## TỔNG QUAN VỀ JAVA

### Giới thiệu về Java

Java là ngôn ngữ lập trình máy tính có tính chất hướng đối tượng, dựa trên các lớp (class), thường được sử dụng cho các hệ thống có tính độc lập cao. Nó được sử dụng để hướng tới các lập trình viên viết ứng dụng "write one, run everywhere" (viết một lần, chạy mọi nơi, nghĩa là đoạn code Java sau khi được biên dịch có thể chạy được trên tất cả các nền tảng hỗ trợ Java mà không cần phải được biên dịch lại. Các ứng dụng Java sau khi đã được biên dịch thành bytecode có thể chạy trên bất kỳ máy ảo Java nào (Java virtual machine)

Cho đến năm 2018, Java là một trong những ngôn ngữ được dùng phổ biến nhất trên thế giới, đặc biệt cho các úng dựng web client- server. Theo thống kê trên thế giới có khoảng 9 triệu lập trình viên Java.

### Java là gì ?

Java là một Ngôn ngữ lập trình và là một Platform.

**Ngôn ngữ lập trình:** Java là một ngôn ngữ lập trình có tính bảo mật cao, hướng đối tượng, bậc cao và mạnh mẽ.

**Platform:**Bất cứ môi trường phần cứng hoặc phần mền nào mà trong đó một chương trình chạy, thì được biết đến như là một Platform. Với môi trường runtime riêng cho mình là JRE và API, Java được gọi là Platform.

### Lịch sử ngôn ngữ lập trình java

Java là ngôn ngữ lập trình hướng đối tượng (tựa C++) do Sun Microsystem đưa ra vào giữa thập niên 90. Chương trình viết bằng ngôn ngữ lập trình java có thể chạy trên bất kỳ hệ thống nào có cài máy ảo java (Java Virtual Machine).

Ngôn ngữ lập trình Java do James Gosling và các cộng sự của Công ty Sun Microsystem phát triển. Đầu thập niên 90, Sun Microsystem tập hợp các nhà nghiên cứu thành lập nên nhóm đặt tên là Green Team. Nhóm Green Team có trách nhiệm xây dựng công nghệ mới cho ngành điện tử tiêu dùng. Để giải quyết vấn đề này, nhóm nghiên cứu phát triển đã xây dựng một ngôn ngữ lập trình mới đặt tên là Oak tương tự như C++ nhưng loại bỏ một số tính năng nguy hiểm của C++ và có khả năng chạy trên nhiều nền phần cứng khác nhau.

Vào năm 1993, hệ thống Web/Internet của thế giới ra đời, chuyển các thông tin Internet dạng văn bản sang mói trường đồ hoạ. Nhóm Java nhận thức rằng Java khi được phát triển sẽ là một ngôn ngữ phục vụ hoàn hảo nhất cho lập trình trên Web. Họ đã xây dựng các nguyên tắc cho những ứng dụng nhỏ nhúng trong trang Web (Web Applets) và hơn nữa, họ triển khai cả một ứng dụng hoàn chinh, một trình duyệt Web (gọi là HotJava), để chứng tỏ khả năng mạnh mẽ của Java.

Tại thời điểm này chỉ những trình duyệt hiểu Java mới có khả năng chạy Applet và HotJava là trình duyệt tiên phong. Nhưng nhà phần mềm khổng lồ Microsoft không đứng ngoài cuộc. Nhận thức rõ sự phát triển của Internet, hãng Microsoft đã lập tức hỗ trợ Java ngay trong trình duyệt IE (Internet Explorer từ phiên bản 3.0 trở đi).

Và đến quý II năm 1995, Sun Microsystems công bố chính thức Java thông qua các ấn phẩm thuộc "Họ sách Java - Java Series" vào năm 1995 trong đó có các cuốn:

Những đặc trưng của ngôn ngữ Java (Java Specifications): Mô tả chi tiết cấu trúc ngữ pháp của Java.

* Cuốn sách trắng về Môi trường Java (The Java Environment - A White Paper): Giới thiệu môi trường làm việc của Java các chi tiết kỹ thuật trong lập trình với Java."
* Chuẩn hóa Java (Java Standardization - A White Paper): Nói về việc chuẩn hóa theo tiêu chuẩn ISO của hệ thống Java.

Java nhanh chóng được đón nhận như là một công cụ mạnh mẽ cho việc phát triển các ứng dụng cho Internet. Các trình duyệt Web lần lượt đưa ra sự hỗ trợ cho Java, mở đầu là Netscape Communications với Netscape Navigator Web (bản 2.0) rồi Microsoft với Internet Explorer 3.0.

### Sự phát triển của java

Java được xây dựng chủ yếu trong bộ công cụ phát triển Java (Java Development Kit - JDK) như là thư viện chuẩn, trong đó chứa các trình biên dịch, thông dịch, giúp đỡ, soạn tài liệu, máy ảo JVM (Java Virtual Machine để chạy chương trình Java), ...

Đó chính là cái nền (platform) của việc phát triển các ứng dụng Java. Với sự phát triển của Java hiện nay, các nhà phát triển phần mềm đã xây dựng thêm rất nhiều nhánh mới như Java - Thư tín (JavaMail), Java-Viễn thông (Java TAPI), ... JDK trở thành môi trường bao trùm của hệ thống Java.

Khởi đầu với phiên bản JDK 1.0 vào năm 1995, JDK 1.1 được công bố vào năm 1996 với nhiều cải tiến như tăng thêm các hàm giao diện (AWT - Abstract Windows Toolkit), xây dựng hệ thống thư viện dùng lại độc lập JavaBeans, thư viện lập trình giao diện JFC (Java Foundation Class), tăng thêm các lớp mới như java.math (lớp phục vụ các hàm toán học), java.net (lớp làm việc trên mạng).

Đến nay JDK 16 của Oracle hiện đã phát hành bản production, từ ngày 16 tháng 3. Các tính năng mới trong JDK 16 khá đa dạng từ các tính năng xem trước đến xử lý rác.

### Khái niệm

#### JAVA

A logo on a green background

Description automatically generated

**Hình 1:**

Java là một ngôn ngữ lập trình được sử dụng rộng rãi để viết mã cho các ứng dụng web. Ngôn ngữ này là lựa chọn phổ biến của các nhà phát triển trong hơn 2 thập niên. Hiện nay có hàng triệu ứng dụng Java đang được sử dụng. Java là một ngôn ngữ đa nền tảng, hướng đến đối tượng, lấy mạng làm trung tâm và có thể được sử dụng như một nền tảng. Đây là một ngôn ngữ lập trình nhanh, bảo mật, đáng tin cậy dùng để viết mã cho mọi thứ từ ứng dụng di động, phần mềm doanh nghiệp cho đến các ứng dụng dữ liệu lớn và công nghệ phía máy chủ.

#### Đặc điểm của ngôn ngữ lập trình java

* **Ngôn ngữ thông dịch**

Ngôn ngữ lập trình thường được chia ra làm 2 loại (tùy theo các hiện thực hóa ngôn ngữ đó) là ngôn ngữ thông dịch và ngôn ngữ biên dịch.

* Thông dịch (Interpreter) : Nó dịch từng lệnh rồi chạy từng lệnh, lần sau muốn chạy lại thì phải dịch lại.
* Biên dịch (Compiler): Code sau khi được biên dịch sẽ tạo ra 1 file thường là .exe, và file .exe này có thể đem sử dụng lại không cần biên dịch nữa.

Ngôn ngữ lập trình Java thuộc loại ngôn ngữ thông dịch. Chính xác hơn, Java là loại ngôn ngữ vừa biên dịch vừa thông dịch. Cụ thể như sau

Khi viết mã, hệ thống tạo ra một tệp .java. Khi biên dịch mã nguồn của chương trình sẽ được biên dịch ra mã byte code. Máy ảo Java (Java Virtual Machine) sẽ thông dịch mã byte code này thành machine code  (hay native code) khi nhận được yêu cầu chạy chương trình.

A diagram of a diagram

Description automatically generated

**Hình 2:** Sơ đồ thông dịch

Ưu điểm : Phương pháp này giúp các đoạn mã viết bằng Java có thể chạy được trên nhiều nền tảng khác nhau. Với điều kiện là JVM có hỗ trợ chạy trên nền tảng này.

Nhược điểm : Cũng như các ngôn ngữ thông dịch khác, quá trình chạy các đoạn mã Java là chậm hơn các ngôn ngữ biên dịch khác (tuy nhiên vẫn ở trong một mức chấp nhận được).

* **Độc lập phần cứng và hệ điều hành**

Một chương trình viết bằng ngôn ngữ Java có thể chạy tốt ở nhiều môi trường khác nhau. Gọi là khả năng “cross-platform”. Khả năng độc lập phần cứng và hệ điều hành được thể hiện ở 2 cấp độ là cấp độ mã nguồn và cấp độ nhị phân.

Ở cấp độ 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 bộ thư viện để hỗ trợ vấn đề này. Chương trình viết bằng ngôn ngữ Java có thể biên dịch trên nhiều loại máy khác nhau mà không gặp lỗi.

Ở cấp độ nhị phân: Một mã biên dịch có thể chạy trên nhiều nền tảng khác nhau mà không cần dịch lại mã nguồn. Tuy nhiên cần có Java Virtual Machine để thông dịch đoạn mã này.

* **Đa luồng**

Java hỗ trợ lập trình đa tiến trình (multithread) để thực thi các công việc đồng thời. Đồng thời cũng cung cấp giải pháp đồng bộ giữa các tiến trình (giải pháp sử dụng priority…).

* **Tính an toàn bảo mật**
* Tính an toàn
* Ngôn ngữ lập trình Java yêu cầu chặt chẽ về kiểu dữ liệu.
* Dữ liệu phải được khai báo tường minh.
* Không sử dụng con trỏ và các phép toán với con trỏ.
* Java kiểm soát chặt chẽ việc truy nhập đến mảng, chuỗi. Không cho phép sử dụng các kỹ thuật tràn. Do đó các truy nhập sẽ không vượt quá kích thước của mảng hoặc chuỗi.
* Quá trình cấp phát và giải phóng bộ nhớ được thực hiện tự động.
* Cơ chế xử lý lỗi giúp việc xử lý và phục hồi lỗi dễ dàng hơn.
* Tính bảo mật
* Java cung cấp một môi trường quản lý chương trình với nhiều mức khác nhau.

Mức 1 : Chỉ có thể truy xuất dữ liệu cũng như phương phức thông qua giao diện mà lớp cung cấp.

Mức 2 : Trình biên dịch kiểm soát các đoạn mã sao cho tuân thủ các quy tắc của ngôn ngữ lập trình Java trước khi thông dịch.

Mức 3 : Trình thông dịch sẽ kiểm tra mã byte code xem các đoạn mã này có đảm bảo được các quy định, quy tắc trước khi thực thi.

Mức 4: Java kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

* **Máy ảo java(jmv – java virtual machine)**

Để đảm bảo tính đa nền, Java sử dụng cơ chế Máy ảo của Java. ByteCode là ngôn ngữ máy của Máy ảo Java tương tự như các lệnh nhị phân của các máy tính thực.

Một chương trình sau khi được viết bằng ngôn ngữ Java (có phần mở rộng là java) phải được biên dịch thành tập tin thực thi được trên máy ảo Java (có phần mở rộng là .class). Tập tin thực thi này chứa các chỉ thị dưới dạng mã Bytecode mà máy ảo Java hiểu được phải làm gì.

Khi thực hiện một chương trình, máy ảo Java lần lượt thông dịch các chỉ thị dưới dạng Bytecode thành các chỉ thị dạng nhị phân của máy tính thực và thực thi thực sự chúng trên máy tính thực (còn gọi là khả năng khả chuyển).

Máy ảo thực tế đó là một chương trình thông dịch. Vì thế các hệ điều hành khác nhau sẽ có các máy ảo khác nhau. Để thực thi một ứng dụng của Java trên một hệ điều hành cụ thể, cần phải cài đặt máy ảo tương ứng cho hệ điều hành đó.

JVM cung cấp môi trường thực thi cho chương trình Java (còn gọi là khả năng độc lập với nền). Có nhiều JVM cho các nền tảng khác nhau chẳng hạn như: Windows, Linux, và Mac.

A diagram of a computer code

Description automatically generated

**Hình 3:**

Tất cả các chương trình muốn thực thi được thì phải được biên dịch ra mã máy. Mã máy của từng kiến trúc CPU của mỗi máy tính là khác nhau (tập lệnh mã máy của CPU Intel, CPU Solarix, CPU Macintosh ... là khác nhau), vì vậy trước đây một chương trình sau khi được biên dịch xong chỉ có thể chạy được trên một kiến trúc CPU cụ thể nào đó (minh hoạ hình 1.4). Đối với CPU Intel, chúng ta có thể chạy các hệ điều hành nhur Microsoft Windows, Unix, Linux, OS/2, ...

Chương trình thực thi được trên Windows được biên dịch dưới dạng file có đuôi (.EXE) còn trên Linux thì được biên dịch dưới dạng file có đuôi (.ELF), vì vậy trước đây một chương trình chạy được trên Windows muốn chạy được trên hệ điều hành khác như Linux chẳng hạn thì phải chỉnh sửa và biên dịch lại.

Ngôn ngữ lập trình Java ra đời, nhờ vào máy ảo Java mà khó khăn nêu trên đã được khắc phục. Một chương trình viết bằng ngôn ngữ lập trình Java sẽ được biên dịch ra mã của máy ảo java (mã java bytecode). Sau đó, máy ảo Java chịu trách nhiệm chuyển mã java bytecode thành mã máy tương ứng. Sun Microsystem chịu trách nhiệm phát triển các máy ảo Java chạy trên các hệ điều hành trên các kiến trúc CPU khác nhau (minh hoạ hình 1.4).

A diagram of a computer process

Description automatically generated

**Hình 1:** Chương trình truyền thống

A diagram of computer components

Description automatically generated

**Hình 1:** Chương trình JAVA

Java bỏ qua cơ chế cấp phát và giải phóng con trỏ bộ nhớ của C vì việc quản lý bộ nhớ trong ứng dụng viết bởi java được chuyển cho máy ảo java đảm nhiệm. Do vậy, một vùng nhớ sau khi được cấp phát sẽ được giải phóng vào lúc nào? Đó chính là khi vùng nhớ không được tham chiếu đến bởi một biến nào. Máy ảo sẽ dò tìm và giải phóng những vùng nhớ vô dụng giúp cho chúng ta khi cần thiết.

#### netbear

NetBeans IDE là một công cụ hỗ trợ lập trình viết mã code miễn phí được cho là tốt nhất hiện nay, được sử dụng chủ yếu cho các lập trình viên phát triển Java tuy nhiên phần mềm có dung lượng khá là nặng dành cho các máy cấu hình có RAM, CPU tương đối cao để vận hành.

NetBeans IDE là môi trường phát triển tích hợp và cực kỳ cần thiết cho các lập trình viên, công cụ này có thể hoạt động tốt với rất nhiều nền tảng hệ điều hành khác nhau như Linux, Windows, MacOS,... là một mã nguồn mở cung cấp các tính năng cần thiết nhất nhăm tại ra các ứng dụng web, thiết bị di động, desktop.

NetBeans IDE hỗ trợ rất nhiều những ngôn ngữ lập trình như Python, Ruby, JavaScript, Groovy, C / C + +, và PHP.

Qua nhiều phiên bản cũ, nay công cụ lập trình NetBeans IDE đã cập nhật lên phiên bản 8.2 mới nhất có rất nhiều thay đổi và nâng cấp so với các phiên bản cũ hơn có thể kể đến như:

* Nâng cao ngôn ngữ lập trình C và C++.
* Nâng cấp, cải tiến các công cụ soạn thảo Profiler và Java.
* Hỗ trợ Docker, PHP 7.
* ECMAScript 6 và hỗ trợ thử nghiệm ECMAScript 7.
* Nâng cấp, cải tiến HTML 5 và Javascript.

Để có thể cài đặt được NetBeans IDE các bạn phải cài đặt và cấu hình sẵn JDK trong máy thì mới cài đặt được. NetBean IDE được chia thành nhiều gói khác nhau để người dùng có thể tải về từng phần, tiết kiệm dung lượng, tùy theo nhu cầu của mình.

Giống với Notepad++, NetBean IDE có gợi ý cú pháp khá tốt, và còn nhiều tính năng khá thú vị như tự động căn chỉnh mã nguồn, Remote FTP, SVN …. Chính vì thế, nó được xem là môi trường phát triển không thể bỏ qua cho các lập trình viên

### Khái quát ứng dụng

**Java và lập trình ứng dụng console**

Ứng dụng Console là ứng dụng nhập xuất ở chế độ văn bản tương tự như màn hình Console của hệ điều hành MS-DOS. Lọai chương trình ứng dụng này thích hợp với những ai bước đầu làm quen với ngôn ngữ lập trình java.

Các ứng dụng kiểu Console thường được dùng để minh họa các ví dụ cơ bản liên quan đến cú pháp ngôn ngữ, các thuật toán, và các chương trình ứng dụng không cần thiết đến giao diện người dùng đồ họa.

**VD 2.1:** Chương trình in ra lời chào “HELLO WORLD”.

|  |
| --- |
| public class vd2{  public static void main(String[] args) {  System.out.println("hello world!");  }  } |
| * **kết quả:** |
| Hello world! |

Thường thì ứng dụng Console được sử dụng rất rộng rãi, nó được dùng thường xuyên để minh họa ngôn ngữ, thuật toán, dùng để xử lý khối lượng công việc ở hậu trường không cần đến giao diện hay tương tác nhiều.

Tuy nhiên, Windows là một thế giới mạnh về giao diện đồ họa và hầu như phần lớn người dùng khi sử dụng chương trình đều thích tương tác với những thành phần đồ họa kiểu nút nhấn, combobox, menu... Tạo một ứng dụng có giao diện đồ họa kiểu Windows sẽ rất dễ nếu bạn dùng những ngôn ngữ trực quan trong môi trường Microsoft Windows như Visual Basic, Delphi, PowerBuilder... Java cũng không thua kém gì.

**Java và phát triển ứng dụng desktop dùng awt và swing(jcf)**

Việc phát triển các chương trình ứng dụng có giao diện người dùng đồ họa trực quan giống như những chương trình được viết dùng ngôn ngữ lập trình VC++ hay Visual Basic đã được java giải quyết bằng thư viện AWT và JFC.

Lập trình ứng dụng Desktop với thư viện JFC (Java Foundation Class) chính là câu trả lời của Java về giao diện đồ họa thay thế Visual Basic của Microsoft.

JFC là thư viện rất phong phú và hỗ trợ mạnh mẽ hơn nhiều so với AWT. JFC giúp cho người lập trình có thể tạo ra một giao diện trực quan của bất kỳ ứng dụng nào.

Đặc điểm nổi bật của giao diện đồ họa viết bằng thư viện JFC của Java là khả năng “Look & Feel” (Cảm quan về giao diện). Bạn có thể dễ dàng thay đổi giao diện theo nhiều kiểu dáng (style) khác nhau mà không cần phải viết lại mã chương trình.

Có thể nói đây là khả năng rất mạnh của Java hơn hẳn những ngôn ngữ và môi trường lập trình trên Windows khác, như Visual Basic, Delphi hay C++. Với khả năng này Java có thể chạy và tương thích giao diện với rất nhiều hệ điều hành. Bản thân ứng dụng Java khi chạy trên một hệ điều hành như Windows vẫn có thể trình bày giao diện đặc thù của mình như trên các hệ điều hành Mac, Macintosh hay Sun Solarix.

Bạn có thể dùng JFC để xây dựng các ứng dụng Windows truyền thống. Nếu muốn sử dụng môi trường giao diện trực quan (IDE) với các công cụ kéo thả như trong Visual Basic, bạn có thể sử dụng Netbeans, Eclipse, Jdeveloper, Visual Studio Code, ...

**Java và ứng dụng applet**

Java Applet là loại ứng dụng có thể nhúng và chạy trong trang web của một trình duyệt web. Từ khi internet mới ra đời, Java Applet cung cấp một khả năng lập trình mạnh mẽ cho các trang web. Nhưng gần đây, khi các chương trình duyệt web đã phát triển với khả năng lập trình bằng VB Script, Java Script, HTML, DHTML, XML, ... cùng với sự canh tranh khốc liệt của Microsoft và Sun đã làm cho Java Applet lu mờ.

Và cho đến bây giờ gần như các lập trình viên đều không còn “mặn mà” với Java Applet nữa (trình duyệt IE đi kèm trong phiên bản Windows 2000 đã không còn mặc nhiên hỗ trợ thực thi một ứng dụng Java Applet).

Micorosoft từ phiên bản Windows 2000 với trình duyệt Internet Explorer 6.0 trở đi không còn hỗ trợ Java Applet mặc định nữa. Muốn chạy Applet bạn phải cài thêm bộ hỗ trợ (plug-in) của Java cho trình duyệt.

Microsoft đưa ra một dạng khác thay thế Applet đó là công nghệ ActiveX với khả năng đưa ứng dụng Desktop truyền thống vào chạy trên trang Web của trình duyệt tương tự Applet. Tuy nhiên, ActiveX vẫn không được đánh giá cao khi chạy chung với trình duyệt bằng Applet.

Applet của Java bảo đảm được tính an toàn, tin cậy (trust) và ít có khả năng gây lỗ hổng về bảo mật, chạy được trên nhiều trình duyệt khác nhau. Applet vẫn là một lựa chọn tốt nhất (tốt hơn ActiveX) khi bạn muốn chuyển ứng dụng Desktop viết bằng Java sang chạy trên nền Web và Internet.

**Java và lập trìnhg mạng**

Có thể nói Java hỗ trợ rất mạnh về lập trình ứng dụng mạng với các lớp thư viện socket giúp đơn giản hóa quá trình kết nối và chuyển dữ liệu trên mạng. Với Java bạn có thể lập trình trên rất nhiều giao thức, TCP/IP, UDP, HTTP, FTP, Telnet, các giao thức tự định nghĩa (User Defined Protocol), ...

Những ứng dụng điển hình về lập trình mạng đó là: các chương trình Chat (tán gẫu), chương trình chuyển dữ liệu (FTP), truyền tin, thông báo tin tức (News), trình duyệt Web (Browser) trao đổi trực tuyến (Instant Messenger), gửi nhận thư điện tử (e- mail), chương trình Mail Server, Web Server ...

**Java và lập trình web**

Java hỗ trợ mạnh mẽ đối với việc phát triển các ứng dụng Web thông qua công nghệ J2EE (Java 2 Enterprise Edition).

Công nghệ J2EE hoàn toàn có thể tạo ra các ứng dụng Web một cách hiệu quả không thua kém công nghệ .NET mà Microsft đang quảng cáo.

Hiện nay có rất nhiều trang Web nổi tiếng ở Việt Nam cũng như khắp nơi trên thế giới được xây dựng và phát triển dựa trên nền công nghệ Java. Số ứng dụng Web được xây dựng dùng công nghệ Java chắc chắn không ai có thể biết được con số chính xác là bao nhiêu, nhưng chúng tôi đưa ra đây vài ví dụ để thấy rằng công nghệ Java của Sun là một “đối thủ đáng gờm” của Microsoft.

**Java và lập trình cơ sỡ dữ liệu**

Cơ sở dữ liệu là yếu tố không thể thiếu trong hầu hết các ứng dụng. Việc hỗ trợ truy xuất cơ sở dữ liệu không phải lúc nào cũng đầy đủ trong các ngôn ngữ lập trình.

Ví dụ như Visual Basic là môi trường lập trình Windows cung cấp rất nhiều cơ chế truy xuất dữ liệu thuận tiện như ODBC, ADO, ADO.NET, OLE DB trong khi với C/C++ thì tiện ích và thư viện truy xuất cơ sở dữ liệu không có nhiều. Lập trình các ứng dụng quản lý liên quan đến dữ liệu bằng C/C++ là công việc không đơn giản.

Java nhận thức rất rõ điều này và hỗ trợ cơ chế truy xuất cơ sở dữ liệu mở với công nghệ JDBC (Java Database Connectivity). JDBC là chuẩn đặc tả cho phép lập trình truy xuất đến rất nhiều loại cơ sở dữ liệu khác nhau như Oracle, DB2, MySQL, SQLServer.

Java hoàn toàn có thể dùng để lập trình các ứng dụng quản lý, các ứng dụng cần truy xuất dữ liệu dễ dàng. JDBC còn cho phép bạn truy cập dữ liệu thông qua mô hình đa tầng (n-tier) rất thích hợp cho các ứng dụng phân tán.

**Java và phát triển các ứng dụng nhúng**

Java Sun đưa ra công nghệ J2ME (The Java 2 Platform, Micro Edition J2ME) hỗ trợ phát triển các chương trình, phần mềm nhúng. J2ME cung cấp một môi trường cho những chương trình ứng dụng có thể chạy được trên các thiết bị cá nhân như: điện thoại di động, máy tính bỏ túi PDA hay Palm, cũng như các thiết bị nhúng khác.

Bạn có thể tìm hiểu chi tiết hơn về công nghệ J2ME tại địa chỉ: **http://java.sun.com/j2me/**

A person touching a digital screen

Description automatically generated

**Hình 1:**

**Java làm ứng dụng android**

A cell phone on a table

Description automatically generated

**Hình 7**

Vài năm trở lại đây, **Android** đã cung cấp nhiều sự thúc đẩy cần thiết và ngày nay nhiều lập trình viên Java là nhà phát triển Ứng dụng Android.

Có thể Android ứng dụng JVM theo cách khác nhau và cách đóng gói khác nhau, nhưng mã vẫn được viết bằng Java.

Có một sự cạnh tranh khi Google hỗ trợ Kotlin như **ngôn ngữ First class trên Android**. Tuy nhiên, lập trình ứng dụng Android với Java vẫn cực kỳ phổ biến.

Ngoài ra, Java còn được ứng dụng rất nhiều trong các lĩnh vực như: dịch vụ tài chính, phát triển phần mềm, dữ liệu lớn (BigData), các ứng dụng khoa học, ...

A diagram of a software application

Description automatically generated

**Hình 1:** Java ột nền (platfrom) hệ điều hành và ứng dụng internet

Java không đơn thuần là ngôn ngữ lập trình, Java còn hơn thế nữa! Java có thể dùng thay thế hệ điều hành với một tập các hàm API (Application Programming Interface) phong phú. Mọi yêu cầu của hệ điều hành Java đều hỗ trợ như xuất nhập tập tin, quản lý người dùng bảo mật, cơ sở dữ liệu, xử lý toán học, truy xuất mạng, kết nối Internet. Máy ảo Java (JVM - Java Vitural Machine) đóng vai trò là một máy tính giúp Java làm được mọi thứ.

**CÁC PHIÊN BẢN JAVA**

A diagram of different devices

Description automatically generated

**Hình 1:**

**Java Standard Edition (Java SE)** – Là một nền tảng cơ bản cho phép phát triển giao diện điều khiển, các ứng dụng mạng và các ứng dụng dạng Win Form.

**Java Enterprise Edition (Java EE)** – Được xây dựng trên nền tảng Java SE, giúp phát triển các ứng dụng web, các ứng dụng ở cấp doanh nghiệp, …

**Java Mobile Edition (Java ME)** – Là một nền tảng cho phép phát triển các ứng dụng nhúng vào các thiết bị điện tử như mobile,…

* **CÁC THÀNH PHẦN CỦA JAVA SE PLATFROM**

Gồm 2 thành phần:

**JRE (Java Runtime Environment)**: cung cấp JVM  (Java Virtual Machine) và thư viện được sử dụng để chạy chương trình Java.

**JDK (Java Development Kit)**: được biết đến như bộ cung cụ phát triển Java, bao gồm: trình biên dịch và trình gỡ rối được sử dụng để phát triển các ứng dụng Java.

A table with different colored text

Description automatically generated with medium confidence

**Hình 1:**

## CÁC LỆNH ĐIỀU KHIỂN CHƯƠNG TRÌNH

### Cấu trúc rẽ nhánh

**khái niệm**

Câu lệnh if được sử dụng để kiểm tra một biểu thức lý luận nào đó có đúng hay không. Nếu đúng thì thực thi khối lệnh; ngược lại nếu sai thì bỏ qua khối lệnh đó.

* **phân loại**

Có 3 loại if trong java

|  |  |  |
| --- | --- | --- |
| **STT** | **Phân loại** | **Chi tiết** |
| 1 | Câu lệnh if | Lệnh if |
| Lệnh if...else |
| Lệnh if...else if..else |
| Các lệnh if lồng nhau |

#### lệnh if

Lệnh if hay còn gọi là lệnh if thiếu, dùng để thực hiện một hành động mà chỉ có một trường hợp duy nhất.

**Cú pháp:**

if(điều kiện) {

câu lệnh 1;

câu lệnh 2;

câu lệnh 3;

...

}

// hoặc

if(điều kiện) câu lệnh 1;

Trong đó:

* Từ khóa if phải viết bằng chữ thường
* Kết quả của biểu thức luận lý phải là đúng hoặc sai.

Lưu ý:

* Không đặt dấu chấm phẩy sau câu lệnh if.
* Biểu thức luận lý phải đặt trong cặp dấu ( ).
* Nếu khối lệnh từ 2 lệnh trở lên thì phải đặt trong dấu { }.

**Lưu đồ:**

Diagram

Description automatically generated

**Hình 1:** Lưu đồ lệnh if

|  |
| --- |
| **VD1:** Nhập vào 2 số nguyên a, b. Tìm và in ra số lớn nhất (sử dụng lệnh if thiếu). |
| import java.util.Scanner;  public class vd1 {  public static void main(String[] args){  int a, b, max;  Scanner sc = new Scanner(System.in);  System.out.print("nhap vao so nguyen a va b:");  a = sc.nextInt();  b = sc.nextInt();  //Tìm số lớn hơn  max = a;  if(b > a)  max = b;  System.out.println("so lon nhat la: " + max);  }  } |
| **Kết quả:** |
| nhap vao so nguyen a va b:4 5  so lon nhat la: 5 |

##### Câu lệnh if...else

Lệnh if … else hay còn gọi là lệnh if đủ, dùng để thực hiện một hành động mà chỉ có hai trường hợp. Một trường hợp khi đúng và một trường hợp khi sai.

**Cú pháp:**

if(điều kiện) {

câu lệnh 1;

câu lệnh 2;

câu lệnh 3;

...

}

else {

câu lệnh 1;

câu lệnh 2;

câu lệnh 3;

...

}

// hoặc

if(điều kiện) câu lệnh 1;

else câu lệnh 1;

Trong đó:

* Từ khóa if, else phải viết bằng chữ thường.
* Kết quả của biểu thức luận lý phải là đúng hoặc sai.

**Lưu ý:**

* Không đặt dấu chấm phẩy sau câu lệnh if, else.
* Biểu thức luận lý phải đặt trong cặp dấu ( ).
* Nếu khối lệnh từ 2 lệnh trở lên thì phải đặt trong dấu { }.

**Lưu đồ:**

**A diagram of a flowchart

Description automatically generated**

**Hình 12** lưu đồ lệnh if...else

Nếu biểu thức logic đúng thì thực hiện khối lệnh 1 và thoát khỏi if. Ngược lại thực hiện khối lệnh 2 và thoát khỏi if.

|  |
| --- |
| **VD2**: nhập váo 1 số nguyên. Tìm chãn lẻ |
| import java.util.Scanner;  public class vd2 {  public static void main(String[] args){  int n;  Scanner sc = new Scanner(System.in);  System.out.print("nhap vao so nguyen n:");  n = sc.nextInt();  //kiem tra tinh chan le  if(n % 2 == 0){  System.out.println(n + " la so chan");  }else{  System.out.println(n + " la so le");  }  }  } |
| * **Kết quả:** |
| nhap vao so nguyen n:10  10 la so chan |

##### lệnh if...else if...else

Lệnh if … else if … else hay còn gọi là lệnh else if, dùng để thực hiện một hành động có từ ba trường hợp trở lên.

**Cú pháp:**

if (điều kiện 1) {

câu lệnh 1;

câu lệnh 2;

...

} else if (điều kiện 2) {

câu lệnh 1;

câu lệnh 2;

...

} else {

câu lệnh 1;

câu lệnh 2;

...

}

Trong đó:

* Từ khóa if, else if, else phải viết bằng chữ thường
* Kết quả của biểu thức luận lý phải là đúng hoặc sai.

**Lưu ý:**

* Không đặt dấu chấm phẩy sau câu lệnh if, else if, else.
* Biểu thức luận lý phải đặt trong cặp dấu ( )
* Nếu khối lệnh từ 2 lệnh trở lên thì phải đặt trong dấu { }.

**Lưu đồ:**

**A diagram of a flowchart

Description automatically generated**

**Hình 13** lưu đồ lệnh if...else if...else

Nếu biểu thức luận lý 1 đúng thì thực hiện khối lệnh 1 và thoát khỏi cấu trúc if.

Ngược lại, nếu biểu thức luận lý 2 đúng thì thực hiện khối lệnh 2 và thoát khỏi cấu trúc if …

Ngược lại, nếu biểu thức luận lý n-1 đúng thì thực hiện khối lệnh n-1 và thoát khỏi cấu trúc if

Ngược lại thì thực hiện khối lệnh n.

|  |
| --- |
| **VD3** so sánh 2 số a và b. Tìm lớn, nhỏ, bằng. |
| import java.util.Scanner;  public class vd3 {  public static void main(String[] args) {  int a, b;  Scanner sc = new Scanner(System.in);  System.out.print("nhap vao a: ");  a = sc.nextInt();  System.out.print("nhap vao b: ");  b = sc.nextInt();  // so sanh  if(a > b){  System.out.print(a +" lon hon " + b);  }else if(a == b){  System.out.print(a + " bang " + b);  }else{  System.out.print(a + " nho hon " +b);  }  }  } |
| * **Kết quả:** |
| nhap vao a: 5  nhap vao b: 9  5 nho hon 9 |

##### Câu lệnh if lồng nhau

Trong lập trình JAVA, ta có thể sử dụng các lệnh if lồng nhau. Thường cấu trúc if lồng nhau càng nhiều cấp thì độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

**Lưu ý:**

Các lệnh if … else lồng nhau thì else sẽ luôn luôn kết hợp với if nào chưa có else gần nhất. Vì vậy khi gặp những lệnh if không có else, bạn phải đặt chúng trong những khối lệnh rõ ràng để tránh bị hiểu sai câu lệnh.

|  |
| --- |
| **VD4:** Tìm x của phương trình ax + b = 0. |
| import java.util.Scanner;  public class vd4 {  public static void main(String[] args) {  double a, b, x;  Scanner sc = new Scanner(System.in);  System.out.print("nhap vao a:");  a = sc.nextDouble();  System.out.print("nhap vao b:");  b = sc.nextDouble();  if(a == 0){  if(b == 0){  System.out.print("pt vo so nghiem!");  }else{  System.out.print("pt vo nghiem!");  }  }else{  x = -b/a;  System.out.print("x cua pt la " + x);  }  }  } |
| * **Kết quả** |
| nhap vao a:2  nhap vao b:1  x cua pt la -0.5 |

#### Câu lệnh switch

**Khái niệm**

Lệnh switch case là một cấu trúc điều khiển và rẽ nhánh hoàn toàn có thể được thay thế bằng cấu trúc if … else. Tuy nhiên, nó cũng có mặt hạn chế là kết quả của biểu thức phải là giá trị hằng nguyên (có giá trị cụ thể).

Việc sử dụng switch sẽ giúp code của chúng ta dễ viết, dễ đọc và có hiệu năng tốt hơn so với sử dụng if … else trong trường hợp có nhiều điều kiện có thể xảy ra.

**Phân loại**

|  |  |
| --- | --- |
| **Tên** | **Cú pháp** |
| switch thiếu | switch (Biểu thức) {  case <Giá trị 1> :  <Khối lệnh 1>  break;  case <Giá trị 2> :  <Khối lệnh 2>  break;  case <Giá trị n> :  <Khối lệnh 3>  break;  } |
| switch đủ | switch (Biểu thức) {  case <Giá trị 1> :  <Khối lệnh 1>  break;  case <Giá trị 2> :  <Khối lệnh 2>  break;  case <Giá trị n> :  <Khối lệnh 3>  break;  default :  <Khối lệnh n>  break;  } |

Trong đó:

* Từ khóa switch, case, default, break phải viết bằng chữ thường.
* Biểu thức bắt buộc là giá trị hằng, có thể là biểu thức nhưng kết quả cần là hằng số nguyên (char, int, long,…)
* Khối lệnh 1, 2… n có thể gồm nhiều câu lệnh, nhưng không cần đặt trong cặp dấu { }

**Lưu đồ:**

A diagram of a flowchart

Description automatically generated

**Hình 14** Lưu đồ switch

|  |
| --- |
| **VD5:** Viết chương trình nhập một ký tự từ (0 🡪 5) đọc ký tự số đó |
| public class vd5 {  public static void main(String[] args) {  try {  System.out.print("hay nhap mot ky tu bat ky:");  int ch = System.in.read();  switch (ch) {  case '0': System.out.println((char)ch + "\tKhong");  break;  case '1': System.out.println((char)ch + "\tMot");  break;  case '2': System.out.println((char)ch + "\tHai");  break;  case '3': System.out.println((char)ch + "\tBa");  break;  case '4': System.out.println((char)ch + "\tBon");  break;  case '5': System.out.println((char)ch + "\tNam");  break;  default:  System.out.println((char)ch+ "khong doc duoc");  break;  }  } catch (Exception e) {  // TODO: handle exception  }  }  } |
| * **Kết quả:** |
| hay nhap mot ky tu bat ky:5   1. Nam |

##### Câu lệnh switch … case

##### Câu lệnh switch … case … default

### Cấu trúc lặp

#### Vòng lặp for

**Khái niệm**

Mục đích của vòng lặp là để giải quyết các công việc có sự lặp đi lặp lại nhiều lần với số lần đã biết trước. Được sử dụng để lặp đi lặp lại 1 khối code có số lần lặp xác định.

**Cú pháp**

for (Khởi tạo giá trị biến lặp; Điều kiện lặp; Cập nhật biến lặp)

{

// Khối lệnh lặp

}

Vòng lặp for được định nghĩa bởi từ khóa for và được chia làm 3 phần chính, mỗi phần được ngăn cách bởi dấu chấm phẩy(;)

* **Phần khởi tạo biến lặp:** Khởi tạo biến vòng lặp có thể được khai báo và khởi tạo giá trị ngay bên trong phần khởi tạo của vòng lặp for. Phần khởi tạo biến được thực thi đầu tiên và chỉ thực thi 1 lần duy nhất trong vòng lặp for.
* **Điều kiện lặp:** Khối lệnh của vòng lặp for sẽ được thực hiện nếu **Biểu thức điều kiện lặp cho giá** **trị đúng**. Vòng lặp for kiểm tra biểu thức điều kiện trước khi thực hiện khối lệnh.
* **Cập nhật biến lặp:** Phần này sẽ được thực thi cuối mỗi lần lặp, sau khi khối lệnh của vòng lặp for được thực thi. Phần này thường chịu trách nhiệm thay đổi giá trị biến vòng lặp được sử dụng trong biểu thức điều kiện (nhằm tránh tình trạng lặp vô hạn). Sau khi thực thi xong phần cập nhật biến vòng lặp, chương trình quay trở lại đánh giá biểu thức điều kiện của vòng lặp for và cứ như thế đến khi biểu thức điều kiện lặp là sai.

**Lưu đồ:**

**A diagram of a work flow

Description automatically generated**

**Hình 15** lưu đồ vòng lặp for

* Bước 1: Khởi tạo giá trị biến lặp (1 lần duy nhất)
* Bước 2: Kiểm tra điều kiện lặp, nếu sai chuyển sang Bước 5, nếu đúng chuyển sang Bước 3.
* Bước 3: Thực hiện nội dung trong thân vòng lặp
* Bước 4: Cập nhật giá trị lặp và quay lại Bước 2
* Bước 5: Kết thúc vòng lặp.

|  |
| --- |
| **VD6**: tính tổng n |
| import java.util.Scanner;  public class vd6 {  public static void main(String[] args) {  int n, sum = 0;  Scanner sc = new Scanner(System.in);  System.out.print("nhap vao sum n: ");  n = sc.nextInt();  for(int i = 1; i <= n; i++)  sum += i;    System.out.print("tong n la: " + sum);  }  } |
| * **Kết quả:** |
| nhap vao sum n: 5  tong n la: 15 |

#### Vòng lặp while

**Khái niệm**

Được sử dụng để lặp đi lặp lại 1 khối lệnh mà không biết trước số lần lặp. Vòng lặp thực hiện lặp lại trong khi biểu thức còn đúng.

**Cú pháp:**

while (biểu thức)

{

// Khối lệnh lặp

}

* Từ khóa while phải viết bằng chữ thường
* Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

**Lưu đồ:**

**A diagram of a flowchart

Description automatically generated**

**Hình 16** lưu đồ vòng lặp while

Kiểm tra biểu thức

* Nếu sai thì kết thúc vòng lặp while (khối lệnh không được thi hành 1 lần nào)
* Nếu đúng thì thực hiện khối lệnh Lặp lại kiểm tra biểu thức

**Lưu ý:**

* Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.
* Trong thân while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.
* Trong thân while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).
* Muốn thoát khỏi vòng lặp while tùy ý có thể dùng các lệnh break, goto, return như lệnh for

|  |
| --- |
| **VD7**: viết chương trình in ra từ 1 🡪 n. |
| import java.util.Scanner;  public class vd7 {  public static void main(String[] args) {  int n, i = 0;  Scanner sc = new Scanner(System.in);  System.out.print("nhap n: ");  n = sc.nextInt();  while (i < n) {  i ++;  System.out.println("gia tri: " + i);  }  }  } |
| * **Kết quả:** |
| nhap n: 6  gia tri: 1  gia tri: 2  gia tri: 3  gia tri: 4  gia tri: 5  gia tri: 6 |

#### Vòng lặp do{...}while

**Khái niệm**

do…while trong java là vòng lặp mà xử lý trong nó được thực thi ít nhất một lần. Giống như vòng lặp While thì vòng lặp do…while cũng sẽ thực thi các lệnh trong nó khi biểu thức điều kiện của nó là True. Tuy nhiên khác với vòng lặp while sẽ tiến hành đánh giá biểu thức điều kiện xem là True hay không rồi chạy các lệnh trong vòng lặp, thì vòng lặp do…while sẽ chạy các lệnh trong nó trước, sau đó mới tiến hành đánh giá biểu thức điều kiện để quyết định có lặp các lần lặp tiếp theo hay không.

**Cú pháp:**

do {

// Khối lệnh

}

while (biểu thức);

* Từ khóa do, while phải viết bằng chữ thường
* Nếu khối lệnh bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

**Lưu đồ:**

**A diagram of a flowchart

Description automatically generated**

**Hình 17** lưu đồ vòng lặp do{...} while

Bước 1: Thực hiện khối lệnh

Bước 2: Kiểm tra biểu thức. Nếu biểu thức là Đúng thì quay lại Bước 1. Nếu biểu thức là Sai thì chuyển sang Bước 3 (khối lệnh được thi hành 1 lần).

Bước 3: Kết thúc vòng lặp

**Lưu ý:**

* Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.
* Trong thân do…while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.
* Trong thân do…while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).
* Muốn thoát khỏi vòng lặp do…while tùy ý có thể dùng các lệnh break, goto, return.

|  |
| --- |
| **VD8:** nhập vào số nguyên n. Nếu không phải thì nhập lại! |
| import java.util.Scanner;  public class vd8 {  public static void main(String[] args) {  int n;  Scanner sc = new Scanner(System.in);  do {  System.out.print("nhap vao so nguyen n: ");  n = sc.nextInt();  } while (n < 0);  System.out.print("xong!");  }  } |
| * **Kết quả:** |
| nhap vao so nguyen n: -1  nhap vao so nguyen n: -10  nhap vao so nguyen n: 5  xong! |

### Câu lệnh điều khiển

#### break

Thông thường lệnh break dùng để thoát khỏi vòng lặp không xác định điều kiện dừng hoặc bạn muốn dừng vòng lặp theo điều kiện do bạn chỉ định. Việc dùng lệnh break để thoát khỏi vòng lặp thường sử dụng phối hợp với lệnh if. Lệnh break dùng trong for, while, do…while, switch. Lệnh break thoát khỏi vòng lặp chứa nó.

|  |
| --- |
| **VD9**: In ra những số < 10 trong khoảng [0,20] |
| public class vd9 {  public static void main(String[] args) {  int i;  for(i = 0; i < 20; i++){  if(i >= 10)  break;  System.out.print(" " + i);  }  }  } |
| * **Kết quả:** |
| 0 1 2 3 4 5 6 7 8 9 |

#### continue

Được dùng trong vòng lặp for, while, do…while. Khi lệnh continue thi hành quyền điều khiển sẽ trao qua cho biểu thức điều kiện của vòng lặp gần nhất. Nghĩa là lộn ngược lên đầu vòng lặp, tất cả những lệnh đi sau trong vòng lặp chứa continue sẽ bị bỏ qua không thi hành.

|  |
| --- |
| **VD10:** In ra số lẻ từ 1 đến 10 |
| public class vd10 {  public static void main(String[] args) {  int i = 0;  for (i = 0; i <= 10; i++) {  if(i % 2 == 0)  continue;  System.out.print(" " + i);  }  }  } |
| * **Kết quả:** |
| 1 3 5 7 9 |

# CHƯƠNG 3: BÀI TẬP ỨNG DỤNG

**Bài 1:** Viết chương trình nhập vào bán kính của một hình tròn. Tính chu vi và diện tích hình tròn, in kết quả ra màng hình.

|  |
| --- |
| import java.util.Scanner;  public class bai1 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.println("Nhap vao ban kinh hinh tron:");  double radius = scanner.nextDouble();  double circumference = 2 \* Math.PI \* radius;  double area = Math.PI \* Math.pow(radius, 2);  System.out.println("Chu vi hinh tron la: " + circumference);  System.out.println("Dien tich hinh tron la: " + area);  }  } |

**Bài 2:** Viết chương trình nhập vào 3 số nguyên a, b, c. Tìm và in ra số lớn nhất.

|  |
| --- |
| import java.util.Scanner;  class bai2 {  public static void main(String[] args)  { int a, b, c, ln;  Scanner bonhap = new Scanner(System.in);  System.out.print("Hay nhap 3 so nguyen");  a = bonhap.nextInt();  b = bonhap.nextInt();  c = bonhap.nextInt();  ln = a;  if (ln < b);  ln = b;  if (ln < c);  ln = c;  System.out.println("Gia tri lon nhat: " + ln);  }  } |

**Bài 3:** Viết chương trình tính n! với n là số nguyên dương được nhập vào từ bàn phím.

|  |
| --- |
| import java.util.Scanner;  public class bai3 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.println("Nhap vao so nguyen duong:");  int n = scanner.nextInt();  long factorial = 1;  for (int i = 1; i <= n; i++) {  factorial \*= i;  }  System.out.println("Giai thua cua " + n + " la: " + factorial);  }  } |

**Bài 4:** viết chương trình tính tổng số tự nhiên từ 1 đến10.

|  |
| --- |
| import java.io.\*;  public class bai4  { public static void main(String[] args) {  int s = 0;  for (int i = 1; i <= 10; i++) {  s += i;  }  System.out.println("Tong so tu nhien tu 1 den 10 la: " + s);  }  } |

**Bài 5:** Viết chương trình in các ký tự từ ‘a’ đến ‘z’ và từ ‘A’ đến ‘Z’.

|  |
| --- |
| import java.io.\*;  public class bai5 {  public static void main(String args[])  { char ch='A';  //chu in  System.out.print("Bang chu in hoa :");  while (ch<='Z')  { System.out.print(" " + ch);  ch++;  }  System.out.println(" ");  //chu thuong  System.out.print("Bang chu cai thuong : ");  ch='a';  while (ch<='z')  { System.out.print(" " + ch);  ch++;  }  System.out.println("");  }  } |

**Bài 6:** Viết chương trình in ra các số nguyên dương nhỏ hơn 100 vừa chia hết cho 2 vừa chia hết cho 5.

|  |
| --- |
| import java.io.\*;  public class bai6  { public static void main(String[] args) {  for (int i = 1; i < 100; i++)  { if (i % 2 == 0 && i % 5 == 0)  { System.out.println(i);  }  }  }  } |

**Bài 7:** Viết chường trình đọc một số nguyên có 2 chữ số. Ví dụ: Nhập số: 18 =>Đọc mười tám.

|  |
| --- |
| import java.util.Scanner;  public class bai7 {  public static void main(String[] args) {  String[] hangChuc = {"", "muoi", "hai muoi", "ba muoi", "bon muoi", "nam muoi", "sau muoi", "bay muoi", "tam muoi", "chin muoi"};  String[] hangDonVi = {"", "mot", "hai", "ba", "bon", "nam", "sau", "bay", "tam", "chin"};  Scanner scanner = new Scanner(System.in);  System.out.print("Nhap vao so gom 2 chu so:");  int so = scanner.nextInt();  int chuc = so / 10;  int donVi = so % 10;  String ketQua = "";  if (chuc == 1 && donVi == 0) {  ketQua = "muoi";  } else if (chuc == 1) {  ketQua = hangChuc[chuc] + " " + hangDonVi[donVi];  } else {  ketQua = hangChuc[chuc] + " " + hangDonVi[donVi];  }  System.out.println("So vua nhap la: " + ketQua);  }  } |

**Bài 8:** Viết chương trình nhập vào hai số nguyên dương n với n là số nguyên được nhập vào từ bàn phím , n = 1..10.

|  |
| --- |
| import java.util.Scanner;  public class bai8  { public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.println("Nhap vao so nguyen duong n (1 <= n <= 10):");  int n = scanner.nextInt();  for (int i = 1; i <= 10; i++) {  System.out.println(n + " x " + i + " = " + n \* i);  }  }  } |

**Bài 9:** Viết chương trình nhập vào hai số nguyên dương a và b. Tìm ước số chung lớn nhất và bội số chung nhỏ nhất của a và b.

|  |
| --- |
| import java.util.Scanner;  public class bai9 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.println("Nhap vao so nguyen a:");  int a = scanner.nextInt();  System.out.println("Nhap vao so nguyen b:");  int b = scanner.nextInt();  int uscln = USCLN(a, b);  int bscnn = BSCNN(a, b, uscln);  System.out.println("Uoc chung lon nhat " + a + " và " + b + " là: " + uscln);  System.out.println("Uoc chung nho nhat " + a + " và " + b + " là: " + bscnn);  }  public static int USCLN(int a, int b) {  if (b == 0) return a;  return USCLN(b, a % b);  }  public static int BSCNN(int a, int b, int uscln) {  return (a \* b) / uscln;  }  } |

**Bài 10:** Viết chương trình yêu cầu nhập vào số chia hết cho 3 mà không chia hết cho 4. Nếu nhập sai thì yêu cầu nhập lại và cho phép nhập lại tối đa 4 lần. Nếu nhập đúng thì xuất hiển dòng chữ “chúc mừng bạn!”, nếu quá số lần quy định thì hiển thị dòng chũ “chúc bạn may mắn lần sau”.

|  |
| --- |
| import java.util.Scanner;  public class bai10 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  int count = 0;  while (count < 4) {  System.out.println("Nhap vao so chia het cho 3 ma khong chia het cho 4:");  int number = scanner.nextInt();  if (number % 3 == 0 && number % 4 != 0) {  System.out.println("Chuc mung ban!");  return;  }  count++;  }  System.out.println("Chuc may man lan sau");  }  } |

# CHƯƠNG 4: ĐÁNH GIÁ KẾT QUẢ THỰC HIỆN

**TÀI LIỆU THAM KHẢO**

[https://vietnix.vn](https://vietnix.vn/java-la-gi/)

[https://kungfutech.edu.vn](https://kungfutech.edu.vn/bai-viet/java/cau-truc-re-nhanh-if-else-trong-java)

[https://howkteam.vn](https://howkteam.vn/course/lap-trinh-java-co-ban-den-huong-doi-tuong/switch-trong-java-3852)

[https://viettuts.vn](https://viettuts.vn/java/menh-de-switch-trong-java)

<https://hoclaptrinh.vn>

Tài liệu học tập lập trình JAVA (biên soạn Châu Ngọc Nhung).