành các xâu phần tử theo dấu hiệu phân cách (delimiter)
  - Delimiter: mặc định là dấu cách trắng \s
  - Có thể định nghĩa lại trong phương thức khởi tạo
- Phương thức khởi tạo
  - Mặc định: StringTokenizer(String input)
  - Định nghĩa lại dấu hiệu phân cách

    StringTokenizer(String input, String delimiter)
- nextToken(): trả lại xâu phần tử tiếp theo
- hasMoreTokens(): trả về false nếu không còn xâu phần tử
- countTokens(): trả về số xâu phần tử tách được

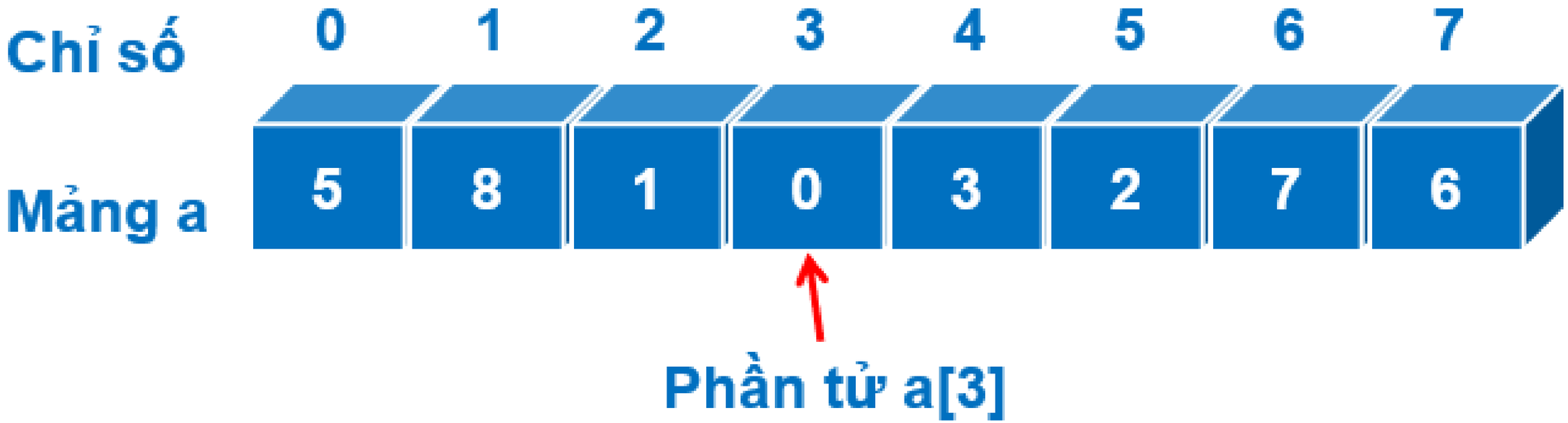
ĐẠI NAM
UNIVERSITY

MẢNG

http://dainam.edu.vn

Mảng là gì?

- Mảng là tập hợp hữu hạn các phần tử cùng kiểu dữ liệu
- Số lượng phần tử xác định khi khai báo, không đổi



Lợi ích của việc sử dụng mảng

- Mảng là cách tốt nhất cho phép quản lý nhiều phần tử dữ liệu có cùng kiểu tại cùng một thời điểm.
- Mảng tạo ra sự tối ưu trong việc quản lý bộ nhớ so với việc sử dụng nhiều biến cho cùng một chức năng.
- Bộ nhớ có thể được gán cho mảng chỉ khi mảng thực sự được sử dụng. Do đó, bộ nhớ không bị tiêu tốn cho mảng ngay khi bạn khai báo mảng.

Khai báo và khởi tạo

- **Cú pháp**

  DataType[] array = new DataType[size];
  DataType array[] = new DataType[size];
  DataType[] array = new DataType[]{value1, value2,...,valueN};
  DataType[] array = {value1, value2,...,valueN};

- **Trong đó:**

  array: Biến mảng
  size: Số phần tử trong mảng, có thể sử dụng giá trị, biến, biểu thức
  value1,... valueN : các giá trị khởi tạo

- **Ví dụ:**

  int[] a = new int[5];

  int a[] = new int[5];

  int[] a = new int[]{2,10,4,8,5};

  int[] a = {2, 10, 4, 8, 5};

Mảng nhiều chiều

- Được coi là mảng của các mảng:
  - Mảng 2 chiều: mảng các mảng 1 chiều
  - Mảng 3 chiều: mảng của các mảng 2 chiều
- Khai báo mảng 2 chiều:

```
DataType[][] array = new DataType[size1][size2];
DataType array[][] = new DataType[size1][size2];
DataType[][] array = {value11,value12,...,value1N}
                     {value21,value22,...,value2N};
```

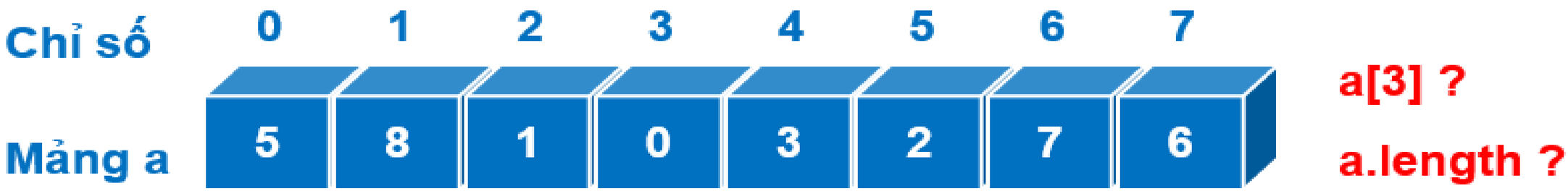
- Tương tự cho mảng nhiều chiều khác

Thao tác với mảng

- Thuộc tính length: Lấy số phần tử của mảng (chiều dài mảng)
- Truy xuất giá trị 1 phần tử của mảng 1 chiều:

TênMảng[chỉ_số]

  - Vị trí của 1 phần tử trong mảng bắt đầu từ 0
  - chỉ_số có giá trị từ 0 → số phần tử - 1



- Truy xuất giá trị 1 phần tử của mảng nhiều chiều:

TênMảng[chỉ_số1][chỉ_số2]...[chỉ_số N]

Duyệt mảng

- Duyệt và xử lý từng phần tử của mảng:

```
for (int i = 0; i < TênMảng.length; i++)
{
        // Xử lý trên phần tử TênMảng[i]
}
```

- Ví dụ: Duyệt và xuất mảng:

```
int[] a = {1,2,3,4,5};
for (int i = 0; i < a.length; i++)
        System.out.println(a[i]);
```

Hạn chế của mảng

- Mảng có kích cỡ và số chiều cố định nên khó khăn trong việc mở rộng ứng dụng.
- Các phần tử được đặt và tham chiếu một cách liên tiếp nhau trong bộ nhớ nên khó khăn cho việc xóa một phần tử ra khỏi mảng.

Lớp Arrays

- **Vấn đề:**

  Cần xử lý mảng một cách nhanh chóng

- **Ví dụ:**
  - Sắp xếp mảng số nguyên tăng dần
  - So sánh hai mảng số nguyên array1 và array
  - Gán giá trị cho các phần tử trong mảng array1
  - Sao chép mảng array1 sang mảng array2
- **Giải quyết:**

  Sử dụng các phương thức của lớp Arrays

Lớp Arrays

```
// import java.util.Arrays;
int array[] = { 2, 5, -2, 6, -3, 8, 0, 7, -9, 4 };
int array1[] = { 2, 5, 6, -3, 8};
// Sắp xếp mảng số nguyên
Arrays.sort(array);
// So sánh hai mảng số nguyên array1 và array
array1.equals(array);
// Gán giá trị cho các phần tử trong mảng array1
Arrays.fill(array1, 10); // 10, 10, 10, 10, 10
// Sao chép mảng array1 sang array2
int[] arr2 = Arrays.copyOf(arr1, 6); // 10 10 10 10 10 0
```