IS216 - LẬP TRÌNH JAVA

Chương 1
TỔNG QUAN
OVERVIEW

Giảng viên: Tạ Việt Phương
E-mail: phuongtv@uit.edu.vn

1

Nội dung

1. Giới thiệu Java
2. Lịch sử phát triển Java
3. Các phiên bản Java
4. Các đặc trưng của Java
5. JDK (Java Development Kit) và các packet API
6. Các ứng dụng viết bằng Java
7. Các khái niệm cơ bản trong Java

2

1. GIỚI THIỆU JAVA

3

Giới thiệu Java

- Java là ngôn ngữ cấp cao (high-level), dựa trên lớp (class-based), là ngôn ngữ lập trình hướng đối tượng (object-oriented), được thiết kế để có càng ít phụ thuộc triển khai (implementation dependencies) càng tốt.
- Là ngôn ngữ lập trình có mục đích chung cho phép các nhà phát triển ứng dụng viết một lần, chạy ở mọi nơi (WORA - write once, run anywhere), Java cho phép phát triển các ứng dụng mà không phụ thuộc nhiều vào nền tảng.

4

Giới thiệu Java

- Các đặc điểm cốt lõi
  - Hướng đối tượng.
  - Độc lập nền tảng.
  - Bảo mật cao.
  - Hỗ trợ đa luồng.
  - Dễ dàng sử dụng.

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


5

Giới thiệu Java

- Xây dựng trên nền tảng C và C++
- Sử dụng các cú pháp của C và các đặc trưng đối tượng của C++
- Ngôn ngữ độc lập với thiết bị
- Không phụ thuộc vào trình biên dịch của CPU
- Java là ngôn ngữ lập trình hướng đối tượng

6

Giới thiệu Java

- Là ngôn ngữ lập trình vừa **biên dịch**, vừa **thông dịch**. Đầu tiên mã nguồn được được biên dịch bằng công cụ **JAVAC** để chuyển thành dạng **bytecode.** Sau đó được thực thi trên từng loại máy cụ thể nhờ chương trình thông dịch
- Chạy trên bất kỳ phần cứng, không phụ thuộc vào hệ điều hành
- Trở thành ngôn ngữ lập trình Internet
- Xây dựng các chương trình điều khiển thiết bị cho điện thoại di động, PDA

7

7

2. LỊCH SỬ PHÁT TRIỂN

8

8

Lịch sử phát triển

- 1991: James Gosling (cùng Mike Sheridan và Patrick Naughton) tại Sun Microsystems phát triển ngôn ngữ OAK nhằm mục đích viết phần mềm điều khiển (phần mềm nhúng) cho các sản phẩm gia dụng. Tên ban đầu là OAK với cú pháp kiểu C/C++
- 1996: Sun đã phát hành bản triển khai công khai đầu tiên tên Java 1.0 (theo tên loại cà phê đến từ Indonesia)

9

9

Lịch sử phát triển

- 2006: Sun đã phát hành phần lớn máy ảo Java (JVM) của mình dưới dạng phần mềm mã nguồn mở và miễn phí (FOSS), theo các điều khoản của Giấy phép Công cộng GNU (GPL).
- 2009-2010: Tập đoàn Oracle mua lại Sun Microsystems
- Hiện nay: Phiên bản mới nhất là Java 23 và 21 (LTS), cập nhật nhiều tính năng mới như Virtual Threads, Pattern Matching.

10

10

3. CÁC PHIÊN BẢN

11

11

Phiên bản

- **Gồm 4 phiên bản:**
  - **Java Card** cho smart-cards.
  - **Java Platform, Micro Edition (Java ME)** – nhắm mục tiêu các môi trường có tài nguyên hạn chế. Đã lỗi thời, hiện chỉ còn Java ME 8 cho IoT.
  - **Java Platform, Standard Edition (Java SE) –** nhắm mục tiêu môi trường máy trạm (workstation)
  - **Java Platform, Enterprise Edition (Java EE) –** nhắm mục tiêu đến các môi trường Internet hoặc doanh nghiệp. Java EE đã đổi thành Jakarta EE từ năm 2018.
  - Chủ yếu sử dụng Java SE trong chương trình học.

12

12

4. CÁC ĐẶC TRƯNG

13

13

Các đặc trưng của Java

1. Đơn giản (Simple)
2. Hướng đối tượng (Object-Oriented)
3. Phân tán (Distributed)
4. Mạnh mẽ (Robust)
5. Bảo mật (Secure)
6. Độc lập phần cứng và hệ điều hành (Architecture-Neutral)
7. Tính di động (Portable)
8. Thông dịch (Interpreted)
9. Hiệu suất cao (High-Performance)
10. Đa luồng (Multithreaded)
11. Động (Dynamic)

14

14

1. Đơn giản

So even though we found that C++ was unsuitable, we designed Java as closely to C++ as possible in order to make the system more comprehensible. Java omits many rarely used, poorly understood, confusing features of C++ that, in our experience, bring more grief than benefit.

Another aspect of being simple is being small. One of the goals of Java is to enable the construction of software that can run stand-alone on small machines. The size of the basic interpreter and class support is about 40K; the basic standard libraries and thread support (essentially a self-contained microkernel) add another 175K

15

15

1. Đơn giản

- Loại bỏ các đặc trưng phức tạp của C và C++ như :
  - Con trỏ
  - Định nghĩa chồng toán tử (operator overloading)
  - Không sử dụng lệnh goto
  - Không định nghĩa file header.h
  - Không dùng struct
  - Không dùng union

16

16

2. Hướng đối tượng

- Thiết kế code xoay quanh mô hình hướng đối tượng
- Dữ liệu và các phương thức của một đối tượng được gói trong một lớp của java
  - Đảm bảo tính bảo mật
  - Thừa kế
  - Đa hình…vv
  - Không cho phép đa thừa kế (multi-inheritance) mà sử dụng các giao diện (interface).

17

17

3. Phân tán

- Java có một thư viện rộng lớn các routines để đáp ứng với các giao thức TCP/IP như HTTP và FTP. Các ứng dụng Java có thể mở và truy cập các đối tượng trên Net thông qua các URL dễ dàng giống như khi truy cập một hệ thống tập tin cục bộ.
- Ngày nay, người ta coi điều này là hiển nhiên - nhưng vào năm 1995, việc kết nối với máy chủ web từ chương trình C++ hoặc Visual Basic là một công việc khá khó khăn.
- Java hỗ trợ **hệ thống phân tán thông qua RMI (Remote Method Invocation)**.

18

18

3. Phân tán

- Các công nghệ phân tán hiện đại:
  - Spring Cloud: Dùng trong kiến trúc Microservices.
  - Jakarta EE (JMS, JAX-RS): Hỗ trợ giao tiếp từ xa.
  - Kubernetes: Chạy ứng dụng Java trong môi trường cloud.

19

19

4. Mạnh mẽ

- Phải khai báo kiểu dữ liệu tường minh trước
- Không sử dụng con trỏ và các phép toán trên con trỏ
- Không phải bận tâm về việc cấp phát và giải phóng vùng nhớ. Java **quản lý bộ nhớ tự động bằng Garbage Collector**.
- Cơ chế bẫy lỗi của java giúp đơn giản quá trình xử lý lỗi và phục hồi sau lỗi

> *Java is intended for writing programs that must be reliable in a variety of ways. Java puts a lot of emphasis on early checking for possible problems, later dynamic (runtime) checking, and eliminating situations that are error-prone. . . . The single biggest difference between Java and C/C++ is that Java has a pointer model that eliminates the possibility of overwriting memory and corrupting data.*

20

20

5. Bảo mật

- Java cung cấp nhiều mức để kiểm soán tính an toàn khi thực thi chương trình
  - Mức 1: Dữ liệu và phương thức được gói trong lớp
  - Mức 2: Trình biên dịch kiểm soát để đảm bảo mã an toàn và tuân theo các nguyên tắc của Java
  - Mức 3: Được đảm bảo bởi trình thông dịch, chúng kiểm tra xem bytecode có đảm bảo các quy tắc an toàn trước khi thực thi
  - Mức 4: Kiểm tra việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống

21

21

6. Độc lập với kiến trúc

- Độc lập với phần cứng và HĐH. Viết một lần, chạy mọi nơi (Write Once, Run Anywhere - WORA).
- Chương trình viết ở một máy nhưng có thể chạy bất kỳ ở đâu. Chúng được thể hiện 2 mức:
  - Mã nguồn
  - Mức nhị phân
- Ở mức mã nguồn
  - Kiểu dữ liệu trong Java nhất quán cho tất cả các hệ điều hành và phần cứng khác nhau
  - Java có riêng một thư viện các lớp cơ sở

22

22

6. Độc lập với kiến trúc

- Tính độc lập ở mức nhị phân:
  - Một chương trình đã biên dịch có thể chạy trên nhiều nền (phần cứng, hệ điều hành) khác mà không cần dịch lại mã nguồn
  - Cần có phần mềm máy ảo java hoạt động như trình thông dịch tại máy thực thi

23

23

7. Tính di động

- Bản thân môi trường Java có thể dễ dàng chuyển sang các nền tảng khác, kiến trúc và hệ điều hành mới.
- Chương trình ứng dụng viết bằng ngôn ngữ Java chỉ cần chạy được trên máy ảo Java là có thể chạy được trên bất kỳ máy tính, hệ điều hành nào có máy ảo Java. “Viết một lần, chạy mọi nơi” (Write Once, Run Anywhere).

24

24

8. Thông dịch

- Biên dịch (compiled) và thông dịch (interpreted)
  - Thông dịch là khi chạy chương trình, ngôn ngữ mới được dịch sang ngôn ngữ máy và thực thi
  - Biên dịch là trước khi chạy, chương trình sẽ dịch toàn bộ thành mã máy rồi mới tiến hành thực thi.
- Trình biên dịch (compiler) và trình thông dịch (interpreter) là các loại trình dịch ngôn ngữ chuyển đổi các đoạn mã lập trình sang ngôn ngữ máy.

25

25

8. Thông dịch

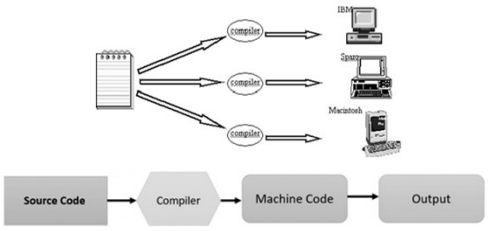
- Ngôn ngữ thông dịch sẽ dễ hiện thực hơn do bỏ qua việc kiểm tra lỗi và tối ưu code thường được thực hiện trong quá trình compiled. Đồng thời hỗ trợ hoạt động đa nền tảng, mã nguồn có thể thực thi mọi nơi mọi lúc mà không cần biên dịch.
- Nhược điểm của thông dịch là:
  - Độ tin cập thấp hơn do không qua bước check syntax tại quá trình complier
  - Tốc độ thực thi chậm hơn đáng kể so với các ngôn ngữ trình biên dịch
  - Dễ bị lọt lộ dịch ngược code

26

26

8. Thông dịch

- Cách biên dịch truyền thống



27

27

8. Thông dịch

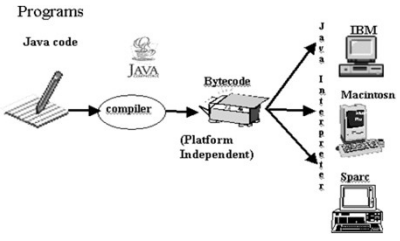
- Cách biên dịch truyền thống
  - Các chương trình viết bằng C, C++,… trình biên dịch sẽ chuyển tập lệnh thành mã máy (machine code), hay lệnh của CPU. Những lệnh này phục thuộc vào CPU trên trên máy thực thi.
  - Với mỗi nền phần cứng khác nhau thì sẽ có một trình biên dịch khác nhau.
  - Dịch lại khi muốn chạy trên nền phần cứng khác khác

28

28

8. Thông dịch

- Chương trình dịch Java



29

29

8. Thông dịch

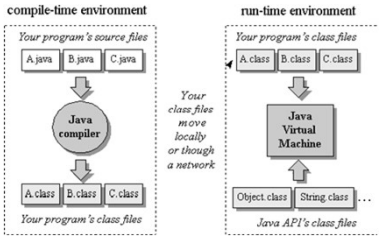
- **Chương trình dịch Java**
- Môi trường phát triển của java gồm 2 phần:
  - Trình biên dịch (Javac)
  - Trình thông dịch
- Trình biên dịch (Javac) chuyển mã nguồn thành dạng bytecode
- Trình thông dịch (máy ảo java)
  - Máy ảo Java chuyển bytecode thành mã CPU

30

30

8. Thông dịch

- **Chương trình dịch Java**



31

31

8. Thông dịch

- **Máy ảo Java (JVM- Java virtual Machine)**
  - Là 1 phần mềm tập hợp các lệnh logic để xác định các hoạt động của máy tính (hệ điều hành thu nhỏ)
  - Trình biên dịch chuyển mã nguồn thành các tập lệnh của máy ảo
  - Trình thông dịch trên mỗi máy chuyển tập lệnh thành chương trình thự thi bằng cách
    - Nạp file .class
    - Quản lý bộ nhớ
    - Dọn rác

32

32

8. Thông dịch

- Mã nguồn được biên dịch thành Java bytecode; sau đó được thông dịch trên JVM thành các mã lệnh thực thi bởi trình thông dịch Just-In-Time (JIT)
- **Trình thông dịch "Just In Time-JIT"** Compiler
  - JIT (Just In Time Compiler) là một trình biên dịch tại thời gian chạy của Java Virtual Machine (JVM) giúp tăng tốc độ thực thi mã Bytecode của Java
  - JIT giúp chuyển đổi Bytecode Java thành mã máy (Machine Code) ngay khi chương trình đang chạy, thay vì phải thông dịch từng lệnh một như cách thức hoạt động của Java Interpreter..
  - Tối ưu: Giảm chi phí dịch mã lặp lại, vì chỉ dịch những đoạn mã thực sự cần thiết. Tối ưu hóa cấu trúc mã lệnh, giúp chương trình chạy nhanh hơn.

33

33

8. Thông dịch

- Java không hoàn toàn là ngôn ngữ thông dịch, vì sử dụng JIT Compiler.
- JIT giúp Java chạy nhanh hơn bằng cách dịch Bytecode thành mã máy

34

34

8. Thông dịch

- **Ví dụ: chương trình Hello World**

○ Dùng Notepad soạn thảo đoạn lệnh bên dưới và lưu lại với tên HelloWorld.java

```
import java.io.*;
class HelloWorld
{
public static void main(String args[])
{
    System.out.print("Hello Class");
}
}
```

Khai báo thư viện java.io

Định nghĩa lớp tên **"HelloWorld"**

Bắt đầu đoạn lệnh

Phương thức **main**

Xuất ra Console thông báo

Kết thúc đoạn lệnh

35

35

8. Thông dịch

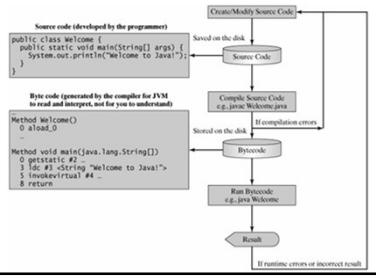
- Java Compiler (javac) biên dịch mã Java thành Bytecode (.class file).
- JVM tải Bytecode và thực thi bằng bộ thông dịch (Interpreter).
- JIT Compiler phân tích mã và phát hiện các đoạn code hay được thực thi nhiều lần (Hot Code).
- JIT Compiler dịch đoạn Bytecode này thành mã máy (Native Code) và lưu vào bộ nhớ cache.
- JVM sử dụng mã đã biên dịch để thực thi nhanh hơn trong các lần tiếp theo.

36

36

8. Thông dịch

- **Chương trình chạy Java**



37

37

9. Hiệu suất cao

- Java nhanh hơn các ngôn ngữ lập trình thông dịch truyền thống khác vì mã Java Bytecode là "gần" với mã gốc. Nó vẫn còn chậm hơn một chút so với ngôn ngữ được biên dịch (C++). Java là một ngôn ngữ thông dịch, đó là lý do tại sao nó chậm hơn các ngôn ngữ được biên dịch.
- Java cải thiện hiệu suất với Graal JIT, Virtual Threads (Java 21+)
- Java 21 hỗ trợ Profile-Guided Optimization (PGO): **tối ưu hóa hiệu suất dựa trên dữ liệu thực tế** về cách chương trình chạy

38

38

10. Đa luồng

- Chương trình Java đa luồng (Multithreading) để thực thi cách công việc song song
- Chương trình Java sử dụng kỹ thuật đa luồng để thực thi các công việc đồng thời, đa luồng cho phép nhiều tiến trình có thể chạy song song cùng một thời điểm và tương tác với nhau.
- Từ Java 21, có Virtual Threads, giúp tạo nhiều thread một cách tối ưu, giảm tải bộ nhớ, không bị hạn chế bởi số lượng CPU cores. Ứng dụng thực tế: Thích hợp cho ứng dụng web, server-side, xử lý đồng thời nhiều request.

39

39

11. Động

- Java là mã nguồn mở. Kiểm soát truy cập lúc chạy giúp liên kết mã động
- Java được thiết kế như một ngôn ngữ động để đáp ứng cho những môi trường mở. Các chương trình Java bổ sung các thông tin cho các đối tượng tại thời gian thực thi. Điều này cho phép khả năng liên kết động các mã.
- Hỗ trợ lập trình đa ngôn ngữ với GraalVM (chạy Java, Python, JS, Ruby)

40

40

5. CÁC THÀNH PHẦN TRONG JDK

41

JDK

Name	Acronym	Explanation
Java Development Kit	JDK	The software for programmers who want to write Java programs
Java Runtime Environment	JRE	The software for running Java programs, without development tools. You do not want that.
Standard Edition	SE	The Java platform for use on desktops and simple server applications. You want that.
Micro Edition	ME	The Java platform for use on small devices.
OpenJDK	–	A free and open source implementation of Java SE.
Hotspot	–	The "just in time" compiler developed by Oracle. If asked, choose this one.
OpenJ9	–	Another "just in time" compiler developed by IBM.
Long Term Support	LTS	A release that is supported for multiple years, unlike the six-month releases that showcase new features. Choose the latest LTS release.

42

42

JDK

- **JDK:** Java Development Kit
- Bộ công cụ phát triển phần mềm Java (JDK) là một bản phân phối Công nghệ Java của Tập đoàn Oracle. Nhằm triển khai JLS (Java Language Specification) và JVMS (Java Virtual Machine Specification), và cung cấp Phiên bản tiêu chuẩn (SE- Standard Edition) của Giao diện lập trình ứng dụng Java (API - Application Programming Interface). Kể từ khi ngôn ngữ Java ra đời, JDK là bộ phát triển phần mềm thông dụng nhất cho Java.

43

43

JDK

- Các phiên bản:

JDK 1.0	- 1996			
JDK 1.1	- 1997			
J2SE 1.2	- 1998			
J2SE 1.3	- 2000			
J2SE 1.4	- 2002			
…		…		
Java SE 18	- 2022	Java SE 21	- 2023	LTS
Java SE 19	- 2022	Java SE 22	- 2024	Hiện đã có 22.0.2
Java SE 20	- 2023	Java SE 23	- 2024	Hiện đã có 23.0.2

44

44

Các packet của Java code API

- **Java.lang**
  - Lớp quan trọng nhất của java
  - Gồm các kiểu dữ liệu cơ bản, ký tự, số nguyên
  - Chứa tập lệnh nhập xuất chuẩn
  - Chứa 1 số lớp quan trọng như: String, StringBuffer
- **Java.applet**
  - Package nhỏ nhất chứa 1 mình lớp Applet
  - Các Applet nhúng trong trang web, hay chạy trong Appletviewer đều thừa kế lớp này

45

45

Các packet của Java code API

- **Java.awt (abstract Window Tookit)**
  - Chứa các lớp tạo giao diện đồ họa
  - Một số lớp bên trong: Button, GridBagLayout, Graphics
- **Java.io**
  - Cung cấp thư viện vào ra chuẩn
- **Java.util**
  - Cung cấp một số công cụ hữu ích như: Date, hashtable, vector, stringTokenizer

46

46

Các packet của Java code API

- **Java.net**
  - Cung cấp khả năng giao tiếp từ xa
  - Cho phép tạo và kết nối tới Socket, URL
- **Java.awt.event**
  - Chứa các lớp, giao diện dùng để xử lý các sự kiện như bàn phím, chuột
- **Java.rmi**
  - Công cụ gọi hàm từ xa

47

47

Các packet của Java code API

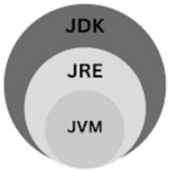
- **Java.security**
  - Cung cấp các công cụ cần thiết để mã hóa dữ liệu
  - Đảm bảo tính an toàn khi dữ liệu truyền đi giữa client và Server
- **Java.sql**
  - Package này chứa Java DataBase Connectivity (JDBC)
  - Dùng để truy xuất cơ sở dữ liệu quan hệ: SQL server, Oracle, MySQL

48

48

Mô hình hoạt động của JVM, JDK, JRE

- JVM (Java Virtual Machine): Máy ảo Java, thực thi mã bytecode.
- JRE (Java Runtime Environment): Chứa JVM + thư viện hỗ trợ để chạy ứng dụng Java.
- JDK (Java Development Kit): Bao gồm JRE + trình biên dịch javac + công cụ phát triển.



49

49

6. CÁC ỨNG DỤNG VIẾT BẰNG JAVA

50

50

Các kiểu chương trình Java

- Ứng dụng Console
- Ứng dụng Applet (đã lỗi thời)
- Ứng dụng Destop
- Ứng dụng Web
- Ứng dụng nhúng

51

51

Ứng dụng Console

- Là loại ứng dụng nhập xuất theo kiểu văn bản thông qua màn hình console tương tự như màn hình Console của hệ điều hành MS-DOS - (CLI - Command Line Interface), không có giao diện đồ họa.
- Được sử dụng để viết các công cụ hệ thống, xử lý dữ liệu, và lập trình máy chủ

52

52

Ứng dụng Console

- Ví dụ

53

53

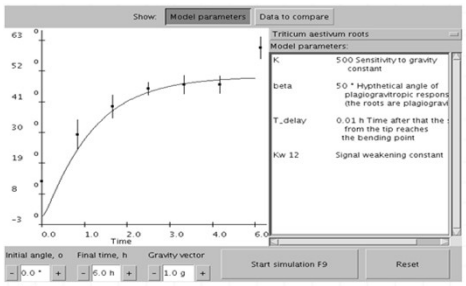
Ứng dụng Applets

- Trước đây, JavaApplet dùng để nhúng trong trang Web, applet được tải về và thực thi khi duyệt web. Chương trình được tạo ra để sử dụng trên Internet thông qua các trình duyệt web hỗ trợ Java: Chrome, IE, Firefox…
- Các trình duyệt Chrome, Firefox, Edge đã ngừng hỗ trợ Applet từ 2017. Java Applet bị loại bỏ từ Java 17.
- Thay thế bằng: WebAssembly, Progressive Web Apps (PWA), và Native JavaFX Applications hoặc Spring Boot + Thymeleaf.

54

54

Ứng dụng Applets



55

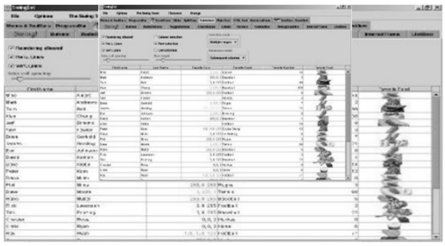
Ứng dụng Desktop

- Dùng các gói thư viện AWT và Swing để tạo ra giao diện cho chương trình
- JFC (Java Foundation Classess) là thư viện phong phú và hỗ trợ mạnh mẽ cho việc thiết kế hơn nhiều so với AWT và Swing. JFC giúp người dùng tạo ra giao diện trực quan cho bất kỳ ứng dụng nào.
- JavaFX - Công nghệ GUI hiện đại. Hiện đang thay thế Swing/AWT, cung cấp UI đẹp hơn, hiệu suất tốt hơn.

56

Ứng dụng Desktop

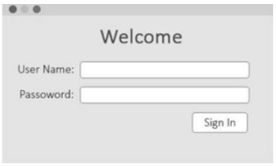
- Ví dụ tạo giao diện bằng JFC



57

Ứng dụng Desktop

- Ví dụ giao diện JavaFX



58

Ứng dụng Web

- Java hỗ trợ mạnh mẽ trong việc phát triển các ứng dụng Web thông qua công nghệ Jakarta EE (trước đây là J2EE - Java 2 Enterprise Edition, đổi tên từ 2018)
- Jakarta EE là nền tảng chính thức để phát triển ứng dụng Web. Hỗ trợ Microservices, API REST với Jakarta RESTful Web Services.
- Hiện nay có rất nhiều Website sử dụng công nghệ Java: Spring Boot phổ biến nhất để xây dựng API REST và web hiện đại, Jakarta EE tiêu chuẩn Java Web chính thức, Thymeleaf là template engine thay thế JSP…

59

Ứng dụng nhúng

- Java đưa ra công nghệ J2ME (Java 2 Platform, Micro Edition) hỗ trợ phát triển các phần mềm nhúng trên các thiết bị di động, PDA cũng như các thiết bị nhúng khác.
- J2ME đã lỗi thời trên mobile.
- Hiện nay Android chiếm lĩnh thị trường và Java chỉ còn dùng trong thiết bị IoT, thiết bị công nghiệp (Java SE Embedded, Java ME 8)
- Android sử dụng Java/Kotlin thay vì J2ME

60

7. JAVA CƠ BẢN

61

MỘT CHƯƠNG TRÌNH CƠ BẢN

```
1   // Tên file : Hello.java
2   /* Tác giả : Chat HQU */
3
4     public class Hello
5     {
6     // Phương thức main, điểm bắt đầu của chương trình
7     public static void main( String args[ ] )
8     {
9       System.out.println( "Hello World" );
10
11    } // Kết thúc phương thức main
12
13  } // Kết thúc lớp Hello
```

Dấu hiệu chú thích => Làm cho chương trình dễ hiểu hơn. Trình biên dịch sẽ bỏ qua những dòng có dấu chú thích

Khai báo lớp
Mỗi CT phải có ít nhất một khai báo lớp
Tên lớp chứa hàm main phải giống tên file

Các câu lệnh phải kết thúc bằng dấu chấm phẩy

Phương thức main() sẽ được gọi đầu tiên. Mỗi CT thực thi phải có một phương thức main()

Hiển thị dãy ký tự ra màn hình

62

Phương thức main

- **Phương thức main():** là điểm bắt đầu thực thi một ứng dụng.
- Mỗi ứng dụng Java phải chứa một phương thức main có dạng như sau: **public static void main(String[] args)**
- Phương thức main chứa ba bổ từ đặc tả sau:
  - **public:** chỉ ra rằng phương thức main có thể được gọi bởi bất kỳ đối tượng nào.
  - **static:** chỉ ra rằng phương thức main là một phương thức lớp.
  - **void:** chỉ ra rằng phương thức main sẽ không trả về bất kỳ một giá trị nào.

63

Phương thức main

- args[] là mảng chứa tham số dòng lệnh khi chạy chương trình.
- Nếu không truyền tham số, args sẽ là mảng rỗng (args.length == 0).
- Nên dùng String[] args hơn String args[].

64

Chú thích trong Java

- Ngôn ngữ Java hỗ trợ ba kiểu chú thích sau:

  /* text */

  // text

  /** documentation */ công cụ javadoc trong bộ JDK sử dụng chú thích này để chuẩn bị cho việc tự động phát sinh tài liệu.
- Dấu mở và đóng ngoặc nhọn "{" và "}" là bắt đầu và kết thúc một khối lệnh.
- Dấu chấm phẩy ";" để kết thúc một dòng lệnh.
- Java được tổ chức theo lớp (class). Các lệnh và các hàm (kể cả hàm main) phải thuộc một lớp nào đó, chúng không được đứng bên ngoài của lớp.

65

Nhập dữ liệu từ bàn phím

- Ví dụ nhập một số nguyên và một số thực

```
import java.io.*;
public class TestInput
{
   public static void main(String[] args) throws Exception
   {
     BufferedReader inStream =
           new BufferedReader(new InputStreamReader(System.in));

     System.out.print("Nhap mot so nguyen:");
     String siNumber = inStream.readLine();
     int iNumber = Integer.parseInt(siNumber);
```


66

Nhập dữ liệu từ bàn phím

```
System.out.print("Nhap mot so thuc:");
  String sfNumber = inStream.readLine();
  float fNumber = Float.parseFloat(sfNumber);

  System.out.println("So nguyen:“ + iNumber);
  System.out.println("So thuc:“ + fNumber);
  }
}
```

67

67

Từ khóa (keyword)

- Từ khóa cho các kiểu dữ liệu cơ bản : **byte, short, int, long, float, double, char, boolean.**
- Từ khóa cho cú pháp lặp: **do, while, for, break, continue.**
- Từ khóa cho cú pháp rẽ nhánh: **if, else, switch, case, default, break.**
- Từ khóa đặc tả đặc tính một method: **private, public, protected, final, static, abstract, synchronized.**
- Hằng (literal): **true, false, null.**
- Từ khóa liên quan đến method: **return, void.**
- Từ khoá liên quan đến package: **package, import.**
- Từ khóa cho việc quản lý lỗi: **try, catch, finally, throw, throws.**
- Từ khóa liên quan đến đối tượng: **new, extends, implements, class, instanceof, this, super.**

68

68

Từ khóa (keyword)

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

69

69

Định danh (Identifier)

- Định danh là dùng biểu diễn tên của biến, của Phương thức, của lớp.
- Trong Java, định danh có thể sử dụng ký tự chữ, ký tự số và ký tự dấu.
- Ký tự đầu tiên phải là ký tự chữ, dấu gạch dưới (_), hoặc dấu dollar ($).
- Có sự phân biệt giữa ký tự chữ hoa và chữ thường.

Ví dụ: Hello, _prime, var8, tvLang

70

70

Biến (Variable)

- Biến là một vùng nhớ để lưu trữ các giá trị của chương trình
- Mỗi biến gắn với 1 kiểu dữ liệu và 1 định danh duy nhất là tên biến
- Tên biến thông thường là một chuỗi các ký tự (Unicode), ký số
  - Trong Java tên biến phân biệt chữ hoa và chữ thường.
  - Tên biến bắt đầu bằng 1 dấu _, $, hay 1 ký tự, không được bắt đầu bằng 1 ký số.
  - Tên biến không có khoảng trắng ở giữa tên.
- Trong java, biến có thể được khai báo ở bất kỳ nơi đâu trong chương trình.

71

71

Biến (Variable)

- **Khai báo**

  <kiểu dữ liệu> <tên biến>;

  <kiểu dữ liệu> <tên biến> = <giá trị>;
- **Gán giá trị**

  <tên biến> = <giá trị>;
- **Ví dụ:**

  int age;

  age = 0; hoặc

  int age = 0;

72

72

Phân loại biến

- Biến trong Java có 3 loại:
  - Local variable
  - Instance variable
  - Static variable

73

73

Phân loại biến

- **Local variable**
  - Được khai báo bên trong các method (phương thức), constructor(phương thức khởi tạo), hoặc block (khối lệnh)
  - Được tạo ra khi method, constructor hoặc block được gọi và biến này sẽ bị destroy (hủy) ngay khi thực hiện xong method, constructor hoặc block.
  - Chúng chỉ xuất hiện nội trong giới hạn của method, constructor hoặc block
  - Không có giá trị mặc định cho các biến local nên chúng nên được khai báo với một giá trị khởi tạo (initial value)

74

74

Phân loại biến

```
public class Test{
    public void pupAge()
    {
        int age = 0;  age = age + 7;
        System.out.println("Puppy age is : " + age);
    }
}
```

```
public static void pupAge()
    {
        int age;
        age = age + 7;
        System.out.println("Puppy age is : " + age);
    }
```

```
Test.java:4:variable number might not
have been initialized
age = age + 7;
        ^
1 error
```

75

75

Phân loại biến

- **Instance variable**
  - Được khai báo trong class (lớp), nhưng bên ngoài các method, constructor hoặc block.
  - Được tạo ra khi một object (đối tượng) được tạo ra và bị hủy khi object bị hủy
  - Giữ các giá trị được tham chiếu tới bởi nhiều method, constructor hoặc block, hoặc là các phần khác.
  - Có thể được khai báo trong lớp trước hoặc sau khi sử dụng nó

76

76

Phân loại biến

- Có giá trị mặc định của chúng: *number* thì là 0, *boolean* thì là false, *object* là null.
- Có thể được truy xuất trực tiếp bằng các gọi tên biến bên trong class. Còn với những static method hoặc lớp khác (khi instance variable được cho phép truy xuất), thì chúng có thể được gọi thông qua

ObjectReference.VariableName (tên_đối_tượng.tên_biến)

77

77

Phân loại biến

```
class Employee{
        public String name;
        private double salary;
        public Employee (String empName){  name = empName;
        }
        public void setSalary(double empSal){
                salary = empSal;
        }
        public void printEmp(){
                System.out.println("name : " + name );
                System.out.println("salary :" + salary);  }  public
        static void main(String args[]){
        Employee empOne = new Employee("Ransika");
empOne.setSalary(1000);
        empOne.printEmp();
}
```

Instance Variable

78

78

Phân loại biến

- **Static variable**
  - Được khai báo với từ khóa static và bên trong class nhưng bên ngoài *method, constructor* hoặc *block*
  - Biến static là duy nhất bên trong một class (tức là không có biến nào khác cùng tên với nó).
  - Được lưu trữ trong bộ nhớ static(vì vậy biến này sẽ còn tồn tại ứng dụng còn đang chạy).
  - Được tạo ra khi chương trình bắt đầu chạy và hủy khi chương trình dừng.
  - Hầu hết các biến static được khai báo public .
  - Giá trị mặc định cũng tương tự như instance variable
  - Có thể được truy xuất bằng cách gọi tên lớp:

ClassName.VariableName (tên_lớp.tên_biến)

79

79

```
public class Employee {
  // salary  variable is a private static variable
  private static double salary;
  public static void main(String args[]){
    salary = 1000;
    System.out.println(salary);
  }
}
public class Test {
  public static void main(String args[]){
        Employee.salary = 2000;
        System.out.println("salary:"+Employe
        e.salary);
        }
}12
```

Gọi trực tiếp biến static salary bên trong lớp Employee

Gọi biến salary từ ngoài

80

80

Biến

- Ví dụ về var: Giúp khai báo biến cục bộ làm mã ngắn gọn hơn mà vẫn đảm bảo kiểu dữ liệu.

```
var message = "Hello, Java 10!"; // Tự động nhận dạng kiểu String
var number = 100; // Nhận dạng là int
```

- Ví dụ về record: Tạo lớp bất biến ngắn gọn hơn, thay vì viết getter/setter thủ công.

```
record Person(String name, int age) {}

public class Main {
    public static void main(String[] args) {
        Person p = new Person("Alice", 25);
        System.out.println(p.name() + " - " + p.age());
    }
}
```

81

81

Hằng (Literal)

- Là một giá trị bất biến trong chương trình
- Tên đặt theo qui ước như tên biến
- Được khai báo dùng từ khóa **final**
- *Hằng số nguyên:* trường hợp giá trị hằng ở dạng long ta thêm vào cuối chuỗi số chữ "l" hay "L".
  - VD: final long y = 20L; // khai báo hằng số long y = 20
- *Hằng số thực:* trường hợp giá trị hằng có kiểu float ta thêm tiếp vĩ ngữ "f" hay "F", còn kiểu số double thì ta thêm tiếp vĩ ngữ "d" hay "D".
  - VD: final float x = 20F; // khai báo hằng số float x= 20
- *Hằng Boolean:* java có 2 hằng boolean là **true, false.**

82

82

Hằng

- **Hằng ký tự:** đặt giữa cặp nháy đơn ''
  - Ví dụ: 'a': hằng ký tự a
- **Hằng chuỗi:** là tập hợp các ký tự được đặt giữa hai dấu nháy kép "". Một hằng chuỗi không có ký tự nào là một hằng chuỗi rỗng.
  - Ví dụ: "Hello World"

(Hằng chuỗi không phải là kiểu dữ liệu cơ sở)

83

83

Ký tự đặc biệt

Ký tự	*Ý nghĩa*
\b	*Xóa lùi (BackSpace)*
\t	*Tab*
\n	*Xuống hàng*
\r	*Dấu enter*
\"	*Nháy kép*
\'	*Nháy đơn*
**	**
\f	*Đẩy trang*
\uxxxx	*Ký tự unicode*

84

84

Kiểu dữ liệu

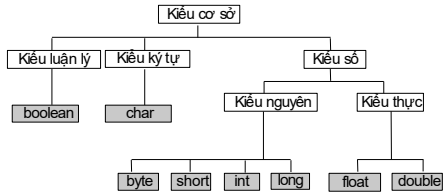
- Kiểu dữ liệu cơ sở (primitive data type)
- Kiểu dữ liệu tham chiếu (reference data type)

85

85

Kiểu dữ liệu cơ sở

- **Kiểu dữ liệu cơ sở (primitive data type)**
  - Có 8 kiểu dữ liệu cơ sở: byte, short, int, long, float, double, boolean và char



86

86

Kiểu dữ liệu cơ sở

Kiểu	Kích thước (bits)	Giá trị	Giá trị mặc định
boolean	[*Note:* The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]	true và false	**false**
char	16	'\u0000' to '\uFFFF' (0 to 65535)	**null**
byte	8	−128 to +127 (-2^7 to $2^7 - 1$)	**0**
short	16	−32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	**0**
int	32	−2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	**0**
long	64	−9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (-263 to $263 - 1$)	**0l**
float	32	1.4012984632481707e−45 to 3.4028234663852886E+38	**0.0f**
double	64	4.94065645841246544e−324 to 1.79769313486231570E+308	**0.0d**

87

87

Kiểu dữ liệu cơ sở

Data Type	Default Value	Size (in bytes)**1 byte = 8 bits**
boolean	FALSE	1 bit
char	" " (space)	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

88

88

Kiểu dữ liệu cơ sở

- **Chuyển đổi kiểu dữ liệu:** khi có sự không tương thích về kiểu dữ liệu (gán, tính toán biểu thức, truyền đối số gọi phương thức)
- Chuyển kiểu hẹp (lớn → nhỏ): cần ép kiểp

  <tên biến 2> = (kiểu dữ liệu) <tên biến 1>;
- Chuyển kiểu rộng (nhỏ → lớn): tự động chuyển

89

89

Kiểu dữ liệu cơ sở

- **Ví dụ:**

  float fNum = 2.2;

  int iCount = (int) fNum

90

90

Kiểu dữ liệu cơ sở

- **Kiểu số nguyên:**
  - Java cung cấp 4 kiểu số nguyên khác nhau là: byte, short, int, long.
  - Kiểu mặc định của các số nguyên là kiểu *int*.
  - Các số nguyên kiểu byte và short rất ít khi được dùng.
  - Trong java không có kiểu số nguyên không dấu như trong ngôn ngữ C/C++.
  - Khai báo và khởi tạo giá trị cho các biến kiểu nguyên:

```
int x = 0;
long y = 100;
```

91

91

Kiểu dữ liệu cơ sở

- Một số lưu ý đối với các phép toán trên số nguyên:
  - Nếu hai toán hạng kiểu *long* thì kết quả là kiểu *long*.
  - Một trong hai toán hạng không phải kiểu *long* sẽ được chuyển thành kiểu *long* trước khi thực hiện phép toán.
  - Nếu hai toán hạng đầu không phải kiểu *long* thì phép tính sẽ thực hiện với kiểu *int*.
  - Các toán hạng kiểu *byte* hay *short* sẽ được chuyển sang kiểu *int* trước khi thực hiện phép toán.

```
byte x = 5;
byte y = 10;
byte z = x + y; // Dòng lệnh báo lỗi kiểu dữ liệu
```

Sửa lại `byte z = (byte) (x + y);`

92

Kiểu dữ liệu cơ sở

- Trong java không thể chuyển biến kiểu int và kiểu boolean như trong ngôn ngữ C/C++.

```
boolean b = false;
if (b == 0)
{
System.out.println("Xin chao");
}
```

Báo lỗi: không được phép so sánh biến kiểu boolean với một giá trị kiểu int.

93

Kiểu dữ liệu cơ sở

- **Kiểu dấu chấm động**
  - Java cung cấp hai kiểu dữ liệu là *float* và *double*.
  - Kiểu *float* có kích thước 4 byte và giá trị mặc định là 0.0f.
  - Kiểu *double* có kích thước 8 byte và giá trị mặc định là 0.0d
  - Số kiểu dấu chấm động không có giá trị nhỏ nhất cũng không có giá trị lớn nhất. Chúng có thể nhận các giá trị:
    - Số âm
    - Số dương
    - Vô cực âm
    - Vô cực dương
  - Khai báo và khởi tạo giá trị cho các biến kiểu dấu chấm động: `float x = 100.0/7;`
    hoặc `float x = 100.0f`

```
double y = 1.56E6;
```

94

Kiểu dữ liệu cơ sở

- Một số lưu ý đối với các phép toán trên số dấu chấm động:
  - Nếu mỗi toán hạng đều có kiểu dấu chấm động thì phép toán chuyển thành phép toán dấu chấm động.
  - Nếu có một toán hạng là double thì các toán hạng còn lại sẽ được chuyển thành kiểu double trước khi thực hiện phép toán.
  - Biến kiểu float và double có thể ép chuyển sang kiểu dữ liệu khác trừ kiểu boolean.

95

95

Kiểu dữ liệu cơ sở

- **Kiểu ký tự (char):**
  - Kiểu ký tự *(char)* trong ngôn ngữ lập trình Java có kích thước là 2 bytes và chỉ dùng để biểu diễn các ký tự trong bộ mã Unicode.
  - Kiểu *char* trong java có thể biểu diễn tất cả 2^{16} = 65536 ký tự khác nhau.
  - Giá trị mặc định cho một biến kiểu *char* là null.
  - Khai báo

```
char kyTu = 'a'
```

96

96

Kiểu dữ liệu cơ sở

- **Kiểu luận lý (boolean):**
  - Kiểu *boolean* chỉ nhận 1 trong 2 giá trị: ***true*** hoặc ***false***
  - Trong java kiểu boolean không thể chuyển thành kiểu nguyên và ngược lại.
  - Giá trị mặc định của kiểu boolean là false.
  - Khai báo

    boolean b = false

97

97

Kiểu dữ liệu tham chiếu

- **Kiểu dữ liệu tham chiếu (reference data type):**
  - Kiểu mảng
  - Kiểu đối tượng (class)

98

98

Kiểu dữ liệu tham chiếu

- **Kiểu mảng:**
  - Mảng là *tập hợp* các phần tử có cùng tên và cùng kiểu dữ liệu.
  - Mỗi phần tử được truy xuất thông qua chỉ số.
  - Khai báo:

```
<kiểu dữ liệu>[ ] <tên mảng>;          // mảng 1 chiều
<kiểu dữ liệu> <tên mảng>[];           // mảng 1 chiều
<kiểu dữ liệu>[ ][ ] <tên mảng>;       // mảng 2 chiều
<kiểu dữ liệu><tên mảng>[ ][ ];        // mảng 2 chiều
```

99

99

Kiểu mảng

- **Khởi tạo:**

```
int     arrInt[] = {1, 2, 3};
char    arrChar[] = {'a','b','c'};
```

- **Cấp phát bộ nhớ cho mảng**

```
int arrInt[100];        // Khai báo này trong Java sẽ bị báo lỗi.
int[] arrInt = new int[100]; //Khai báo dùng từ khóa new
```

100

100

Kiểu mảng

- **Truy cập mảng**
  - Chỉ số mảng ***n*** phần tử: từ ***0*** đến ***n-1***
  - Các phần tử được truy xuất thông qua chỉ số của nó đặt giữa cặp dấu ngoặc vuông ([]).

```
int arrInt[] = {1, 2, 3};
int x = arrInt[0]; // x sẽ có giá trị là 1.
int y = arrInt[1]; // y sẽ có giá trị là 2.
int z = arrInt[2]; // z sẽ có giá trị là 3.
```

101

101

Kiểu mảng

Nhập và xuất giá trị các phần tử của một mảng các số nguyên:

```
class ArrayDemo{
    public static void main(String args[]){
    int [] arrInt = new int[10];
    int i;
    for(i = 0; i < 10; i = i+1)
        arrInt[i] = i;
    for(i = 0; i < 10; i = i+1)
        System.out.println("This is arrInt[" + i +"]: " + arrInt[i]);
    }
}
```

102

102

Kiểu mảng

Tìm phần tử có giá trị nhỏ nhất (Min) và lớn nhất (Max) trong một mảng.

```
class MinMax2
{   public static void main(String args[])
    {     int nums[] = { 99, -10, 100123, 18, -978, 5623, 463, -9, 287, 49 };
          int min, max;
          min = max = nums[0];
          for(int i=1; i < 10; i++)
          {    if(nums[i] < min) min = nums[i];
               if(nums[i] > max) max = nums[i];
          }
          System.out.println("Min and max: " + min + " " + max);
    }
}
```


Kiểu mảng

Nhập và xuất giá trị của các phần tử trong một mảng hai chiều.

```
class TwoD_Arr
{     public static void main(String args[])
      {     int t, i;
            int [][] table = new int[3][4];
            for(t=0; t < 3; ++t)
            {       for(i=0; i < 4; ++i)
                    {       table[t][i] = (t*4)+i+1;
                            System.out.print(table[t][i] + " ");
                    }
                    System.out.println();
            }
      }
}
```


Kiểu đối tượng

Dữ liệu kiểu đối tượng do người dùng định nghĩa. Chứa tập các thuộc tính và phương thức.

- **Khai báo đối tượng**
  <Kiểu đối tượng> <biến ĐT>;
  - *VD: Animal animal;*
- **Khởi tạo đối tượng**

<Kiểu đối tượng> <biến ĐT> = new <Kiểu đối tượng>;

  - *VD: Animal animal = new Animal("giraffe");*
- **Truy xuất thành phần đối tượng**
  <biến ĐT>.<thuộc tính>
  - *VD: animal.tall;*
  <biến ĐT>.<phương thức>
  - *VD: animal.addanimal();*

Chuỗi (String)

- String là một đối tượng trong Java. Nhưng những đối tượng này không thể thay đổi được - giá trị của chúng một khi đã được khởi tạo thì không thể thay đổi.
- Ví dụ:

```
//msg trỏ tới vùng nhớ 1 có giá trị là
String msg = "Hello";
//msg trỏ tới vùng nhớ 2 có giá trị là "Hello world"
"Hello" msg += " world";
```

như ví dụ trên, String "Hello" không bị thay đổi. Thay vào đó, một String mới được tạo với giá trị là "Hello world" và msg được gán bằng String mới này.

Chuỗi (String)

- Nếu khởi tạo 2 String với cùng một giá trị mà không dùng từ khóa new thì sẽ chỉ có một đối tượng String được tạo.
- Ví dụ:

String s1="hello";//s1 trỏ tới vùng nhớ 1 có giá trị là "hello"
String s2="hello";//s2 cũng trỏ tới vùng nhớ 1 có giá trị là "hello"

Ở đây, s1 và s2 cùng trỏ tới một đối tượng String trong bộ nhớ. Bởi vậy, nếu thay đổi giá trị của s1 thì một đối tượng String mới sẽ được tạo còn đối tượng String ban đầu mà s2 trỏ tới thì sẽ không thay đổi

Chuỗi (String)

1. Tạo chuỗi

- String message = "Welcome to Java!" // tạo tắt - String literal
- String message = new String("Welcome to Java!");
- String s = new String();

Chuỗi (String)

2. Sử dụng phương thức **intern** của lớp String để trả về một string literal , tương tự như string được tạo bởi lệnh tắt:

- 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

109

109

Toán tử và biểu thức

- Toán tử số học
- Toán tử trên bit
- Toán tử quan hệ và logic
- Toán tử gán
- Toán tử ép kiểu
- Toán tử điều kiện

110

110

Toán tử và biểu thức

- **Toán tử số học:**

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia nguyên
%	Chia dư
++	Tăng 1
--	Giảm 1

111

111

Toán tử và biểu thức

- **Toán tử trên bit:**

Toán tử	Ý nghĩa
&	AND
|	OR
^	XOR
<<	Dịch trái
>>	Dịch phải
~	Bù bit

112

112

Toán tử và biểu thức

- **Toán tử quan hệ và logic:**

Toán tử	Ý nghĩa
= =	So sánh bằng
!=	So sánh khác
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hay bằng
<=	So sánh nhỏ hơn hay bằng
||	OR (biểu thức logic)
&&	AND (biểu thức logic)
!	NOT (biểu thức logic)

113

113

Toán tử và biểu thức

- **Toán tử gán:**

Toán tử	Ví dụ	Ý nghĩa
=	a = b	gán a = b
+=	a += 5	a = a + 5
-=	b -= 10	b = b – 10
*=	c *= 3	c = c * 3
/=	d /= 2	d = d/2
%=	e %= 4	e = e % 4

114

114

Toán tử và biểu thức

- **Toán tử ép kiểu:**
  - Ép kiểu rộng (*widening conversion*): từ kiểu nhỏ sang kiểu lớn (không mất mát thông tin)
  - Ép kiểu hẹp (*narrow conversion*): từ kiểu lớn sang kiểu nhỏ (có khả năng mất mát thông tin)

    <tên biến> = (kiểu_dữ_liệu) <tên_biến>;
- ***Ví dụ***:

```
float fNum = 2.2;
int iCount = (int) fNum; // (iCount = 2)
```

115

115

Toán tử và biểu thức

- **Toán tử điều kiện:**

Cú pháp: *<điều kiện> ? <biểu thức 1> : < biểu thức 2>*

  - Nếu điều kiện đúng thì thực hiện <biểu thức 1>, còn ngược lại là <biểu thức 2>.
  - <điều kiện>: là một biểu thức logic
  - <biểu thức 1>, <biểu thức 2>: có thể là hai giá trị, hai biểu thức hoặc hai hành động.

- ***Ví dụ:***

```
int x = 10;
int y = 20;
int z = (x<y) ? 50 : 80;
// Kết quả z = 50 do biểu thức (x < y) là đúng.
```

116

116

Độ ưu tiên

Thứ tự ưu tiên từ trái qua phải và từ trên xuống dưới

1	.	[]	()	
2	++	--	!	~
3	*	/	%	
4	+	-		
5	<<	>>	>>>	
6	<	>	<=	>=
7	==	!=		
8	&			
9	^			
10	&&			
11	||			
12	?:			
13	=			

117

117

CÂU LỆNH – CẤU TRÚC LỆNH

118

118

Cấu trúc điều khiển

- Cấu trúc **if….else**
- Cấu trúc **switch… case**
- Cấu trúc lặp (loop)
- Cấu trúc lệnh nhảy **jump**

119

119

Cấu trúc điều khiển

- Cấu trúc **if….else**
  - **Dạng 1:**

```
if (<điều_kiện>) {
      <khối_lệnh>;}
```

  - **Dạng 2:**

```
if (<điều_kiện>){
      <khối _lệnh1>;
}
else {
      <khối _lệnh2>;
}
```

120

120

Cấu trúc điều khiển

```
import java.util.Date;
public class TestIf
{
        public static void main( String args[ ] )
        {
            Date today = new Date();
            if( today.getDay() == 0 )
              System.out.println("Hom nay la chu nhat\n");
            else
              System.out.println("Hom nay khong la chu nhat\n" );
        }
}
```

121

121

Cấu trúc điều khiển

- Cấu trúc **switch….case**

```
switch (<biến>) {
        case <giátrị_1>:
                <khối_lệnh_1>;
                break;
        ....
        case <giátrị_n>:
                <khối_lệnh_n>;  break;
        default:
                <khối lệnh default>;
```

Là giá trị hằng

Dùng để thoát khỏi cấu trúc switch

122

122

Cấu trúc điều khiển

```
switch(wash) {
        case 1: // wash is 1 for Cotton
                System.out.println("Cotton selected");
        break;
        case 2: // wash is 2 for Linen
                System.out.println("Linen selected");
        break;
        case 3: // wash is 3 forWool
                System.out.println("Wool selected");
        break;
        default: // Not a valid value for wash
                System.out.println("Selection error");
        break;
}
```

123

123

Cấu trúc điều khiển

- Cấu trúc lặp
  - **Dạng 1:**

```
while (<điều_kiện_lặp>) {
        <khối _lệnh>;
}
```

  - **Dạng 2:**

```
do {
        <khối_lệnh>;
} while (điều_kiện);
```

  - **Dạng 3:**

```
for (khởi_tạo_biến_đếm;đk_lặp;tăng_biến) {
        <khối _lệnh>;
}
```

124

124

Cấu trúc điều khiển

```
// Tính tổng các số lẻ từ 1 đến 100
  int tong = 0, i = 1;
  while (i<=100)
  {
      tong+=i; i+=2;
  }
  System.out.println(tong);
```

125

125

Cấu trúc điều khiển

```
// Tính tổng các số lẻ từ 1 đến 100
  int tong = 0, i=1;
  do
  {
      tong+=i; i+=2;
  }   while (i<=100);
  System.out.println(tong);
```

126

126

Cấu trúc điều khiển

```
// Chương trình tính tổng các số lẻ từ 1 đến 100
public class TestFor {
   public static void main(String[] args)  {
      int tong = 0;
      for(int i=1; i<=100; i+=2)
           tong+=i;
      System.out.println(tong);
   }
}
```

127

127

Cấu trúc điều khiển

- **FOR**

Cú pháp:

```
For(vaiable : collection){ Statements; }
```

Ví dụ:

```
int[] a={1,2,3,4,5,6,7,8,9,0};
for(int i : a) {
  System.out.println(i);
 }
```

128

128

Cấu trúc điều khiển

- Cấu trúc lệnh nhảy jump:
  - Lệnh **break**: trong cấu trúc lặp, câu lệnh break dùng để thóat khỏi cấu trúc lặp trong cùng chứa nó.
  - Lệnh **continue**: dùng để tiếp tục vòng lặp trong cùng chứa nó (ngược với break).
  - Dùng kết hợp nhãn (**label**) với từ khóa ***break*** và ***continue*** để thay thế cho lệnh **goto** (trong C).

129

129

Cấu trúc điều khiển

- Cấu trúc lệnh nhảy jump:

Ví dụ:

```
label:
for (...) {
        for (...) {
                if (<biểu thức điều kiện>)
                        break label;
                else
                        continue label;
                }
}
```

130

130

Lớp bao kiểu dữ liệu

Data type	Wrapper Class (java.lang.*)	Ghi chú
boolean	Boolean	-Gói (package): chứa nhóm nhiều class. - Ngoài các Wrapper Class, gói java.lang còn cung cấp các lớp nền tảng cho việc thiết kế ngôn ngữ java như: String, Math, …
byte	Byte	
short	Short	
char	Character	
int	Integer	
long	Long	
float	Float	
double	Double	

131

131

Ví dụ lớp bao kiểu dữ liệu

```
Float a=new Float(5.5);
System.out.println(a.floatValue());
Float fObj1 = new Float("5.35");
Float fObj2 = new Float("5.34");
int i2 = fObj1.compareTo(fObj2);
if(i2 > 0){
        System.out.println("First is greater");
}else if(i2 < 0){
        System.out.println("Second is greater");
}else{
        System.out.println("Both are equal");
}
}
```

132

132

Ví dụ lớp bao kiểu dữ liệu

- Ví dụ lớp String:
  - Biểu diễn chuỗi

String s1 = "abc";
String s2 = new String("def");
String s3 = s1 + s2;

  - So sánh chuỗi

s1.equals("abc")
s2.equalIgoredCase("def");

133

133

NHẬP XUẤT

134

134

Nhập xuất trong Java

- Xuất dữ liệu ra màn hình có 3 loại:
  - System.out.println(" ");
    // Xuất kết quả ra màn hình đồng thời con trỏ chuột nhảy xuống dòng tiếp theo
  - System.out.print(" ");
    // Xuất kết quả ra màn hình nhưng con trỏ chuột không xuống dòng.
  - System.out.printf(" ");
    // Xuất ra màn hình kết quả đồng thời có thể định dạng được kết quả đó nhờ vào các đối số thích hợp.

135

135

Nhập xuất trong Java

- %c: Ký tự
- %d: Số thập phân (số nguyên) (cơ số 10)
- %e: Dấu phẩy động theo cấp số nhân
- %f: Dấu phẩy động
- %i: Số nguyên (cơ sở 10)
- %o: Số bát phân (cơ sở 8)

136

136

Nhập xuất trong Java

- %s: Chuỗi
- %u: Số thập phân (số nguyên) không dấu
- %x: Số trong hệ thập lục phân (cơ sở 16)
- %t: Định dạng ngày / giờ
- %%: Dấu phần trăm
- \%: Dấu phần trăm

137

137

Nhập xuất trong Java

- Nhập dữ liệu từ bàn phím có 3 cách:
  - Cách đầu tiên là sử dụng lớp **BufferReader**
  - Cách 2 là sử dụng lớp **Scanner** (hay dùng)
  - Cách 3 là sử dụng **JOptionPane**

138

138

Lớp BufferedReader

- **BufferReader** là một lớp dùng để đọc dữ liệu từ bàn phím hay từ file.Có thể dùng lớp này để đọc một *chuỗi*, một *mảng* hoặc một *ký tự*.
- Tham số đầu vào của BufferReader có thể là **InputStreamReader** hoặc **FileReader** (Dùng để đọc file).
- Một số phương thức của lớp **BufferReader**:
  - **read()** : đọc một kí tự.
  - **readLine()** : đọc một dòng text.
- Khai báo

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

139

139

Lớp BufferedReader

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class NhapXuat {
    public static int a, b,tong;
    public static void main(String[] args) {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
         String str;
        System.out.println("Nhap a:"); str=br.readLine();
        a=Integer.parseInt(str);
        System.out.println("Nhap b:");  str=br.readLine();
        b=Integer.parseInt(str);  tong=a+b;
       System.out.println("tong "+tong);
}
}
```

140

140

Lớp Scanner

- Phân loại được dữ liệu mà người dùng nhập vào: có thể đọc kiểu int hay float, double,…
- Một số phương thức của lớp **Scanner:**
  - **next() //dùng cho String**
  - **nextInt()**
  - **nextFloat()**
  - **nextBoolean()**
  - **nextByte()**
  - **nextLine()**

141

141

Lớp Scanner

- Lớp java.util.Scanner

public boolean	nextBoolean()
public byte	nextByte()
public byte	nextByte(int radix)
public double	nextDouble()
public float	nextFloat()
public int	nextInt()
public int	nextInt(int radix)
public String	nextLine()
public long	nextLong()
public long	nextLong(int radix)
public short	nextShort()
public short	nextShort(int radix)

142

142

Lớp Scanner

- **Ví dụ:**

```
Scanner in = new Scanner(System.in);
int number = in.nextInt();
String string = in.nextLine();
float real = in.nextFloat();
 String string2 = in.nextLine();
```

143

143

Lớp JOptionPane

- JOptionPane là một lớp thừa kế từ lớp **JComponent**.
- Khi biên dịch program thì nó sẽ hiện lên một **dialog box** cho phép cho ta nhập dữ liệu.
- Một số phương thức của lớp **JOptionPane:**
  - **showConfirmDialog()** : Hiển thị một câu hỏi lựa chọn giống như *yes, no, cancel*
  - **showInputDialog()** : Hiển thị box nhập
  - **showMessageDialog()** : Báo cho người dùng một sự kiện vừa xảy ra.
- Dữ liệu nhập vào chỉ có kiểu String

144

144

Lớp JOptionPane

```
import javax.swing.JOptionPane;
public class InputFromKeyboardJOptionPane
{
		public static void main(String[] args)
		{
		String name = "";	name=JOptionPane.showInputDialog("Please enter your
name");
		String msg = "Hello " + name + "!";
		JOptionPane.showMessageDialog(null, msg);		System.out.println("Name
is:"+msg);
		}
}
```

145

145

Q & A

Giảng viên: Tạ Việt Phương
E-mail:phuongtv@uit.edu.vn

146

146

Bài tập

1. TestArray: Nhập vào 6 số, kiểm tra xem có phải dãy tăng hay không?
2. Viết chương trình đảo ngược chuỗi họ tên. Ví dụ: Liễu Như Yên → Yên Liễu Như
3. Nhập vào 1 chuỗi, in ra nghịch đảo chuỗi vừa nhập
4. Nhập vào một số nguyên n, in ra bảng cửu chương của n.

147

147

Homework

- Tìm hiểu thêm về Maven và Gradle - Công cụ quản lý dự án Java, để quản lý thư viện và build ứng dụng, tích hợp với CI/CD
- Cài đặt JDK và NetBeans (hoặc VS Code)

148

148