

# THAM KHẢO NHANH JAVA

OOP, collections, streams, xử lý exception cơ bản

## Cơ Bản

### Hello World

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

### Biên Dịch & Chạy

```
javac Main.java # compile
java Main # run
java Main.java # single-file (Java 11+)
```

### Quy Tắc Đặt Tên

**ClassName** PascalCase cho class và interface  
**methodName** camelCase cho method và biến  
**CONSTANT\_NAME** UPPER\_SNAKE cho hằng số  
**com.example.pkg** Tên miền ngược viết thường cho package

### Kiểu Dữ Liệu

#### Kiểu Nguyên Thủy

**byte** 8-bit có dấu (-128 đến 127)  
**short** 16-bit có dấu  
**int** 32-bit có dấu (số nguyên mặc định)  
**long** 64-bit có dấu (hầu to "L")  
**float** 32-bit IEEE-754 (hầu to "f")  
**double** 64-bit IEEE-754 (thập phân mặc định)  
**boolean** true / false  
**char** Ký tự Unicode 16-bit

#### Strings

```
String s = "hello";
String joined = s + " world"; // concatenation
int len = s.length();
String sub = s.substring(0, 3); // "hel"
boolean eq = s.equals("hello"); // content equality
```

#### Ép Kiểu

```
int i = (int) 3.14; // narrowing cast
double d = 1; // widening (auto)
int n = Integer.parseInt("42"); // string to int
String s = String.valueOf(42); // int to string
```

#### Mảng

```
int[] nums = {1, 2, 3};
String[] names = new String[5];
int[][] matrix = new int[3][4];
Arrays.sort(nums);
```

#### Luồng Điều Khiển

##### If / Else

```
if (x > 0) {
    System.out.println("positive");
} else if (x == 0) {
    System.out.println("zero");
} else {
    System.out.println("negative");
}
```

##### Switch

```
// Traditional
switch (day) {
    case "Mon": doWork(); break;
    case "Sat": case "Sun": rest(); break;
    default: routine();
}
// Switch expression (Java 14+)
String type = switch (day) {
    case "Sat", "Sun" -> "weekend";
    default -> "weekday";
};
```

##### Vòng Lặp

```
for (int i = 0; i < 10; i++) { }
for (String s : list) { } // enhanced for
while (condition) { }
do { } while (condition);
```

## Methods

### Định Nghĩa

```
public static int add(int a, int b) {
    return a + b;
}
```

### Varargs & Nạp Chồng

```
static int sum(int... nums) {
    int total = 0;
    for (int n : nums) total += n;
    return total;
}
// sum(1, 2) sum(1, 2, 3) both work
```

### Access Modifiers

**public** Truy cập từ bất kỳ đâu  
**protected** Cùng package + subclass  
**(default)** Chỉ cùng package (không có từ khóa)  
**private** Chỉ cùng class

## Classes & Objects

### Định Nghĩa Class

```
public class User {
    private String name;
    private int age;
    public User(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() { return name; }
}
```

### Records (Java 16+)

```
public record Point(double x, double y) {
    // auto: constructor, getters, equals, hashCode, toString
    public double distance() {
        return Math.sqrt(x * x + y * y);
    }
}
```

### Static & Final

**static** Thuộc về class, không phải instance  
**final field** Không thể gán lại sau khi khởi tạo  
**final method** Không thể override  
**final class** Không thể kế thừa

### Kế Thừa

#### Extends

```
public class Animal {
    public void speak() { System.out.println("..."); }
}
public class Dog extends Animal {
    @Override
    public void speak() { System.out.println("Woof!"); }
}
```

#### Abstract Classes

```
public abstract class Shape {
    abstract double area();
    public void describe() {
        System.out.println("Area: " + area());
    }
}
```

#### Khái Niệm Chính

**super** Gọi constructor hoặc method cha  
**@Override** Kiểm tra override lúc biên dịch  
**instanceof** Kiểm tra kiểu lúc chạy  
**sealed (17+)** Hạn chế class nào có thể extend

## Interfaces

### Định Nghĩa

```
public interface Printable {
    void print(); // abstract
    default String format() { // default method
        return toString();
    }
}
static Printable of(String s) { // static method
    return () -> System.out.println(s);
}
```

### Triển Khai

```
public class Report implements Printable, Serializable {
    @Override
    public void print() {
        System.out.println("Report");
    }
}
```

### Functional Interfaces

**Runnable** `() -> void`  
**Supplier<T>** `() -> T`  
**Consumer<T>** `**Function<T,R>** `**Predicate<T>** `**Comparator<T>** `

## Collections

### List

```
List<String> list = new ArrayList<>();
list.add("a");
list.get(0); // "a"
list.size(); // 1
List<String> immutable = List.of("a", "b", "c");
```

### Map

```
Map<String, Integer> map = new HashMap<>();
map.put("key", 42);
map.getOrDefault("key", 0); // 42
map.containsKey("key"); // true
map.forEach((k, v) -> { });
```

### Set

```
Set<String> set = new HashSet<>();
set.add("a");
set.contains("a");
set.contains("a"); // true
Set<String> immutable = Set.of("a", "b", "c");
```

### Triển Khai Phổ Biến

**ArrayList** Mảng có thể mở rộng, truy cập ngẫu nhiên nhanh  
**LinkedList** Danh sách liên kết đôi, chèn/xóa nhanh  
**HashMap** Bảng hash, get/put O(1)  
**TreeMap** Sắp xếp theo key, O(log n)  
**HashSet** Phân tử duy nhất, lookup O(1)  
**LinkedHashMap** HashMap theo thứ tự chèn

## Xử Lý Exception

### Try / Catch / Finally

```
try {
    int result = 10 / 0;
} catch (ArithmeticException e) {
    System.err.println(e.getMessage());
} finally {
    // always executes
}
```

### Try-with-Resources

```
try (var reader = new BufferedReader(new FileReader(path))) {
    String line = reader.readLine();
} // auto-closes reader
```

### Phân Cấp Exception

**Throwable** Gốc của tất cả error và exception  
**Error** Vấn đề nghiêm trọng (OutOfMemoryError)  
**Exception** Exception đã kiểm tra (phải xử lý)  
**RuntimeException** Chưa kiểm tra (NullPointerException, IndexOutOfBoundsException)

## Exception Tùy Chỉnh

```
public class AppException extends Exception {
    public AppException(String msg) { super(msg); }
    public AppException(String msg, Throwable cause) {
        super(msg, cause);
    }
}
```

## Streams & Lambdas

### Cú Pháp Lambda

Comparator<String> byLen = (a, b) -> a.length() - b.length();  
Runnable task = () -> System.out.println("run");  
Function<String, Integer> len = String::length; // method ref

### Stream Pipeline

```
List<String> result = names.stream()
    .filter(n -> n.length() > 3)
    .map(String::toUpperCase)
    .sorted()
    .collect(Collectors.toList());
```

### Thao Tác Stream Phổ Biến

**.filter(pred)** Giữ phần tử khớp predicate  
**.map(func)** Biến đổi từng phần tử  
**.flatMap(func)** Map và làm phẳng stream lồng nhau  
**.sorted()** Sắp xếp (tự nhiên hoặc với Comparator)  
**.distinct()** Xóa trùng lặp  
**.limit(n)** Lấy n phần tử đầu tiên  
**.collect()** Terminal: gom vào collection  
**.forEach()** Terminal: thực hiện hành động trên từng phần tử  
**.reduce()** Terminal: kết hợp thành một giá trị  
**.count()** Terminal: đếm phần tử

## Generics

### Generic Class & Method

```
public class Box<T> {
    private T value;
    public Box(T value) { this.value = value; }
    public T get() { return value; }
}
public static <T> List<T> listOf(T... items) {
    return List.of(items);
}
```

### Kiểu Ràng Buộc & Wildcard

**<T extends Number>** T phải là Number hoặc subclass  
**<T extends A & B>** Nhiều ràng buộc (class + interface)  
**<?>** Kiểu không xác định (chỉ đọc)  
**<? extends T>** Wildcard căn trên (producer)  
**<? super T>** Wildcard căn dưới (consumer)

## Optional & Java Hiện Đại

### Optional

```
Optional<String> opt = Optional.ofNullable(getValue());
String result = opt.orElse("default");
opt.ifPresent(v -> System.out.println(v));
String upper = opt.map(String::toUpperCase).orElse("");
```

### Text Blocks (Java 15+)

```
String json = """
    {
        "name": "Alice", "age": 30
    }
    """;
```

### Tiện Ích Hữu Ích

**var (10+)** Suy luận kiểu biến cục bộ  
**record (16+)** Class mang dữ liệu bất biến  
**sealed (17+)** Phân cấp class bị hạn chế  
**pattern matching (21+)** `instanceof` với auto-cast  
**virtual threads (21+)** Luồng nhẹ qua `Thread.ofVirtual()`