Java (phiên âm Tiếng Việt: "Gia-va") là một ngôn ngữ lập trình hướng đối tượng, dựa trên lớp được thiết kế để có càng ít phụ thuộc thực thi càng tốt. Nó 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),[9] nghĩa là mã Java đã biên dịch có thể chạy trên tất cả các nền tảng hỗ trợ Java mà không cần biên dịch lại.[10] Các ứng dụng Java thường được biên dịch thành bytecode có thể chạy trên bất kỳ máy ảo Java (JVM) nào bất kể kiến trúc máy tính bên dưới. Cú pháp của Java tương tự như C và C++, nhưng có ít cơ sở cấp thấp hơn các ngôn ngữ trên. Java runtime cung cấp các khả năng động (chẳng hạn như phản ánh và sửa đổi mã thời gian chạy) thường không có sẵn trong các ngôn ngữ biên dịch truyền thống. Tính đến năm 2019 Java là một trong những ngôn ngữ lập trình phổ biến nhất được sử dụng theo GitHub,[11][12] đặc biệt cho các ứng dụng web máy khách-máy chủ, với 9 triệu nhà phát triển đã được báo cáo.[13]

Java ban đầu được James Gosling tại Sun Microsystems (sau đó đã được Oracle mua lại) phát triển và được phát hành vào năm 1995 như một thành phần cốt lõi của nền tảng Java của Sun Microsystems. Các trình biên dịch Java, máy ảo và thư viện lớp thực thi gốc và tham chiếu ban đầu được Sun phát hành theo giấy phép độc quyền. Kể từ tháng 5 năm 2007, tuân theo các thông số kỹ thuật của Quy trình Cộng đồng Java, Sun đã cấp phép hầu hết các công nghệ Java của mình theo Giấy phép Công cộng GNU. Oracle cung cấp Máy ảo Java HotSpot của riêng mình, tuy nhiên việc triển khai tham chiếu chính thức là OpenJDK JVM, là phần mềm mã nguồn mở miễn phí và được hầu hết các nhà phát triển sử dụng và là JVM mặc định cho hầu hết các bản phân phối Linux.

Tính đến tháng 9 năm 2020, phiên bản mới nhất là Java 15, với Java 11, một phiên bản hỗ trợ dài hạn (LTS), được phát hành vào 25 tháng 9 năm 2018. Oracle phát hành bản cập nhật miễn phí cho công chúng với phiên bản kế thừa Java 8 LTS vào tháng 1 năm 2019 cho mục đích sử dụng thương mại, mặc dù nếu không nó sẽ vẫn hỗ trợ Java 8 với các bản cập nhật công khai cho mục đích sử dụng cá nhân vô thời hạn. Các nhà cung cấp khác đã bắt đầu cung cấp các bản miễn phí của OpenJDK 8 và 11 mà vẫn đang nhận được bảo mật và các nâng cấp khác.

Oracle (và những công ty khác) khuyên người dùng nên gỡ cài đặt các phiên bản Java đã lỗi thời vì những rủi ro nghiêm trọng do các vấn đề bảo mật chưa được giải quyết.[14] Vì Java 9, 10, 12, 13 và 14 không còn được hỗ trợ, Oracle khuyên người dùng nên chuyển ngay sang phiên bản mới nhất (hiện tại là Java 15) hoặc bản phát hành LTS. James Gosling, Mike Sheridan và Patrick Naughton khởi xướng dự án

ngôn ngữ Java vào tháng 6 năm 1991.[15] Java ban đầu được thiết kế cho truyền hình tương tác, nhưng nó quá tiên tiến đối với ngành truyền hình cáp kỹ thuật số vào thời điểm đó.[16] Ban đầu ngôn ngữ này được gọi là Oak theo tên một cây sồi bên ngoài văn phòng của Gosling. Sau đó, dự án có tên là Green và cuối cùng được đổi tên thành Java, từ cà phê Java, loại cà phê đến từ Indonesia.[17] Gosling đã thiết kế Java với cú pháp kiểu C/C++ mà các lập trình viên hệ thống và ứng dụng đã quen thuộc.[18

Sun Microsystems đã phát hành bản triển khai công khai đầu tiên dưới dạng Java 1.0 vào năm 1996.[19] Nó hứa hẹn khả năng Viết một lần, Chạy mọi nơi (WORA), cung cấp thời gian chạy miễn phí trên các nền tảng phổ biến. Khá an toàn và có tính năng bảo mật có thể định cấu hình, nó cho phép các hạn chế truy cập mạng và tệp. Các trình duyệt web lớn đã sớm kết hợp khả năng chạy các ứng dụng Java trong các trang web và Java nhanh chóng trở nên phổ biến. Trình biên dịch Java 1.0 được viết lại bằng Java bởi Arthur van Hoff để tuân thủ nghiêm ngặt đặc tả ngôn ngữ Java 1.0.[20] Với sự ra đời của Java 2 (ban đầu được phát hành với tên gọi J2SE 1.2 vào tháng 12 năm 1998 – 1999), các phiên bản mới có nhiều cấu hình được xây dựng cho các loại nền tảng khác nhau. J2EE bao gồm các công nghệ và API cho các ứng dụng doanh nghiệp thường chạy trong môi trường máy chủ, trong khi các API đặc trưng của J2ME được tối ưu hóa cho các ứng dụng di động. Phiên bản dành cho máy tính để bàn được đổi tên thành J2SE. Năm 2006, vì mục đích tiếp thị, Sun đã đổi tên các phiên bản J2 mới lần lượt là Java EE, Java ME và Java SE.

Năm 1997, Sun Microsystems đã tiếp cận cơ quan tiêu chuẩn ISO/IEC JTC 1 và sau đó là Ecma International để chính thức hóa Java, nhưng sau đó công ty nhanh chóng rút khỏi quy trình này.[21][22][23] Java vẫn là một tiêu chuẩn thực tế, được kiểm soát thông qua Quy trình cộng đồng Java.[24] Đã có lúc, Sun cung cấp hầu hết các triển khai Java của mình mà không tính phí, bất chấp trạng thái phần mềm độc quyền của họ. Sun đã tạo ra doanh thu từ Java thông qua việc bán giấy phép cho các sản phẩm chuyên biệt như Hệ thống Doanh nghiệp Java.

Vào ngày 13 tháng 11 năm 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). Vào ngày 8 tháng 5 năm 2007, Sun đã hoàn thành quá trình, cung cấp tất cả mã cốt lõi của JVM theo các điều khoản phân phối phần mềm miễn phí / nguồn mở, ngoại trừ một phần nhỏ mã mà Sun không giữ bản quyền.[25][26] Phó chủ tịch Rich Green của Sun nói rằng vai trò lý

tưởng của Sun đối với Java là như một nhà truyền giáo.[27] Sau khi Tập đoàn Oracle mua lại Sun Microsystems vào năm 2009–10, Oracle đã tự mô tả mình là người quản lý công nghệ Java với cam kết không ngừng thúc đẩy cộng đồng tham gia và minh bạch.[28] Điều này không ngăn được Oracle đệ đơn kiện Google ngay sau đó vì đã sử dụng Java bên trong Android SDK.

Ngày 2 tháng 4 năm 2010, James Gosling từ chức tại Oracle.[29]

Vào tháng 1 năm 2016, Oracle đã thông báo rằng môi trường thời gian chạy Java dựa trên JDK 9 sẽ ngừng cung cấp plugin trình duyệt.[30]

Phần mềm Java chạy trên mọi thứ, từ máy tính xách tay đến trung tâm dữ liệu, bảng điều khiển trò chơi đến siêu máy tính khoa học.[31]

Nguyên tắc

Có 5 mục tiêu chính trong việc xây dựng ngôn ngữ Java:[32]Nó phải đơn giản, hướng đối tượng và quen thuộc.

Nó phải mạnh mẽ và an toàn.

Nó phải là kiến trúc trung lập và di động.

Nó phải thực thi với hiệu suất cao.

Nó phải được thông dịch, phân luồng và động.

Java JVM và bytecode

Một mục tiêu thiết kế của Java là tính di động, có nghĩa là các chương trình được viết cho nền tảng Java phải chạy tương tự nhau trên bất kỳ sự kết hợp nào giữa phần cứng và hệ điều hành với hỗ trợ thời gian chạy thích hợp. Điều này đạt được bằng cách biên dịch mã ngôn ngữ Java sang một đại diện trung gian được gọi là Java bytecode, thay vì trực tiếp tới mã máy cụ thể về kiến trúc. Các lệnh mã bytecode trong Java tương tự như mã máy, nhưng chúng được thiết kế để thực thi bởi một máy ảo (VM) được viết riêng cho phần cứng máy chủ. Người dùng cuối thường sử dụng Java Runtime Environment (JRE) được cài đặt trên máy của họ cho các ứng dụng Java độc lập hoặc trong trình duyệt web cho các ứng dụng Java.

Các thư viện tiêu chuẩn cung cấp một cách chung để truy cập các tính năng dành riêng cho máy chủ như đồ họa, phân luồng và mạng. Việc sử dụng bytecode phổ biến làm cho việc chuyển cổng trở nên đơn giản. Tuy nhiên, chi phí của việc thông dịch bytecode thành các lệnh máy làm cho các chương trình được thông dịch hầu

như luôn chạy chậm hơn các chương trình thực thi gốc. Các trình biên dịch Just-in-time (JIT) biên dịch mã byte thành mã máy trong thời gian chạy đã được giới thiệu từ giai đoạn đầu. Bản thân Java độc lập với nền tảng và được điều chỉnh cho phù hợp với nền tảng cụ thể mà máy ảo Java (JVM) chạy trên nó, máy này sẽ dịch mã bytecode của Java sang ngôn ngữ máy của nền tảng.[39] Hiệu suất

Các chương trình được viết bằng Java nổi tiếng là chậm hơn và đòi hỏi nhiều bộ nhớ hơn các chương trình được viết bằng C++.[40][41] Tuy nhiên, tốc độ thực thi của các chương trình Java được cải thiện đáng kể với sự ra đời của tính năng biên dịch đúng lúc vào năm 1997/1998 cho Java 1.1,[42] việc bổ sung các tính năng ngôn ngữ hỗ trợ phân tích mã tốt hơn (chẳng hạn như các lớp bên trong, lớp StringBuilder, các xác nhận tùy chọn, v.v.) và tối ưu hóa trong máy ảo Java, chẳng hạn như HotSpot trở thành mặc định cho JVM của Sun vào năm 2000. Với Java 1.5, hiệu suất đã được cải thiện với việc bổ sung gói java.util.concurrent, bao gồm khóa các triển khai miễn phí của ConcurrentMaps và các bộ sưu tập đa lõi khác và nó đã được cải thiện hơn nữa với Java 1.6.

Không JVM

Một số nền tảng cung cấp hỗ trợ phần cứng trực tiếp cho Java; có những bộ điều khiển vi mô có thể chạy Java bytecode trong phần cứng thay vì máy ảo Java phần mềm,[43] và một số bộ xử lý dựa trên ARM có thể có hỗ trợ phần cứng để thực thi Java bytecode thông qua tùy chọn Jazelle của chúng, mặc dù hỗ trợ hầu hết đã bị loại bỏ trong các triển khai hiện tại của ARM.

Quản lý bộ nhớ tự động

Java sử dụng bộ thu gom rác tự động (AGC) để quản lý bộ nhớ trong vòng đời đối tượng. Lập trình viên xác định thời điểm các đối tượng được tạo và thời gian chạy Java chịu trách nhiệm khôi phục bộ nhớ khi các đối tượng không còn được sử dụng. Khi không còn tham chiếu đến một đối tượng, bộ nhớ không thể truy cập sẽ đủ điều kiện để được giải phóng tự động bởi bộ thu gom rác. Một cái gì đó tương tự như rò rỉ bộ nhớ vẫn có thể xảy ra nếu mã của lập trình viên giữ một tham chiếu đến một đối tượng không còn cần thiết, thường là khi các đối tượng không còn cần thiết được lưu trữ trong các bộ chứa vẫn đang được sử dụng. Nếu các phương thức cho một đối tượng không tồn tại được gọi, một ngoại lệ con trỏ null sẽ được đưa ra.[44][45] Một trong những ý tưởng đằng sau mô hình quản lý bộ nhớ tự động của Java là các lập trình viên có thể không phải chịu gánh nặng khi phải thực hiện quản lý bộ nhớ thủ công. Trong một số ngôn ngữ, bộ nhớ để tạo các đối tượng được cấp phát ngầm trên ngăn xếp hoặc được cấp phát và phân bổ rõ ràng từ heap. Trong

trường hợp thứ hai, trách nhiệm quản lý bộ nhớ thuộc về lập trình viên. Nếu chương trình không phân bổ một đối tượng, một rò rỉ bộ nhớ sẽ xảy ra. Nếu chương trình cố gắng truy cập hoặc phân bổ bộ nhớ đã được phân bổ, kết quả là không xác định và khó dự đoán, và chương trình có thể trở nên không ổn định hoặc gặp sự cố. Điều này có thể được khắc phục một phần bằng cách sử dụng các con trỏ thông minh, nhưng chúng làm tăng thêm chi phí và sự phức tạp. Lưu ý rằng việc thu gom rác không ngăn chặn rò rỉ bộ nhớ logic, tức là những nơi bộ nhớ vẫn được tham chiếu nhưng không bao giờ được sử dụng

Việc thu gom rác có thể xảy ra bất cứ lúc nào. Lý tưởng nhất, nó sẽ xảy ra khi một chương trình không hoạt động. Nó được đảm bảo sẽ được kích hoạt nếu không có đủ bộ nhớ trống trên heap để cấp phát một đối tượng mới; điều này có thể khiến một chương trình bị dừng trong giây lát. Không thể quản lý bộ nhớ rõ ràng trong Java.Java không hỗ trợ số học con trỏ kiểu C/C++, trong đó địa chỉ đối tượng có thể được thao tác số học (ví dụ: bằng cách thêm hoặc trừ một phần bù). Điều này cho phép bộ thu gom rác di chuyển các đối tượng được tham chiếu và đảm bảo an toàn và bảo mật kiểu.Giống như trong C++ và một số ngôn ngữ hướng đối tượng khác, các biến của kiểu dữ liệu nguyên thủy của Java hoặc được lưu trữ trực tiếp trong các trường (đối với các đối tượng) hoặc trên ngăn xếp (đối với các phương thức) chứ không phải trên heap, điều này thường đúng đối với dữ liệu không nguyên thủy các loại (nhưng hãy xem phân tích thoát). Đây là một quyết định có ý thức của các nhà thiết kế của Java vì lý do hiệu suất.

Java chứa nhiều loại trình thu gom rác. Theo mặc định, HotSpot sử dụng bộ thu gom rác quét song song.[46] Tuy nhiên, cũng có một số trình thu gom rác khác có thể được sử dụng để quản lý đống rác. Đối với 90% ứng dụng trong Java, bộ thu gom rác đồng thời Mark-Sweep (CMS) là đủ.[47] Oracle đặt mục tiêu thay thế CMS bằng Garbage-First Collector (G1).[48]

Giải quyết được vấn đề quản lý bộ nhớ không giúp lập trình viên bớt gánh nặng xử lý đúng cách các loại tài nguyên khác, như kết nối mạng hoặc cơ sở dữ liệu, xử lý tệp, v.v., đặc biệt là khi có lỗi.

Các lớp đặc biệt

Apple Bài chi tiết: Java appletJava applet là các chương trình được nhúng vào trong các ứng dụng khác, thường là trong một trang web hiển thị trong trình duyệt web. API của Java applet hiện không còn được dùng nữa kể từ ngày 9 tháng 1 năm 2017. Servlet Bài chi tiết: Java servlet Công nghệ Java servlet cung cấp cho các nhà phát triển Web một cơ chế nhất quán, đơn giản để mở rộng chức năng của máy

chủ Web và để truy cập các hệ thống kinh doanh hiện có. Servlet là các thành phần Java EE phía máy chủ tạo ra các phản hồi (thường là các trang HTML) cho các yêu cầu (thường là các yêu cầu HTTP) từ máy khách.

Ở một mức độ nào đó, API của Java servlet đã được thay thế bởi hai công nghệ Java cho dịch vụ web:

Java API cho RESTful Web Services (JAX-RS 2.0) hữu ích cho AJAX, JSON và dịch vụ REST, và

Java API cho XML Web Services (JAX-WS) hữu ích cho dịch vụ web SOAP.