O'REILLY®

IntelliJ IDEA Dojo

Become a more effective JVM developer

Daniel Hinojosa

About this course

- This course is a thorough introduction to IntelliJ IDEA
- We will cover items like
  - Navigation and Tool Windows
  - Effective Test Driven Development
  - Maneuvering Efficiently
  - Multicursors
  - Templates
  - Version Control Management
  - Artificial Intelligence in IntelliJ and other JetBrains products
- We will have some labs to get those fingers working

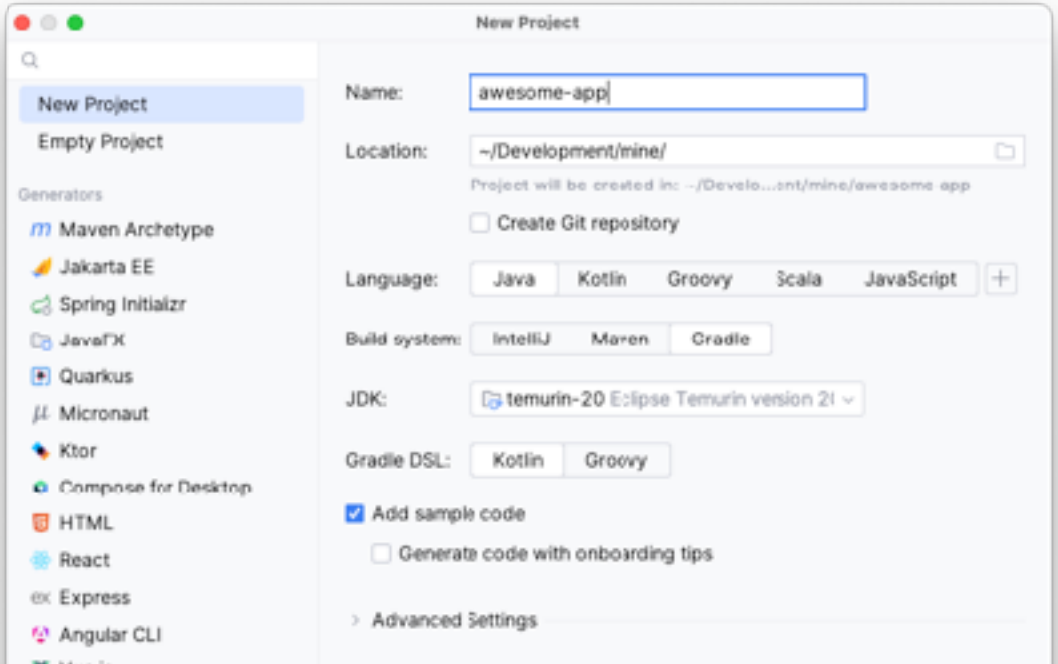
O'REILLY

IntelliJ Purpose, Tool Windows, and Files

New Projects

New Projects

- Typically, projects are created outside the IDE.
- Created and maintained by a build tool, i.e., Maven, Gradle, etc.
- But, IntelliJ offers some good choices without resorting to “IntelliJ-only” type projects



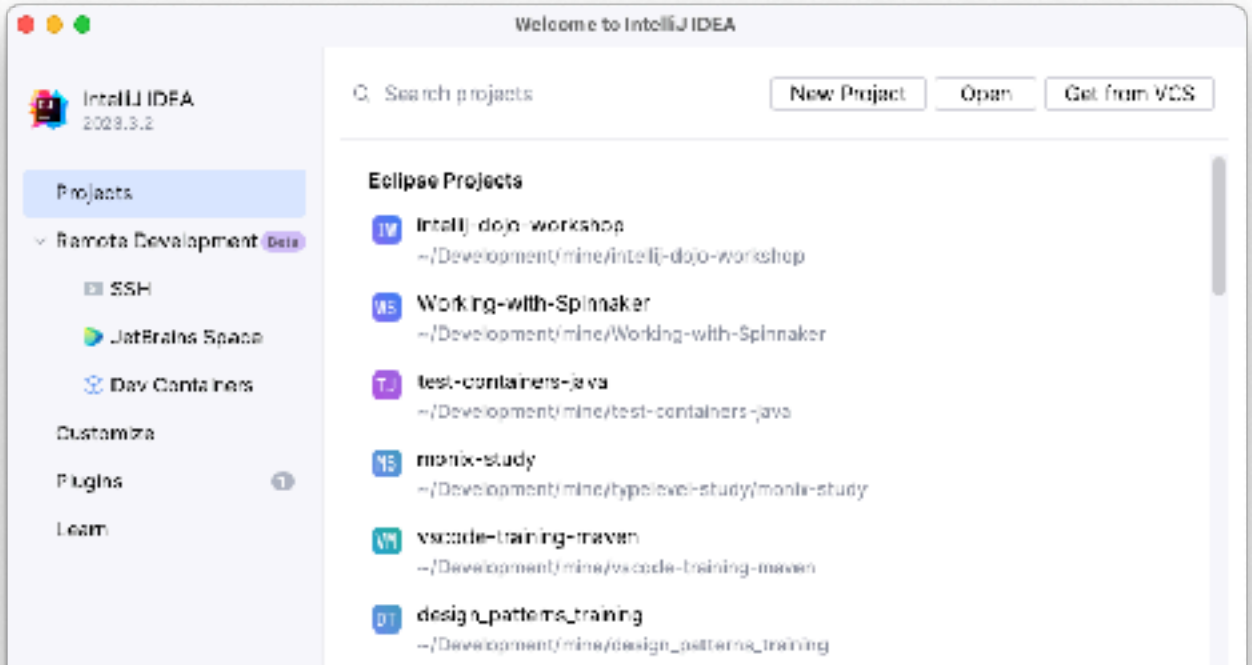
Opening Projects

Opening Projects

- Since projects are created outside the IDE with a build tool. i.e., Maven, Gradle, etc.
- All that needs to be done is open the project.
- This can be done with **[File > Open]** or **[File > Recent Projects]**

Opening Projects

- When you start IntelliJ for the first time, or when you close out of currently open projects, you can open your projects in the Welcome to IntelliJ window.



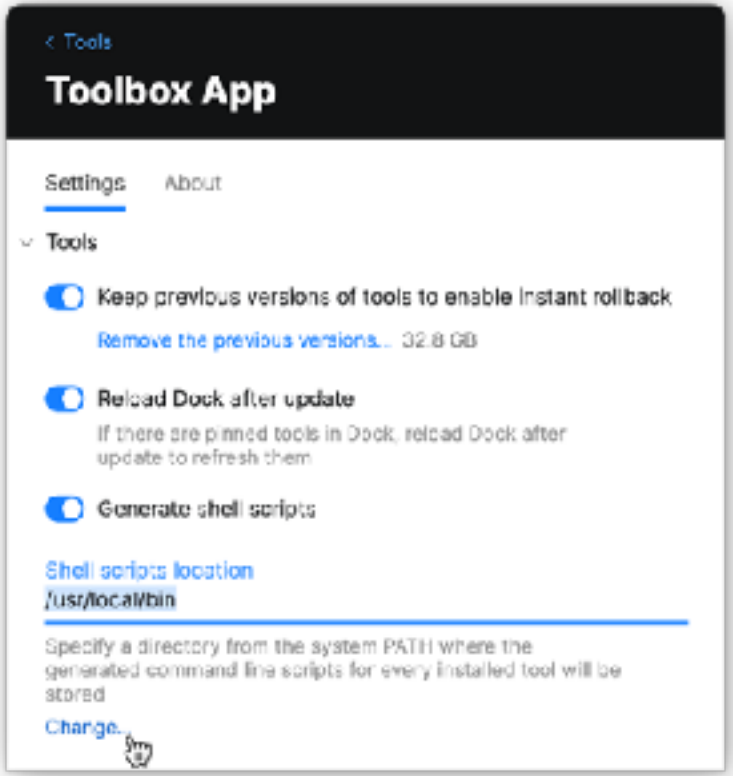
Command Line

Opening Projects using the Command Line

- If you want to open a project into IntelliJ from command line you can do so from command line by typing `idea .`
- After some setup…

Generating Scripts

- In your JetBrains toolbox, click on the bolt.
- Click on **Settings**
- Toggle on “Generate Shell Scripts” and either keep or override the location of the scripts
- For Windows users, put this location on your PATH



Tool Windows

Tool Windows & Navigation

ALT-0...ALT-9
Open Corresponding Tool Window
CTRL-E
View Recent Files, Includes Tool Windows
CTRL-SHIFT+E
View Recent Edited Locations
ESC
Return to the Editor
CTRL+SHIFT+F12
Maximize the Editor
SHIFT+ESCAPE
Close Current Tool Window
CTRL+ALT+SHIFT+LEFT/ CTRL+ALT+SHIFT+RIGHT
Shrink or Expand Tool Window

Tool Windows & Navigation

COMMAND(⌘) + 0...COMMAND(⌘) +9
Open Corresponding Tool Window
COMMAND(⌘)-E
View Recent Files, Includes Tool Windows
COMMAND(⌘)-SHIFT+E
View Recent Edited Locations
ESCAPE(⎋)
Return to the Editor
COMMAND(⌘)+SHIFT(⇧)+F12
Maximize the Editor
SHIFT(⇧)+ESCAPE(⎋)
Close Current Tool Window
CONTROL(^) + OPTION(⌥) + LEFT(←)/
CONTROL(^) + OPTION(⌥) + RIGHT(→)
Shrink or Expand Tool Window

Find Action

Action Keys

CTRL-SHIFT-A Find Action
CTRL-CTRL Run Anything
SHIFT-SHIFT Search Everywhere

Action Keys

SHIFT(⇧)+COMMAND(⌘) + A Find Action
CONTROL(^) + CONTROL(^) Run Anything
SHIFT(⇧)+ SHIFT(⇧) Search Everywhere

Tab Management

Tab Management Keys

ALT+RIGHT/LEFT Go to Next or Previous Tab
CTRL+F4 Close Current Tab
[Action Only] Close All Tabs

Tab Management Keys

CONTROL(^) + RIGHT(→)/LEFT(←) Go to Next or Previous Tab
COMMAND(⌘) + W Close Current Tab
COMMAND(⌘) +SHIFT(⇧) + W Close All Tabs

Finding Items

Finding Items

CTRL-N Find Class
CTRL-SHIFT-N Find File
CTRL-ALT-SHIFT-N Find Symbol

Finding Items

COMMAND(⌘) + O Find Class
COMMAND(⌘) + SHIFT(⇧) + O Find File
COMMAND(⌘) + OPTION(⌥) + O Find Symbol


Dojo: Tool Windows

Q&A

O'REILLY

Test Driven Development

Test Driven Development

Test Driven Development

1. Quickly add a test.
2. Run all tests and see the new one fail.
3. Make a little change.
4. Run all tests and see them all succeed.
5. Refactor to remove duplication.

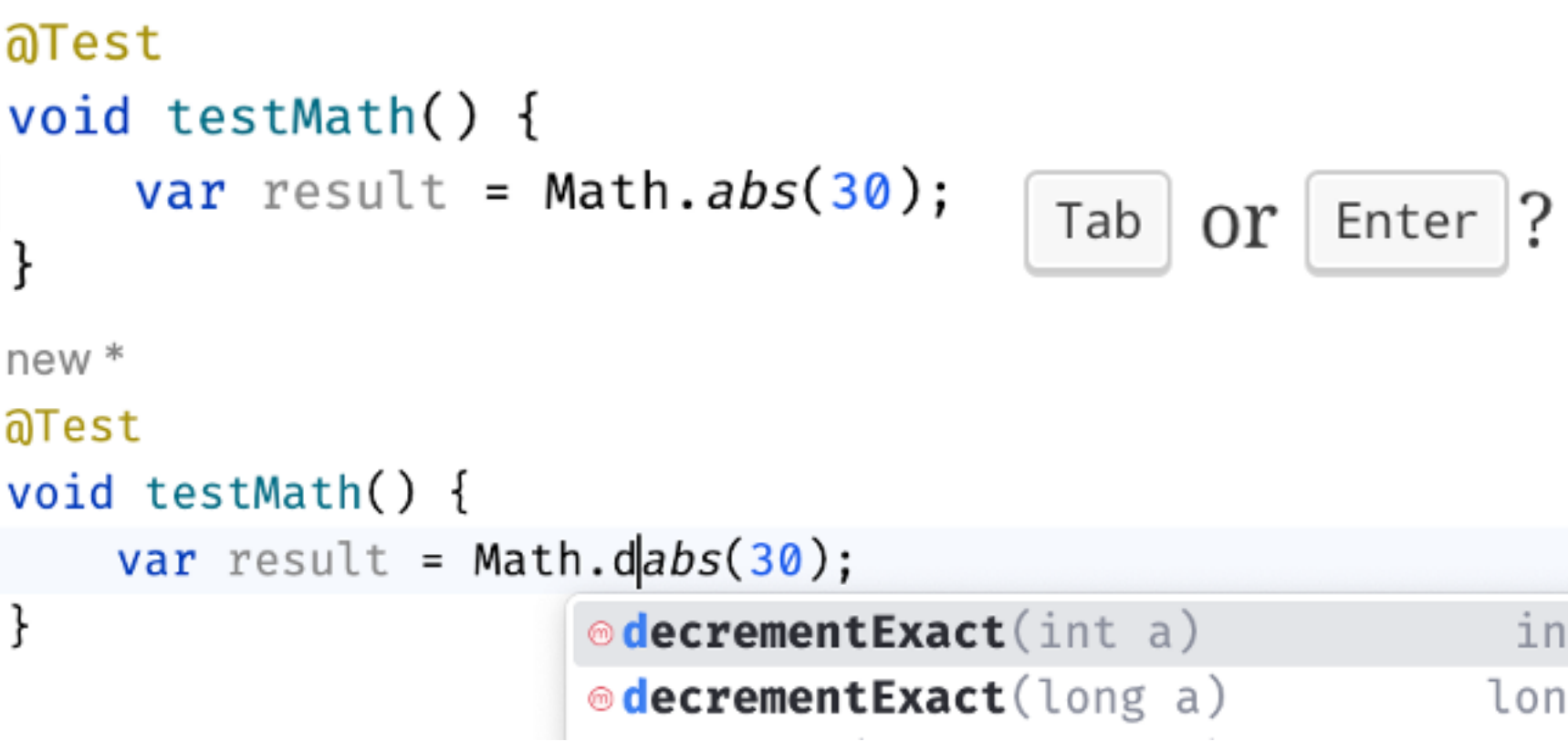
Kent Beck – Test Driven Development By Example 2003

Test Driven Development Benefits

- Promotes design decisions up front
- Allows you and your team to understand your code
- Model the API the way you want it to look
- Means of communicating an API before implementation
- Avoids technical debt
- Can be used with any programming language

Prefer Tabs Over Enter

Prefer Tab over Enter

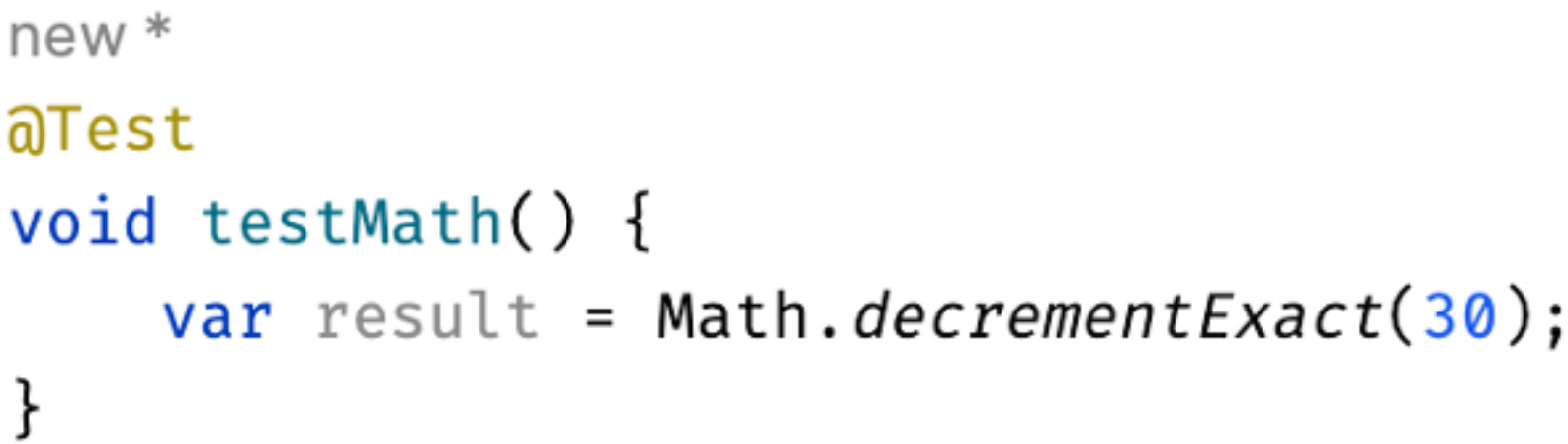


Results with Enter

```
new *
@Test
void testMath() {
    var result = Math.decrementExact()abs(30);
}
```

int a
long a

Results with Tab



Generating Files and Code

Generating Files and Code

ALT+INSERT Generate File or Code

Generating Files and Code

COMMAND(⌘) + N or CONTROL(^) + RETURN(↵)

Generate File or Code

Navigating Errors

Navigating Errors

F2

Move to the Next Highlighted Issue

ALT+ENTER

Propose a Fix

Navigating Errors

F2

Move to the Next Highlighted Issue

OPTION(⌥) + RETURN(↵)

Propose a Fix

Testing

Navigating Errors

CTRL + SHIFT + T

Toggle Between Test and Production

Navigating Errors

COMMAND(⌘) + SHIFT (⇧) + T

Toggle Between Test and Production

Code Completion

Code Completion

CTRL + SPACE

Basic Code Completion

CTRL + ALT + SPACE

Code Completion that shows inaccessible members

CTRL + SHIFT + SPACE

Type Sensitive Code Completion

CTRL + SHIFT + ENTER

Complete Statement

CTRL + P

Parameter Info

CTRL + SHIFT+P

Type Info

Code Completion

CONTROL(^) + SPACE

Basic Code Completion

CONTROL(^) + OPTION (⌥) + SPACE

Code Completion that shows inaccessible members

CONTROL(^) + SHIFT(⇧) + SPACE

Type Sensitive Code Completion

COMMAND(⌘) + SHIFT(⇧) + RETURN(↵)

Complete Statement

COMMAND(⌘) + P

Parameter Info

CONTROL(^) + SHIFT(⇧) +P

Type Info

Running

Running

All runs are contextual, this includes main methods, tests, test folders, HTTP files, and more.

CTRL + SHIFT + F10

Run and mark your context configuration

SHIFT + F10

Run the last context configuration

ALT+SHIFT+F10

Select context configuration and run

Running

All runs are contextual, this includes main methods, tests, test folders, HTTP files, and more.

CONTROL(^) + SHIFT (⇧) + R

Run and mark your context configuration

CONTROL(^) + R

Run the last context configuration

CONTROL(^) + OPTION(⌥) + R

Select context configuration and run


Dojo: Test Driven Development

Q&A

Break

O'REILLY

Essential Maneuvers

Formatting Code

Formatting Code

CTRL + ALT + L

Format All Code (includes imports)

CTRL + ALT + O

Optimize Imports

CTRL + ALT + I

Format Line by Line

Formatting Code

COMMAND(⌘) + OPTION(⌥) + L

Format All Code (includes imports)

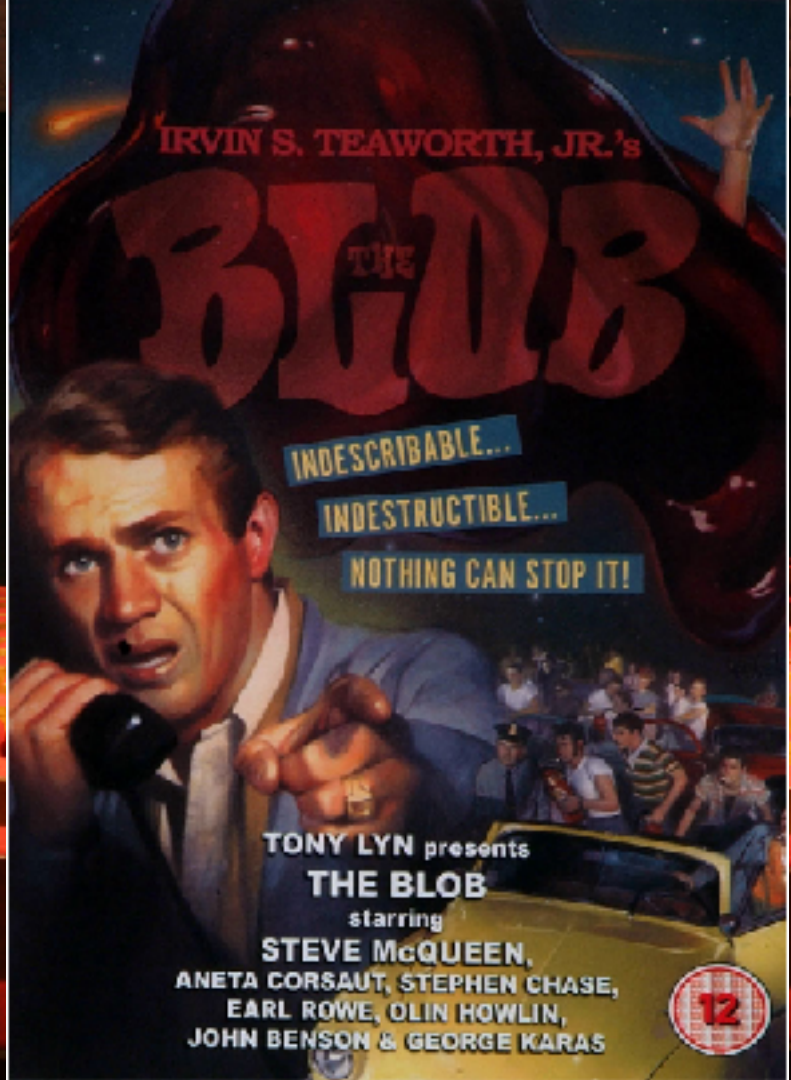
CONTROL(^) + OPTION(⌥) + O

Optimize Imports

CONTROL(^) + OPTION(⌥) + I

Format Line by Line

Selecting Code Blocks


IRVIN S. TEAWORTH, JR.'s
THE BLOB
INDESCRIBABLE...
INDESTRUCTIBLE...
NOTHING CAN STOP IT!
TONY LYN presents
THE BLOB
starring
STEVE McQUEEN,
ANETA CORSAUT, STEPHEN CHASE,
EARL ROWE, OLIN HOWLIN,
JOHN BENSON & GEORGE KARAS
12

Selecting Code Blocks

CTRL + W

Increase Code Block

CTRL + SHIFT + W

Decrease Code Block

Selecting Code Blocks

OPTION(⌥) + UP(↑)

Increase Code Block

OPTION(⌥) + DOWN(↓)

Decrease Code Block

Editing Basics

Editing Basics

CTRL+Y
Delete Line at Caret
CTRL+D
Duplicate Line at Caret
CTRL+X
Cut Line at Caret
CTRL+C
Copy Line at Caret
CTRL+V
Paste Line at Caret
CTRL+SHIFT+V
Paste From History

Editing Basics

COMMAND(⌘)+DELETE(⌫)
Delete Line at Caret
COMMAND(⌘)+D
Duplicate Line at Caret
COMMAND(⌘)+X
Cut Line at Caret
COMMAND(⌘)+C
Copy Line at Caret
COMMAND(⌘)+V
Paste Line at Caret
COMMAND(⌘)+SHIFT(⇧)+V
Paste From History

Surround Blocks and Selections

Surround Blocks and Selections

CTRL + ALT + T

Surround Block or Selection

Surround Blocks and Selections

COMMAND(⌘) + OPTION(⌥) + T

Surround Block or Selection

Refactoring

Refactoring (Part 1)

CTRL + ALT+ SHIFT + T

Refactor This…

F5

Copy item

F6

Move item

SHIFT+F6

Rename globally

ALT+DELETE

Safe Delete

Refactoring (Part 2)

Ctrl + Alt + N
Inline
Ctrl + Alt + M
Extract Method
Ctrl + Alt + V
Extract Variable
Ctrl + Alt + F
Extract Field
Ctrl + Alt + C
Extract Constant
Ctrl + Alt + P
Extract Parameter

Refactoring (Part 1)

CONTROL(^) + T

Refactor This

F5

Copy item

F6

Move item

SHIFT(⇧)+F6

Rename globally

COMMAND(⌘)+DELETE

Safe Delete

Refactoring (Part 2)

COMMAND(⌘) + OPTION(⌥) + N

Inline

COMMAND(⌘) + OPTION(⌥) + M

Extract Method

COMMAND(⌘) + OPTION(⌥) + V

Extract Variable

COMMAND(⌘) + OPTION(⌥) + F

Extract Field

COMMAND(⌘) + OPTION(⌥) + C

Extract Constant

COMMAND(⌘) + OPTION(⌥) + P

Extract Parameter


Dojo: Editing and Refactoring

O'REILLY

Navigating Code

Usages

Usages

CTRL + ALT + B

Go to Implementation

CTRL + B

Go to Declaration

ALT + F7

Find Usages

CTRL + F7

Find in Files

CTRL + ALT + LEFT / RIGHT

Navigate back / forward

Usages

COMMAND(⌘) + OPTION(⌥) + B

Go to Implementation

COMMAND(⌘) + B

Go to Declaration

OPTION (⌥) + F7

Find Usages

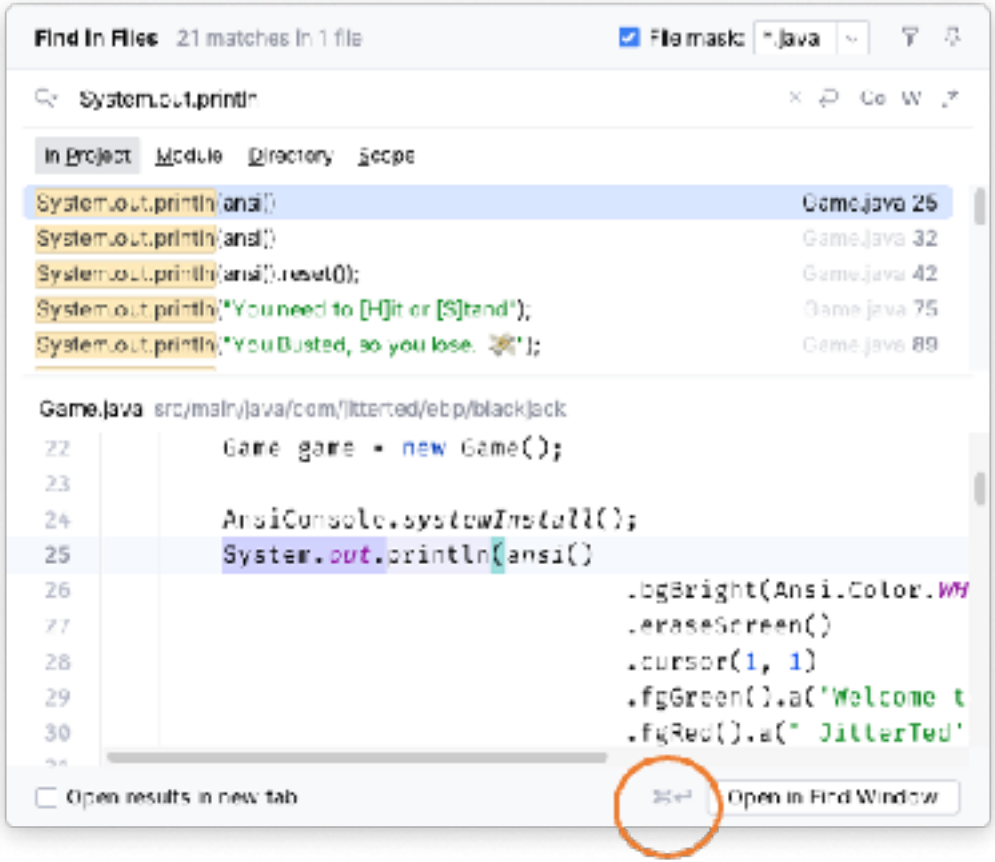
COMMAND(⌘) + SHIFT(⇧) + F7

Find In Files

COMMAND(⌘) + OPTION(⌥) + LEFT(←) / RIGHT(→)

Navigate back / forward

Route “Find in Files” to Find Window



Moving Code

Moving Code

CTRL + SHIFT + UP/DOWN

Move either line or method, depending on caret location

Moving Code

SHIFT (⇧) + COMMAND(⌘) + UP(↑)/DOWN(↓)

Move either line or method, depending on caret location

Code Documentation and Help

Code Documentation and Help

CTRL + Q

Quick Documentation Lookup

CTRL + SHIFT + I

Quick Definition Lookup

ALT+Q

Context Info

CONTROL(^) + SHIFT (⇧) + P

Type Info (Twice to show more)

Code Documentation and Help

CONTROL(^) + J

Quick Documentation

OPTION(⌥) + SPACE or COMMAND (⌘) +Y

Quick Definition Lookup

CONTROL(^) + SHIFT (⇧) + Q

Context Info

CONTROL(^) + SHIFT (⇧) + P

Type Info (Twice to show more)

Hierarchies

Hierarchies

CTRL+H

Type Hierarchy - Show Parent and Child Classes

CTRL+SHIFT+H

Method Hierarchy – Show subclasses where the method overrides the selected method as well as what is overridden

CTRL+ALT+H

Call Hierarchy – Callers (Supertypes) or Callees (Subtypes) of a method, if on a field it will show the the methods that interact with it

Hierarchies

CONTROL(^) +H

Type Hierarchy - Show Parent and Child Classes

COMMAND (⌘) + SHIFT(⇧) +H

Method Hierarchy – Show subclasses where the method overrides the selected method as well as what is overridden

CONTROL(^) + OPTION(⌥) +H

Call Hierarchy – Callers (Supertypes) or Callees (Subtypes) of a method, if on a field it will show the the methods that interact with it

Structure

Structure

ALT + 7

View the Structure of the Class

CTRL+F12

File Structure Popup

Structure

COMMAND(⌘) + 7

View the Structure of the Class

COMMAND(⌘) +F12

File Structure Popup


Dojo: Navigating

O'REILLY

Additional Cool Stuff

Debugging

Debugging

CTRL+F8

Toggle Breakpoint

CTRL + SHIFT+ F8

View Breakpoints

CTRL+SHIFT+F9

Debug and Mark your Context Configuration

SHIFT+F9

Debug the Last Context Configuration

ALT+SHIFT+F9

Select Context Configuration and Debug

Debugging

COMMAND(⌘) + F8

Toggle Breakpoint

COMMAND(⌘) + SHIFT (⇧) + F8

View Breakpoints

CONTROL(^) + SHIFT (⇧) + D

Debug and Mark your Context Configuration

CONTROL(^) + D

Debug the Last Context Configuration

CONTROL(^) + OPTION(⌥) + R

Select Context Configuration and Debug

Live Templates

Live Templates

- Use live templates to insert common constructs into your code, such as:
  - Loops
  - Conditions
  - Various Declarations
  - Print Statements
- Configure Live Templates by going to going to **[Settings > Editor > Live Templates]**
- You can find **Live Templates** by using **Find Action**

Live Template Variables

- `$END$` indicates the position of the caret when the code snippet is complete, and you can no longer press Tab to jump to the next variable.
- `$SELECTION$` is used in surround templates and denotes the code fragment to be wrapped. After the template expands, it wraps the selected text as specified in the template.

Live Template Functions

- IntelliJ has wide selection of Live Template Functions available
- The functions are used with your templates to refine what to do with live template variable. Functions include:
  - `enum`
  - `time`
  - `suggestIndexName`
  - `snakeCase`
  - more..
- You can use **Edit Template Variables** to add these custom expressions.

File Templates

File Templates

- Instead of code snippets that can be created in a live template, a *File Template*
- Provides initial code as a file, this can be anything from:
  - Corporate Templates
  - Favorite Test File Structure
  - Model, View, Controller Templates
  - Common Constructs from your favorite framework
- Configure Live Templates by going to going to
  [Settings > Editor > File and Code Templates]
- You can find **File Templates** by using **Find Action**

File Template Variables

- Variables, are also available for file templates
- This can be used to substitute pertinent information into your template at the time of creation
- Variables include
  - `${DATE}`
  - `${PACKAGE_NAME}`
  - `${FILE_NAME}`
  - more..

Multicursors

Multicursors

ALT+SHIFT+ [Click]

- Start Multicursors
- Delete Single Multicursor
- Note: Can be used in selections (blobs) as well

ALT+J

- Select Next Word Occurrence

ALT+SHIFT+J

- Select All Word Occurrences
- Can be used with Find…

ALT + SHIFT + G

- Add Carets At The End of Lines in Selected Block

Multicursors

OPTION(⌥) + SHIFT(⇧) + [Click]

Start Multicursors

Delete Single Multicursor

Note: Can be used with selections (blobs) as well

CONTROL(^) + G

Select Next Word Occurrence

CONTROL(^) + COMMAND(⌘) + G

Select All Word Occurrences

Can be used with Find…

OPTION(⌥) + SHIFT(⇧) + G

Add Carets At The End of Lines in Selected Block

Column Selection Mode

Column Selection Mode

ALT+SHIFT+INSERT

Column Selection Mode

CTRL+CTRL

Column Selection Mode (Alternate)

Column Selection Mode

SHIFT(⇧) + COMMAND(⌘) + 8

Column Selection Mode

OPTION(⌥) + OPTION(⌥)

Column Selection Mode (Alternate)

HTML/XML Editing

HTML/XML Editing with Emmet

- http://docs.emmet.io/abbreviations/syntax/
- https://docs.emmet.io/cheat-sheet/

Version Control

Version Control

ALT + 0

Version Control Window

CTRL + K

Commit

CTRL+T

Update Project

CTRL + SHIFT + K

Push

ALT + ` (Backquote)

Version Control Quick Help

Version Control

COMMAND(⌘) + 0

Version Control Window

COMMAND(⌘) + K

Commit

COMMAND(⌘) + T

Update Project

COMMAND(⌘) +SHIFT(⇧) + K

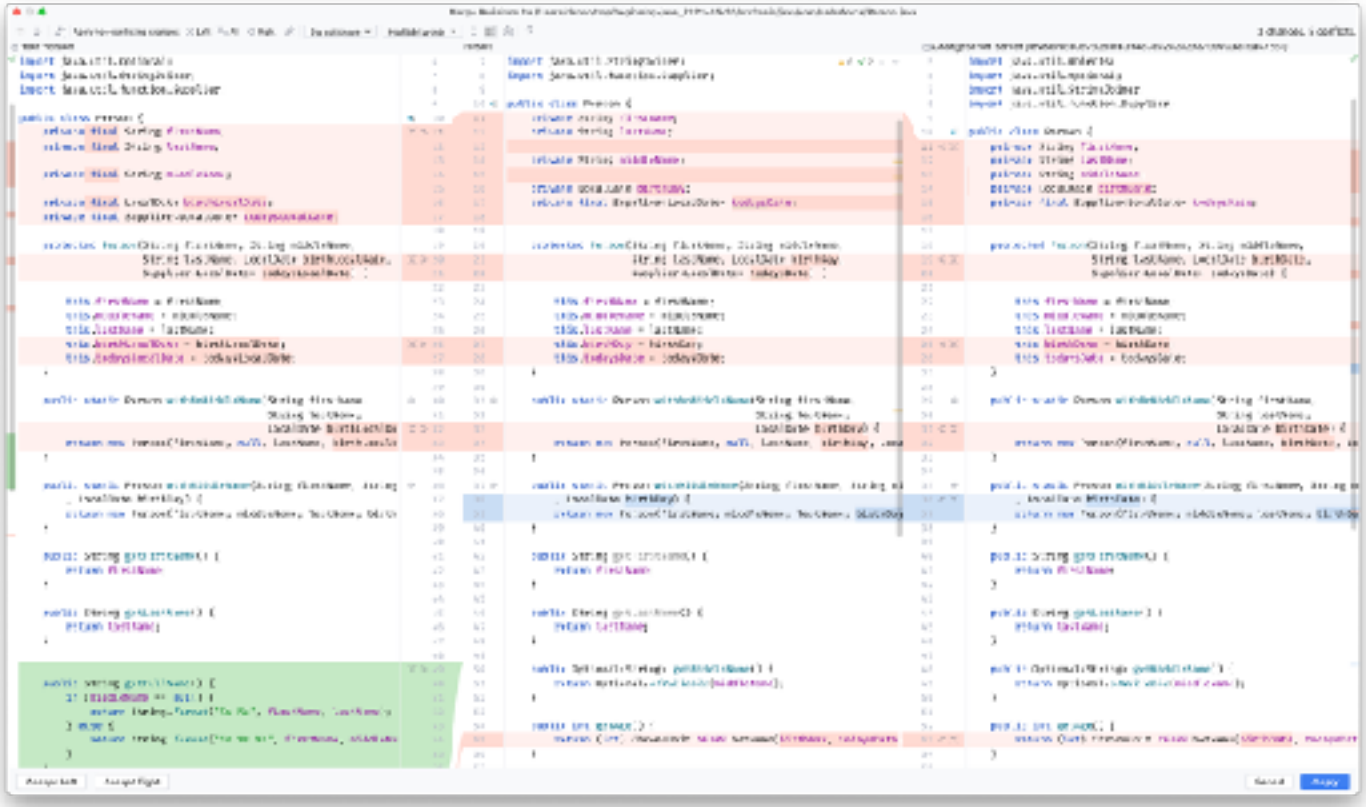
Push

CONTROL(^) + V

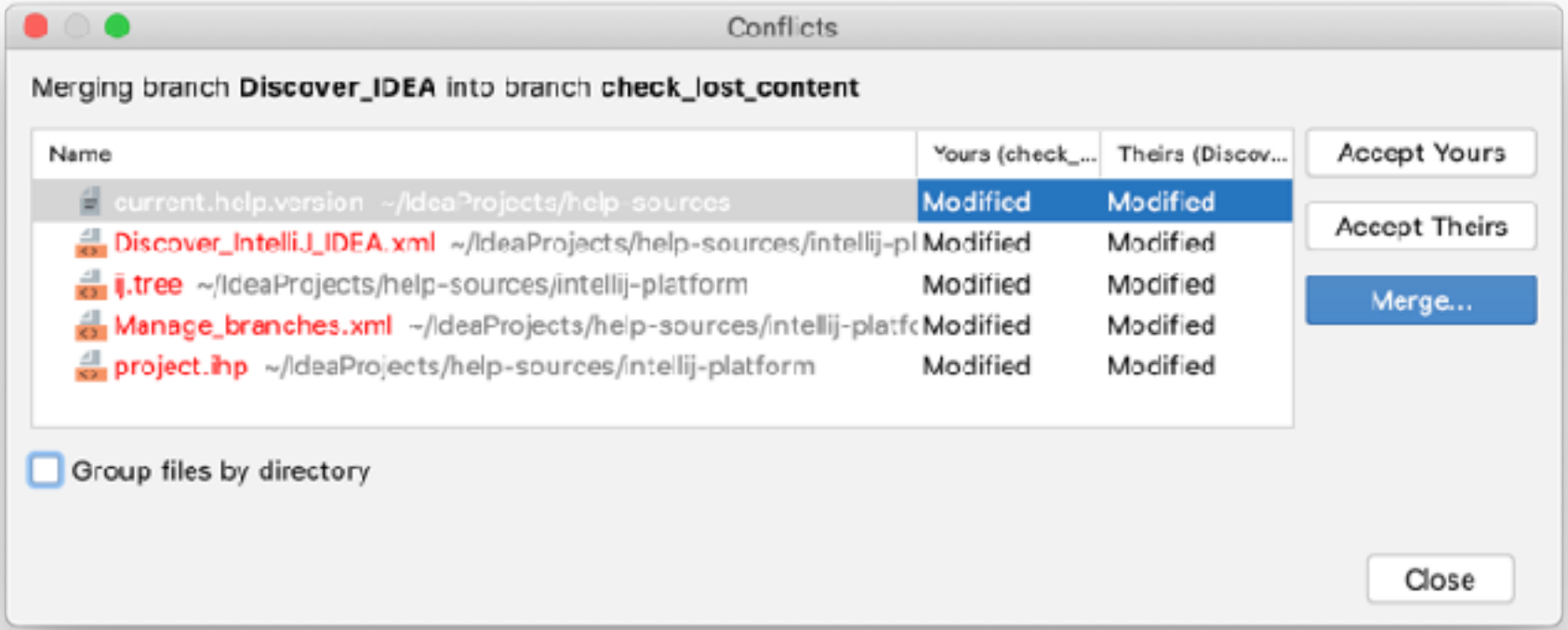
Version Control Quick Help

Version Control Conflicts

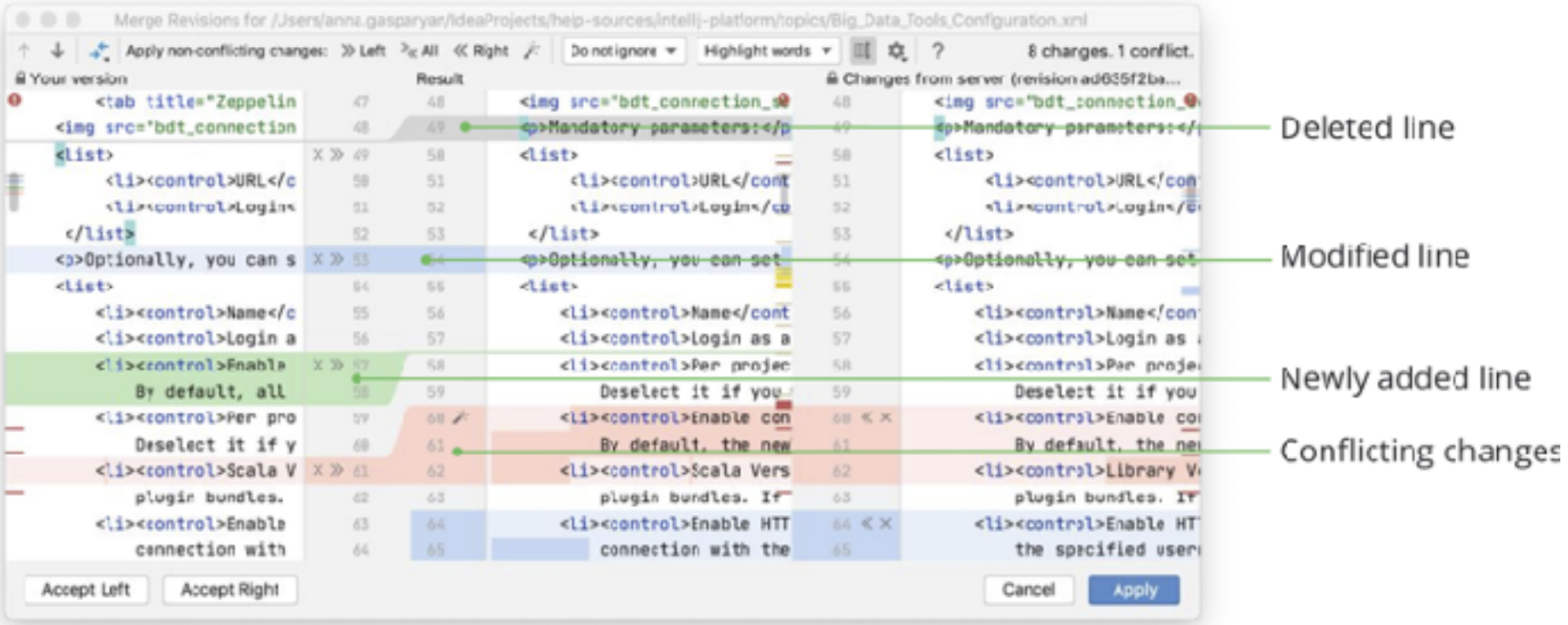
Version Control Conflicts



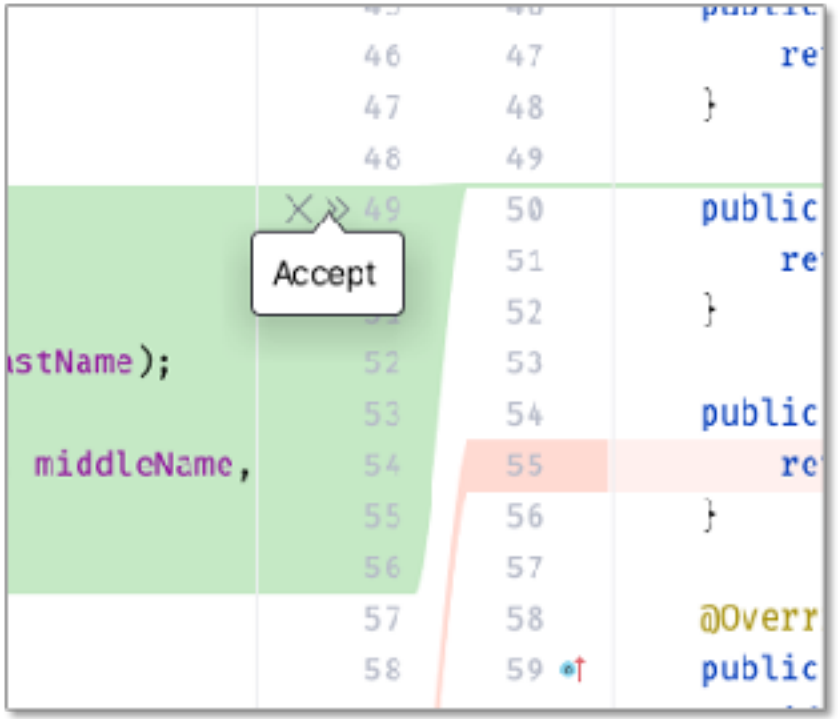
Accept or Merge



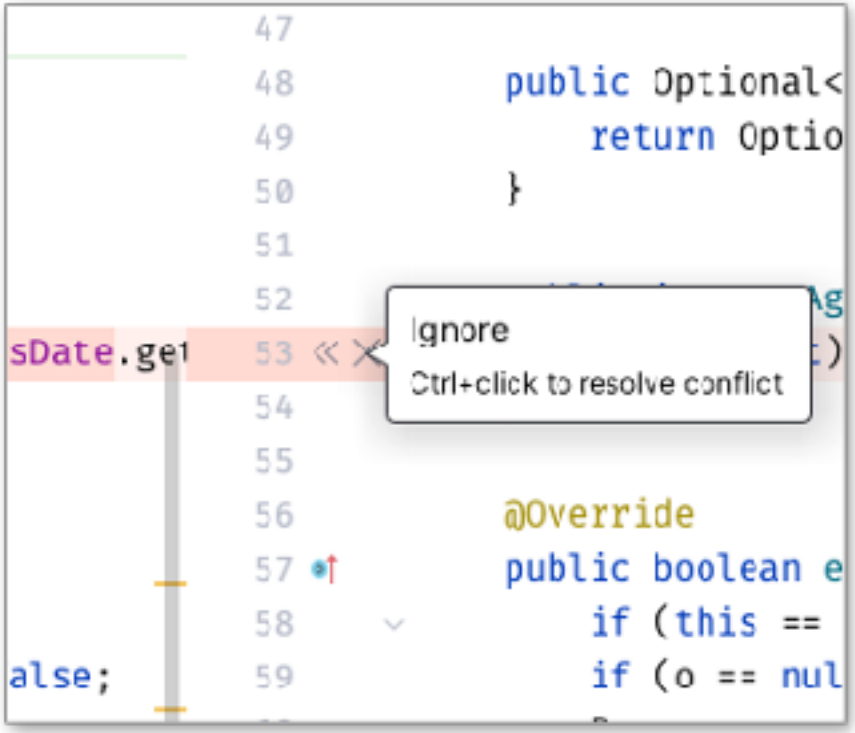
Color Meanings



Accepting Conflicts



Ignoring Conflicts



Version Control Conflicts

F7

Next Difference

SHIFT+F7

Previous Difference

Version Control Conflicts

F7

Next Difference

SHIFT(⇧) + F7

Previous Difference


Dojo: Multicursors and Templates

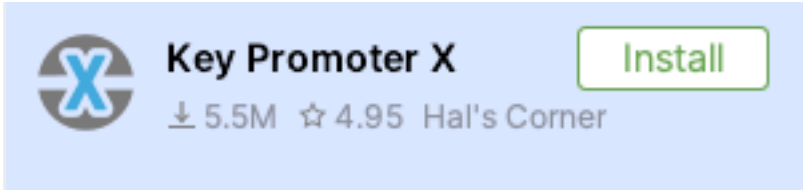
O'REILLY

Conclusion

Continual Learning

You don’t have to grok all this in one day

- There is a lot to take in when learning a new editor and becoming efficient
- This presentation is to
  - Show all the possibilities. Even if you don’t have it all memorized
  - Serve as a reference so you can know what to demand from other editors/IDEs
- When using IntelliJ after today, some things to consider:
  - Keep a personal cheat sheet handy, but only the items that you don’t have memorized
  - Keep in the back of your mind that there must be a key binding, and look it up using **Find Action**
  - Try **Key Promoter X**
  - For Presentations try **Presentation Assistant**
- **Subscribe to the JetBrains IntelliJ channel on YouTube:**
  https://www.youtube.com/@intellijidea



AI and IntelliJ IDEA

AI Assistant

- JetBrains now offers AI solutions as either a trial or as an ultimate package
- AI Assistant Services include:
  - Chat Question and Answers
  - Assistance in writing tests, commit messages, and documents
  - Research common bugs
  - Ask for solutions to common problems
  - Provide naming assistance
  - Provide refactoring help
- Levels
  - Trial - Equal to the Pro Plan, but with limitations
  - Pro - $8.33 per month/$100.00 per year
  - Enterprise - Multi seat pro plan and customization

Poll Question

What are some of the favorite things that you've learned?

Q&A


O'REILLY®