Complete IdeaVim + Vim Key Mapping and Behavior Reference

English Edition — Comprehensive explanations, examples, and cheat sheet

Part A — IdeaVim (Leader: ,)

Every mapping and behavior from your JetBrains IdeaVim configuration, explained clearly.

Core Options & Enhancements

Setting	Explanation
clipboard+=unnamedplus	Use the system clipboard for all yanks/deletes/pastes.
ideajoin	Improved `J` join that respects the IDE formatter.
highlightedyank	Temporarily highlights yanked text for feedback.
surround	Add/change/delete surrounding characters quickly.
easymotion	Two■key on■screen jump navigation (`s`).
matchit	Smarter `%` to jump matching brackets/tags.
notimeout	No timeout while typing multi■key chords (leader combos).

Better Behaviors

Mapping	Purpose
<C-o>	Back in IDE navigation history.
<C-i>	Forward in IDE navigation history.
K	Show hover information / quick documentation.

Disabled Shortcuts (prevents accidents)

Mapping	Why
<A-S-f>, <C-S-f>	Avoid accidentally formatting or global find operations.
<C-n>, <C-p> (Normal/Insert)	Avoid conflicts with completion and keep behavior consistent.

Plugin Shortcuts

Mapping	Action
s (Normal/Visual)	EasyMotion jump.
gm (Normal/Visual)	Matchit: jump to matching bracket/tag.

CamelCase Text Objects

Mapping	Effect
vic / cic / dic / yic	Operate on a camelCase component: select / change / delete / yank.
ci → "_ci	Force change into black■hole register to avoid clobbering yanks.

Split Helpers — Function Args (,sfN)

Key	Description + Example

,sf2	Split the first 2 arguments onto new lines; keep the rest inline.
	`Before: foo(a, b, c, d, e, f, g)`
	`Press:  ,sf2`
	`After:`
	`foo(`
	`a,`
	`b,`
	`c, d, e, f, g`
	`)`

,sf3	Split the first 3 arguments onto new lines; keep the rest inline.
	`Before: foo(a, b, c, d, e, f, g)`
	`Press:  ,sf3`
	`After:`
	`foo(`
	`a,`
	`b,`
	`c,`
	`d, e, f, g`
	`)`

,sf4	Split the first 4 arguments onto new lines; keep the rest inline.
	`Before: foo(a, b, c, d, e, f, g)`
	`Press:  ,sf4`
	`After:`
	`foo(`
	`a,`
	`b,`
	`c,`
	`d,`
	`e, f, g`
	`)`

,sf5	Split the first 5 arguments onto new lines; keep the rest inline.
	`Before: foo(a, b, c, d, e, f, g)`
	`Press:  ,sf5`
	`After:`
	`foo(`
	`a,`
	`b,`
	`c,`
	`d,`
	`e,`
	`f, g`
	`)`

,sf6	Split the first 6 arguments onto new lines; keep the rest inline.
	`Before: foo(a, b, c, d, e, f, g)`
	`Press:  ,sf6`
	`After:`
	`foo(`
	`a,`
	`b,`
	`c,`
	`d,`
	`e,`
	`f,`
	`g`
	`)`

,sf7	Split the first 7 arguments onto new lines; keep the rest inline.
	`Before: foo(a, b, c, d, e, f, g)`
	`Press:  ,sf7`
	`After:`
	`foo(`
	`a,`
	`b,`
	`c,`
	`d,`
	`e,`
	`f,`
	`g,`
	`)`

Split Helpers — Chained Calls (,scN)

Key	Description + Example

,sc2	Stack the first 2 calls of a dot■chain; keep the rest on the last line.
	`Before: obj.first().second().third().fourth().fifth().sixth()`
	`Press:  ,sc2`
	`After:`
	`obj`
	`.first()`
	`.second()`
	`.third().fourth().fifth().sixth()`

,sc3	Stack the first 3 calls of a dot■chain; keep the rest on the last line.
	`Before: obj.first().second().third().fourth().fifth().sixth()`
	`Press:  ,sc3`
	`After:`
	`obj`
	`.first()`
	`.second()`
	`.third()`
	`.fourth().fifth().sixth()`

,sc4	Stack the first 4 calls of a dot■chain; keep the rest on the last line.
	`Before: obj.first().second().third().fourth().fifth().sixth()`
	`Press: ,sc4`
	`After:`
	`obj`
	`.first()`
	`.second()`
	`.third()`
	`.fourth()`
	`.fifth().sixth()`

,sc5	Stack the first 5 calls of a dot■chain; keep the rest on the last line.
	`Before: obj.first().second().third().fourth().fifth().sixth()`
	`Press: ,sc5`
	`After:`
	`obj`
	`.first()`
	`.second()`
	`.third()`
	`.fourth()`
	`.fifth()`
	`.sixth()`

,sc6	Stack the first 6 calls of a dot■chain; keep the rest on the last line.
	`Before: obj.first().second().third().fourth().fifth().sixth()`
	`Press: ,sc6`
	`After:`
	`obj`
	`.first()`
	`.second()`
	`.third()`
	`.fourth()`
	`.fifth()`
	`.sixth()`

,sc7	Stack the first 7 calls of a dot■chain; keep the rest on the last line.
	`Before: obj.first().second().third().fourth().fifth().sixth()`
	`Press: ,sc7`
	`After:`
	`obj`
	`.first()`
	`.second()`
	`.third()`
	`.fourth()`
	`.fifth()`
	`.sixth()`

Split Helpers — Ternary (,st)

Key	Description + Example

,st	Split a `? :` ternary across lines to improve readability and future diffs.
	`Before: result = condition ? valueA : valueB`
	`Press: ,st`
	`After:`
	`result = condition`
	`? valueA`
	`: valueB`

Config Maintenance

Key	Action
,ei	Open ~/.ideavimrc in the editor.
,si	Reload (source) ~/.ideavimrc.

General IDE Shortcuts

Key	Action
gs / gc / gn	Goto Symbol / Class / Navigate (ReSharper)
<C-p> / <C-e>	Goto File / Recent Files
,fu / ,ff	Find Usages / Find in Path
,re / ,ao / ,rp / ,iv / ,il	Rename / Select all occurrences / Replace / Introduce variable / Inline
<S-Space>	Call inline completion action
,oe / ,os / ,cs / ,ta	Reveal in / Recent projects / Close project / Find action by ID
,og	Open terminal (`wt lg`)
,ie / ,oa	Sweep AI prompt bar / new chat

Save / Cleanup / Imports

Key	Action
<C-s>	Save all (works in Insert mode too)
,fd / ,cc / ,oi	Reformat / Silent code cleanup / Optimize imports

Tool Windows & Popups

Key	Action
,sp / ,op / ,of / ,ot / ,od / ,oc	Select in Project / Project / Find / Terminal / Debug / Commit tool windows
,hw	Hide all windows
<C-m>	Show popup menu
,pi / ,fs / ,rf	Parameter Info / File Structure / Refactorings quick list

Editor Navigation & Tabs

Key	Action
,ne / ,pe	Next / previous error
[d /]d	Prev / next error in solution (ReSharper)
[f /]f	Method up / down
<C-h> / <C-l>	Previous / next tab (also in Insert mode)
,tc / ,to / ,tp / ,tm	Close / Close others / Pin / Close unmodified tabs

Window (Split) Navigation

Key	Action
,wc / ,wo	Close current split / keep only current split
,wj / ,wk / ,wh / ,wl	Move focus (down/up/left/right)
,ws / ,wv	Horizontal / Vertical split

Build / Run / Debug

Key	Action
,ba / ,ra / ,da / ,sa	Build solution / Run / Debug / Stop
,rt / ,dt	Run / Debug tests in context
,tb	Toggle breakpoint
<C-S-A-j/k/h/l>	Step over / Resume / Step out / Step into
,ee	Quick evaluate expression

Code Folding & Bracket Collapsing

Environment	Collapse	Expand	Collapse All	Expand All	Description
IdeaVim / JetBrains	,zc	,zo	,zc	,zo	Uses IDE folding regions (functio
Vim / Neovim	zc	zo	zM	zR	Fold by indent/syntax/markers de

Part B — Vim / Neovim (Leader:)

Native Vim mappings and habits for efficient editing.

Leader & Core Options

Mapping/Option	Description
Leader=<Space>	Disable bare <Space> (noop), then repurpose as leader prefix.
clipboard+=unnamed	Sync unnamed register with OS clipboard where available.
scrolloff=10	Keep 10 lines of context while moving.
incsearch + hlsearch	Live search preview; highlight all matches.
number + relativenumber	Absolute current line, relative others for motion counts.
ignorecase + smartcase	Case■insensitive unless pattern has uppercase.

Better Behaviors

Mapping	Purpose
<C-d>zz / <C-u>zz	Half■page scroll then center cursor.
nzz / Nzz	Center search hits.
Y → y$	Yank to end of line (more intuitive than `yy`).
x / s / c / C → "_	Deletes/changes don’t clobber your last yank.
Visual p → "_dP	Paste over selection without losing previous yank.
Visual </> keep selection	Use `<gv` / `>gv` to reselect after indent.

Productivity Shortcuts

Mapping	Action
<C-j>/<C-k> (Normal)	Smooth scroll (`<C-e>`/`<C-y>`) without moving cursor.
<C-j>/<C-k> (Insert)	Completion next/prev (`<C-n>`/`<C-p>`).
gh / gl	Jump to first non■blank (`^`) / end (`$`) of line across modes.
+ / -	Increment / decrement numbers under cursor.
U	Redo (`<C-r>`).
Q	Replay macro in register q (`@q`).
[[/]]	Jump to previous/next section (e.g., at `{` / `}`).

Leader Utilities & Config

<Space>ns	Clear search highlights (:nohlsearch).

<Space>ev / <Space>sv	Open / reload ~/.vimrc.

Prevent Bad Habits (Disabled keys)

Keys	Reason
<Up>/<Down>/<Left>/<Right>	Encourages hjkl movement and keyboard■centric navigation.
<C-n>/<C-p> in Insert & <C-n> in Normal	Prevents accidental completion/register side■effects.

Folding in Vim

Command	Effect
zc / zo	Close / open current fold under the cursor.
zM / zR	Close / open all folds in the buffer.
za	Toggle current fold.
zd / zE	Delete one / all folds you created manually.
set foldmethod=indent|syntax|marker	Choose how folds are detected/created.

Appendix — One■Page Cheat Sheet

Quick lookup for the most useful mappings (IdeaVim and Vim).

Category	IdeaVim (,)	Vim (<Space>)
Navigate	<C-o>/<C-i>, K; ,op (Project), ,of (Find)	gh/gl (^/$), n/N zz
Split Helpers	,sfN / ,scN / ,st	—
Folding	,zc / ,zo	zc / zo / zM / zR
Run/Debug	,ra / ,da / ,sa / ,tb	—
Refactor	,re / ,iv / ,il / ,ao	—
Save/Cleanup	<C-s>, ,fd, ,cc, ,oi	—
Tabs/Windows	<C-h>/<C-l>, ,tc/ ,to / ,tp / ,tm; ,ws/ ,wv	—
Config	,ei / ,si	<Space>ev / <Space>sv
Search	,fu / ,ff	<Space>ns
AI	<Tab> accept Sweep AI	—