MODERNE IDE DEBUG, NAVIGATION & KÜRZEL-SPICKZETTEL

Architektur-Referenz für IntelliJ, VS Code/VSCodium, Eclipse & Terminal (2026)

1. Universelle IDE-Tastenkürzel-Matrix

Tastenbelegung

Aktion	IntelliJ IDEA	VS Code / VSCodium	Eclipse
Befehlspalette	`Ctrl + Shift + A`	`Ctrl + Shift + P`	`Ctrl + 3`
Datei öffnen	`Ctrl + Shift + N`	`Ctrl + P`	`Ctrl + Shift + R`
Symbol / Klasse	`Ctrl + N / Alt+Home`	`Ctrl + T`	`Ctrl + Shift + T`
Zeile duplizieren / löschen	`Ctrl + D / Ctrl + Y`	`Shift+Alt+Down / Ctrl+Shift+K`	`Ctrl + Alt + Down / Ctrl + D`
Mehrfach-Cursor	`Alt + Shift + Click`	`Alt + Click`	`Alt + Shift + A`
Zuletzt geöffnete Dateien	`Ctrl + E`	`Ctrl + Tab`	`Ctrl + E`

2. Professioneller Debugging-Workflow

Debug-Engine

Haltepunkte (Breakpoints): Klick auf den Zeilenrand oder `Ctrl + F8` / `F9`.

Bedingte Haltepunkte: Rechtsklick auf Haltepunkt: Ausdruck definieren wie `user.getId() == 42`.

Step Over (Überspringen): `F8` (IntelliJ) / `F10` (VS Code) — Nächste Zeile ausführen.

Step Into (Hineinspringen): `F7` (IntelliJ) / `F11` (VS Code) — In Funktionsaufruf wechseln.

Step Out (Herausspringen): `Shift + F8` / `Shift + F11` — Funktion beenden und zurückspringen.

Ausdruck auswerten (Evaluate): `Alt + F8` — Zur Laufzeit dynamischen Code ausführen.

Variablen überwachen (Watch): Rechtsklick • "Add to Watches" zur dauerhaften Überwachung.

3. Automatisierte Refactoring-Muster

Sauberer Code

Operation	Tastenkürzel (IntelliJ / VSC)	Architektonischer Nutzen
Umbenennen	`Shift + F6` / `F2`	Aktualisiert alle Referenzen im gesamten Projekt sicher.
Methode extrahieren	`Ctrl + Alt + M`	Überführt Codeblock in eigenständige Funktion.
Variable extrahieren	`Ctrl + Alt + V`	Weist Ausdruck einer lokalen Variablen zu.
Signatur ändern	`Ctrl + F6`	Passt Parameterreihenfolge an allen Aufrufstellen an.
Inlining (Einbetten)	`Ctrl + Alt + N`	Ersetzt temporäre Variable direkt durch Ausdruck.

4. Versionskontrolle in der IDE

VCS-Tools

Git Blame / Annotate: Rechtsklick auf Rand • "Annotate with Git Blame" zur Einsicht der Autoren.

Visuelles 3-Wege-Merge: Konflikte visuell zwischen Lokal, Basis und Eingehend lösen.

Mit Zweig vergleichen: Side-by-Side Diff gegen Remote-Zweige durchführen.

Shelve / Stash: Unfertige Änderungen sicher parken, ohne Commits zu verunreinigen.

5. Language Server Protocol (LSP) Architektur

Sprachanalyse

```
IDE (LSP Client) <-- JSON-RPC --> Language Server
Rust: rust-analyzer | Python: pyright / basedpyright | C++: clangd
Go: gopls | TypeScript: vtsls / tsserver | Java: eclipse.jdt.ls
```

Navigationsziel	IntelliJ IDEA	VS Code / Zed
Zur Definition	`Ctrl + B` / `Ctrl + Click`	`F12` / `Ctrl + Click`
Alle Verwendungen	`Alt + F7`	`Shift + F12`
Zur Typdefinition	`Ctrl + Shift + B`	`Ctrl + F12`
Zur Implementierung	`Ctrl + Alt + B`	`Ctrl + F12`

6. Speicher- & Leistungsoptimierung

Performance

JVM Heap konfigurieren: In der `vmoptions`-Datei anpassen:
`-Xms2048m -Xmx4096m -XX:+UseG1GC -XX:SoftRefLRUPolicyMSPerMB=50`

Verzeichnisse ausschließen: `node_modules`, `.venv`, `target` als Excluded markieren, um Hintergrundindizierung um 70% zu senken.

Language Server neustarten: Befehlspalette • `Restart Language Server` bei Speicherlecks.

Tab-Begrenzung: Geöffnete Tabs auf 10 begrenzen; Navigation primär über `Ctrl + E` abwickeln.

7. Erweiterte Regex-Muster in der IDE

Suchen & Ersetzen

Projektweites Suchen / Ersetzen: `Ctrl + Shift + F` / `Ctrl + Shift + R`

Erfassungsgruppen nutzen:
Suche: `public void test(\w+)\(\)` → Ersetzen: `void should$1()` (Bezug auf $1)

Zeilenenden matchen: `;\s*$` — Semikolons mit überflüssigen Leerzeichen aufspüren.

Dateimaskenfilter: `*.ts, !*.spec.ts` — Testdateien ausschließen.

8. KI-Sicherheit & Erweiterungshygiene

Best Practices

Bereich	Empfohlene Praxis	Sicherheits- & Geschwindigkeitsgewinn
KI-Assistenten	Lokale Modelle (Ollama / Continue.dev)	Kein Quellcode verlässt das Unternehmen.
Plugins	Unbenutzte Erweiterungen deaktivieren	Verkürzt den IDE-Kaltstart um bis zu 60%.
Formatierung	Format on Save / Git Pre-Commit	Verhindert Stil-Diskussionen im Code Review.
Terminal	Integriertes Terminal (Alt+F12 / Ctrl+`)	Verhindert Ablenkung durch Fensterwechsel.