# Chương 1

## Cấu trúc cơ bản của chương trình Java

Gồm các thành phần chính:

* Kiểu dữ liệu nguyên thủy
* Câu lệnh
* Mảng
* Phương thức tĩnh (static methods)
* Chuỗi (strings)
* Nhập/xuất (I/O)

## Kiểu dữ liệu nguyên thủy và biểu thức

Gồm số nguyên (int), số thực (double), boolean (boolean), ký tự (char)

**Biểu thức:** Thường có dạng trung tố (toán hạng - toán tử - toán hạng).

**Chuyển đổi kiểu dữ liệu:** Các số sẽ tự động được nâng cấp lên kiểu dữ liệu rộng hơn nếu không làm mất thông tin.

**So sánh: Các toán tử so sánh trong Java gồm:** bằng (==), không bằng (!=), nhỏ hơn (<), nhỏ hơn hoặc bằng (<=), lớn hơn (>), và lớn hơn hoặc bằng (>=).

**Các kiểu dữ liệu nguyên thủy khác:** Để tăng tính linh hoạt, Java cung cấp thêm năm kiểu dữ liệu nguyên thủy khác long, short, char, byte ,float.

## Câu lệnh

**Khai báo (Declarations):** Java yêu cầu sử dụng khai báo để xác định tên và kiểu của các biến, giúp chúng ta rõ ràng hơn về các phép toán mà chúng ta đang chỉ định.

**Câu lệnh gán:** Câu lệnh gán liên kết một giá trị kiểu dữ liệu (được xác định bởi một biểu thức) với một biến.

**Câu lệnh điều kiện**: Trong Java, câu lệnh điều kiện cho phép thực hiện các hành động khác nhau dựa trên đầu vào khác nhau.

**Vòng lặp**: Nhiều phép tính có tính chất lặp đi lặp lại.

**Break và Continue**: Java cung cấp hai câu lệnh bổ sung để kiểm soát luồng lặp trong vòng lặp.

## Ký hiệu rút gọn

Có nhiều cách để diễn đạt một phép tính, và chúng ta nên ưu tiên viết mã rõ ràng, gọn gàng và hiệu quả.

**Khai báo khởi tạo**: Chúng ta có thể kết hợp khai báo với gán giá trị để khởi tạo biến ngay tại thời điểm tạo.

**Gán giá trị ngầm**

**Khối lệnh đơn**: Nếu một khối lệnh trong điều kiện hoặc vòng lặp chỉ có một câu lệnh, các dấu ngoặc nhọn có thể được bỏ qua.

**Cú pháp vòng lặp for:** Hỗ trợ cách lập trình khởi tạo và tăng.

## Mảng

Một mảng lưu trữ một chuỗi giá trị có cùng kiểu dữ liệu. Chúng ta không chỉ muốn lưu trữ các giá trị mà còn muốn truy cập từng giá trị riêng lẻ

**Tạo và khởi tạo mảng**: Để tạo một mảng trong chương trình Java, có ba bước khác nhau:

Khai báo mảng, tạo mảng, khởi tạo giá trị của mảng

**Hình thức ngắn**: Để tiết kiệm mã, chúng ta thường tận dụng quy ước khởi tạo mảng mặc định của Java.

**Sử dụng mảng**: Mã xử lý mảng điển hình cho thấy cách làm việc với mảng sau khi đã khai báo và tạo.

**Aliasing:** là hiện tượng khi hai tên mảng hoặc biến tham chiếu cùng một mảng, dẫn đến việc thay đổi giá trị của một tên sẽ ảnh hưởng đến giá trị của tên còn lại.

**Mảng hai chiều:** trong Java là một mảng của các mảng một chiều.

**Phương thức tĩnh:** cho phép thực hiện một chuỗi các lệnh khi được gọi, mà không cần tạo đối tượng.

**Đinh nghĩa** **phương thức (method)** trong Java bao gồm một chuỗi các lệnh được định nghĩa để thực hiện một phép toán.

Khi **gọi phương thức tĩnh**, các biến tham số của nó sẽ được khởi tạo với các giá trị tương ứng từ các biểu thức trong lời gọi.

**Tính chất của phương thức** trong Java: Tham số được truyền theo giá trị, tên phương thức có thể được nạp chồng, phương thức chỉ có một giá trị trả về, phương thức có thể có tác động phụ,

**Đệ quy:** Một kỹ thuật lập trình trong đó một phương thức gọi chính nó.

**Mô hình lập trình cơ bản:** trong Java thường xoay quanh việc phát triển một chương trình giải quyết một nhiệm vụ tính toán cụ thể

**Lập trình mô-đun:** một khái niệm quan trọng trong mô hình lập trình Java, nơi mà các thư viện các phương thức tĩnh (modules) cho phép sự tổ chức mã nguồn hiệu quả hơn.

**Unit testing** là một phần quan trọng trong quy trình phát triển phần mềm, giúp đảm bảo rằng từng phần của mã nguồn hoạt động đúng như mong đợi.

**Thư viện bên ngoài** đóng vai trò quan trọng trong lập trình Java bằng cách cung cấp mã nguồn, hàm và cấu trúc dữ liệu có thể tái sử dụng, giúp đơn giản hóa quá trình phát triển.

## APIs

là một phần quan trọng trong lập trình mô-đun, cung cấp tài liệu giải thích cách hoạt động của các phương thức thư viện mà người khác có thể sử dụng. Bao gồm tên thư viện, chữ ký phương thức, mô tả ngắn

**Mô hình lập trình:**

Lập trình mô-đun không chỉ là việc tạo ra các hàm riêng lẻ, mà còn bao gồm việc chia nhỏ các bài toán phức tạp thành các phần nhỏ dễ quản lý. Mỗi phần được xử lý bởi một thư viện riêng, và các thư viện này được thiết kế để dễ dàng tái sử dụng trong các ứng dụng khác.

**Lợi ích của việc sử dụng thư viện:**

* **Tối ưu hóa**: Thư viện thường cung cấp các cài đặt tối ưu và hiệu quả hơn so với mã tự viết.
* **Độ tin cậy**: Thư viện đã được kiểm thử kỹ lưỡng, giảm thiểu lỗi khi sử dụng.
* **Tính mô-đun**: Phân chia mã thành các phần độc lập giúp việc bảo trì và phát triển dễ dàng hơn.

## String

Văn bản trình bày về kiểu dữ liệu chuỗi (String) trong Java, nhấn mạnh rằng chuỗi là một tập hợp các ký tự và được định nghĩa bằng cách sử dụng dấu ngoặc kép. Mặc dù String không phải là kiểu dữ liệu nguyên thủy, nhưng nó là một loại dữ liệu cơ bản thường được sử dụng trong các chương trình Java.

chương trình, với các tham số được truyền vào dưới dạng một mảng chuỗi.

## Mô Hình Nhập và Xuất Dữ Liệu trong Java

* Mô hình nhập/xuất
* Giao diện giữa Java và hệ điều hành
* Các lệnh và tham số
* Thư viện StdOut
* Đầu ra định dạng
* Thư viện StdIn
* Chuyển hướng và ống dẫn

## Tìm Kiếm Nhị Phân

* **Tìm Kiếm Nhị Phân Là Gì?**
  1. Tìm kiếm nhị phân là một thuật toán tìm kiếm dùng để tìm vị trí của một giá trị mục tiêu trong một mảng đã được sắp xếp.
  2. Thuật toán này hoạt động bằng cách chia khoảng tìm kiếm thành hai nửa. Nếu giá trị mục tiêu nhỏ hơn phần tử ở giữa khoảng, thuật toán sẽ thu hẹp khoảng tìm kiếm xuống nửa dưới. Ngược lại, nó sẽ thu hẹp xuống nửa trên.
  3. Quá trình này tiếp tục cho đến khi tìm thấy giá trị mục tiêu hoặc khoảng tìm kiếm trống.
* **Tại Sao Nên Sử Dụng Tìm Kiếm Nhị Phân?**
  1. Tìm kiếm nhị phân hiệu quả hơn so với tìm kiếm tuần tự, đặc biệt với các tập dữ liệu lớn. Nó yêu cầu ít phép so sánh hơn (độ phức tạp thời gian logarithmic, O(log⁡n)O(\log n)O(logn)) so với độ phức tạp của tìm kiếm tuần tự là O(n)O(n)O(n).