



# IntelliJ Super Productivity in 45 Minutes

Dr Heinz M. Kabutz

Last Updated 2025-02-13

# Why IntelliJ IDEA?

## ● My story

- Started with Borland JBuilder 1997
- Then used Eclipse for a year in 2002
- 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!**
- ◎ **Interrupt me at any time**
  - Questions that are off-topic might be delayed until later
- ◎ **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**

# Disclaimer

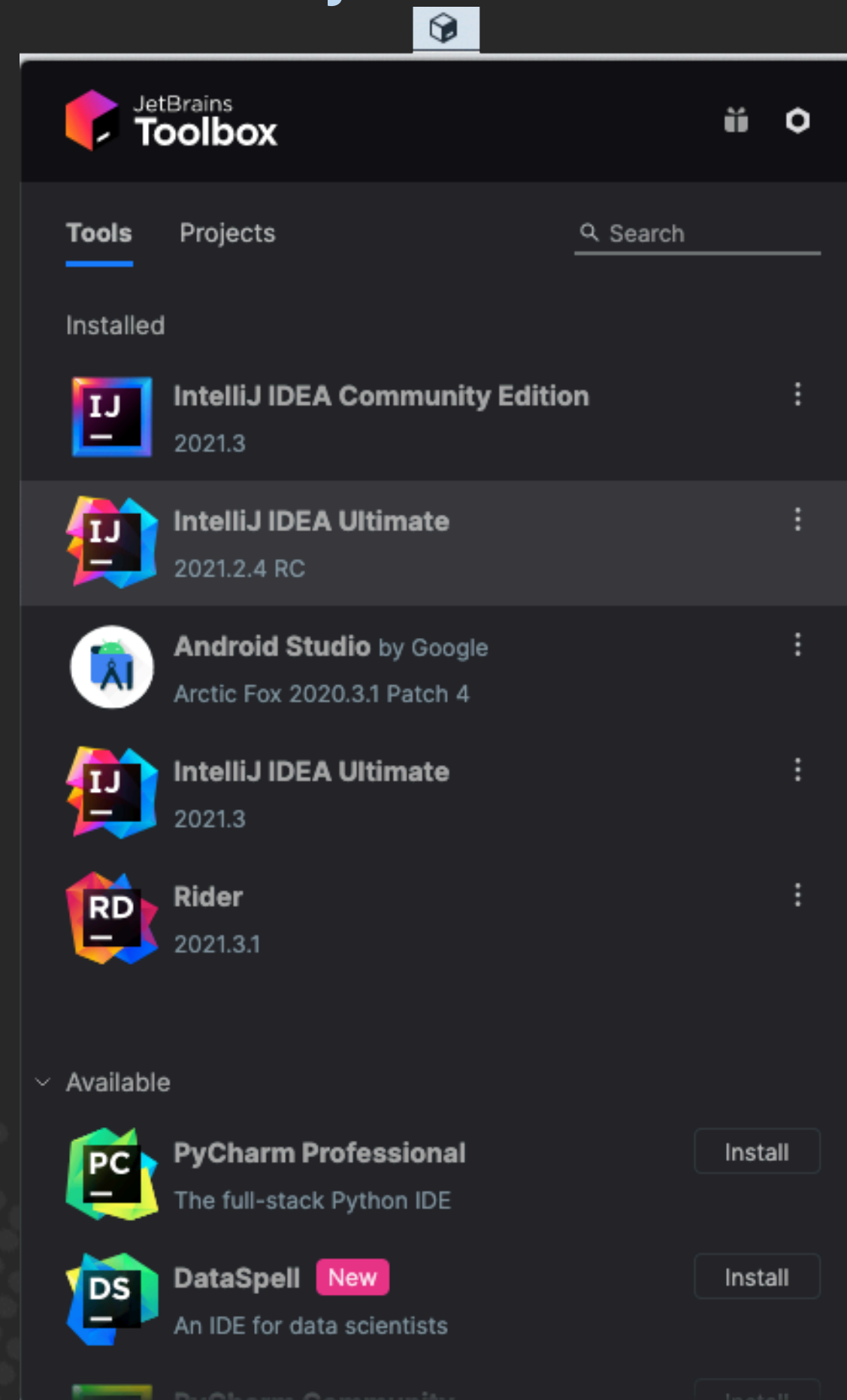
## © I do not work for JetBrains

- Cannot speak on behalf of the company
- Please do not ask about pricing and future plans
  - e.g. Why is AI Assistant not free? Why no free student version?

# Download JetBrains Toolbox

## ● Keeps the IDEs up to date

– <https://www.jetbrains.com/toolbox-app/>



# Versions of IntelliJ

## ◎ **Ultimate Edition**

- Paid version, has many great features
- AI Assistant as extra subscription

## ◎ **Community Edition**

- Free version, excellent tool for editing Java code

# AI Assistant

- ◎ **Plugin available since 2024.1 for all versions**
  - Free 7 day trial for paid IDEs
  - [www.jetbrains.com/ai/#plans-and-pricing](http://www.jetbrains.com/ai/#plans-and-pricing)
- ◎ **Our code is not used for LLM training**
  - [www.jetbrains.com/help/ai/data-collection-and-use-policy.html](http://www.jetbrains.com/help/ai/data-collection-and-use-policy.html)
  - They have options for on-prem LLMs
- ◎ **More information here**
  - [www.jetbrains.com/ai/#general-faq](http://www.jetbrains.com/ai/#general-faq)

# Setting up IntelliJ IDEA

## ◎ Appearance & Behavior

- Darcula Scheme most popular, but errors harder to see

## ◎ Keymap, Mac OS X

- IntelliJ IDEA classic has more finger friendly shortcuts



# Editor Settings

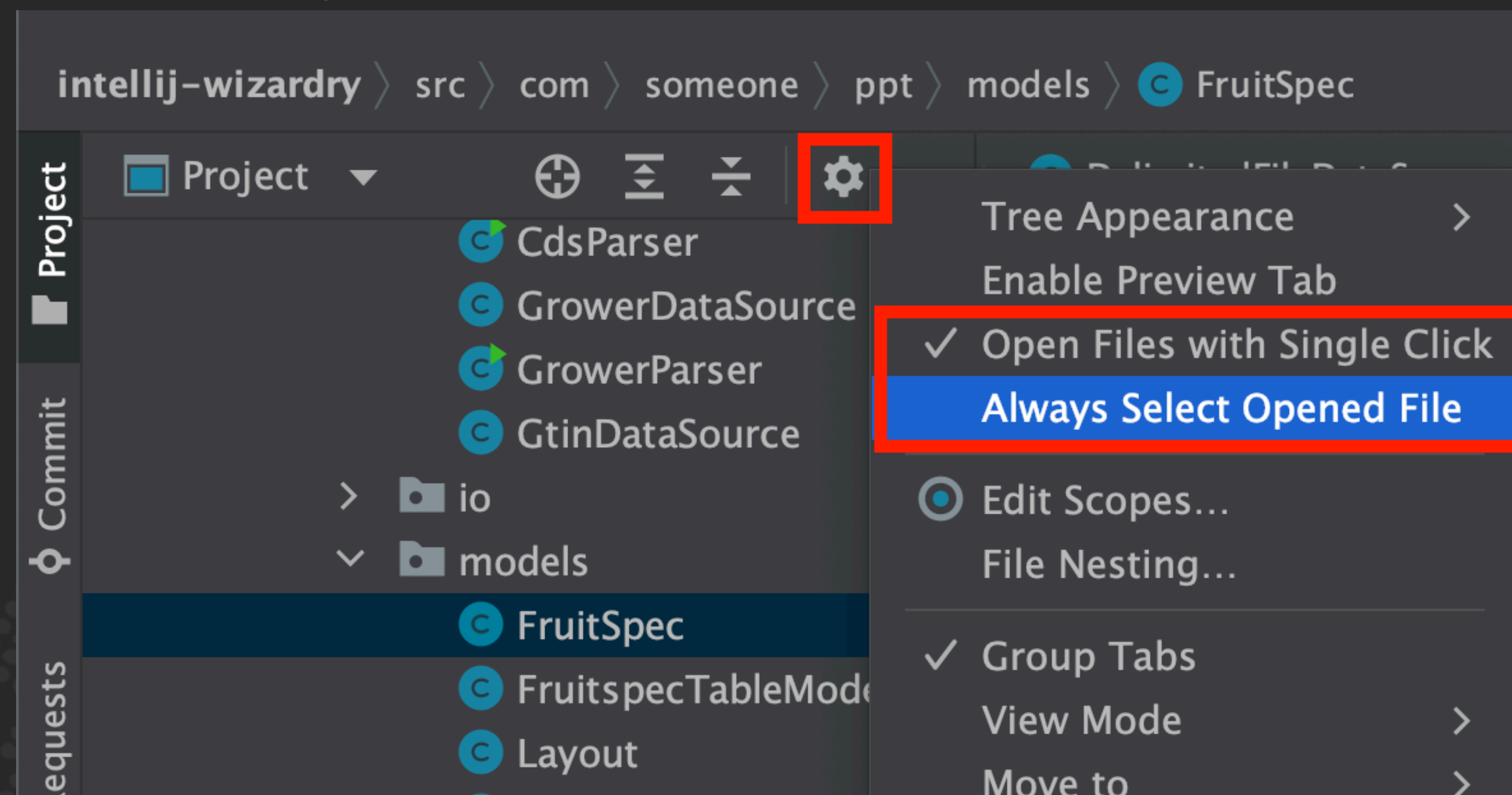
- **Editor** →

- General → Smart Keys: Select Use "CamelHumps" words

# 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





# IntelliJ IDEA Philosophy

# IntelliJ IDEA Philosophy

- © IntelliJ designed to be used mostly without mouse
- © Hotkeys for almost everything
  - Print out Help → Keyboard Shortcuts 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
Quick switch current scheme	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
Resume expression	Alt + F8
Resume program	F9
Toggle breakpoint	Ctrl + FB
New breakpoints	Ctrl + Shift + FB

**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
Deselect 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
Explain Doc	Shift + F1
Start line	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
Autoscroll 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 all code block end / start	Ctrl + Shift + J / [
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 line 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 + J / [
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
Get source / view source	F4, Ctrl + Enter
Show navigation bar	Alt + Home
Toggle bookmarks	F11
Toggle bookmarks with mnemonic	Ctrl + F11
Go to numbered bookmark	Ctrl + #[0-9]
Show bookmarks	Shift + F11

**COMPILE AND RUN**

Build project	Ctrl + F9
Compile selected file, 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**

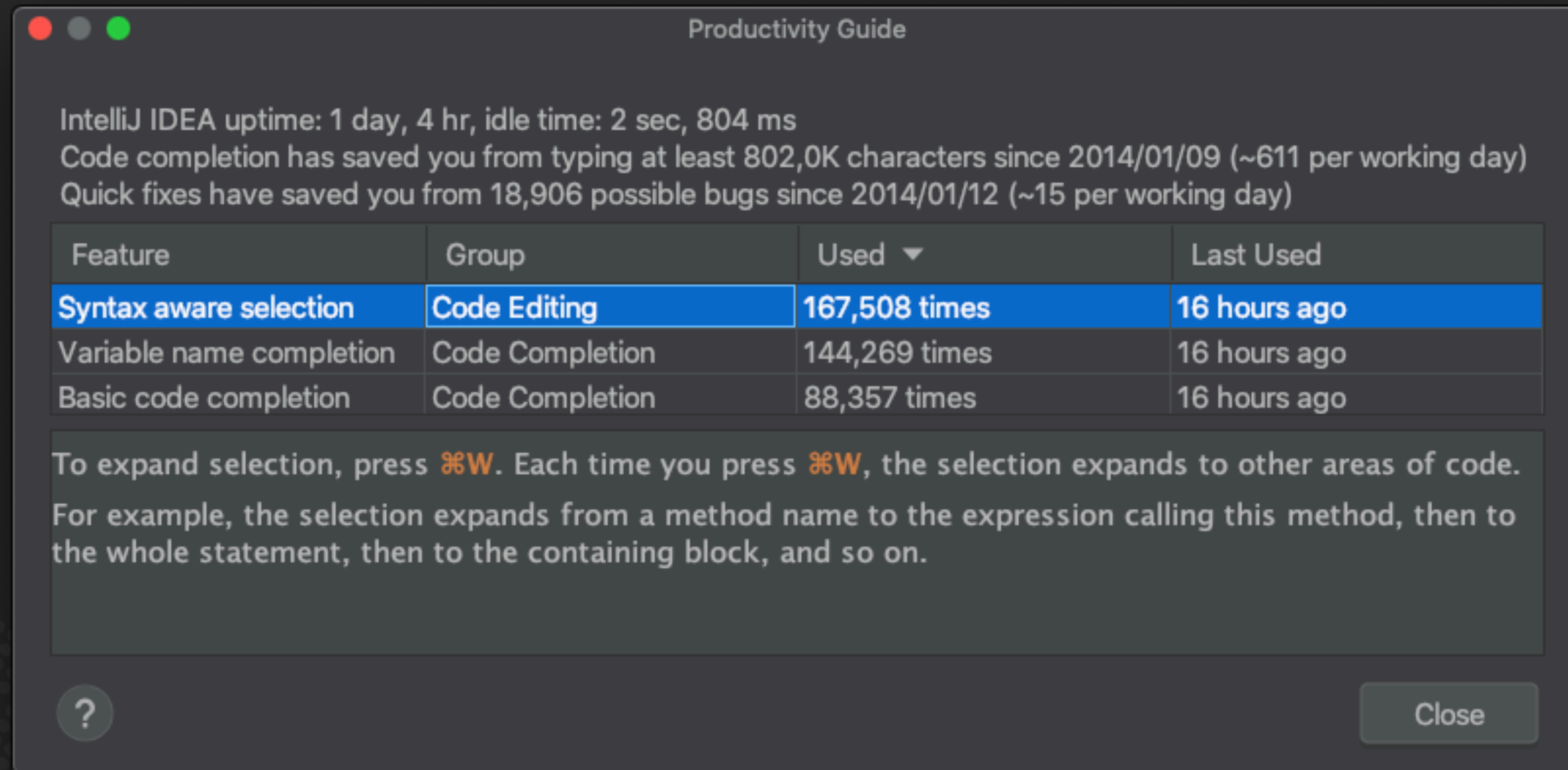
Commit project to VCS	Ctrl + K
Update project from VCS	Ctrl + T
Push commit	Ctrl + Shift + K
VCS quick popup	Alt + BackQuote ( ` )

**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



The screenshot shows a 'Productivity Guide' dialog box with a title bar containing three window control buttons (red, grey, green) and the text 'Productivity Guide'. The main content area contains the following text:

IntelliJ IDEA uptime: 1 day, 4 hr, idle time: 2 sec, 804 ms  
Code completion has saved you from typing at least 802,0K characters since 2014/01/09 (~611 per working day)  
Quick fixes have saved you from 18,906 possible bugs since 2014/01/12 (~15 per working day)

Feature	Group	Used ▼	Last Used
Syntax aware selection	Code Editing	167,508 times	16 hours ago
Variable name completion	Code Completion	144,269 times	16 hours ago
Basic code completion	Code Completion	88,357 times	16 hours ago

To expand selection, press **⌘W**. Each time you press **⌘W**, the selection expands to other areas of code. For example, the selection expands from a method name to the expression calling this method, then to the whole statement, then to the containing block, and so on.

At the bottom left, there is a question mark icon in a circle. At the bottom right, there is a 'Close' button.

# 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 ⇧**



# When IntelliJ IDEA Hangs Up

# Livelock from Corrupt Local Files

The screenshot shows the IntelliJ IDEA IDE interface. The main editor displays the code for `WildFactorizer.java`. The code is as follows:

```
1 package concurrency.ch06_task_execution.solution_6_1;
2
3 import concurrency.math.*;
4
5 import java.util.concurrent.atomic.*;
6
7 public class WildFactorizer {
8     public static void main(String[] args) {
9         long start = (1 << 19) - 1; // known Mersenne prime
10        AtomicLong next = new AtomicLong(start);
11        for (int i = 0; i < 10000; i++) {
12            long number = next.getAndIncrement();
13            long[] factors = Factorizer.factor(number);
14            if (factors.length == 1) {
15                System.out.println(factors[0]);
16            }
17        }
18    }
19 }
```

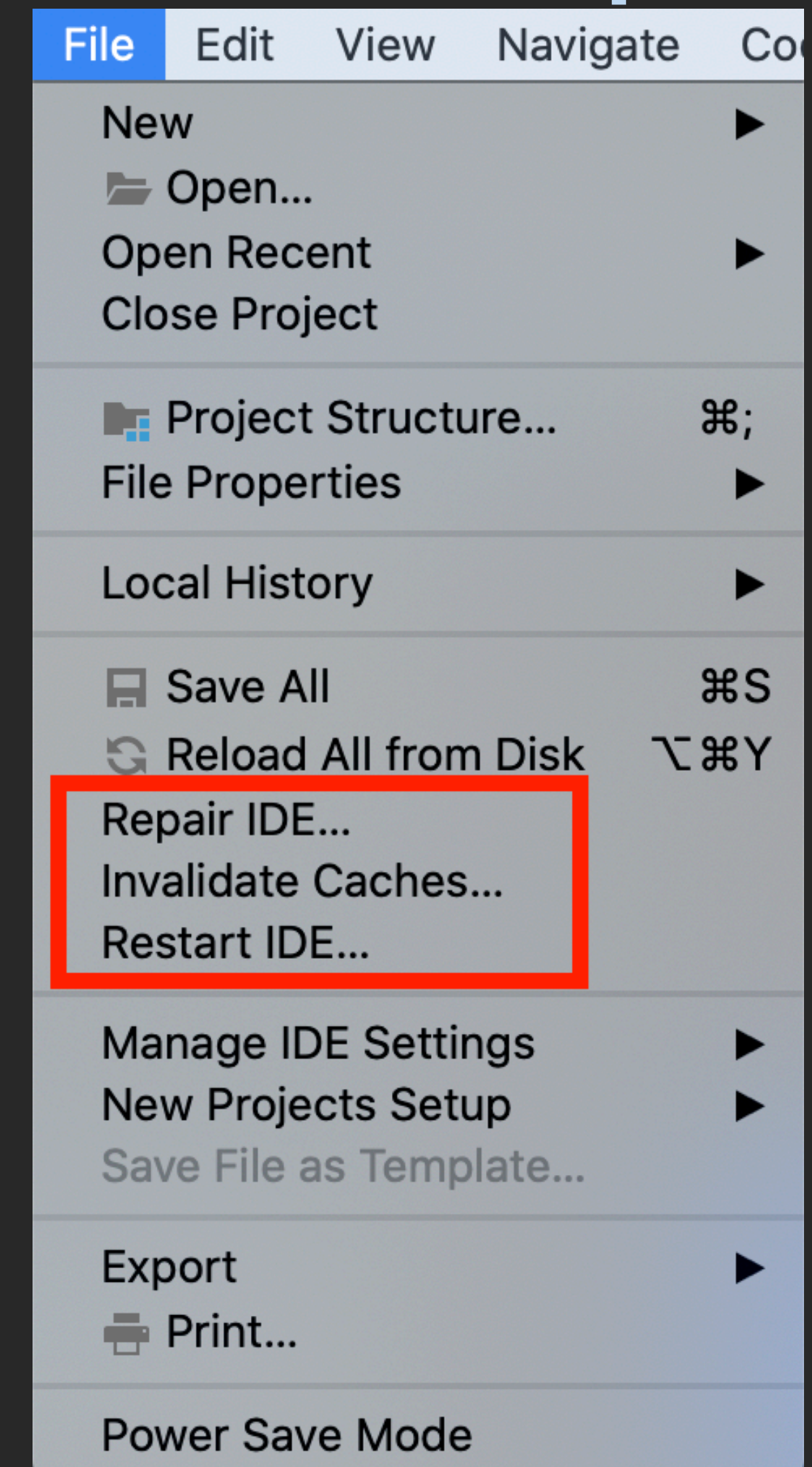
The IDE shows a compilation error at the bottom: "Compilation aborted (moments ago)". The error message is partially visible as "171M of 2043M". The project structure on the left shows the file is located in `concurrency.ch06_task_execution.solution_6_1`. The status bar at the bottom indicates the time is 7:20 and the encoding is UTF-8.



# 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



# Opening old projects

- ◎ **Modern versions of IntelliJ IDEA might go into infinite loops**
  - Need to kill with -9
- ◎ **Create new project with existing sources and overwrite old project files**



# Essential Shortcuts

# Superkey for fixing almost anything

- ◎ "Show intention actions and quick-fixes"
  - Windows/Linux: Alt + Enter
  - Mac OS X:  $\text{⌘} \leftarrow$

# Generate new code

## ◎ "Generate Code"

- Windows/Linux: Alt + Ins
- Mac OS X (Official): ⌘N
- Mac OS X (Heinz): ^↵ or ^N

## ◎ Quick demo creating playground.Person

- Add final field, show difference between ^↵ and ⌘↵

## Before we continue ...

### ◎ Get our Data Structures in Java Course here

- <https://tinyurl.com/jbiltong>
- Coupon expires at 1pm South African Time
  - But you have life-time access once you have redeemed it



# Live Templates

- ◎ **We can generate code quickly with live templates**
  - psvm or main: Main method
  - sout, soutv, soutm, soutp: Output
  - iter, fori, itco, itar: Iteration

[tinyurl.com/jbiltong](https://tinyurl.com/jbiltong)



# Navigation

## ◎ "Go to declaration"

- Windows/Linux: Ctrl + B or Ctrl + Click
- Mac OS X: ⌘B or ⌘Click

## ◎ "Navigate back / forward"

- Windows/Linux: Ctrl + Alt + Left / Right
- Mac OS X: ⌘⇧← / ⌘⇧→

[tinyurl.com/jbiltong](https://tinyurl.com/jbiltong)





# 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

# Bookmarks

- ◎ **Quick navigation between locations in project**
- ◎ **Set with**
  - Windows/Linux: Ctrl + Shift + #[0-9]
  - Mac OS X:  $\wedge\uparrow 0 \dots \wedge\uparrow 9$
- ◎ **Navigate with**
  - Windows/Linux: Ctrl + #[0-9]
  - Mac OS X:  $\wedge 0 \dots \wedge 9$

# Syntax Aware Selection

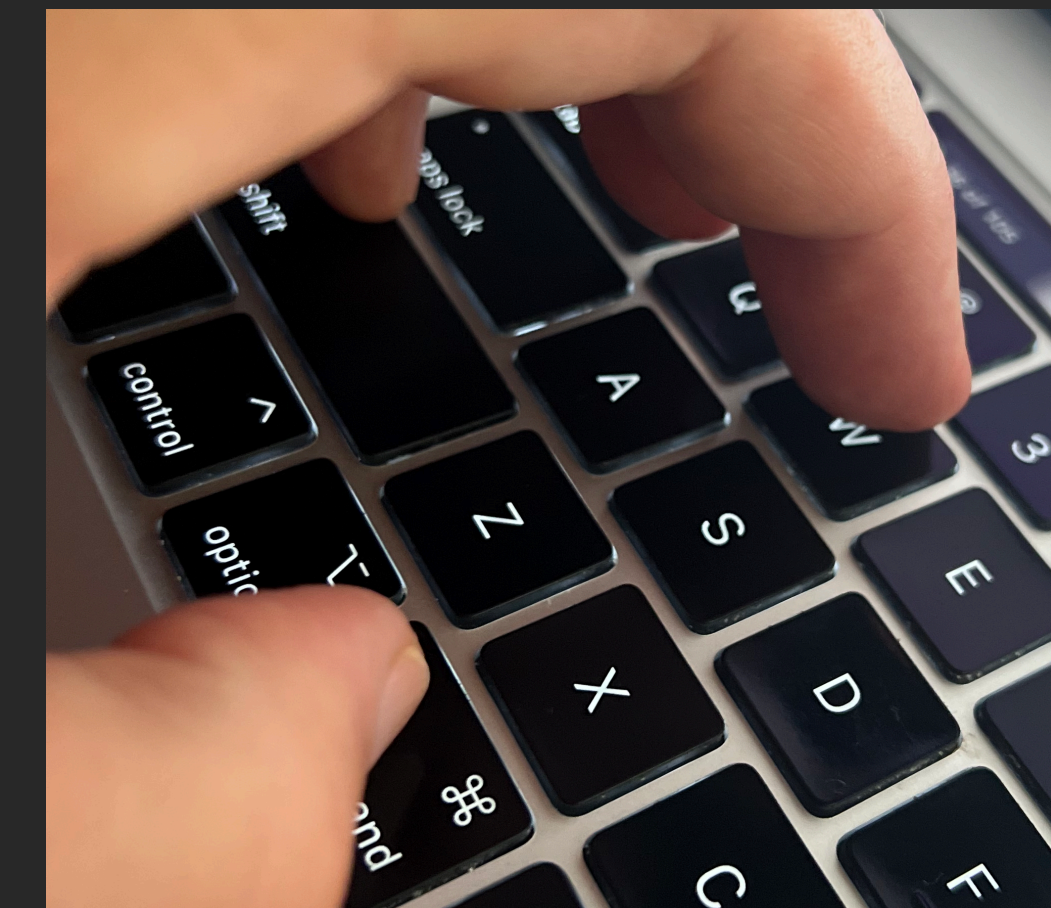
## ◎ "Extend Selection"

- 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

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



# 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**



# Turbo-boosted Productivity

# 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
- ◎ **Demo: changing main(String[]) to main(String...)**

# 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));
}
```

# Column Select Editing

## ● With the mouse

- Windows/Linux: Alt + Drag Mouse
- Mac OS X:  $\text{⌘} + \text{drag mouse}$

## ● With keyboard toggle column selection mode

- Windows/Linux: Alt + Shift + Insert
- Mac OS X (Official):  $\text{⌘} \uparrow 8$
- Mac OS X (Heinz):  $\text{⌘} \wedge \uparrow C$
- Make sure to turn column selection mode off afterwards



## Demo

- © 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;
    }
}
```

# AI Assistant

- **Let's try with the AI Assistant**



# Code Completion

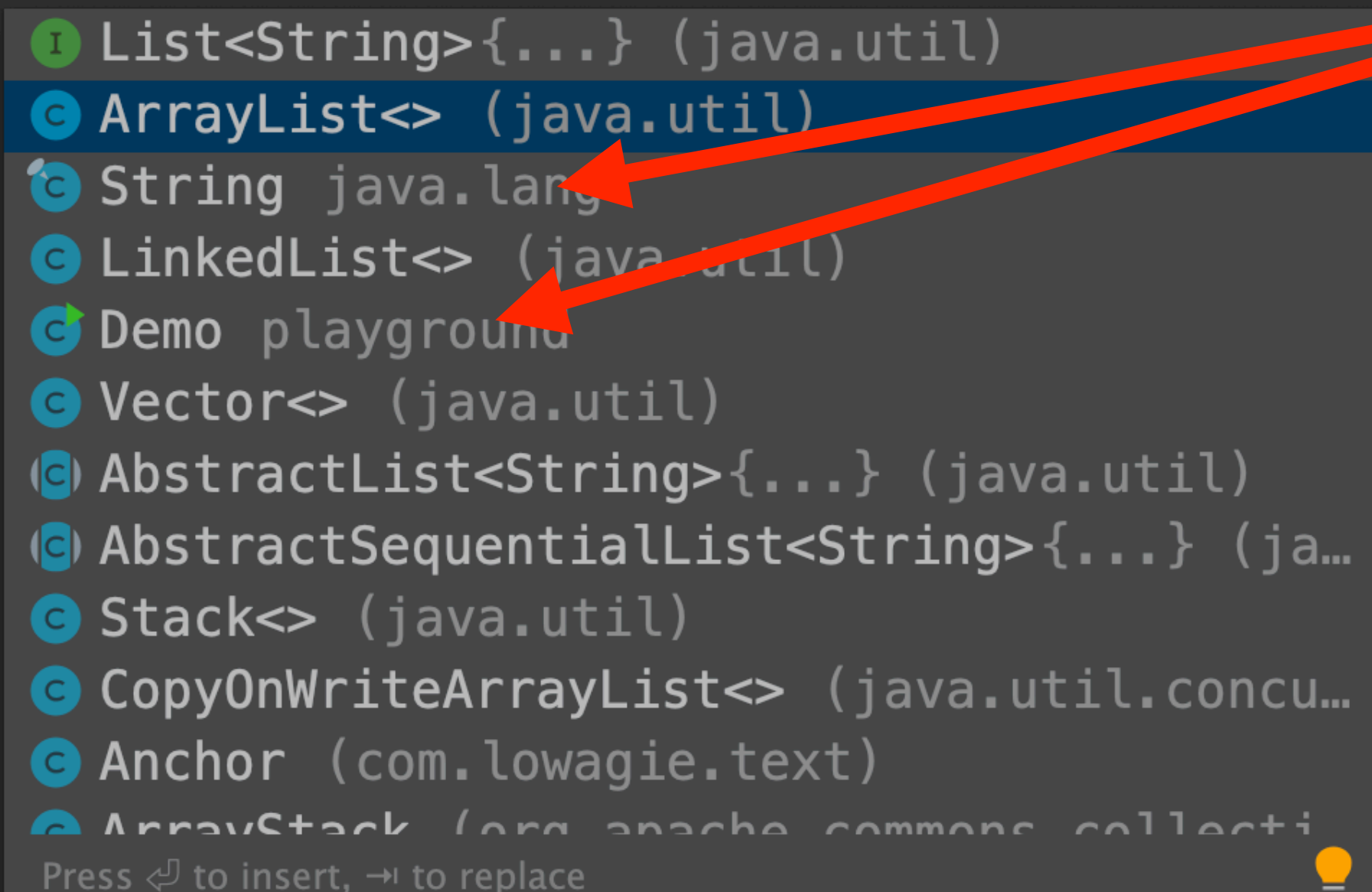
# Basic Code Completion

## ◎ Code completion traditionally uses Ctrl+Space

- Windows/Linux: Ctrl + Space
- Mac OS X: ^Space

```
import java.util.List;
```

```
public class Demo {  
    public static void main(String[] args) {  
        List<String> names = new  
    }  
}
```



The screenshot shows the IntelliJ IDEA code completion menu for the line `List<String> names = new`. The menu lists several options, with `List<String>{...} (java.util)` selected. Two red arrows point from a large red question mark on the right towards the menu items `String java.lang` and `Demo playground`. At the bottom of the menu, it says "Press ↵ to insert, → to replace".

- List<String>{...} (java.util)
- ArrayList<> (java.util)
- String java.lang
- LinkedList<> (java.util)
- Demo playground
- Vector<> (java.util)
- AbstractList<String>{...} (java.util)
- AbstractSequentialList<String>{...} (java.util)
- Stack<> (java.util)
- CopyOnWriteArrayList<> (java.util.concurrent)
- Anchor (com.lowagie.text)
- ArrayStack (org.apache.commons.collections)

Press ↵ to insert, → to replace

# Smart Code Completion

## ◎ This gives much better result - I always use this

- Windows/Linux: Ctrl + Shift + Space
- Mac OS X: ^⇧Space

```
import java.util.List;
```

```
public class Demo {  
    public static void main(String[] args) {  
        List<String> names = new |  
    }  
}
```

```
I List<String>{...} (java.util)  
C Anchor (com.lowagie.text)  
C ArrayList<> (java.util)  
C LinkedList<> (java.util)  
C AbstractList<String>{...} (java.util)  
C Vector<> (java.util)  
C AbstractSequentialList<String>{...} (ja...  
C Stack<> (java.util)  
C Phrase (com.lowagie.text)  
C CopyOnWriteArrayList<> (java.util.concu...  
C ArrayStack (org.apache.commons.collecti...  
C Chapter (com.lowagie.text)
```

Press ↵ to insert, → to replace





# Refactoring

# Alter Class / Field / Method

## ● "Move"

- Windows/Linux: F6
- Mac OS X: F6

## ● "Rename ..."

- Windows/Linux: Shift + F6
- Mac OS X: ⌘F6

## ● "Change Signature" - applies to methods

- Windows/Linux: Ctrl + F6
- Mac OS X: ⌘F6

# 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



# Demo

- ◎ **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**



# 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

# Java Language Migration Aids

## © Demos

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



# Conclusion

# The Java Specialists' Newsletter

- ◎ **Make sure to subscribe**

- [www.javaspecialists.eu/archive/subscribe/](http://www.javaspecialists.eu/archive/subscribe/)

- ◎ **Readers in 150+ countries**

- Began in Cape Town

- ◎ **Over 24 years of newsletters on advanced Java**

- All previous newsletters available on [www.javaspecialists.eu](http://www.javaspecialists.eu)
- Courses, additional training, etc.

# Remember please

## ◎ Get our Data Structures in Java Course here

- <https://tinyurl.com/jbiltong>
- Coupon expires at 1pm South African Time
  - But you have life-time access once you have redeemed it





# Conclusion IntelliJ Super Productivity

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