# JAVA MINDMAP

**🔹 1. Core Java**

* Data Types, Variables, Operators
* Control Structures: if, switch, for, while, do-while
* Arrays and String handling (StringBuilder, StringBuffer)
* OOP Pillars:
  + **Encapsulation** – private fields + public getters/setters
  + **Inheritance** – extends, super
  + **Polymorphism** – Method Overriding + Overloading
  + **Abstraction** – Abstract classes & Interfaces
* Constructors, static/instance blocks, this, super

**🔹 2. Exception Handling**

* try-catch-finally
* throw vs throws
* Checked vs Unchecked exceptions
* Custom exception classes
* Exception chaining and logging best practices

**🔹 3. Collections Framework**

* Core Interfaces: List, Set, Map, Queue
* Implementations:
  + ArrayList, LinkedList, HashSet, TreeSet
  + HashMap, TreeMap, LinkedHashMap
* Sorting: Collections.sort(), Comparator, Comparable
* Concurrent collections: ConcurrentHashMap, CopyOnWriteArrayList

**🔹 4. Java 8 Features (Deep)**

* **Lambda Expressions** – (a, b) -> a + b
* **Functional Interfaces** – Predicate, Function, Consumer, Supplier
* **Stream API** – filter(), map(), flatMap(), reduce(), collect()
* **Optional<T>** – Avoid null with orElse, ifPresent, map()
* **Method References** – Class::method
* **Date & Time API** – LocalDate, LocalTime, ZonedDateTime, Duration
* **Interface Enhancements** – default, static methods
* **CompletableFuture** – Async workflows (thenApply, thenCombine)

**🔹 5. Multithreading & Concurrency**

* Thread, Runnable, Callable, ExecutorService
* Thread lifecycle (start(), run(), sleep(), join())
* Synchronization, volatile, wait/notify
* Advanced tools: Semaphore, CountDownLatch, BlockingQueue, CyclicBarrier

**🔹 6. I/O & Serialization**

* File Handling: File, FileReader, BufferedReader, Scanner
* Streams: Byte Stream vs Character Stream
* Serialization: Serializable, transient, ObjectOutputStream
* try-with-resources for auto-closing

**🔹 7. Java Memory Management**

* Stack vs Heap memory
* Garbage Collection basics
* JVM options: -Xms, -Xmx, GC tuning flags
* finalize() (deprecated), System.gc(), OutOfMemoryError

**🔹 8. JDBC & Database Access**

* Steps: Load driver → Create connection → Execute statement → Handle result
* Interfaces: Connection, Statement, PreparedStatement, ResultSet
* Batch updates: addBatch(), executeBatch()
* Concept: Connection pooling (HikariCP, DBCP)

**🔹 9. Reflection & Annotations**

* Reflection API: Class.forName(), Method.invoke()
* Accessing fields/methods dynamically
* Built-in annotations: @Override, @SuppressWarnings, @Deprecated
* Custom annotations: @interface, @Target, @Retention

# JAVA 8 MIND MAP

**☀️ 1. Lambda Expressions**

Lambda Syntax

- (parameter) -> expression

- (a, b) -> a + b

- () -> System.out.println("Hi")

Use Cases

- Functional interfaces

- Event handling, sorting, filtering

Key Benefit: Write cleaner, inline function logic

**🔁 2. Functional Interfaces**

Functional Interface = Interface with only 1 abstract method

Built-in Interfaces

- Predicate<T> → boolean test(T)

- Function<T, R> → R apply(T)

- Consumer<T> → void accept(T)

- Supplier<T> → T get()

- UnaryOperator<T> → T apply(T)

- BinaryOperator<T> → T apply(T, T)

Annotation: @FunctionalInterface (optional but recommended)

**🧱 3. Streams API**

Stream = Sequence of elements supporting pipeline operations

Operations:

- filter(), map(), sorted(), distinct()

- collect(), reduce(), count(), forEach()

Stream Types:

- Stream<T>

- IntStream, LongStream, DoubleStream

Terminal vs Intermediate:

- Terminal: collect(), count(), reduce()

- Intermediate: filter(), map(), sorted()

Parallel Streams:

- stream().parallel() for multithreaded processing

**🧭 4. Method References**

arduino

CopyEdit

Syntax: ClassName::methodName or object::instanceMethod

Types:

- Static: Math::sqrt

- Instance: System.out::println

- Constructor: ArrayList::new

Shortcut for Lambda expressions

**📦 5. Optional<T>**

css

CopyEdit

Avoids NullPointerException

Common Methods:

- Optional.of(value)

- Optional.empty()

- Optional.ofNullable(value)

- isPresent(), get(), orElse(), orElseGet(), orElseThrow()

- ifPresent(consumer)

Chaining: optional.map(...).filter(...).orElse(...)

**📆 6. Date and Time API (java.time)**

sql

CopyEdit

New Classes:

- LocalDate → date only

- LocalTime → time only

- LocalDateTime → date + time

- ZonedDateTime → with timezone

- Period → difference in dates

- Duration → difference in time

Parsing & Formatting:

- DateTimeFormatter.ofPattern("dd-MM-yyyy")

**📚 7. Default & Static Methods in Interfaces**

csharp

CopyEdit

- Default methods → Provide method body in interface

default void log() { ... }

- Static methods → Called using InterfaceName.method()

static void help() { ... }

Reason: Interface evolution without breaking old code

**🪝 8. Collectors & Collect()**

css

CopyEdit

Used with Streams to convert into list/map/set

Collectors.toList(), toSet(), toMap(), joining(), groupingBy(), partitioningBy()

Example:

List<String> names = list.stream()

.filter(...)

.collect(Collectors.toList());

**🧪 9. Stream Best Practices**

diff

CopyEdit

- Avoid modifying data inside stream

- Prefer method references when possible

- Don’t use parallel streams for small collections

- Don’t mix terminal operations

- Use flatMap() for nested collections

**🔁 10. forEach vs forEachOrdered**

scss

CopyEdit

- forEach → Order not guaranteed (esp. in parallel streams)

- forEachOrdered → Maintains order (even in parallel streams)

**🧵 7. Lambda in Multi-threading**

diff

CopyEdit

Runnable r = () -> { logic }

Executors.newSingleThreadExecutor().submit(() -> compute());

Use in:

- Stream.parallel()

- CompletableFuture

- Callback design

Avoid:

- Holding locks inside lambdas

- Shared variables without synchronization