

Referensi Cepat Java

OOP, collection, stream, penanganan exception

Dasar

Hello World

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

Compile & Jalankan

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

Konvensi Penamaan

ClassName	PascalCase untuk class dan interface
methodName	camelCase untuk method dan variabel
CONSTANT_NAME	UPPER_SNAKE untuk konstanta
com.example.pkg	Domain terbalik huruf kecil untuk package

Tipe Data

Primitif

byte	8-bit bertanda (-128 hingga 127)
short	16-bit bertanda
int	32-bit bertanda (integer default)
long	64-bit bertanda (sufiks L)
float	32-bit IEEE-754 (sufiks f)
double	64-bit IEEE-754 (desimal default)
boolean	true / false
char	Karakter Unicode 16-bit

String

```
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
```

Type Casting

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

Array

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

Alur Kontrol

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";
};
```

Perulangan

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

Method

Definisi

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

Varargs & Overloading

```
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 Modifier

public	Dapat diakses dari mana saja
protected	Package yang sama + subclass
(default)	Package yang sama saja (tanpa keyword)
private	Class yang sama saja

Class & Objek

Definisi 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; }
}
```

Record (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	Milik class, bukan instance
final field	Tidak bisa diubah setelah inisialisasi
final method	Tidak bisa di-override
final class	Tidak bisa di-subclass

Pewarisan

Extends

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

Abstract Class

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

Konsep Utama

super	Panggil constructor atau method parent
@Override	Pemeriksaan override saat compile
instanceof	Pemeriksaan tipe saat runtime
sealed (17+)	Batasi class mana yang bisa extends

Interface

Definisi

```
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);
    }
}
```

Implementasi

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

Functional Interface

Runnable	() -> void
Supplier<T>	() -> T
Consumer<T>	T -> void
Function<T,R>	T -> R
Predicate<T>	T -> boolean
Comparator<T>	(T, T) -> int

Collection

List

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

Referensi Cepat Java

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"); // true
Set<String> immutable = Set.of("a", "b", "c");
```

Implementasi Umum

ArrayList	Array resizable, akses acak cepat
LinkedList	Doubly-linked, insert/remove cepat
HashMap	Hash table, O(1) get/put
TreeMap	Diurutkan berdasarkan key, O(log n)
HashSet	Elemen unik, O(1) lookup
LinkedHashMap	HashMap dengan urutan penyisipan

Penanganan 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
```

Hierarki Exception

Throwable	Root dari semua error dan exception
Error	Masalah serius (OutOfMemoryError)
Exception	Checked exception (harus ditangani)
RuntimeException	Unchecked (NullPointerException, IndexOutOfBounds)

Custom Exception

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

Stream & Lambda

Sintaks Lambda

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

Pipeline Stream

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

Operasi Stream Umum

.filter(pred)	Simpan elemen yang cocok dengan predicate
.map(func)	Transformasi setiap elemen
.flatMap(func)	Map dan ratakan stream bersarang
.sorted()	Urutkan (natural atau dengan Comparator)
.distinct()	Hapus duplikat
.limit(n)	Ambil n elemen pertama
.collect()	Terminal: kumpulkan ke dalam collection
.forEach()	Terminal: lakukan aksi pada setiap elemen
.reduce()	Terminal: gabungkan menjadi satu nilai
.count()	Terminal: hitung elemen

Generics

Class & Method Generic

```
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);
}
```

Tipe Terbatas & Wildcard

<T extends Number>	T harus Number atau subclass
<T extends A & B>	Batasan ganda (class + interface)
<?>	Tipe tidak diketahui (read-only)
<? extends T>	Upper bound wildcard (producer)
<? super T>	Lower bound wildcard (consumer)

Optional & Java Modern

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 Block (Java 15+)

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

Utilitas Berguna

var (10+)	Inferensi tipe variabel lokal
record (16+)	Class pembawa data immutable
sealed (17+)	Hierarki class terbatas
pattern matching (21+)	instanceof dengan auto-cast
virtual threads (21+)	Thread ringan via Thread.ofVirtual()