JFOKUS 2014

Effective IDE Usage

Rabea Gransberger

Dipl.-Inf.

@rgransberger

Rabea Gransberger

Speaker Bio

- Diploma in Computer Science 2008 (University Bremen, Germany)
- Developer, Project and Department Lead at MEKO-S GmbH, Bremen, Germany

- Mainly working with **eclipse** Technology (RCP, RAP, Tabris)
  - Developing OTIS (Oil Trading and Information System)

- Passionate Java developer with focus on clean code

Motivation: Effective IDE Usage

Explore the IDE and be more productive!

- Developers type too much (and Copy & Paste)
- Features of the IDE are unknown only used as simple text editor
- Settings are not made according to programming style used

Disclaimer

- No introduction to IDEs in general
- No comparison of IDEs
- Only Java SE development Tools (`*.java` no `*.jsp` etc.)
- Not all features covered, look at the slides after the talk to find more
- Not a: “Do you know this shortcut”-talk!

Agenda

- General IDE Setup
- Settings & Menus
- Editor Basics & Features
- Navigation Basics & Important Shortcuts
- Syntax Coloring, Templates & Formatting
- Content Assist & Refactoring
- Code Analysis, Console & External Tools
- Debugger
- Q&A

(E) = Eclipse 4.3.1 (Kepler SR1)*
(JI) = IntelliJ Idea Community 13.0.2
(N) = NetBeans 8.0 beta
Demos in Eclipse

* With Code Recommenders 2.0 and JDT 8 Beta

GENERAL IDE SETUP

Sharing Settings

Share settings for a team or whole company

- Option 1: Export/Import (E)(JI)(N)
  - Export settings
  - Send E-Mail with exported file
  - Let members (re-)import them
- Option 2: Auto Sharing (E)(JI)
  - (E) eclipse.ini: `-pluginCustomization customization.ini`
  - (JI) Config Server or Version Control

Sharing Settings: Eclipse

- File → Export: General → Preferences and choose desired options
- Save to a file, e.g. `test1.epf`
- Change the option you are looking for, export to `test2.epf`
- Do a diff between `test1.epf` and `test2.epf`
  Copy change to new file, remove the `\instance\` qualifier in front
- Save file as `plugin_customization.ini` on network drive
- Add lines to `<installdir>\eclipse.ini`:

```
-pluginCustomization
<pathTo…>plugin_customization.ini
```

Sharing Settings: IntelliJ Idea

- Team / Company
  - IntelliJ Configuration Server Plugin
- Team
  - Check-In `.idea` directory of project to version control

    Exclude `workspace.xml` from version control
- For yourself
  - Template Project Settings (apply for new projects)
  - File →Other Settings →Default Settings

Working with Project Sets

- Working for different customers or in different teams
- Open multiple IDE instances with different project sets
- Create links on the desktop to start the IDE with a project set
- (E) Workspace

  `-data D:\workspaces\JFokus` (General → Workspace → Workspace Name)
- (JI) Project

  `"D:\IdeaProjects\JFokus“`
- (N) User Dir

  `--userdir D:\netbeans\Jfokus`

Appearance

- Change the theme/Look & Feel of your IDE
- (**E**) Can be set per workspace to distinguish them easily
  - General → Appearance: Theme
  - CSS based, can be widely customized
- (**JI**) General setting
  - Appearance → UI options: Theme
- (**N**) User Dir setting
  - Appearance → Look and Feel
- Customizing of editor/console colors (e.g. dark theme) also possible

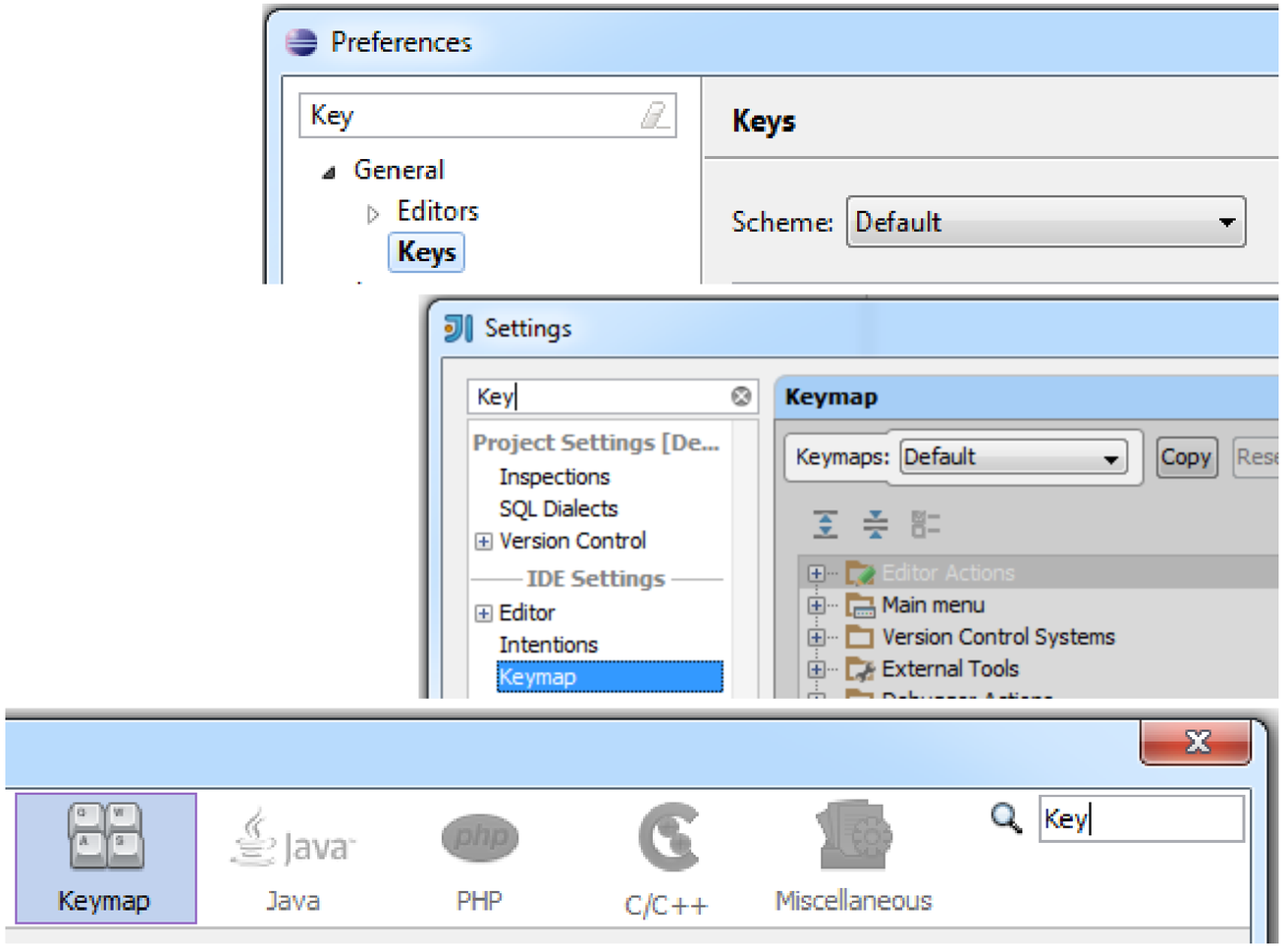
Updates

- Auto-Update for plugins and minor versions available (E)(JI)(N)
- Goal: Updating IDE to new version while preserving installed plugins
- (E) Only update to minor version possible, for major
  - Install Software Items from File: File → Export/Import → Install
  - Download + re-install + zip for team members
  - Install from command line or Yoxos Profile
- (JI) Direct updates for program and plugins if possible
  - Possible to choose channel (major/minor/updates only) for updates in settings
- (N) Download new installer: picks up and checks old plugins

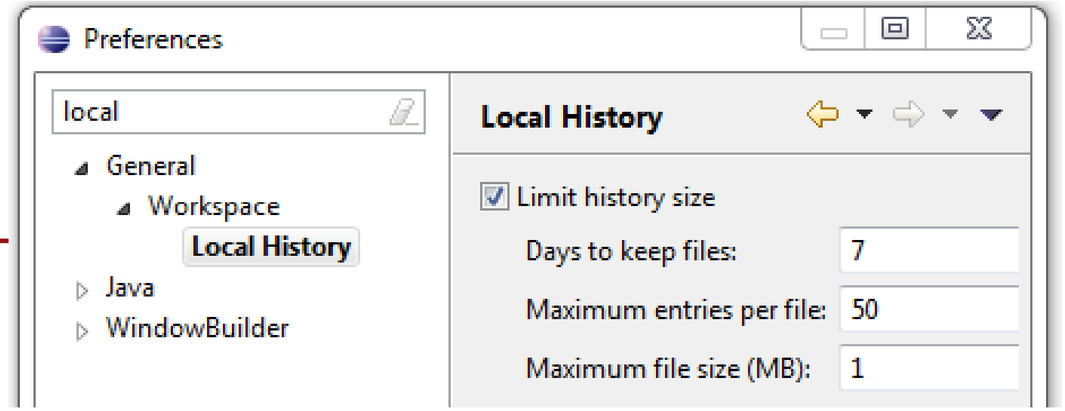
Settings & Menus

Settings

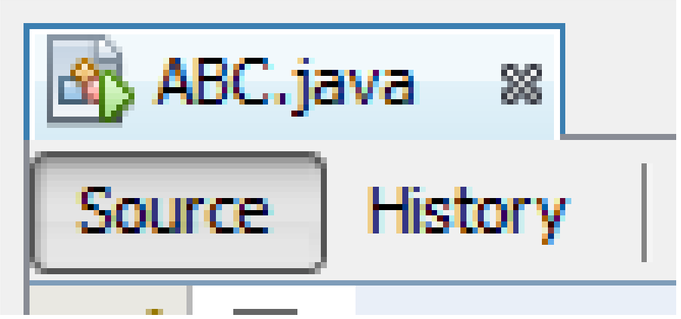
- Search by keyword
  - (E) Menu: Window → Preferences
  - (JI) Menu: File → Settings
  - (N) Menu: Tools → Options
- Shortcuts options
  - (E) General → Keys
  - (JI) Keymap
  - (N) Keymap



Local History

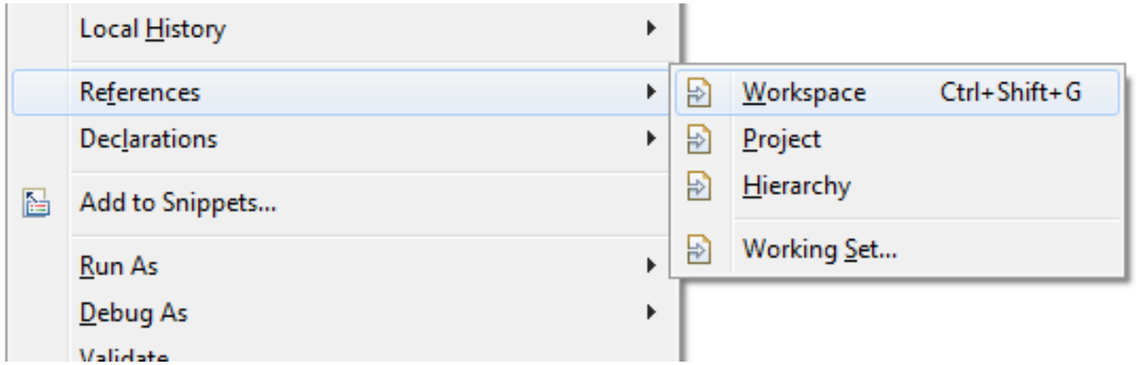


- IDE keeps local history of files
- Get back a revision not in version control
- Restore when deleted (context menu, parent)
- Check settings of version history size
  - (**E**) General → Workspace → Local History
  - (**JI**) Version Control → Limit history
  - (**N**) Misc → Versioning → Local History



Explore the Menus

- Some commands have no (default) shortcut but are in the menu



Examples:

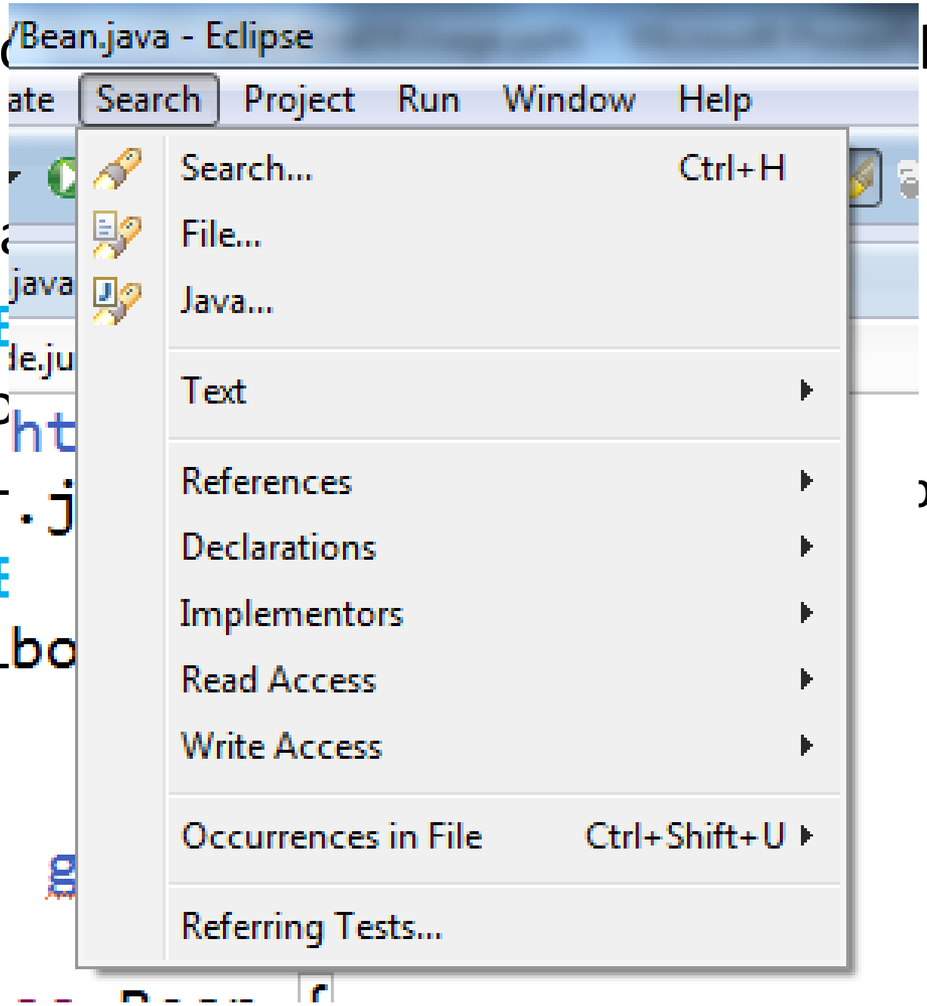
- (E) Search menu has more options then context menu:
  - Referring tests, read / write access, implementors
- (E) Navigate:
  - Open (super) implementation

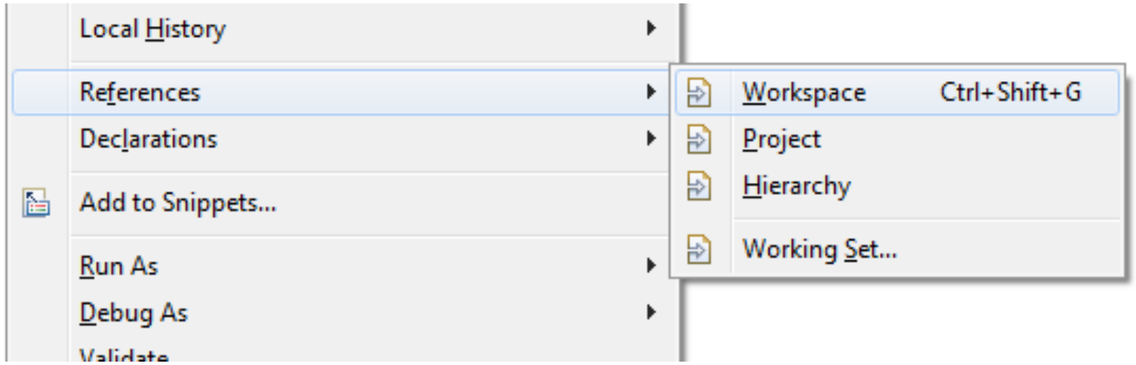
Explore the Menus

- S hortcut but are in the menu

Ex

- (E
  c
  ht
  _.j
- (E
  _bo





plementors

Editor Basics & Features

Editor Basics

- Double click tab to maximize or use shortcuts
- Show line numbers (**E**) General → Editors → Text Editors (**JI**)(**N**) on
- Compare two files (**E**)(**JI**) Context Menu: Compare (**N**) Tools → Diff
- Move to new window, drag & drop, align multiple editors alongside
- Paste text on package creates Java file automatically (**E**)(**JI**)
- Column/Block selection mode, e.g. for creating maps (**E**)(**JI**)(**N**)
- Auto close editors tabs (**E**)(**JI**)
- Task Tags like `TODO` and `FIXME` can be used
- Go to line (**E**) `Ctrl+L` (**JI**) `Ctrl+G` (**N**) `Ctrl+G`

Editor Features

- Escape on paste into String (E) Java →Editor →Typing (JI)(N) auto on
- Mark Occurrences: e.g. highlights selected variable in file
- Show matching brackets / Highlight current scope
  - (E) Java →Editor →Bracket-Highlighting: Enclosing
  - (JI) Editor →Highlight current: Scope
- Folding of certain areas in editor, e.g. hide comments
  - (E) Java →Editor →Folding
  - (JI) Editor →Code Folding
  - (N) Editor →General →Code Folding
- Copy qualified name (E)(JI) Context Menu (N) via Plugin

Navigation Basics
& Important Shortcuts

Navigation Basics

- Open Type/Resource: With CamelCase search
  (E) `Ctrl+Shift+T/Ctrl+Shift+R` (JI) `Ctrl+N/Ctrl+Shift+N`
  (N) `Ctrl+O/Alt+Shift+O`
- Quick Open Type/Hierarchy
  (E) `F3/F4` (JI) `Ctrl+B/Ctrl+H`(N) `Ctrl+B/Alt+Shift+F12`
- Next/Prev match
  (E) `Ctrl+./Ctrl+,` (JI) `Ctrl+F3/Shift+F3` (N)`Alt+Down/Alt+Up`
- Last edit location
  (E) `Ctrl+Q` (JI) `Ctrl+Shift+Back` (N) `Ctrl+Q`
- Breadcrumbs ((E)(JI)with context menu for actions like new class on package)
  (E) `Alt+Shift+B` (JI) `Alt+Shift+B` (N) Menu: Views

Important Shortcuts

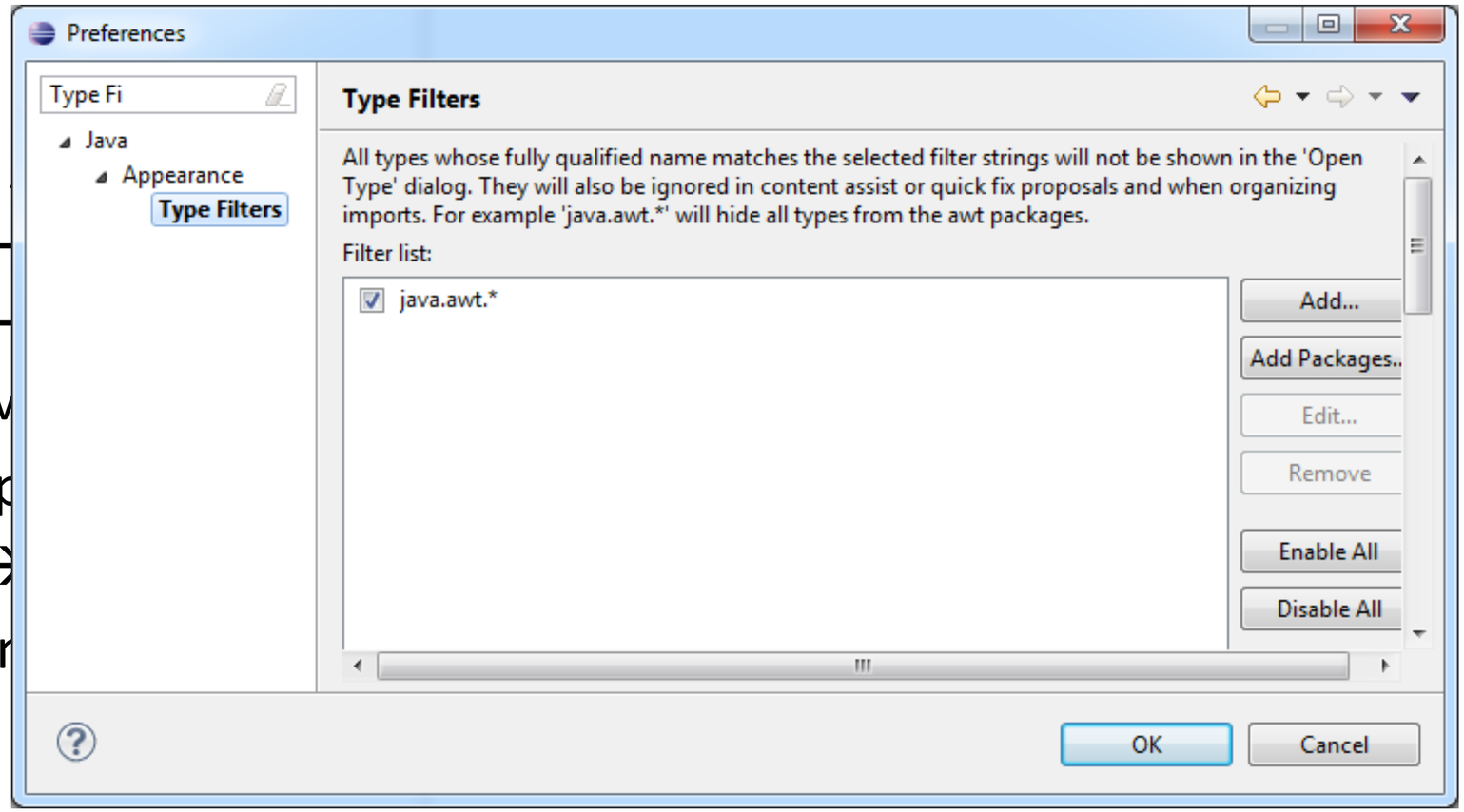
• Content Assist:	(E)Ctrl+Space	(JI)Ctrl+(Shift)+Space	(N)Ctrl+Space
• Quick fix:	(E) Ctrl+1	(JI) Alt+Enter	(N) Alt+Enter
• Quick Access:	(E) Ctrl+3	(JI) Ctrl+Shift+A	(N) Ctrl+I
• Find editor:	(E) Ctrl+E	(JI) Ctrl+E	(N)Ctrl+Shift+T
• Incr. Search:	(E) Ctrl+J / K	(JI) Ctrl+F	(N) Ctrl+F
• (Re-)Run:	(E) Ctrl+F11	(JI) Shift+F10	(N) F6
• Format:	(E) Ctrl+Shift+F	(JI) Ctrl+Alt+L	(N) Alt+Shift+F
• Delete Line:	(E) Ctrl+D	(JI) Ctrl+Y	(N) Ctrl+E
• Comment:	(E) Ctrl+Shift+C	(JI) Ctrl+/	(N) Ctrl+/

Imports

- Exclude e.g. `java.awt.*` from import proposals
  - (**E**) Java → Appearance → Type Filters
  - (**JI**) Editor → Auto Import → Java: Exclude from Import and Completion
  - (**N**) Editor → Code Completion → Java: Package/Classes Exclude
  - Or use a Java 8 compact profile ☺
- Organize imports on paste (**E**)(**N**) auto on
  - (**JI**) Editor → Auto Import
- Number of imports for `*` for normal and static imports (**E**)(**JI**)(**N**)

Imports

- Exclude e.g.
  - (E) Java →
  - (JI) Editor
  - (N) Editor
  - Or use a Jav
- Organize imp
  - (JI) Editor →
- Number of ir



Syntax Coloring, Templates & Formatting

Syntax Coloring

See possible issues immediately by coloring or font settings

Examples:

- (E) Java → Editor → Syntax Coloring
  - Mark Boxing/Unboxing expressions in red
- (JI) Editor → Colors & Fonts → Java
  - Reassigned parameter
- (N) Fonts & Color → Syntax
  - Use different color for parameter

Syntax Coloring

See possibl

Examples:

- (E) Java →
  - Mark Bo
- (JI) Editor
  - Reassign
- (N) Fonts &
  - Use diffe

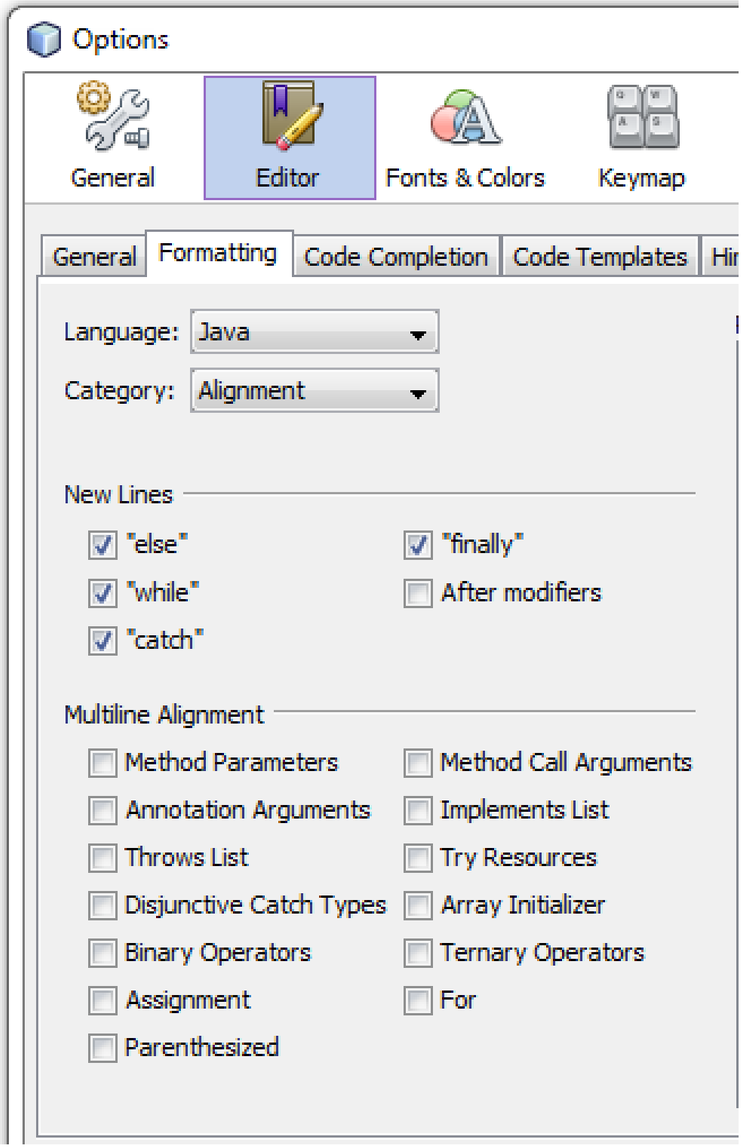
```java
/**
 * This is about <code>ClassName</code>. {@link com.yourCompany.aPackage.Interface}
 * @author author
 * @deprecated use <code>OtherClass</code>
 */
public class ClassName<E> extends AnyClass implements InterfaceName<String> {
  enum Color {
    RED, GREEN, BLUE
  };

  /* This comment may span multiple lines. */
  static Object staticField;

  // This comment may span only this line
  private E field;

  private AbstractClassName field2;

  // TASK: refactor
  @SuppressWarnings(value = "all")
  public int foo(Integer parameter) {
    abstractMethod(inheritedField);
    int local = 42 * hashCode();
    staticMethod();
    return bar(local) + parameter + parameter.intValue();
  }


```

Formatting

- Team should work with same settings for code style
- May set pre-/suffix for variables, e.g. `m_field`
- Detailed formatting options for code: braces, line wraps, spaces, etc.
  - (**E**) Java → Code Style → Formatter
  - (**JI**) Code Style → Java
  - (**N**) Editor → Formatting
- Format on save for edited lines only
  - (**E**) Java → Editor → Save Actions: Format edited lines
  - (**JI**) Editor → Smart keys → Reformat on Paste: Format Block
  - (**N**) Editor → On Save: Reformat

Formatting

- Team should work with same settings for code style
- May set pre-/suffix for variables, e.g. `m_field`
- Detailed formatting options for code: braces, line wraps, sp
  - (**E**) Java → Code Style → Formatter
  - (**JI**) Code Style → Java
  - (**N**) Editor → Formatting
- Format on save for edited lines only
  - (**E**) Java → Editor → Save Actions: Format edited lines
  - (**JI**) Editor → Smart keys → Reformat on Paste: Format Block
  - (**N**) Editor → On Save: Reformat

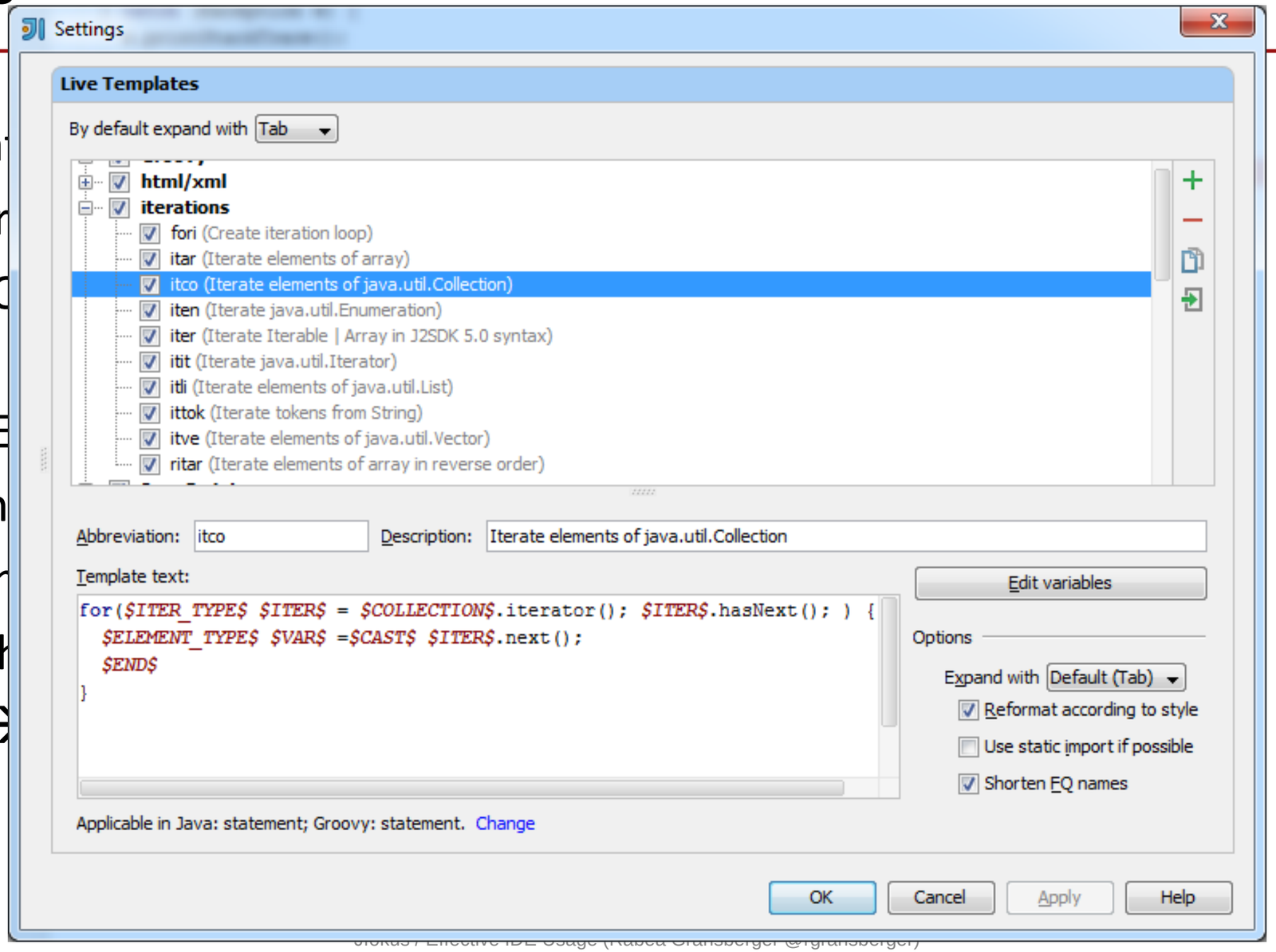


Templates

- Use templates to generate expressions and blocks
- Create your own: e.g. Exceptions to slf4j instead of console
- (E) Java → Code Style → Code Templates
  - Structure, comments, copyright
- (E) Java → Editor → Templates: smart code snippets
- (JI) File Templates: File structure
- (JI) Live Templates: smart code snippets
- (JI) Copyright → Copyright Profiles: Copyright
- (N) Editor → Code Templates → Java: smart code snippets

Templates

- Use templa
- Create your
- (E) Java → C
  - Structure,
- (E) Java → E
- (JI) File Tem
- (JI) Live Tem
- (JI) Copyrigh
- (N) Editor →



Save Actions

- Actions which can be applied on save (**E**)(**JI**)(**N**)
- In **E**clipse, e.g.:
  - Java → Editor → Save Actions
  - Format + Organize imports
  - Use parentheses in statements
  - Use qualified access with `this`
  - Add `@Override` and `@Deprecated`
- (**E**) Additional Clean-Up profile which can be triggered manually
  - Menu: Source → Clean Up

Content Assist
& Refactoring

Content Assist

Let the IDE generate code, don‘t type yourself

- Constructors/Fields/Getters/Setters/toString/equals/hashCode
- Override/Implement methods
- Call API which does not exist
- Call method and generate assignment
- Create fields from parameters
- Use quick fixes / intentions when shown by IDE
- Can use abbreviations of type name for completion
- (E) Completion overwrites, best guessed arguments, trigger = 50ms

Refactoring

- Refactoring is a safe operation to restructure the code
- Common: Rename, Move, change signature
  - Extract variable, method, constant
  - Create superclass, interface, pull up, push down
- Safe delete is not present in (**E**) but can be simulated:
  - Empty the body of the method to delete
  - Inline the method, which results in replacing it with nothing
- NetBeans provides additional transformation to Java 7 and Java 8
  - Refactor → Inspect and Transform

Refactorings Overview

- Change Class Signature
- Change Signature
- Convert Anonymous to Inner
- Convert to Instance Method
- Copy
- Encapsulate Fields
- Extract Delegate
- Extract Include File
- Extract Interface
- Extract Method
- Extract Method Object
- Extract Parameter Object
- Extract Superclass
- Generify Refactoring
- Inline
- Extract Constant
- Extract Field
- Extract Parameter
- Extract Property
- Extract Variable
- Invert Boolean
- Make Class Static
- Make Method Static
- Migrate
- Move Refactorings
- Pull Members up
- Push Members down
- Remove Middleman
- Rename Refactorings
- Replace Constructor with Builder
- Replace Constructor with Factory Method
- Replace Inheritance with Delegation
- Replace Method Code Duplicates
- Replace Temp with Query
- Safe Delete
- Type Migration
- Use Interface Where Possible
- Wrap Return Value

Code Analysis, Console & External Tools

Static code analysis

Find potential errors before running

- Plugins for PMD/FindBugs (E)(JI)(N)

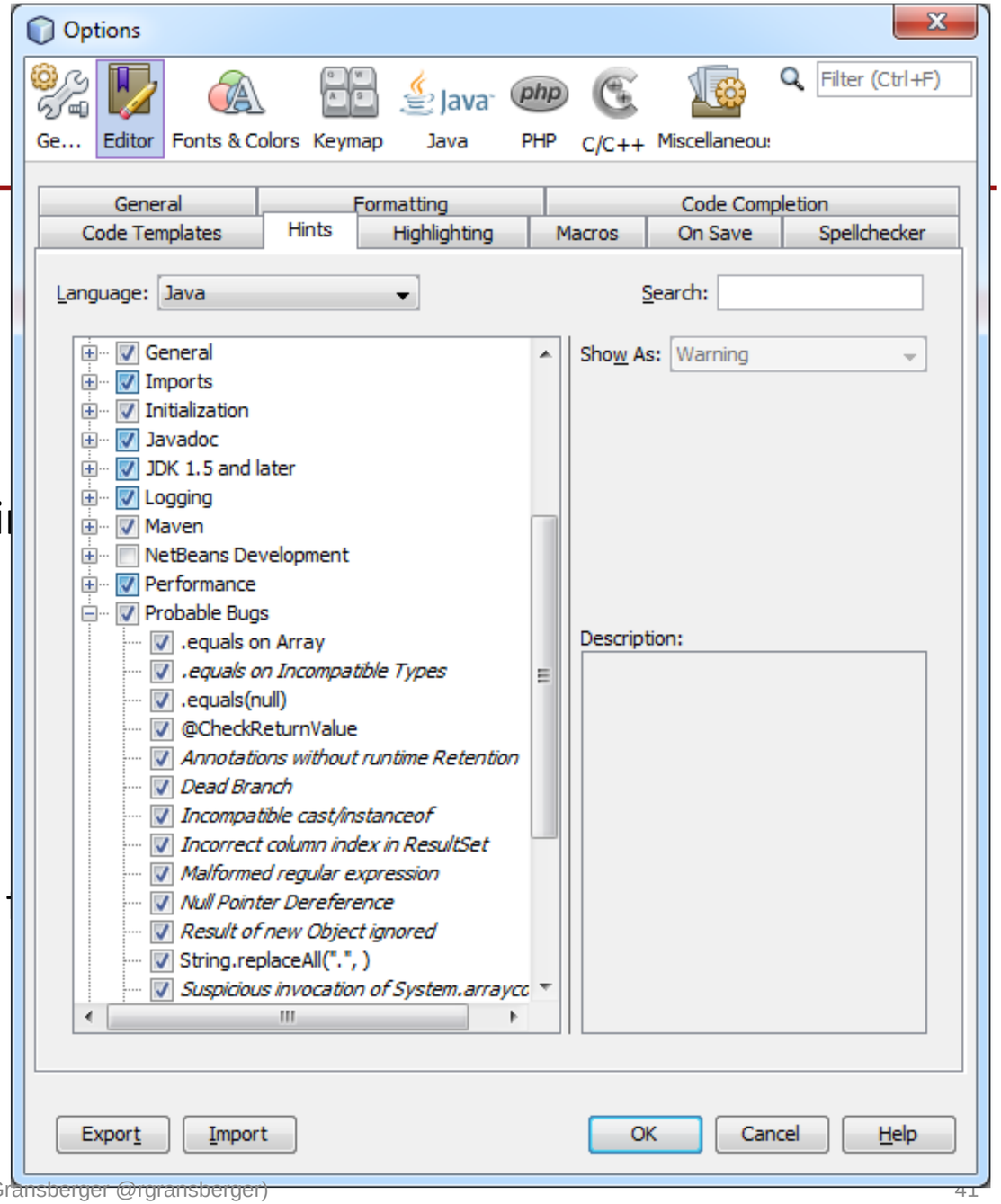
- Eclipse
  - ECJ: incremental compiler with additional warnings/errors
  - Java → Compiler → Errors / Warnings
  - Plugin: *CodePro Analytix*
- IntelliJ Idea: Inspections
- NetBeans: Editor → Hints
  - Powerful Inspect & Transform: e.g. apply Java 8 features

Static code analysis

Find potential errors before running

- Plugins for PMD/FindBugs (E)(JI)(N)

- Eclipse
  - ECJ: incremental compiler with additional warni
  - Java → Compiler → Errors / Warnings
  - Plugin: *CodePro Analytix*
- IntelliJ Idea: Inspections
- NetBeans: Editor → Hints
  - Powerful Inspect & Transform: e.g. apply Java 8



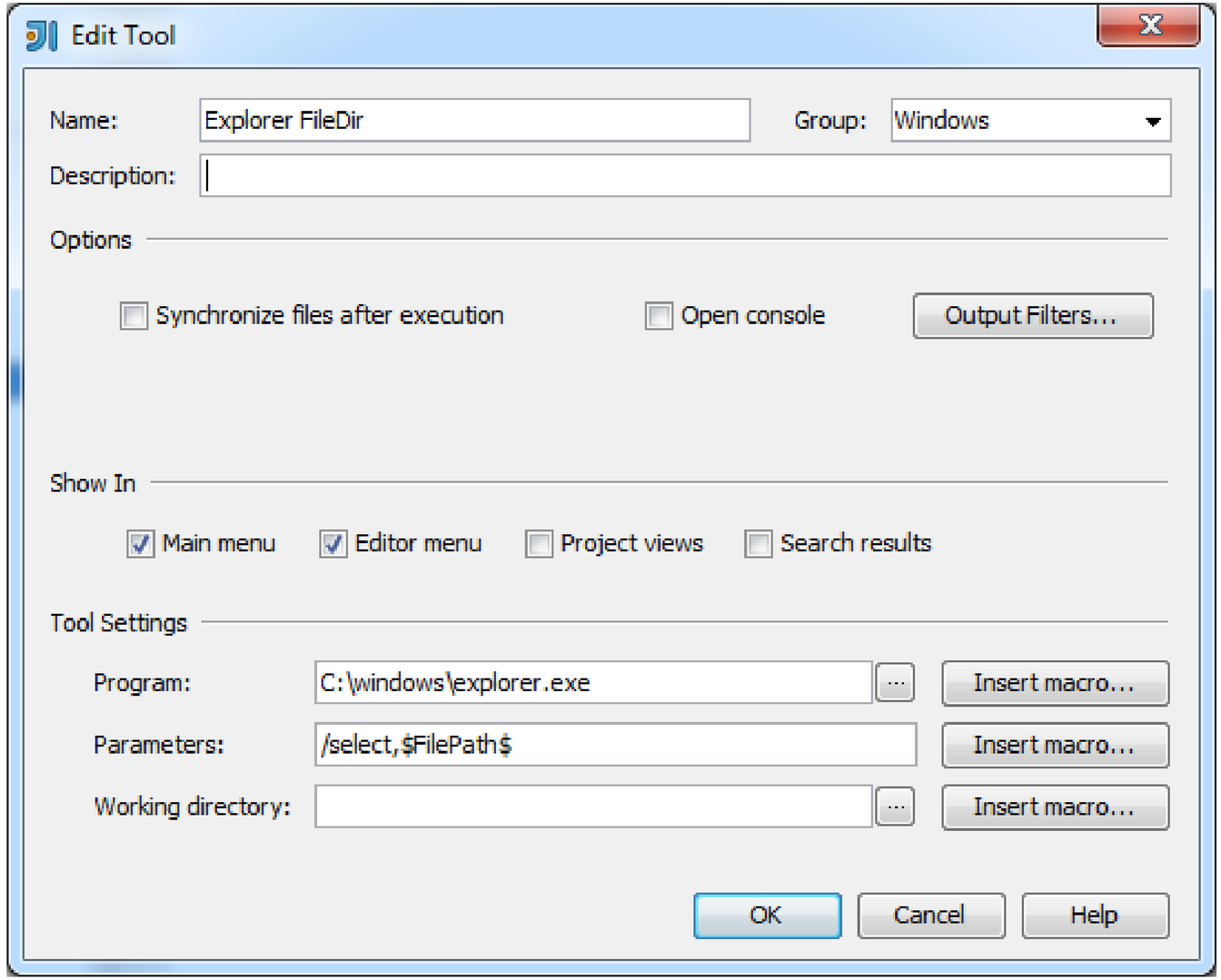
Console

- Settings for Console
  - (**E**) Run/Debug → Console
  - (**JI**) various locations, use search
  - (**N**) Miscellaneous → Terminal
- Change character limit of console
- Change Colors & Fonts
- Show on output and/or error

- (**E**) StackTrace Console (Console Toolbar), paste trace from log

External Tools

- Open explorer folder containing selected file
- Open console in IDE
- (E) Run → External Tools → External Tools Configuration
- (JI) External Tools
- (N) via Plugin
  - *Cool Editor Actions*
  - *Command Shortcuts*



Debugger

Debugger

- If HotSwap enabled when debugging, changes applied immediately
- Breakpoints can be set on class, methods, fields and exceptions too
- Conditional breakpoints
- Step-Filters: Skip simple getters, setters, certain classes
  - (**E**) Java → Debug → Step Filters
  - (**JI**) Debugger → Stepping
  - (**N**) Java → Java Debugger → Step Filters
- Shortcuts: Step over, Step into, Step return, Continue
  - (**E**) `F6/F5/F7/F8` (**JI**) `F8/F7/Shift+F8/F9` (**N**) `F8/F7/Ctrl+F7/F5`

Debugger II

- Pause Debugger to find current execution location
- Formatting for variables view
  - (**E**) Java → Debug → Detail Formatters
  - (**JI**) Debugger → Data Type Renderers
  - (**N**) Java → Java Debugger → Variable Formatters
- Different view for collections and default view `toString()`
  - (**E**) Toolbar: Debug → Variables: Show Logical Structure
  - (**JI**) Debugger → Data Views
  - (**N**) Part of variable formatters

Debugger III

- Evaluate Expression
  - (E) `Ctrl+Shift+I` (JI) (Ctrl+) `Alt+F8` (N) `Ctrl+F9`
- Recall previous method
  - (E) Context Menu: Drop to frame
  - (JI) Context Menu: Drop frame
  - (N) Context Menu: Pop
- References / Labels
  - (E) Variables View menu (JI) Use label to name an object (N) Variables Context Menu
- Monitors, Thread Groups, Deadlock Detection
  - (E) Debug View menu (JI) Debugger/Threads Context Menu → Customize
    (N) Debugging View Context Menu → Options

Miscellaneous

Java 8

- Release in March, 18th
  - Current beta versions are ready to test
- NetBeans and IntelliJ have Java 8 support in Release Versions
- Eclipse has Update Site for Java Development Tools for Java 8 preview

- IntelliJ does collapse anonymous classes to lambdas by default
  - Code is more readable but still compatible with Java 7 and below
  - Analyse → Inspect Code to find and replace with lamba expressions
- NetBeans: Source → Inspect
- Eclipse: Source → Clean Up

Eclipse Misc

- Fast incremental compiler, projects run even with errors
- Working Sets to group projects and use as top-level in navigation
- If a command doesn‘t work, look at the Error View (`Ctrl+Shift+V`)
- Breakpoints-View allows to group by project or create working sets
- Search View allows deleting and pinning
- File → New → Scrapbook Page: Test incomplete Statements in running JVM
- Find in search results trick:
  - Search → File… execute normally
  - Context menu in Search Results view: Expand All, Show In → Package Explorer
  - Search → File: Scope = Selected Resources

IntelliJ Misc

- Productivity Guide to see features which aren‘t used yet
- Supports *Language Injections* to get content assist in embedded SQL
- Use `Alt+Shift+C` to quickly review your recent changes to the project
- Structural search and replace
- Presentation Mode
- “Open copy in editor” option to prevent editing the wrong file ☺
- Search only in strings and comments
- Context aware content assist

NetBeans Misc

- You can profile your application from inside netbeans with the embedded profiler known from JVisualVM
- Load snapshots/heapdumps and compare snapshots. JConsole and JMeter are provided as plugins
- Clipboad history `Ctrl+Shift+D`
- JDK 8 compact profile support with warnings

More Information

- Did you know your IDE has a lot of build-in help pages?
  - (E) http://help.eclipse.org
  - (JI) http://www.jetbrains.com/idea/webhelp
  - (N) http://netbeans.org/kb/index.html
- Look for sections like
  - "Getting started"
  - "Tips and Tricks"
  - "Important Shortcuts"
- Read the Release Notes which mention new features

Bugs & "Missing" Features

- **E**clipse, Intelli**J** **I**dea and **N**etBeans can be extended via plugins
- File a feature request in the respective issue tracker
- Have a look at the API and develop your own plugin

- If you find a bug in the IDE, please report it:
  - (**E**) http://bugs.eclipse.org
  - (**JI**) http://youtrack.jetbrains.com/dashboard
  - (**N**) http://netbeans.org/bugzilla

Summary

- Explore the features of your favorite IDE: Know your Tools!
- Be lazy, let the IDE do the work with Content Assist / Quick Fix / Templates
- Create settings and share for the Team
- Do *“Learning shortcuts”*-sessions in regular time intervals

Questions?

Slides:

https://speakerdeck.com/rgra

GitHub (modified MouseFeed Plugin)**:**

https://github.com/rgra/mousefeed

Contact Information:

Rabea Gransberger

Twitter: @rgransberger