IntelliJ Wizardry with Heinz Kabutz

Dr Heinz M. Kabutz

Last Updated 2022-03-01

Javaspecialists.eu
java training

Copyright Notice

1: Introduction

Why IntelliJ IDEA?

- My story
  - Started with Borland JBuilder
  - Then used Eclipse for a year
  - Needed something to work with horrible messy Java code
    - And create some Swing GUIs at the same time
  - Downloaded IntelliJ IDEA
    - No free version at the time
    - Used it for 30 days
    - Bought it
  - Paid twice
    - Once for the license and then reduced hours worked
      - But work was far more pleasant, less frustration, better life

Questions

- Please please please please ask questions!
  - For self-study, please leave comment in sections
- Interrupt me at any time
  - Questions that are off-topic might be delayed until later
  - If so, please write down question and we can look at it during exercise time
- There are some stupid questions
  - They are the ones we did not ask
  - Once we have asked them, they are not stupid anymore
- The more we ask, the more everyone learns

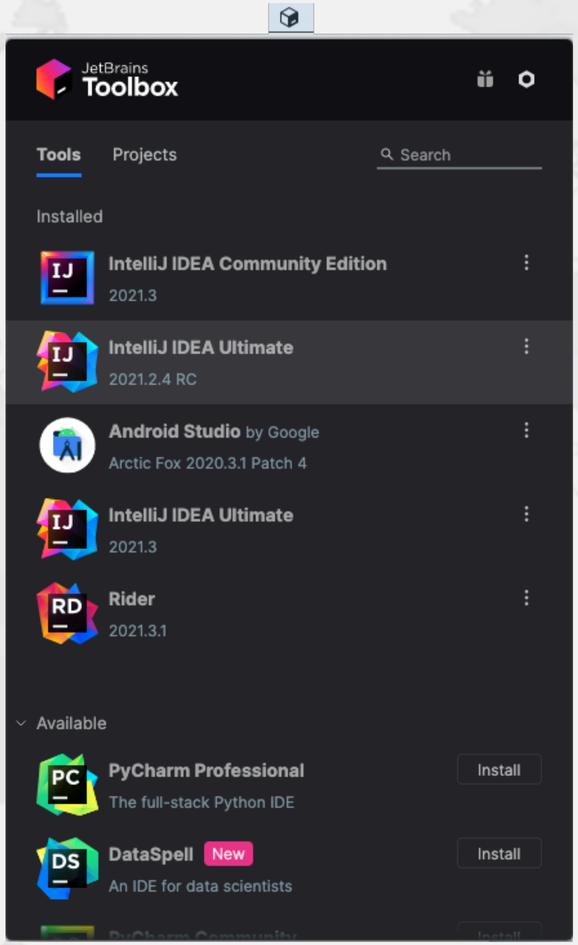
Exercises

- We learn cycling by falling
  - Listening to lectures is not enough
  - We need to build up muscle memory in our fingers
- Exercises help us to practice the concepts
  - We have a git repository for this course
    - My work will be in the "kabutz" branch
    - Please don't push to master :-)

2: Setting Up IntelliJ IDEA

Download JetBrains Toolbox

- **Keeps the IDEs up to date**
  - **https://www.jetbrains.com/toolbox-app/**



Install IntelliJ IDEA Community Edition

- Everything we do today will work in the free IDE
  - But there are a lot of fantastic features in Ultimate version
- Simply open your JetBrains Toolbox and install
  - IntelliJ IDEA Community Edition
- Or download it directly here
  - https://www.jetbrains.com/idea/download/

Importing Project from VCS

- Start IntelliJ IDEA and select "Get from VCS"
  - Version control: Git
  - URL: https://user:pass@javaspecialists.eu/git/safari/intellij-wizardry-####.git
    - Link is *not* browseable
  - Directory: Wherever you like
- "Open Project Structure dialog"
  - Windows/Linux: Ctrl + Alt + Shift + S
  - Mac OS X: ⌘;
  - Set up SDK to be JDK 17
    - You might have to download that

Settings Dialog

- **"Open Settings dialog"**
  - **Windows/Linux: Ctrl + Alt + S**
  - **Mac OS X: ⌘,**

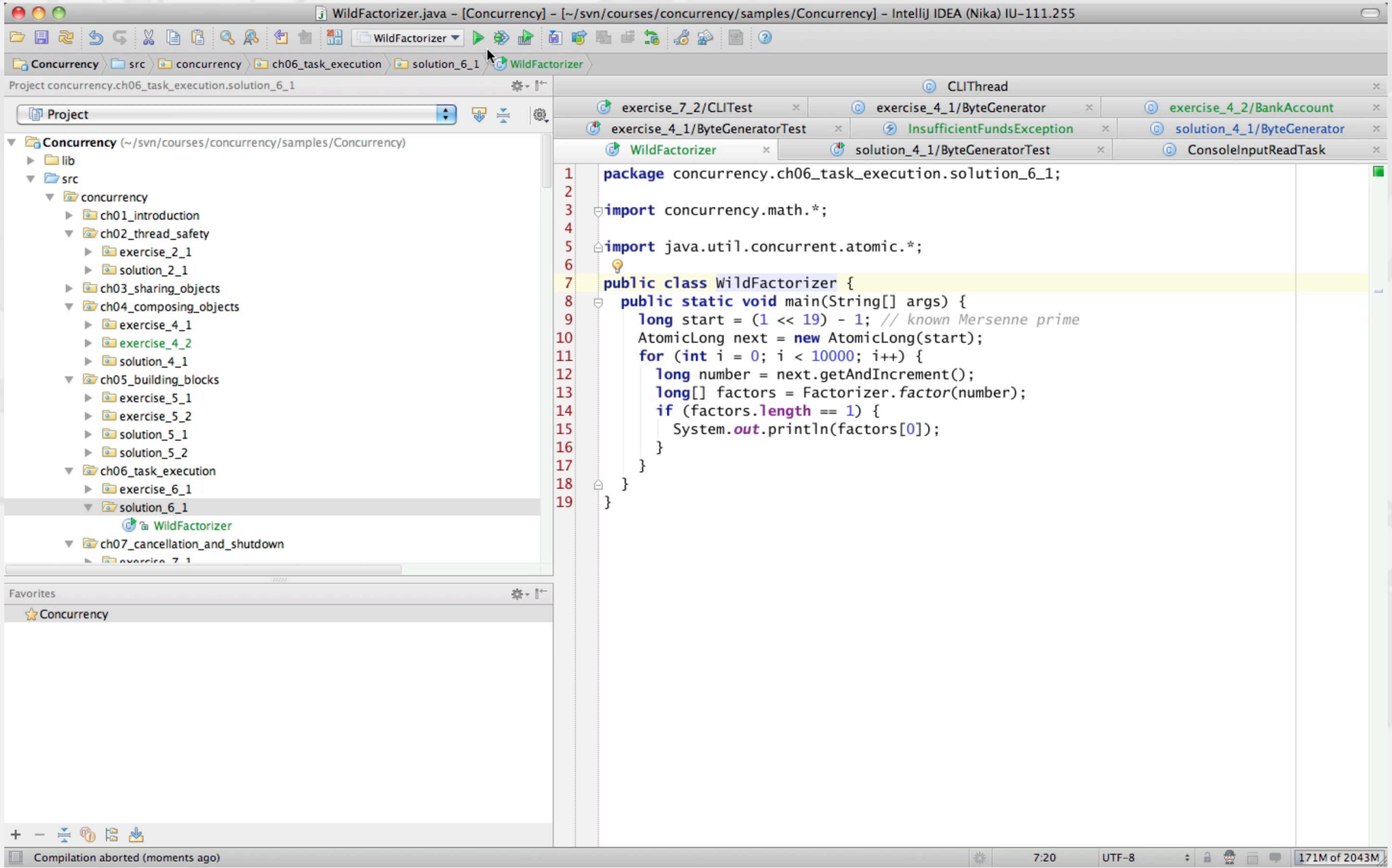
Setting up IntelliJ IDEA Community Edition

- **Appearance & Behavior**
  - Darcula Scheme most popular, but errors harder to see
- **Keymap, copy Mac OS X**
  - Copy "Mac OS X Heinz.xml" to IDEA/keymaps folder
    - Based on original IntelliJ keymap, similar to Windows
      - Allows more easy transition between Mac OS X and Windows
    - Column selection, presentation, terminal, copy plain text
  - Some global Mac OS X keyboard shortcuts clash
    - Mission Control ^← / ^→
    - Input Sources ^Space
    - Services → Text → Convert Text to Traditional Chinese

Setting up IntelliJ IDEA Community Edition

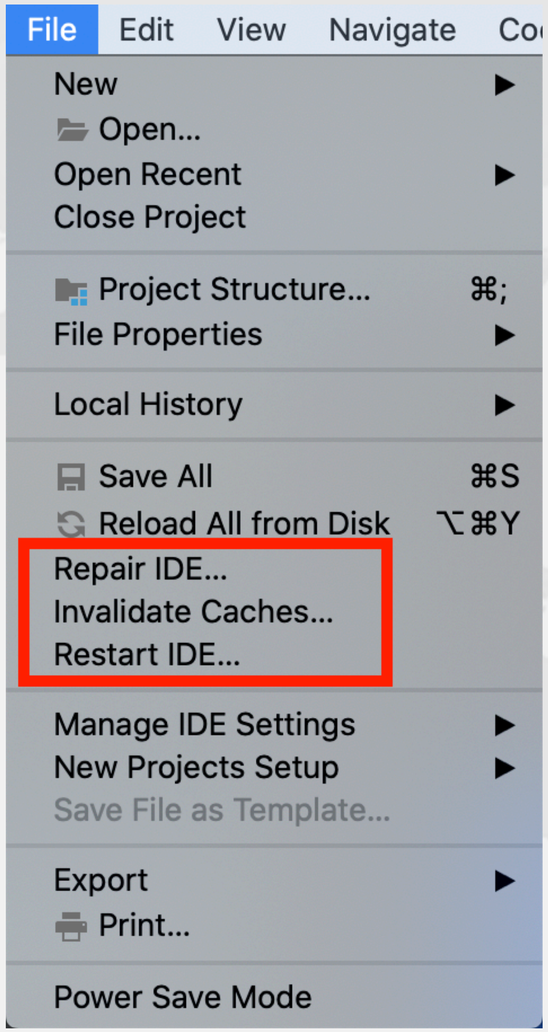
- **Editor →**
  - **General → Appearance: Show method separators**
  - **General → Code folding**
  - **General → Smart Keys: Select Use "CamelHumps" words**
  - **Color Scheme: Darcula Courses better for showing errors**
  - **Live Templates (more about this later)**

Livelock from Corrupt Local Files



Recovering from Broken Indexes

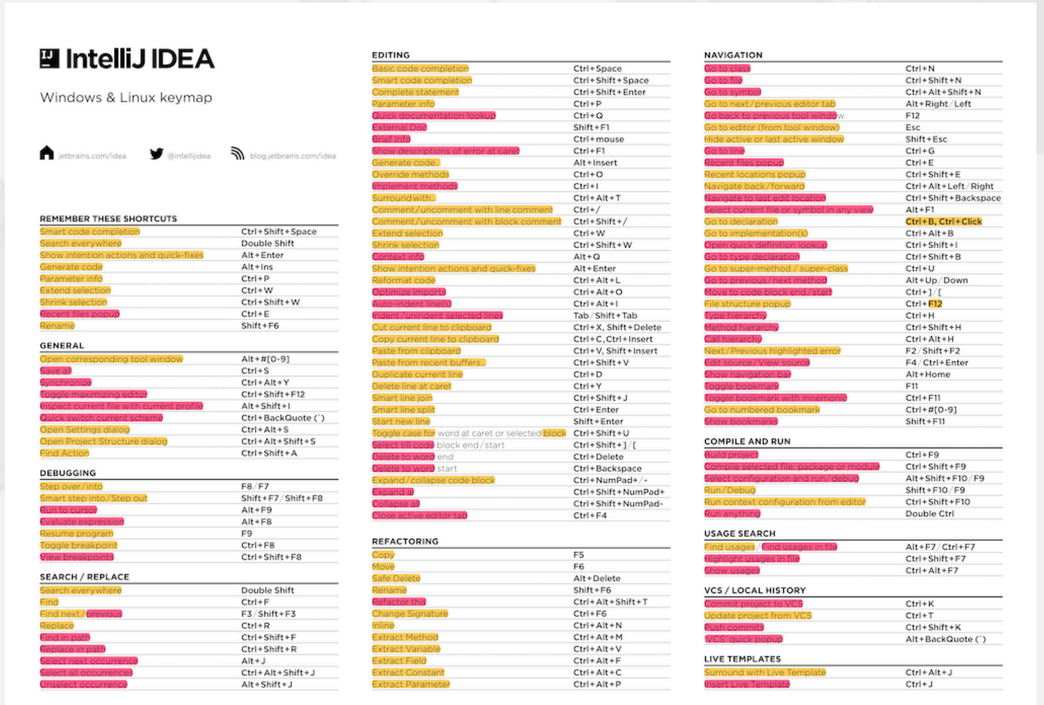
- IntelliJ gets into a bad state if indexes are corrupted
  - File → Repair IDE...
    - Try this first, stepping through until everything works
  - File → Invalidate Caches...
    - Usually takes a while to reindex
      - But solves most of the problems
      - Try to not lose local history



3. IntelliJ IDEA Philosophy

IntelliJ IDEA Philosophy

- **IntelliJ designed to be used mostly without mouse**
- **Hotkeys for almost everything**
  - **Print out resources/IntelliJIDEA_ReferenceCard.pdf**
  - **Memorize one new shortcut per day ≈ 6 months**



IntelliJ IDEA

Windows & Linux keymap

jetbrains.com/idea @intellijidea blog.jetbrains.com/idea

REMEMBER THESE SHORTCUTS

Smart code completion	Ctrl + Shift + Space
Search everywhere	Double Shift
Show intention actions and quick-fixes	Alt + Enter
Generate code	Alt + Ins
Parameter info	Ctrl + P
Extend selection	Ctrl + W
Shrink selection	Ctrl + Shift + W
Recent files popup	Ctrl + E
Rename	Shift + F6

GENERAL

Open corresponding tool window	Alt + #[0-9]
Save all	Ctrl + S
Synchronize	Ctrl + Alt + Y
Toggle maximizing editor	Ctrl + Shift + F12
Inspect current file with current profile	Alt + Shift + I
Select In... / Reveal / Scope	Ctrl + BackQuote (`)
Open Settings dialog	Ctrl + Alt + S
Open Project Structure dialog	Ctrl + Alt + Shift + S
Find Action	Ctrl + Shift + A

DEBUGGING

Step over / into	F8 / F7
Smart step into / Step out	Shift + F7 / Shift + F8
Run to cursor	Alt + F9
Evaluate expression	Alt + F8
Resume program	F9
Toggle breakpoint	Ctrl + F8
View breakpoints	Ctrl + Shift + F8

SEARCH / REPLACE

Search everywhere	Double Shift
Find	Ctrl + F
Find next / previous	F3 / Shift + F3
Replace	Ctrl + R
Find in path	Ctrl + Shift + F
Replace in path	Ctrl + Shift + R
Select next occurrence	Alt + J
Select all occurrences	Ctrl + Alt + Shift + J
Unselect occurrence	Alt + Shift + J

EDITING

Basic code completion	Ctrl + Space
Smart code completion	Ctrl + Shift + Space
Complete statement	Ctrl + Shift + Enter
Parameter info	Ctrl + P
Quick documentation lookup	Ctrl + Q
External doc	Shift + F1
Brief info	Ctrl + mouse
Show descriptions of error at caret	Ctrl + F1
Generate code...	Alt + Insert
Override methods	Ctrl + O
Implement methods	Ctrl + I
Surround with...	Ctrl + Alt + T
Comment / uncomment with line comment	Ctrl + /
Comment / uncomment with block comment	Ctrl + Shift + /
Extend selection	Ctrl + W
Shrink selection	Ctrl + Shift + W
Context info	Alt + Q
Show intention actions and quick-fixes	Alt + Enter
Reformat code	Ctrl + Alt + L
Optimize imports	Ctrl + Alt + O
Auto-indent lines	Ctrl + Alt + I
Indent / unindent selected lines	Tab / Shift + Tab
Cut current line to clipboard	Ctrl + X, Shift + Delete
Copy current line to clipboard	Ctrl + C, Ctrl + Insert
Paste from clipboard	Ctrl + V, Shift + Insert
Paste from recent buffers...	Ctrl + Shift + V
Duplicate current line	Ctrl + D
Delete line at caret	Ctrl + Y
Smart line join	Ctrl + Shift + J
Smart line split	Ctrl + Enter
Start new line	Shift + Enter
Toggle case for word at caret or selected block	Ctrl + Shift + U
Select till code block end / start	Ctrl + Shift +], [
Delete to word end	Ctrl + Delete
Delete to word start	Ctrl + Backspace
Expand / collapse code block	Ctrl + NumPad+ / -
Expand all	Ctrl + Shift + NumPad+
Collapse all	Ctrl + Shift + NumPad-
Close active editor tab	Ctrl + F4

REFACTORING

Copy...	F5
Move...	F6
Safe Delete...	Alt + Delete
Rename...	Shift + F6
Refactor this...	Ctrl + Alt + Shift + T
Change Signature...	Ctrl + F6
Inline	Ctrl + Alt + N
Extract Method...	Ctrl + Alt + M
Extract Variable...	Ctrl + Alt + V
Extract Field...	Ctrl + Alt + F
Extract Constant...	Ctrl + Alt + C
Extract Parameter...	Ctrl + Alt + P

NAVIGATION

Go to class...	Ctrl + N
Go to file...	Ctrl + Shift + N
Go to symbol...	Ctrl + Alt + Shift + N
Go to next / previous editor tab	Alt + Right / Left
Go back to previous tool window	F12
Go to editor (from tool window)	Esc
Hide active or last active window	Shift + Esc
Go to line...	Ctrl + G
Recent files popup	Ctrl + E
Recent locations popup	Ctrl + Shift + E
Navigate back / forward	Ctrl + Alt + Left / Right
Navigate to last edit location	Ctrl + Shift + Backspace
Select current file or symbol in any view	Alt + F1
Go to declaration	Ctrl + B, Ctrl + Click
Go to implementation(s)	Ctrl + Alt + B
Open quick definition lookup	Ctrl + Shift + I
Go to type declaration	Ctrl + Shift + B
Go to super-method / super-class	Ctrl + U
Go to previous / next method	Alt + Up / Down
Move to code block end / start	Ctrl +], [
File structure popup	Ctrl + F12
Type hierarchy	Ctrl + H
Method hierarchy	Ctrl + Shift + H
Call hierarchy	Ctrl + Alt + H
Next / Previous highlighted error	F2 / Shift + F2
Edit source / View source	F4 / Ctrl + Enter
Show navigation bar	Alt + Home
Toggle bookmark	F11
Toggle bookmark with mnemonic	Ctrl + F11
Go to numbered bookmark	Ctrl + #[0-9]
Show bookmarks	Shift + F11

COMPILE AND RUN

Build project	Ctrl + F9
Compile selected files, package or module	Ctrl + Shift + F9
Select configuration and run / debug	Alt + Shift + F10 / F9
Run / Debug	Shift + F10 / F9
Run context configuration from editor	Ctrl + Shift + F10
Run anything	Double Ctrl

USAGE SEARCH

Find usages / Find usages in file	Alt + F7 / Ctrl + F7
Highlight usages in file	Ctrl + Shift + F7
Show usages	Ctrl + Alt + F7

VCS / LOCAL HISTORY

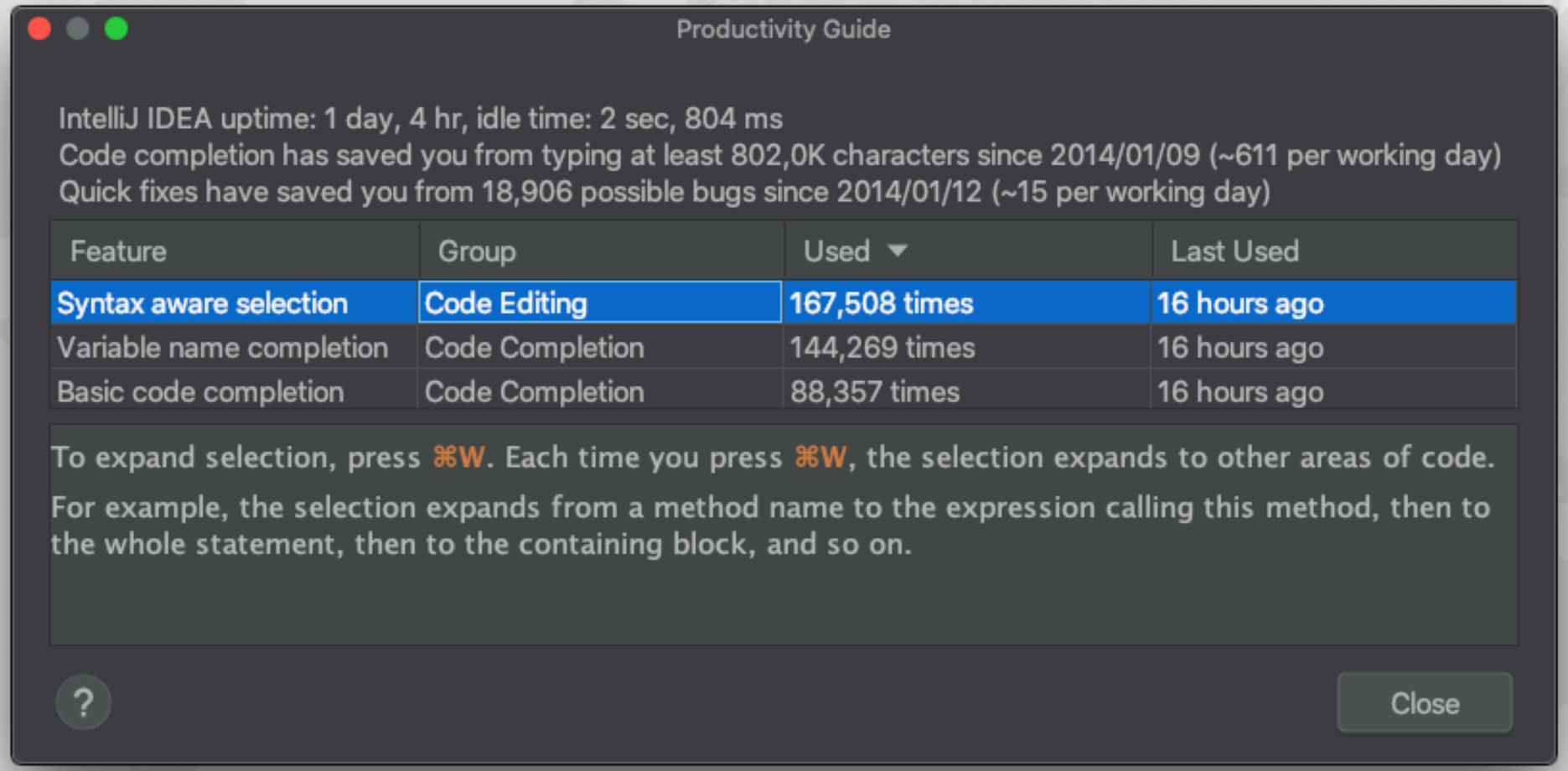
Commit project to VCS	Ctrl + K
Update project from VCS	Ctrl + T
Push commits	Ctrl + Shift + K
VCS quick popup	Alt + BackQuote (`)

LIVE TEMPLATES

Surround with Live Template	Ctrl + Alt + J
Insert Live Template	Ctrl + J

Help → My Productivity

- **Track progress in how productive you have become**



Useful Plugins

- **Presentation Assistant**
  - Shows what the keyboard shortcuts are that we are typing
- **Key Promoter X**
  - Gives hint of when we could have used a shortcut
- **Keymap Exporter**
  - Prints a PDF of whichever keymap you want to learn
- **Keystroke Counter**
  - Fun plugin counting how much typing we do
- **JOL and JMH**
  - Useful for measuring object size and to run benchmarks

Find Action

- "Find Action" discovers an action in IDEA
  - Windows/Linux: Ctrl + Shift + A
  - Mac OS X: ⌘⇧A
- e.g. "Select All"

Searching

- "Search Everywhere"
  - Windows/Linux: Double Shift
  - Mac OS X: ⇧⇧
- "Search Everywhere and Include non-project items"
  - Windows/Linux: Quadruple Shift
  - Mac OS X: ⇧⇧⇧⇧
- Kept on hitting this by mistake when pressing ↑

Tool Windows

- **Project Tool Window**
  - Windows/Linux: Alt + 1
  - Mac OS X: ⌘1
- **Terminal Tool Window**
  - Windows/Linux: Alt + F12
  - Mac OS X (Official): ⌥ F12
  - Mac OS X (Heinz): ⌘2
- **Run Tool Window**
  - Windows/Linux: Alt + 4
  - Mac OS X: ⌘4

Tool Windows

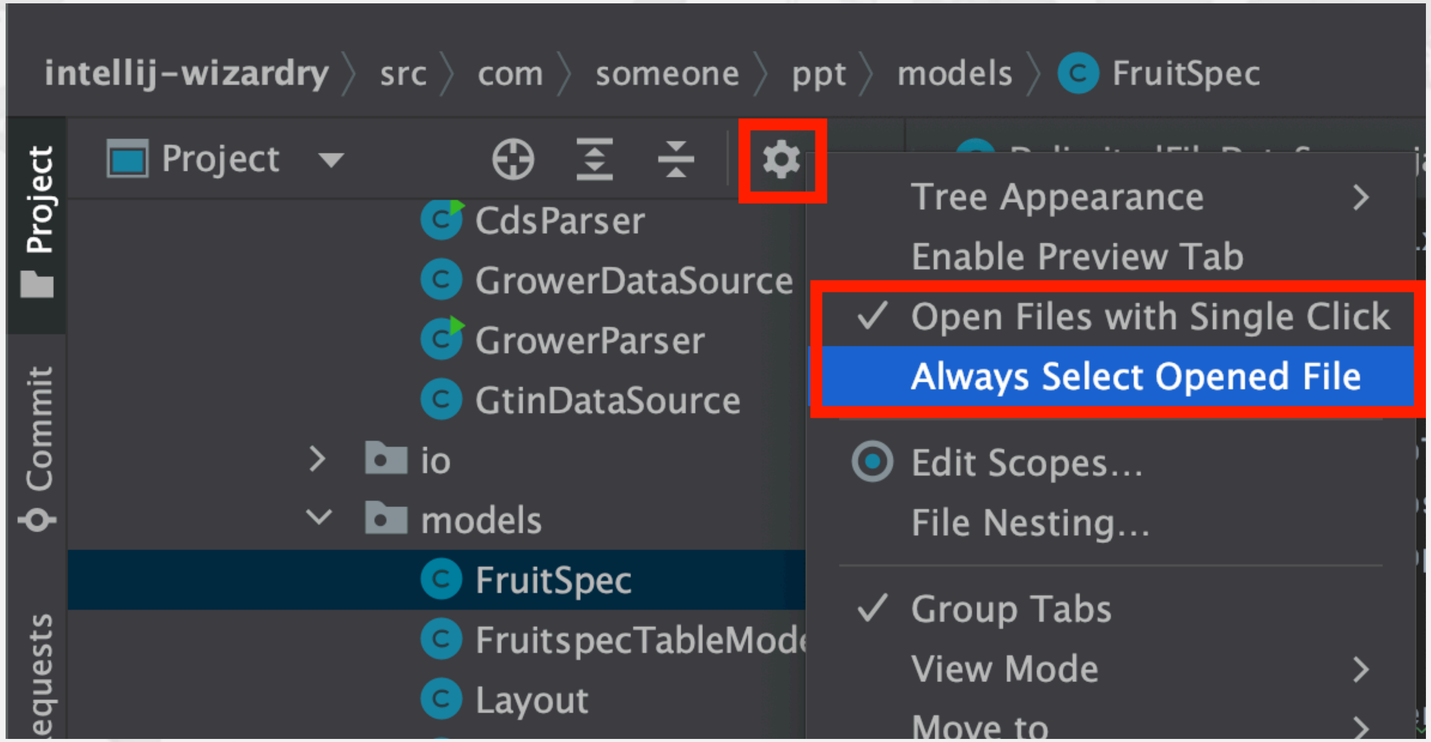
- **Debug Tool Window**
  - **Windows/Linux: Alt + 5**
  - **Mac OS X: ⌘5**
- **Structure Tool Window**
  - **Windows/Linux: Alt + 7 or Ctrl + F12 for popup**
  - **Mac OS X: ⌘7 or ⌘F12 for popup**

Switching Between Editor & Tool Windows

- "Go to editor" (from tool window)
  - Windows/Linux: Esc
  - Mac OS X: ⎋
- "Hide active or last active window"
  - Only the last one, not all of them
  - Windows/Linux: Shift + Esc
  - Mac OS X: ⇧⎋
- "Toggle maximizing editor"
  - Windows/Linux: Ctrl + Shift + F12
  - Mac OS X: ⌘⇧F12

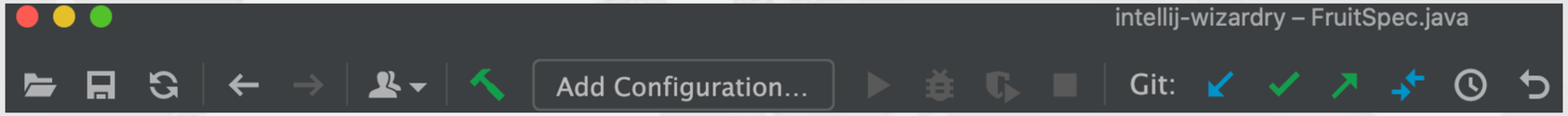
Autoscroll to/from source

- **Should be on by default**
  - **Lots of times saw programmers editing the wrong file**
  - **– ✓ Open Files with Single Click**
  - **– ✓ Always Select Opened File**



View → Appearance → Toolbar

- **Shows a useful toolbar at the top**



4. Essential Shortcuts

Superkey for fixing almost anything

- **"Show intention actions and quick-fixes"**
  - **Windows/Linux: Alt + Enter**
  - **Mac OS X: ⌥↩**
- **Quick demo fixing com.someone.ppt.cds.CdsGenerator**

Generate new code

- "Generate Code"
  - Windows/Linux: Alt + Ins
  - Mac OS X (Official): ⌘N
  - Mac OS X (Heinz): ^↩ or ^N
- Quick demo creating playground.Demo
  - Add final field, show difference between ^↩ and ⌥↩

Live Templates

- We can generate code quickly with live templates
  - psvm or main: Main method
  - sout, soutv, soutm, soutp, souf, serr, soutc: Output
  - iter, fori, itco, itar, ritar: Iteration
  - ifn, inn: if == null / if != null
  - prsf, psf, psfi, psfs: private/public final fields

Navigation

- "Go to declaration"
  - Windows/Linux: Ctrl + B or Ctrl + Click
  - Mac OS X: ⌘B or ⌘Click
- "Go to super-method / super-class" 
  - Windows/Linux: Ctrl + U
  - Mac OS X: ⌘U
- "Go to implementation(s)" 
  - Windows/Linux: Ctrl + Alt + B
  - Mac OS X: ⌘⌥B

Should you throw away your mouse?

- **Everything can be done with keyboard in IDEA**
  - **It is useful to learn to touch type**
  - **I usually have left hand on keyboard and right on mouse**
    - **Easy enough to find the correct keys - index fingers on F & J**
- **For navigating, I find the mouse faster**
  - **Hold down Ctrl or ⌘ and everything becomes a hyperlink**
  - **Scrolling with mouse or touchpad smoother**

Find Usages

- **Find Usages**
  - **Windows/Linux: Alt + F7**
  - **Mac OS X: ⌥F7**
- **We can specify the scope of where to search** 
  - **All places**
  - **Project files**
  - **Project and libraries**
  - **Recently viewed files**
  - **etc.**
- **"Open results in new tab" also quite useful**

Back and Forth Navigation

- "Navigate back / forward"
  - Windows/Linux: Ctrl + Alt + Left / Right
  - Mac OS X: ⌘⌥← / ⌘⌥→
- "Recent files/locations popup"
  - Windows/Linux: Ctrl + E / Ctrl + Shift + E
  - Mac OS X: ⌘E / ⌘⇧E
- "Go to next / previous editor tab"
  - Windows/Linux: Alt + Right / Left
  - Mac OS X: ^← / ^→
    - Clashes with Mac OS X Default Keymap for Mission Control

Bookmarks

- Quick navigation between locations in project
- Set with
  - Windows/Linux: Ctrl + Shift + #[0-9]
  - Mac OS X: ^⇧0 ... ^⇧9
- Navigate with
  - Windows/Linux: Ctrl + #[0-9]
  - Mac OS X: ^0 ... ^9

Search and Replace

- "Find"
  - Allows regular expressions with .*
  - Windows/Linux: Ctrl + F
  - Mac OS X: ⌘F
- "Find next"
  - Windows/Linux/Mac OS X (Heinz): F3
  - Mac OS X (Official): ⌘G
- "Replace"
  - Windows/Linux: Ctrl + R
  - Mac OS X: ⌘R

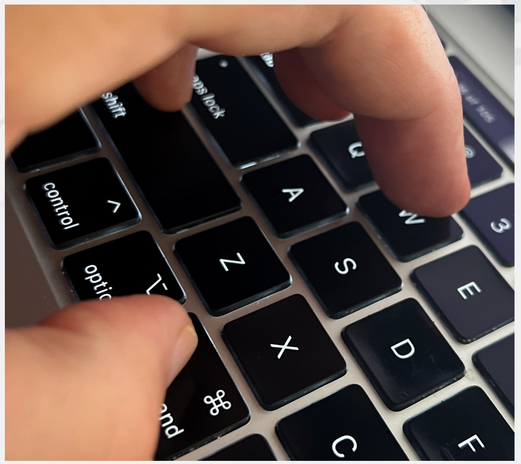
Jump out of line onto new line

- **"Start New Line"**
  - **Windows/Linux: Shift + Enter**
  - **Mac OS X: ⇧↩**
- **Even in the middle of some code, jump onto new line**

Syntax Aware Selection

Feature	Group	Used
Syntax aware selection	Code Editing	167,508 times
Variable name completion	Code Completion	144,269 times
Basic code completion	Code Completion	88,357 times

- **"Extend Selection"**
  - **Windows/Linux: Ctrl + W**
  - **Mac OS X (Official): ⌥↑**
  - **Mac OS X (Heinz): ⌘W**
    - **Closes windows in other Mac OS X programs**
    - **But my left thumb and middle finger and pinkie do this nicely**
    - **By FAR my most used shortcut, 167k times since 2014**
- **"Shrink Selection"**
  - **Windows/Linux: Ctrl + Shift + W**
  - **Mac OS X (Official): ⌥↓**
  - **Mac OS X (Heinz): ⌘⇧W**



Move code up / down

- "Move Code Up"
  - Windows/Linux: Ctrl + Shift + Up
  - Mac OS X: ⌘⇧↑
- "Move Code Down"
  - Windows/Linux: Ctrl + Shift + Down
  - Mac OS X: ⌘⇧↓
- Note: If nothing is selected, then we consider the current line to be selected

5. Turbo-boosted Productivity

Surround with ...

- "Surround with ..."
  - Windows/Linux: Ctrl + Alt + T
  - Mac OS X: ⌘⌥T
- Context aware, for example with Java

1. if
2. if / else
3. while
4. do / while
5. for
6. try / catch
7. try / finally
8. try / catch / finally
9. synchronized
0. Runnable

Surround with Live Template

- **"Surround with Live Template"**
  - **Windows/Linux: Ctrl + Alt + J**
  - **Mac OS X: ⌘⌥J**
- **Again context aware, for example with Java**

C. Surround with Callable
RL. Surround with ReadWriteLock.readLock
WL. Surround with ReadWriteLock.writeLock
I. Iterate Iterable or array

Define your own Live Templates

- **Preferences -> Editor -> Live Templates**
- **e.g. Wrap code in System.nanoTime()**
  - **Fantastic for demos, use JMH for serious benchmarks**

```
Abbreviation: nanoTime
Description: System.nanoTime()
Template text:

long $TIME$ = System.nanoTime();
try {
  $SELECTION$
} finally {
  $TIME$ = System.nanoTime() - $TIME$;
  System.out.printf("$TIME$ = %dms%n",
    ($TIME$/1_000_000));
}
```

Error Based Coding

- **Make deliberate mistakes, followed by**
  - **Next highlighted error (F2)**
    - **Previous highlighted error with Shift + F2 or ⇧F2**
  - **Superkey (Alt + Enter or ⌥↩)**
  - **Tab**
- **Quick demo**

Copy & Paste

- **An April Fools joke that became real**
  - **Sorry, you can't have one - they are sold out!**

Line Based Editing

- By default, actions apply to our current line
  - Windows/Linux:
    - Ctrl + D - Duplicate current line
    - Ctrl + C, Ctrl + Insert - Copy current line to clipboard
    - Ctrl + X, Shift + Delete - Cut current line to clipboard
    - Ctrl + V, Shift + Insert - Paste from clipboard
    - Ctrl + Shift + V - Paste from recent buffers...
    - Ctrl + Y - Delete line
  - Mac OS X:
    - ⌘D - Duplicate
    - ⌘V - Paste
    - ⌘⌫ - Delete line
    - ⌘C - Copy
    - ⌘X - Cut
    - ⌘⇧V - Paste from recent buffers

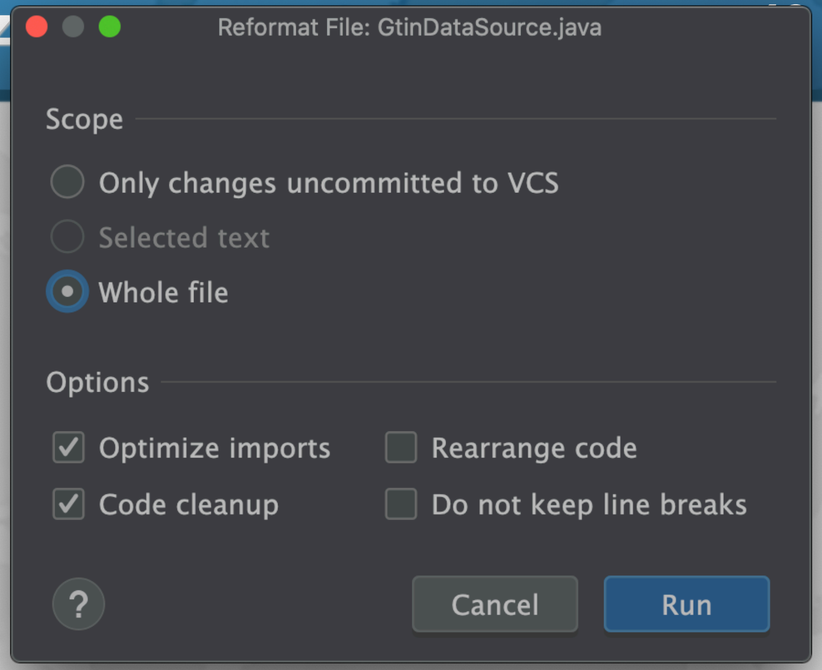
Column Select Editing

- Let's make all fields in CdsGenerator final
  - One at a time is tedious
- With the mouse
  - Windows/Linux: Alt + Drag Mouse
  - Mac OS X: ⌥+drag mouse
- With keyboard toggle column selection mode
  - Windows/Linux: Alt + Shift + Insert
  - Mac OS X (Official): ⌘⇧8
  - Mac OS X (Heinz): ⌘^⇧C
- Make sure to turn column selection mode off

Toggle Case

- Toggles upper/lower case
  - hello → HELLO → hello
  - Hello → hello
  - helloWorld → helloworld → HELLOWORLD
- Shortcut
  - Windows/Linux: Ctrl + Shift + U
  - Mac OS X: ⌘⇧U

Reformatting Code



- **"Reformat Code"**
  - **Windows/Linux: Ctrl + Alt + L**
  - **Mac OS X: ⌘⌥L**
- **"Reformat File"**
  - Can be configured for "Optimize Imports" and "Code Cleanup"
  - These settings then are also used for "Reformat Code"
  - **Windows/Linux: Ctrl + Alt + Shift + L**
  - **Mac OS X: ⌘⌥⇧L**

Exercise

- **Inside the com.someone.ppt.models.FruitSpec, create an enum that holds all the fields**

```
public enum Field {
    BARCODE("BarCode"),
    RUNNUMBER("RunNumber"),
    PUC("PUC"),
    /* etc. ... */
    GTIN("GTIN");
    private final String name;

    Field(String name) {
        this.name = name;
    }
}
```

6. Code Completion

Joining and Splitting Lines

- **"Smart line join"**
  - **Windows/Linux: Ctrl + Shift + J**
  - **Mac OS X: ^⇧J**
  - **e.g.**

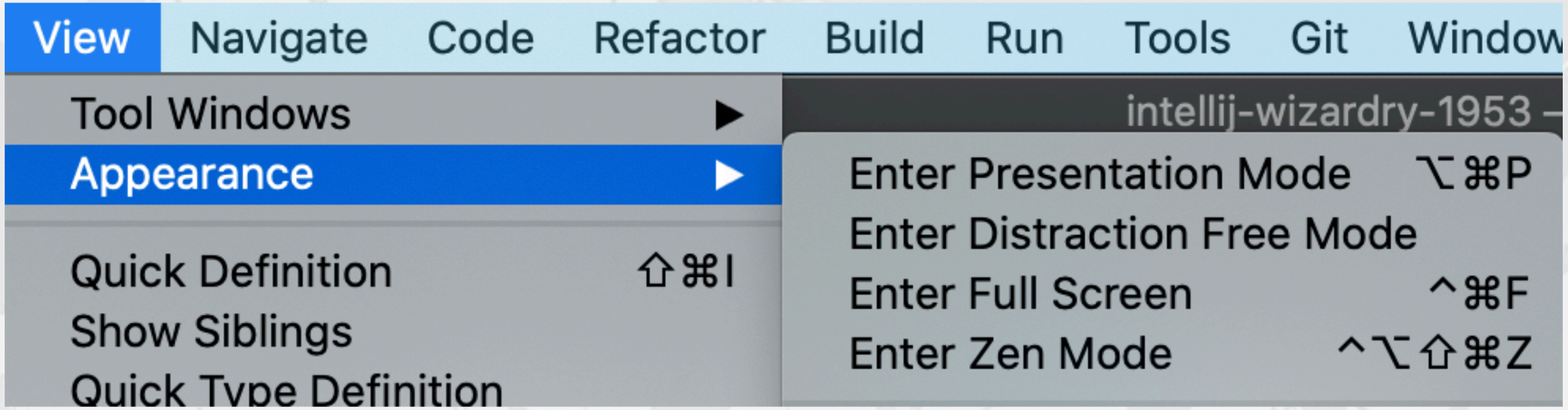
```
String[][] data = null;
data = source.getData();
```

Smart line join

```
String[][] data = source.getData();
```

- **"Smart line split"**
  - **Not sure what is "smart" about it**
  - **Windows/Linux: Ctrl + Alt + Shift + L**
  - **Mac OS X: ⌘⌥⇧L**

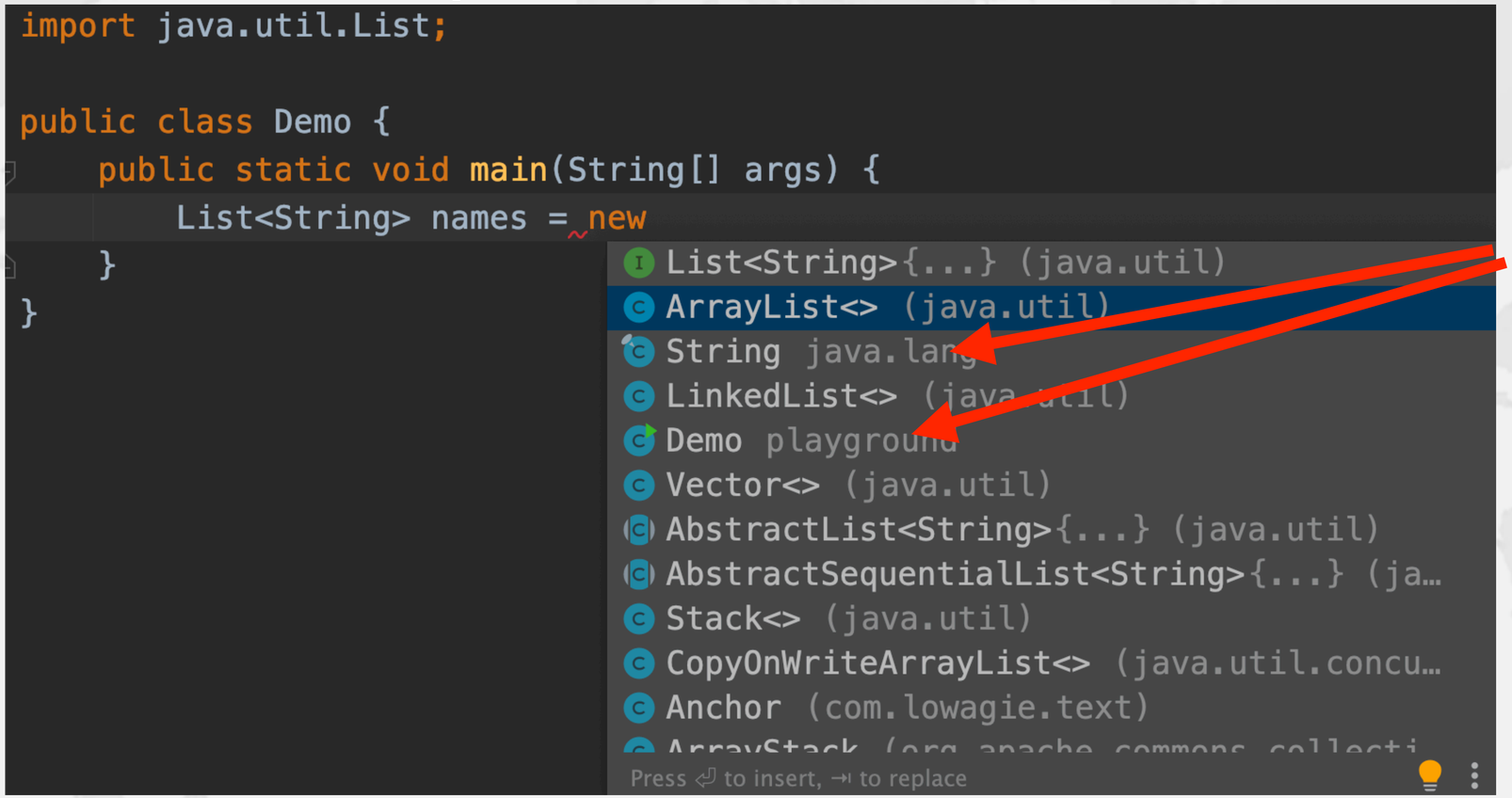
Different Appearances



- **Mac OS X (Heinz): ⌥⌘P for presentation mode**
  - **Incredibly useful when speaking**
  - **Font size is in Settings → Appearance & Behavior → Appearance → Presentation Mode**

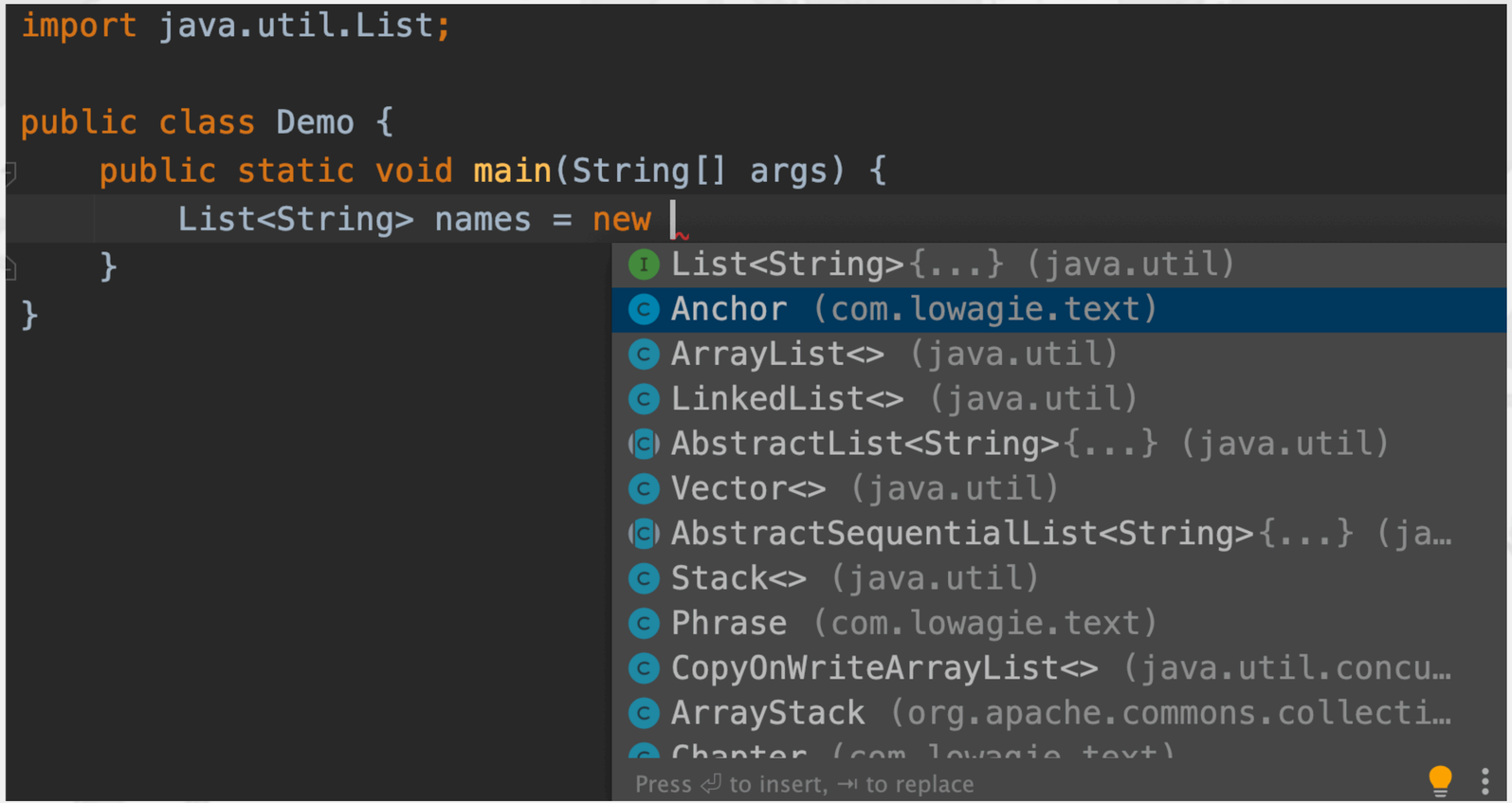
Basic Code Completion

- **Code completion traditionally uses Ctrl+Space**
  - **Windows/Linux: Ctrl + Space**
  - **Mac OS X: ^Space**



Smart Code Completion

- **This gives much better result - I always use this**
  - **Windows/Linux: Ctrl + Shift + Space**
  - **Mac OS X: ^⇧Space**



Complete Statement

- **Completes the current statement**
  - **Windows/Linux: Ctrl + Shift + Enter**
  - **Mac OS X: ⌘⇧↩**

```
while|
```

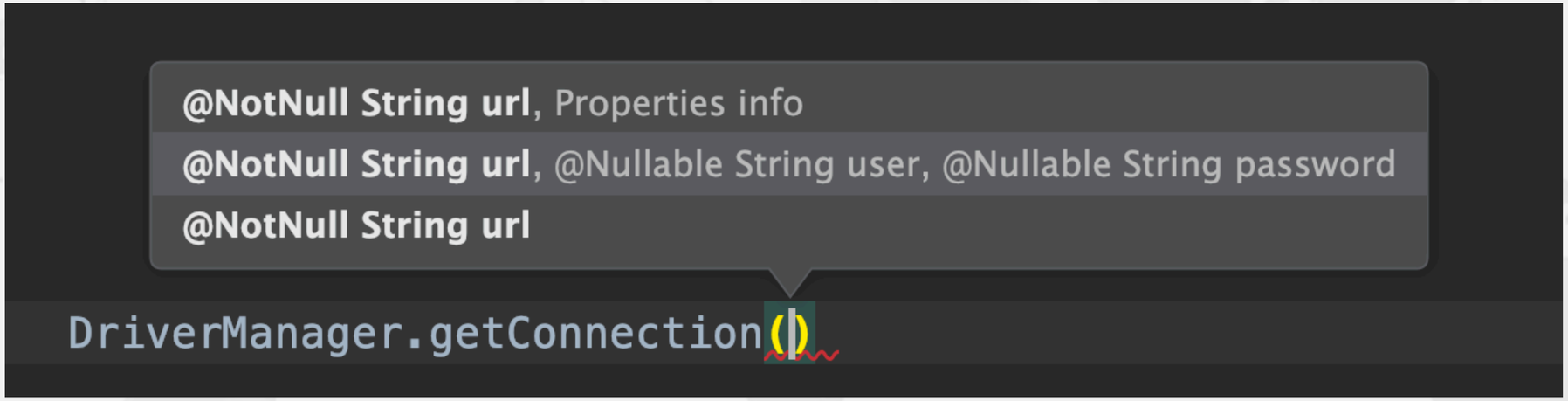
```
while (|) {
}
```

```
list.add(42|)
```

```
list.add(42);|
```

Parameter Info

- Should appear automatically when you type "("
- Shows the parameters for a method
  - Windows/Linux: Ctrl + P
  - Mac OS X: ⌘P

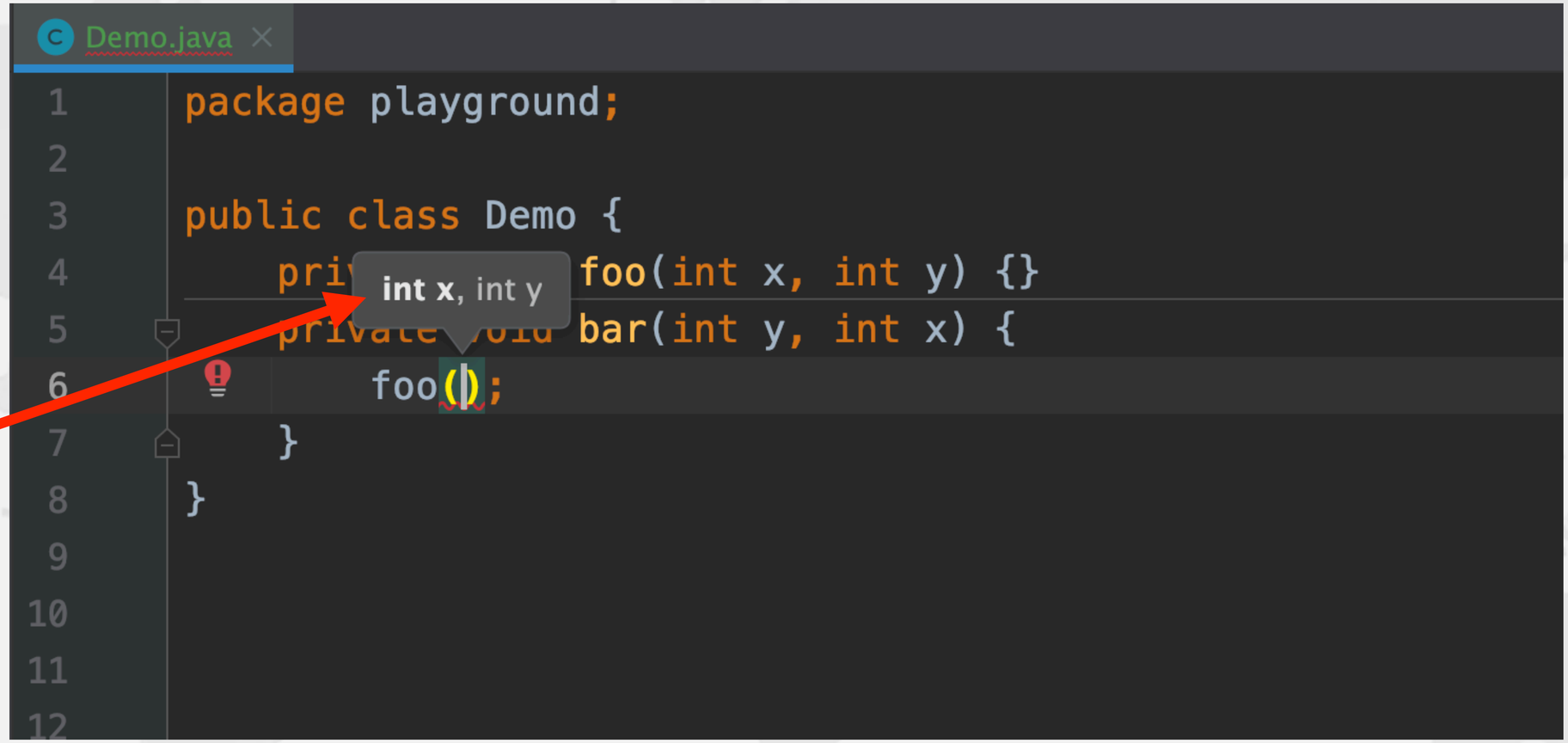


Beware of Parameter Reordering

Demo.java

```
1  package playground;
2
3  public class Demo {
4      private void foo(int x, int y) {}
5      private void bar(int y, int x) {
6          foo();
7      }
8  }
9
10
11
12
```

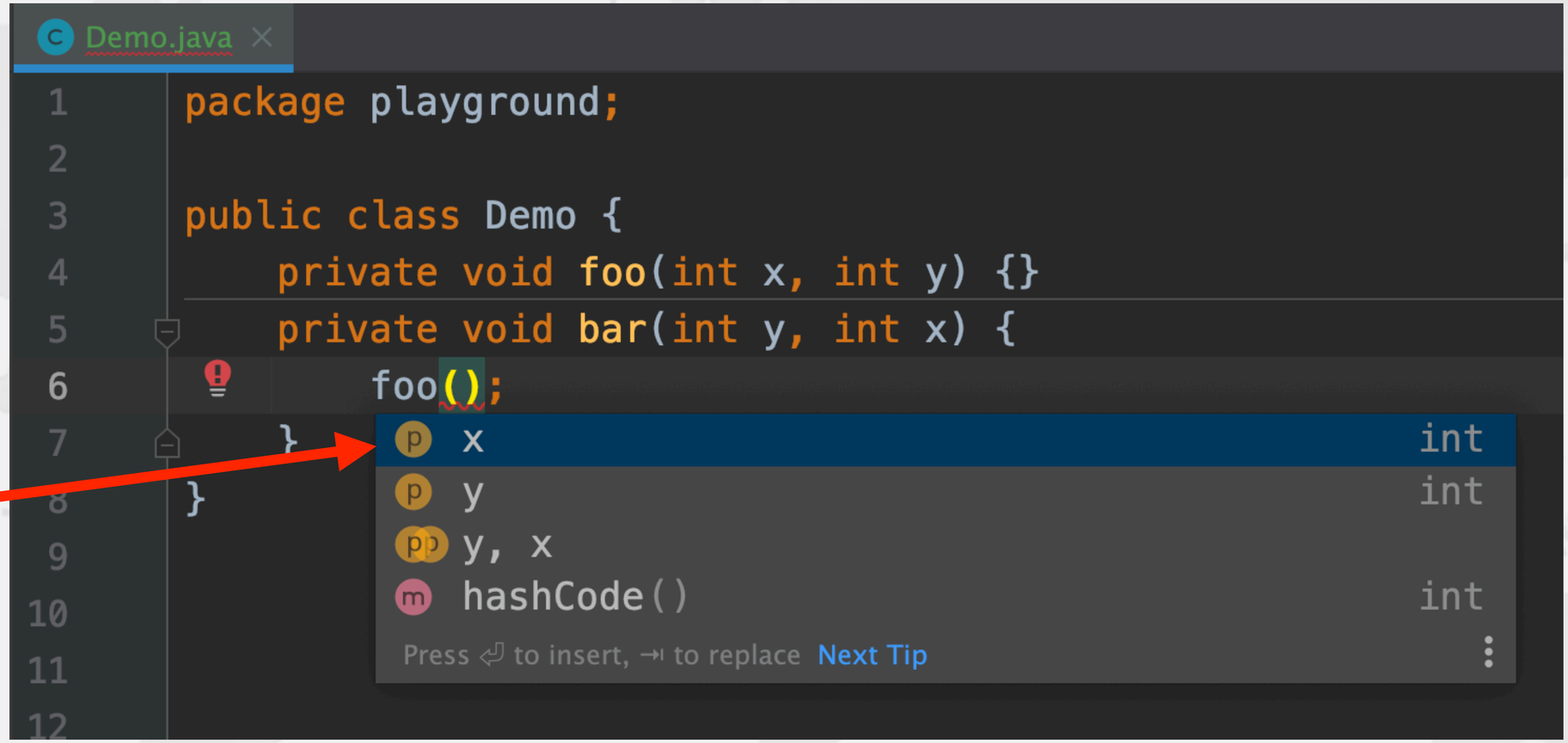
Beware of Parameter Reordering



⌘P

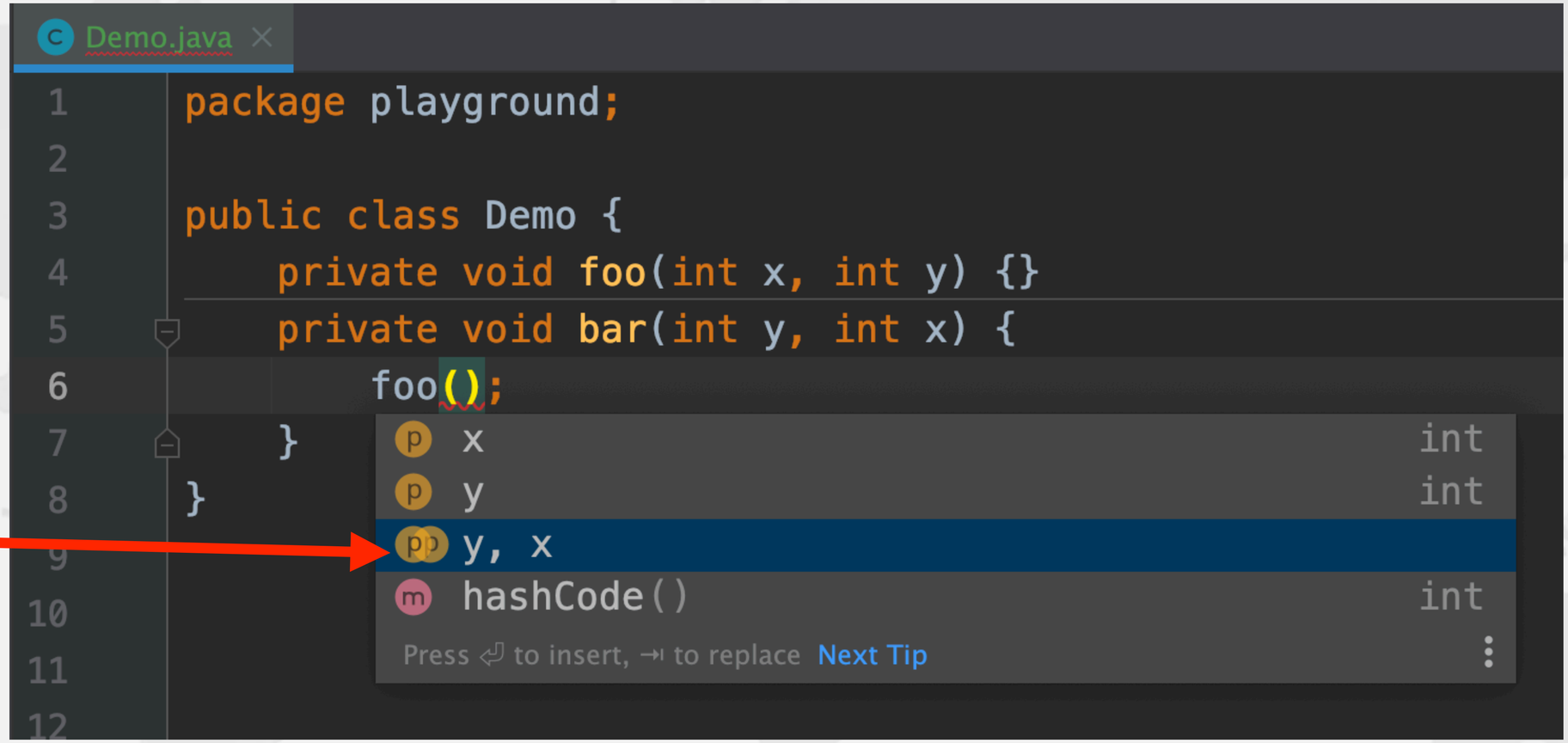
Beware of Parameter Reordering

^⇧Space



Beware of Parameter Reordering

Hmmm



Beware of Parameter Reordering

Ooops

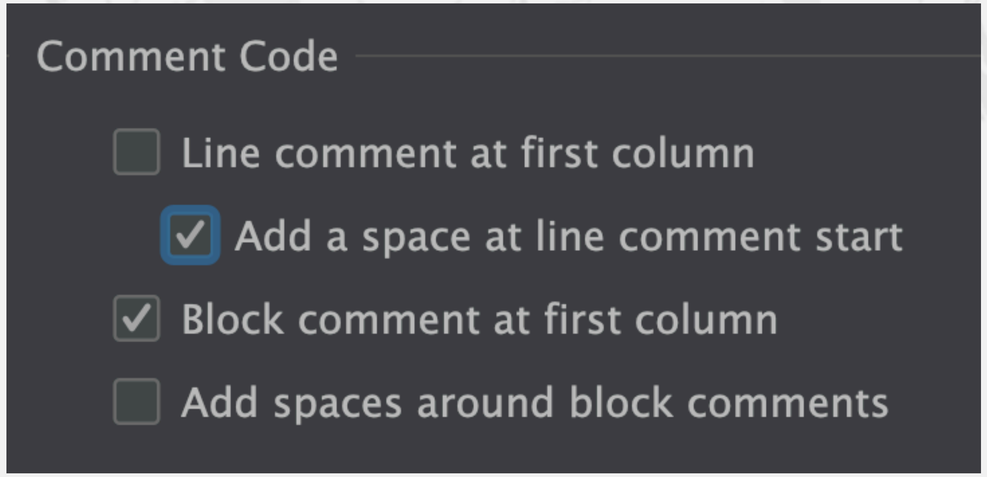
Demo.java

```
1  package playground;
2
3  public class Demo {
4      private void foo(int x, int y) {}
5      private void bar(int y, int x) {
6          foo(y, x);
7      }
8  }
9
10
11
12
```

- Bug reported to JetBrains on 2022-01-04

Commenting Out Code

- "Comment / uncomment with line comment"
  - Windows/Linux: Ctrl + /
  - Mac OS X: ⌘ /
- By default, puts the // at the beginning of line
  - Change in Settings → Editor → Code Style → Java → Code Generation →



Block Comments

- "Comment / uncomment with block comment"
  - Windows/Linux: Ctrl + Shift + /
  - Mac OS X: ⌘⌥ /
- Nice trick for easily uncommenting blocks of code

```
/*
    while ((line = fis.readLine()) != null) {
        if (!line.equals("")) {
            lines.add(line);
        }
    }
//*/
```

Block Comments

- "Comment / uncomment with block comment"
  - Windows/Linux: Ctrl + Shift + /
  - Mac OS X: ⌘⌥ /
- Nice trick for easily uncommenting blocks of code

Simply add a single / and the code is uncommented

```
//*
    while ((line = fis.readLine()) != null) {
        if (!line.equals("")) {
            lines.add(line);
        }
    }
//*/
```

Block Comments

- "Comment / uncomment with block comment"
  - Windows/Linux: Ctrl + Shift + /
  - Mac OS X: ⌘⌥ /
- Nice trick for easily uncommenting blocks of code

Simply add a single / and the code is uncommented

```
//*
    while ((line = fis.readLine()) != null) {
        if (!line.equals("")) {
            lines.add(line);
        }
    }
//*/
```

No change needed here

Code Folding

- Hide non-essential information, such as
  - File headers, imports, Javadocs
  - Method bodies
    - One-line methods, getters/setters, anonymous types
  - Sections of a method
  - etc.
  - Settings → Editor → General → Code Folding
- "Expand / collapse code block"
  - Windows/Linux: Ctrl + NumPad+ / NumPad-
  - Mac OS X: ⌘+ / ⌘-

Override and Implement Methods

- "Override methods"
  - Windows/Linux: Ctrl + O
  - Mac OS X: ⌘O
- "Implement methods"
  - Windows/Linux: Ctrl + I
  - Mac OS X: ⌘I
- We can also do that with "Generate Code"
  - Windows/Linux: Alt + Ins
  - Mac OS X (Official): ⌘N
  - Mac OS X (Heinz): ^↩ or ^N

CamelCase in Code Completion

- **Instead of C**opy**O**n**W**rite**A**rray**L**ist**, use COWAL**
  - **IntelliJ starts searching as we type**
  - **Simply use the capital letters in the CamelCase class**

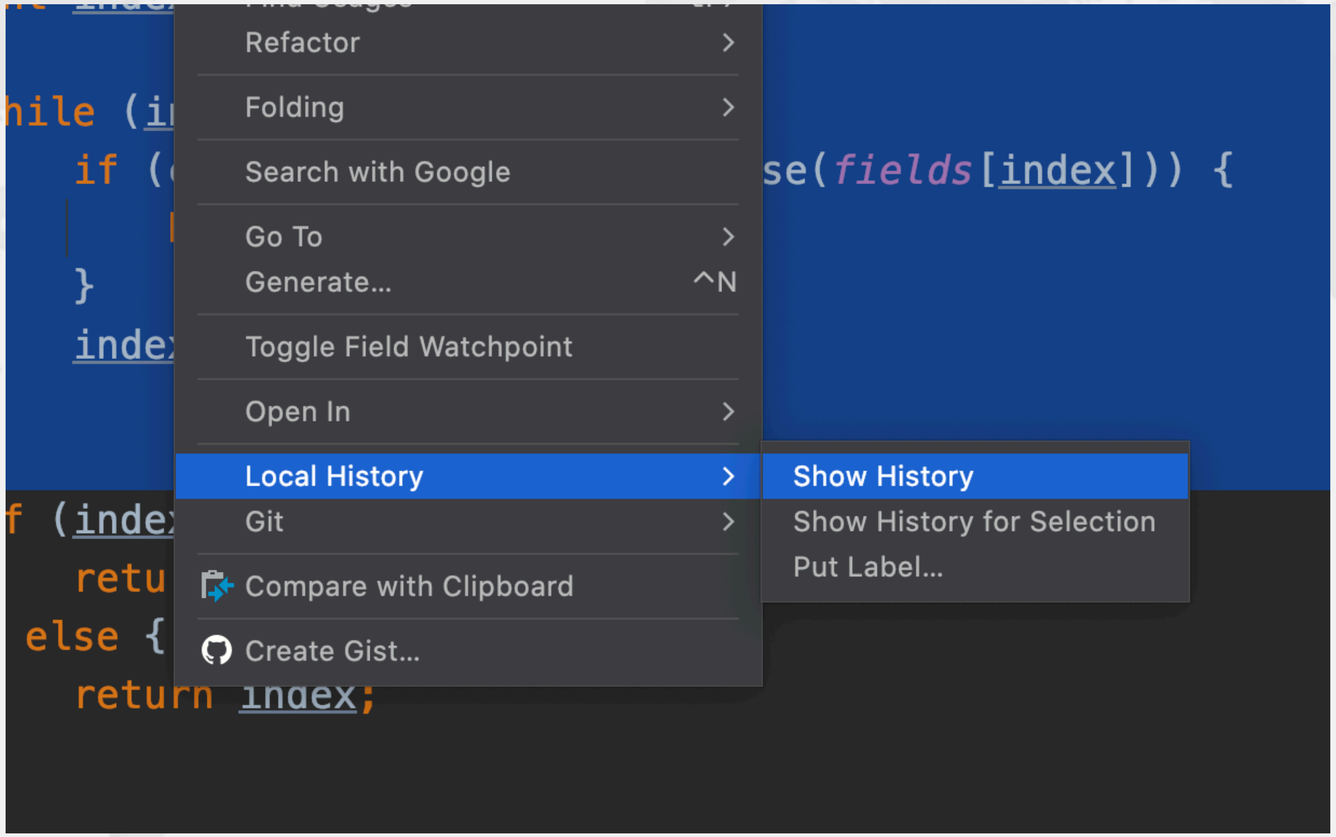
Imports

- Imports are managed mostly automatically
  - Can be configured in settings for single imports or multiple
    - Depends on your company standards
    - java.base has
      - 13679 explicit imports
      - 876 wildcard imports, mostly java.util, java.security, java.io (6%)
      - 585 static explicit imports
      - 180 static wildcard imports (24%)
  - I usually fold away the imports and never look at them

7. Source Management

Source Management

- **IntelliJ maintains local history of changes**
  - **Can be a life saver**
  - **Avoid deleting it when repairing the IDE**



Integration with Version Control System

- To commit a change
  - Windows/Linux: Ctrl + K
  - Mac OS X: ⌘K
    - Once you have typed the commit message, ^↩
- To push commits
  - Windows/Linux: Ctrl + Shift + K
  - Mac OS X: ⌘⇧K
- Update project from VCS
  - Windows/Linux: Ctrl + T
  - Mac OS X: ⌘T

Git Tool Window

- **Git Tool Window**
  - **Windows/Linux: Alt + 9**
  - **Mac OS X: ⌘9**
- **Quick Demo**

8. Refactoring

Refactoring

- Pioneered by Martin Fowler
  - Based on research by William Opdyke
- What it is
  - Improving the design of existing code
    - Without adding new functionality
    - Introduce good design patterns
- Unit testing
  - Bad refactorings often introduce bugs

Copy Class

- "Copy"
  - Windows/Linux: F5
  - Mac OS X: F5
  - Turn on function keys for IntelliJ IDEA on MacBook Pro
    - Settings → Keymap → Show F1, F2 etc. keys on the Touch Bar
- Used more often than I thought
  - On average, once a day
  - But that includes a lot of teaching, rather than just new code
- Ok, let's move on ....

Move Class / Field / Method

- "Move"
  - Windows/Linux: F6
  - Mac OS X: F6
- Exercise
  - Move PrtGenerator.calculateFakeGTIN() and calculateFakeGTIN2() to a new class com.someone.ppt.cds.GTINGenerator

Renaming Things

- "Rename ..."
  - Windows/Linux: Shift + F6
  - Mac OS X: ⇧F6
- Applies to almost anything:
  - Classes, methods, parameters, variables, fields
- Another trick:
  - Rename it and then propagate all changes with our superkey
    - Windows/Linux: Alt + Enter
    - Mac OS X: ⌥↩
    - Do it before leaving that particular line
  - Exercise: Rename Tickable to Tickeable

Change Signature

- "Change Signature"
  - Windows/Linux: Ctrl + F6
  - Mac OS X: ⌘F6
- Applies mainly to methods for changing:
  - Return type
  - Parameter types and names
  - Order of parameters
- Can in some cases use the same trick as previous
  - Change method and then propagate with Alt + Enter or ⌥↩

Exercise

- **Remove parameters from the buildQuery() method in TableModelFactory whose values are always null**
  - **Also change "StringBuffer result" to use StringBuilder**

Extract Variable

- Local variables tend to increase method length
- "Extract Variable"
  - Windows/Linux: Ctrl + Alt + V
  - Mac OS X: ⌘⌥V
- Break up long chain of method calls into variables
  - e.g. Java 8 streams
- Can also help to reduce duplicate code
- Exercise
  - generatePcdTemplate() in PcdGenerator
    - "" + eventID

Extract Method

- Select a block of code and "Extract Method"
  - Windows/Linux: Ctrl + Alt + M
  - Mac OS X: ⌘⌥M
- Some restrictions
  - Cannot have more than one return value
  - Block must represent a set of statements or expressions
- Additional benefits
  - Extracting a method can discover other, similar, code

Exercise

- **Extract snippets from generatePcdRemarks() method in PcdGenerator into separate method**

```
String remark = resultSet.getString("Remark1");
if (!"".equals(remark)) {
    remarks.put(remark, remark);
}

remark = resultSet.getString("Remark2");
if (!"".equals(remark)) {
    remarks.put(remark, remark);
}

remark = resultSet.getString("Remark3");
if (!"".equals(remark)) {
    remarks.put(remark, remark);
}
```

Inline Code

- Applies to methods, fields, local variables
- "Inline"
  - Windows/Linux: Ctrl + Alt + N
  - Mac OS X: ⌘⌥N
- Conveniently close to "Extract Method" shortcut

Extract Field

- "Extract Field" applies to expressions and variables
  - Windows/Linux: Ctrl + Alt + F
  - Mac OS X: ⌘⌥F
- Not that useful
  - Extracted fields would typically not be final, set in methods
  - May introduce race conditions

Extract Constant

- **"Extract Constant" uses Java naming convention**
  - **Windows/Linux: Ctrl + Alt + C**
  - **Mac OS X: ⌘⌥C**

Extract Parameter

- **"Extract Parameter"**
  - **Windows/Linux: Ctrl + Alt + P**
  - **Mac OS X: ⌘⌥P**

Safe Delete

- "Safe Delete" - searches whether the code is used
  - Can also search in comments and Strings
  - Windows/Linux: Alt + Delete
  - Mac OS X: ⌘⌦
    - ⌦ is fn⌫ on laptop keyboard
- We can also search for unused code with analyzer
  - More in next section

Postfix Completion

- Write a postfix after your expression and press tab
  - ! - Negates a boolean expression
  - nn - Adds a check verifying that an expression is not null
  - null - Adds a check verifying that an expression is null
  - try - Inserts a statement in a try-catch block
  - var - Introduces a local variable for an expression
- Demo

9. Analyzer

Analyzer

- IntelliJ shines with its code analyzer and refactoring
- Code → Inspect Code ...
  - Whole project
    - Or: Right-click in project → Analyze → Inspect Code ...
  - Inspection profile: Default IDE
- It checks for the most glaring code inconsistencies
  - Let's try it out together

Making Fields and Methods more **private**

- **Not on by default**
  - **Declaration redundancy → Declaration access can be weaker**
- **Good approach is to tighten up the access control**
  - **Encapsulation one of the best weapons against complexity**
- **Let's do this together**

Final / non-final Parameters and Variables

- Some like final parameters and local variables
  - Others do not like the extra clutter
- In the java.base module, we find
  - 83680 local variables or parameters that could be final
  - 4909 unnecessary final on local variables or parameters (6%)
- Since Java 8, we have "effectively final"
- Let's get rid of unnecessary final in the project
  - Java → Code style issues → Unnecessary 'final' on local variable or parameter

Making fields final

- Final fields have a big benefit
  - Make objects as immutable as possible
  - Reduces bugs, both in correctness and concurrency
- Let's make all fields final that we can
  - Java → Declaration redundancy → Declaration can have final modifier

Removing unused code

- Don't waste effort maintaining dead code
  - 28% to 45% of a system's functionality is never used
- Find and eliminate unused code
  - Java → Declaration redundancy → Unused declaration
- Relies on accurate entry points into the application
  - Careful: Might delete too much

Error Handling

- Incorrect catch blocks source of many bugs
- Several "Error handling" inspections
  - Catch block may ignore exception
    - Best to then name the exception parameter "ignored"
  - Caught exception is immediately rethrown
  - 'finally' block which can not complete normally
  - Many others, depend on company coding convention
    - e.g. "Unchecked exception class", "Checked exception class"
- Code maturity inspection
  - Call to 'printStackTrace()'

Migrating to newer Java versions

- Java syntax has come a long way since 1.0
- Java 5
  - Generics, Enums, Autoboxing, Enhanced 'for' loop
- Java 7
  - Diamond generics operator <>
  - Multi-catch in try-catch, try-with-resource
  - Switching on String
- Java 8
  - Default and static interface methods
  - Lambdas and streams

Java Language Changes

- **Java 11**
  - var **for local variable types**
- **Java 17**
  - **Switch Expressions (JEP 361)**
  - **Text blocks (JEP 378)**
  - **Records (JEP 395)**
  - **Pattern Matching for instanceof (JEP 394)**
  - **Strongly Encapsulate JDK Internals by Default (JEP 396)**
  - **Sealed classes (JEP 409)**

Java Language Migration Aids

- **Exercises**
  - **Use enhanced switch in**
    - **CdsGenerator#generateCfg()**
    - **PcdGenerator#generatePcdTemplate()**
  - **Use pattern variable in ElegantTable#initGui()**
  - **Use try-with-resource in LineSocket#receiveFile()**

Find Duplicate Code

- Code → Analyze Code → Locate Duplicates ...
  - Only available in IntelliJ IDEA Ultimate Edition
- Can reduce duplicate code
  - But resultant code might be harder to understand

10. Running and Debugging Code

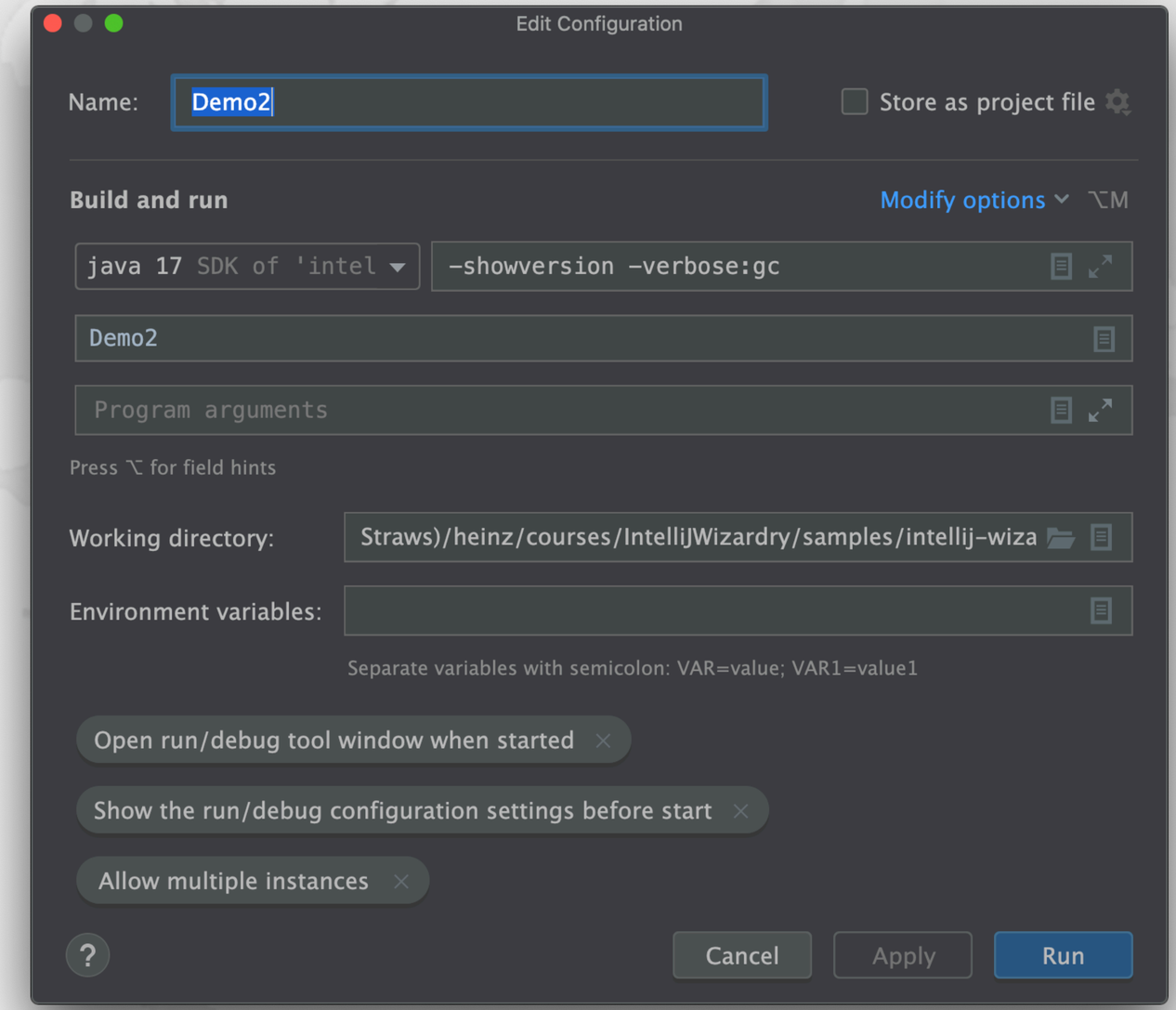
Running and Debugging Code

- To run / debug the current class
  - Windows/Linux: Ctrl + Shift + F10 / Ctrl + Shift + F9
  - Mac OS X (Official): ^⇧R / ^⇧D
  - Mac OS X (Heinz): ^⇧F10 / ^⇧F9
- To run / debug the previous class
  - Windows/Linux: Shift + F10 / Shift + F9
  - Mac OS X (Official): ^R / ^D
  - Mac OS X (Heinz): ⇧F10 / ⇧F9

Run Configuration

- **Templates used for default run configuration**
  - **Run → Edit configurations... → Edit configuration templates...**
  - **Select Application → Modify options → Show the run/debug settings before start**
    - **This always opens up the configuration window before run**
  - **To set a VM Option, use Modify options → Add VM options**

Demo


Edit Configuration
Name:
Demo2
Store as project file
Build and run
Modify options
⌥M
java 17 SDK of 'intel
-showversion -verbose:gc
Demo2
Program arguments
Press ⌥ for field hints
Working directory:
Straws)/heinz/courses/IntelliJWizardry/samples/intellij-wiza
Environment variables:
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Show the run/debug configuration settings before start
Allow multiple instances
?
Cancel
Apply
Run

Debugging a Class

- Toggle breakpoint
  - Windows/Linux: Ctrl + F8
  - Mac OS X: ⌘F8
  - Or just click in the margin
- Resume program
  - Windows/Linux/Mac OS X: F9
- Step over
  - Windows/Linux/Mac OS X: F8
- Step into
  - Windows/Linux/Mac OS X: F7

Debugging a Class

- Smart step into
  - Windows/Linux: Shift + F7
  - Mac OS X: ⇧F7
- Force step into - useful for debugging JDK classes
  - Windows/Linux: Alt + Shift + F7
  - Mac OS X: ⌥⇧F7
- Step out
  - Windows/Linux: Shift + F8
  - Mac OS X: ⇧F8

Demo

- Debug debugdemo.Factorial to make it work
  - Show how the debugger works
  - Show breakpoints, evaluate expressions, add watch

11. Conclusion

Conclusion to IntelliJ Wizardry

- **Many more keystrokes and features to learn**
- **One new one per day**
- **Happy coding!**

The Java Specialists' Newsletter

- Make sure to subscribe
  - www.javaspecialists.eu/archive/subscribe/
- Readers in 150+ countries
- Over 21 years of newsletters on advanced Java
  - All previous newsletters available on www.javaspecialists.eu
  - Courses, additional training, etc.