



Generative AI Programming Assistant

Danny Dig

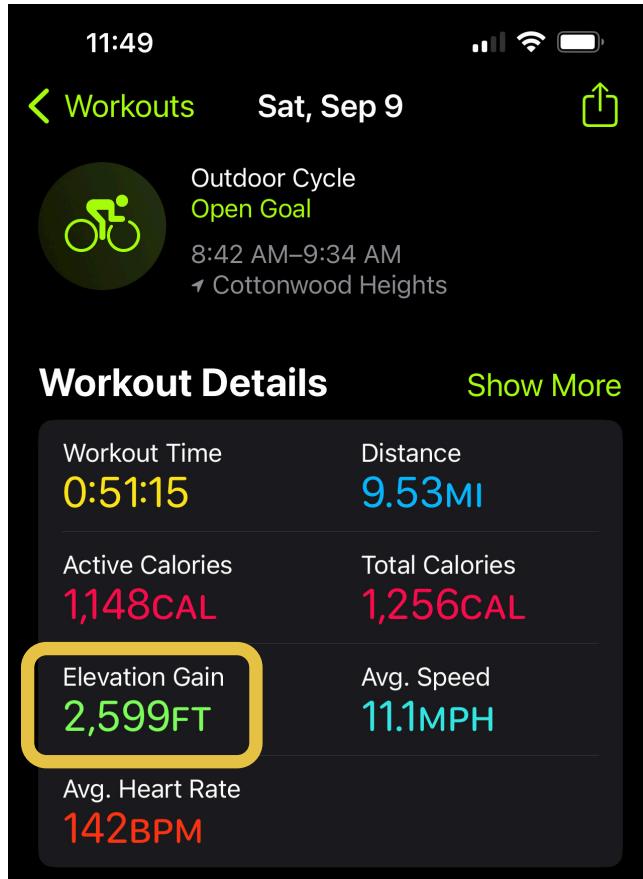
danny.dig@colorado.edu



Boulder



E-Assistant augments our capacity



85 Nm, 500 Wh

9 hardest things programmers have to do

Writing tests

Writing documentation

Implementing functionality you disagree with ...

Working with someone else's code

Dealing with other people ...

Estimating time to complete a task

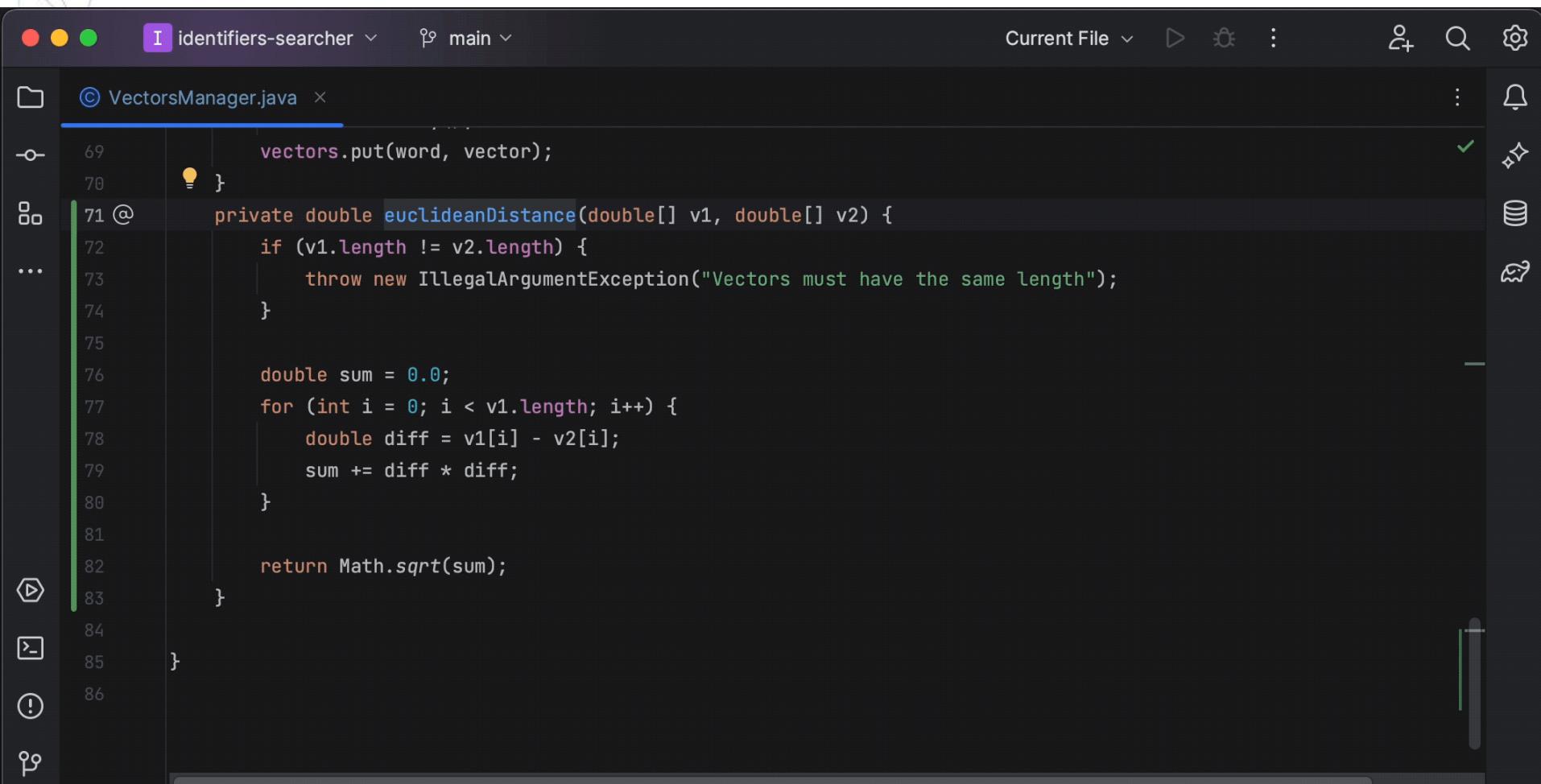
Explaining what I do (or don't do)

Naming things

source: Elizabeth Churchill, UX Director at Google

Generative AI explains code

<https://blog.jetbrains.com/idea/tag/ai-assistant/>



The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** Shows "identifiers-searcher" and "main".
- File List:** Shows "VectorsManager.java".
- Code Editor:** Displays the following Java code:

```
69         vectors.put(word, vector);
70     }
71 @    private double euclideanDistance(double[] v1, double[] v2) {
72     if (v1.length != v2.length) {
73         throw new IllegalArgumentException("Vectors must have the same length");
74     }
75
76     double sum = 0.0;
77     for (int i = 0; i < v1.length; i++) {
78         double diff = v1[i] - v2[i];
79         sum += diff * diff;
80     }
81
82     return Math.sqrt(sum);
83 }
84
85 }
86 }
```
- Toolbars and Icons:** Standard IntelliJ IDEA icons for file operations, search, and settings.
- Sidebar:** Shows navigation icons for project, file, and search.

Explain code, bugs and offer fixes, summarize recent code changes,
perform code reviews

Generative AI explains Runtime Errors

The screenshot shows a Java development environment with the following interface elements:

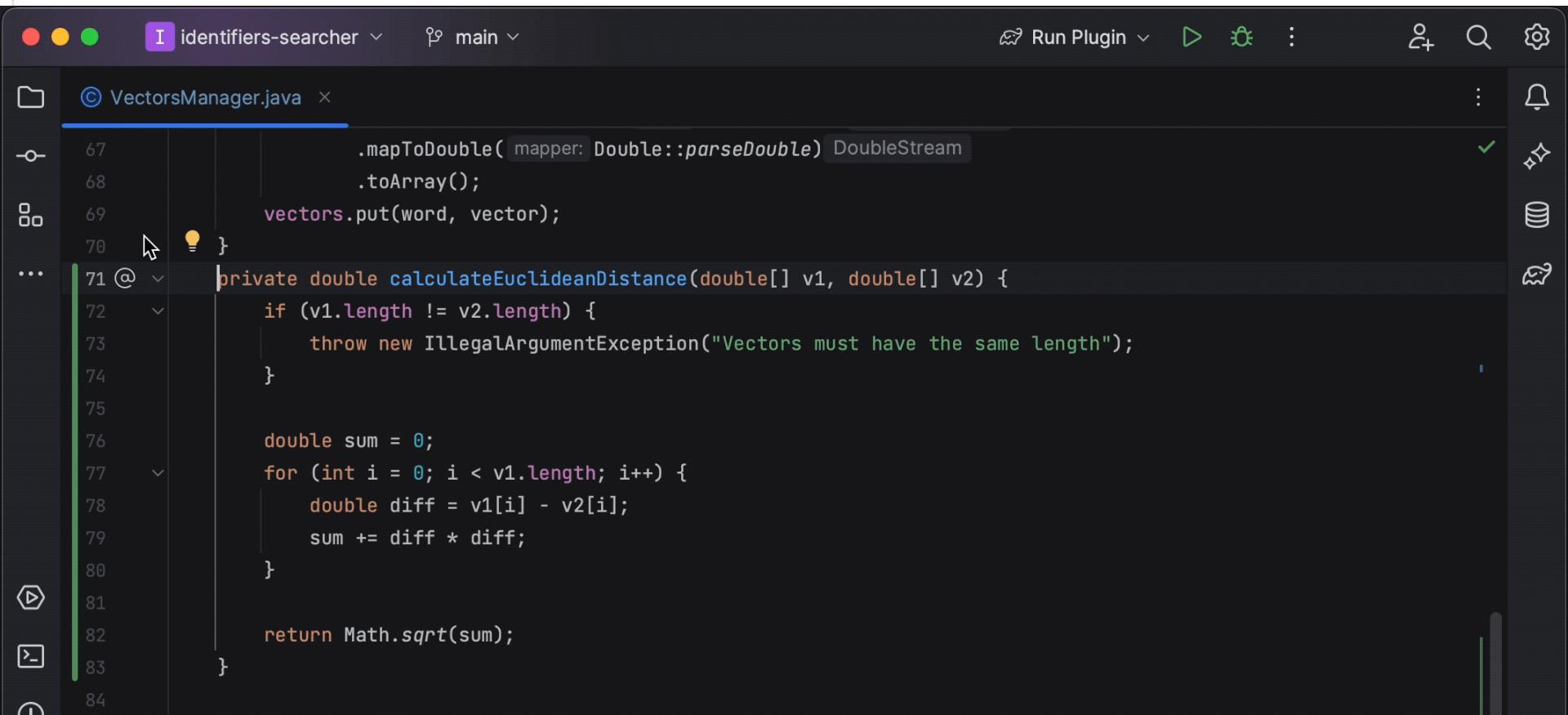
- Top Bar:** Shows the project name "Petclinic" and the file "main".
- Code Editor:** Displays the file "Main.java" containing the following code:

```
1 package org.example;
2
3 import java.util.stream.Stream;
4
5 public class Main {
6     public static void main(String[] args) {
7         var list = Stream.iterate( seed: 1, i -> i + 1 )
8             .toList();
9         System.out.println(list.size());
10    }
11 }
```
- Run Tab:** Shows the process has finished with exit code 1.
- Output Tab:** Displays the error stack trace:

```
Exception in thread "main" java.lang.OutOfMemoryError Create breakpoint : Java heap space Explain with AI ↗
at java.base/java.lang.Integer.valueOf(Integer.java:1081)
at org.example.Main.main$0(Main.java:7)
at org.example.Main$$Lambda$14/0x00000008000c0940.apply(Unknown Source)
at java.base/java.util.stream.Stream$1.tryAdvance(Stream.java:1464)
> at java.base/java.util.Spliterator.forEachRemaining(Spliterator.java:332) <3 internal lines>
> at java.base/java.util.stream.AbstractPipeline.evaluateToArrayNode(AbstractPipeline.java:260) <3 internal lines>
at org.example.Main.main(Main.java:8)
```
- Bottom Status Bar:** Shows the current time as 11:2, the file type as LF, the encoding as UTF-8, and the code style as 4 spaces.

Generative AI generates documentation

Programmers often don't document their code, so AI generates documentation



A screenshot of a Java IDE interface. The title bar shows "identifiers-searcher" and "main". The main window displays a Java file named "VectorsManager.java". The code shown is:

```
    .mapToDouble( mapper: Double::parseDouble) DoubleStream
    .toArray();
    vectors.put(word, vector);
}
private double calculateEuclideanDistance(double[] v1, double[] v2) {
    if (v1.length != v2.length) {
        throw new IllegalArgumentException("Vectors must have the same length");
    }

    double sum = 0;
    for (int i = 0; i < v1.length; i++) {
        double diff = v1[i] - v2[i];
        sum += diff * diff;
    }

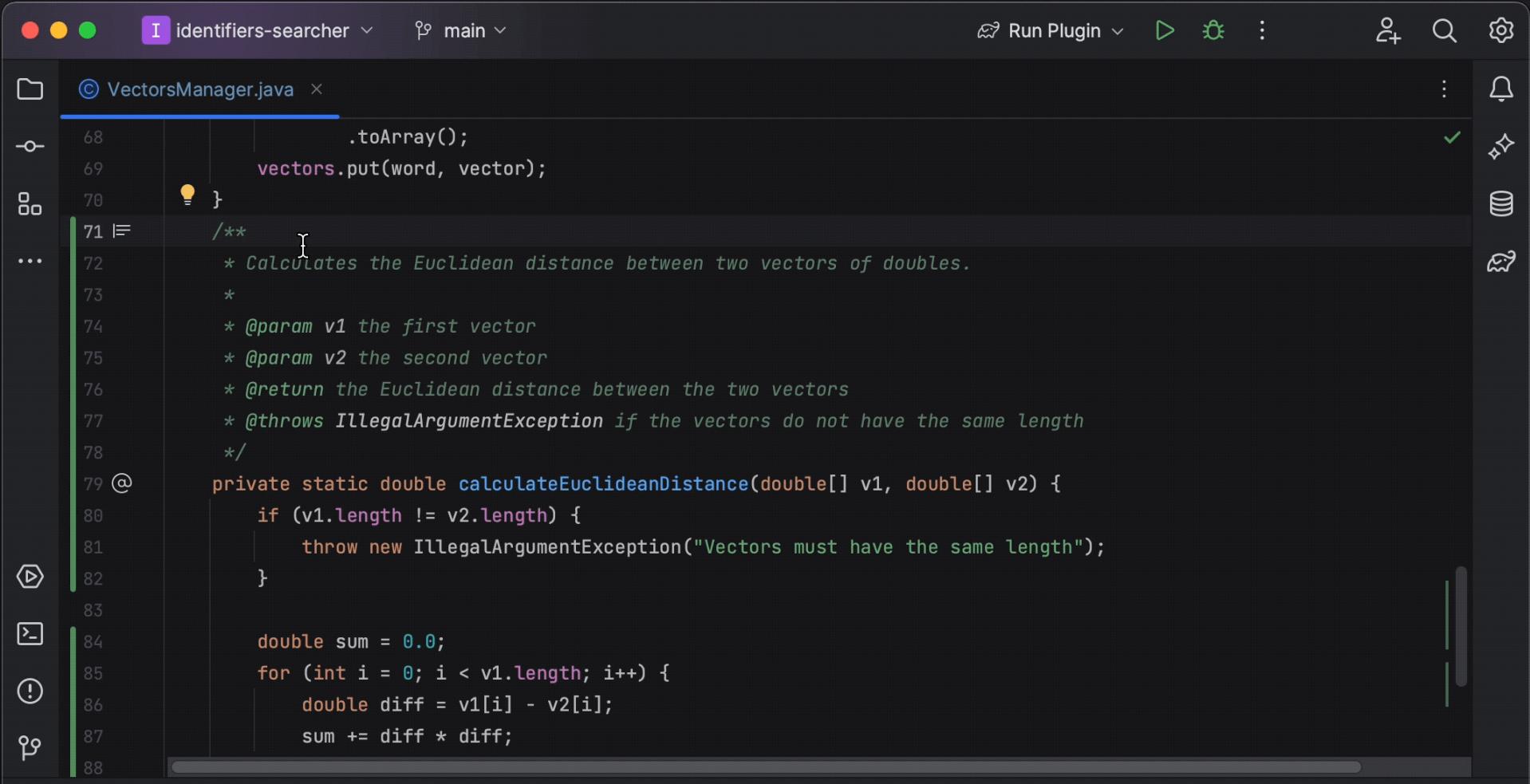
    return Math.sqrt(sum);
}
```

The code editor has a dark theme with syntax highlighting. A tooltip or code completion suggestion "DoubleStream" is visible above the cursor at line 67. The IDE interface includes various toolbars and panels typical of a developer environment.

Generate: documentation, code, test cases, configuration files, comments, commit messages, CLI commands, suggestions to improve code

Generative AI generates commit messages

Programmers don't write good commit messages when they push their changes into code repositories



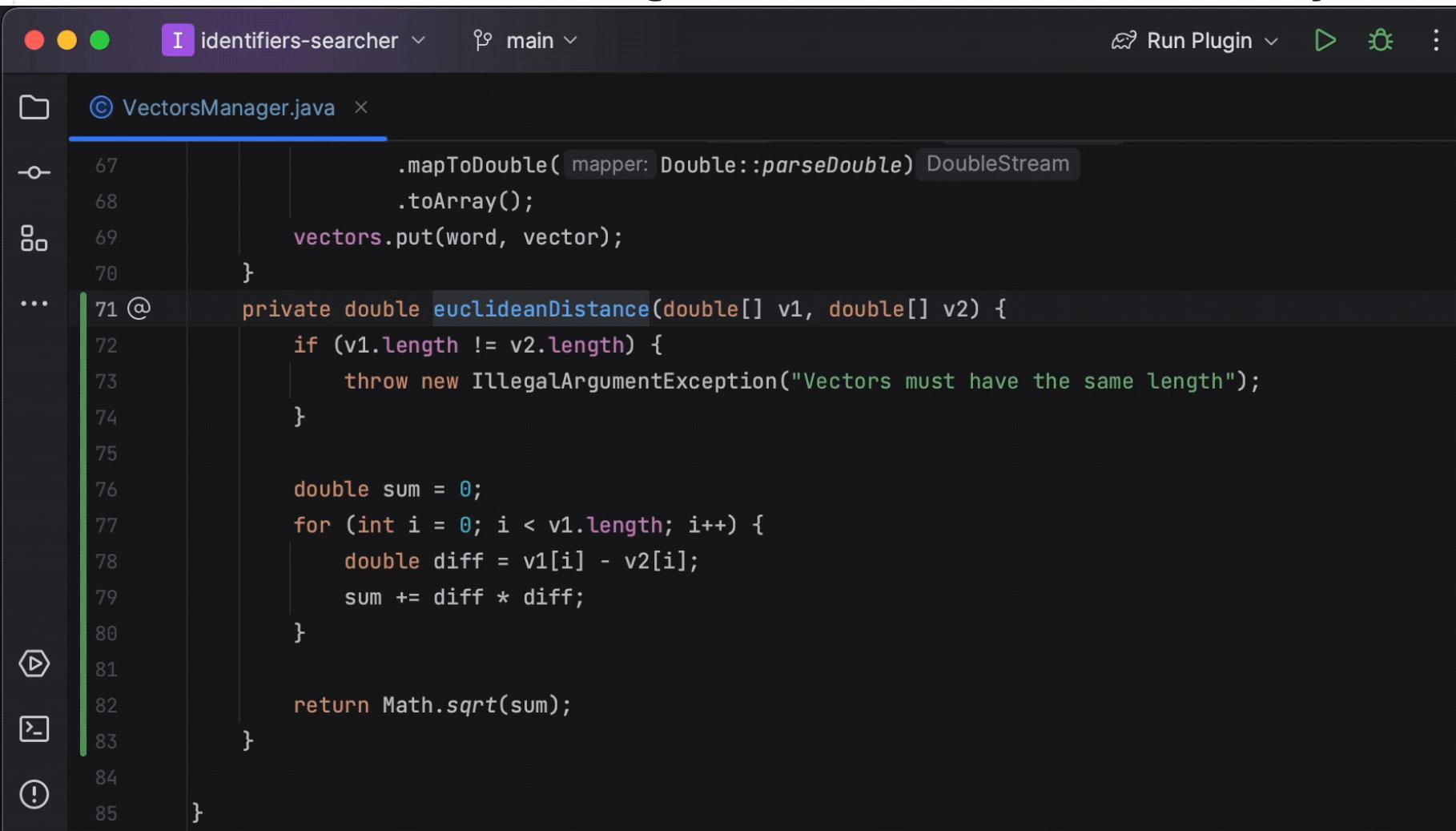
A screenshot of a Java code editor showing the file `VectorsManager.java`. The code implements a method to calculate the Euclidean distance between two vectors. The code is annotated with Javadoc comments and includes exception handling for vectors of different lengths.

```
identifiers-searcher main
VectorsManager.java

68     .toArray();
69     vectors.put(word, vector);
70 }
71 /**
72  * Calculates the Euclidean distance between two vectors of doubles.
73  *
74  * @param v1 the first vector
75  * @param v2 the second vector
76  * @return the Euclidean distance between the two vectors
77  * @throws IllegalArgumentException if the vectors do not have the same length
78 */
79 private static double calculateEuclideanDistance(double[] v1, double[] v2) {
80     if (v1.length != v2.length) {
81         throw new IllegalArgumentException("Vectors must have the same length");
82     }
83
84     double sum = 0.0;
85     for (int i = 0; i < v1.length; i++) {
86         double diff = v1[i] - v2[i];
87         sum += diff * diff;
88     }
89 }
```

Generative AI name suggestions

Programmers don't spend enough time giving intention revealing names to code entities, though it is crucial for code readability



A screenshot of a Java code editor showing a file named `VectorsManager.java`. The code implements a method to calculate the Euclidean distance between two vectors. The code is annotated with intentions from an AI tool, such as `parseDouble` for the `mapper` parameter and `sqrt` for the `Math.sqrt` call.

```
67     .mapToDouble( mapper: Double::parseDouble) DoubleStream
68     .toArray();
69     vectors.put(word, vector);
70 }
71 @private double euclideanDistance(double[] v1, double[] v2) {
72     if (v1.length != v2.length) {
73         throw new IllegalArgumentException("Vectors must have the same length");
74     }
75
76     double sum = 0;
77     for (int i = 0; i < v1.length; i++) {
78         double diff = v1[i] - v2[i];
79         sum += diff * diff;
80     }
81
82     return Math.sqrt(sum);
83 }
84
85 }
```

Addressing top challenge with Generative AI

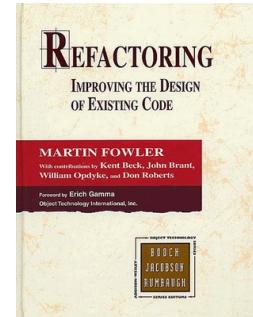
Top concerns of surveyed developers are security, safety, and trustworthiness of AI Generative tools



**Combine creativity of Generative AI with
Safety analysis of Programming Environments**

My Refactoring Story

"A change made to the **internal structure of software to make it **easier to understand** and **cheaper to modify** without changing its observable behaviour" – M. Fowler [1999]**



Top-level menu in all modern IDEs

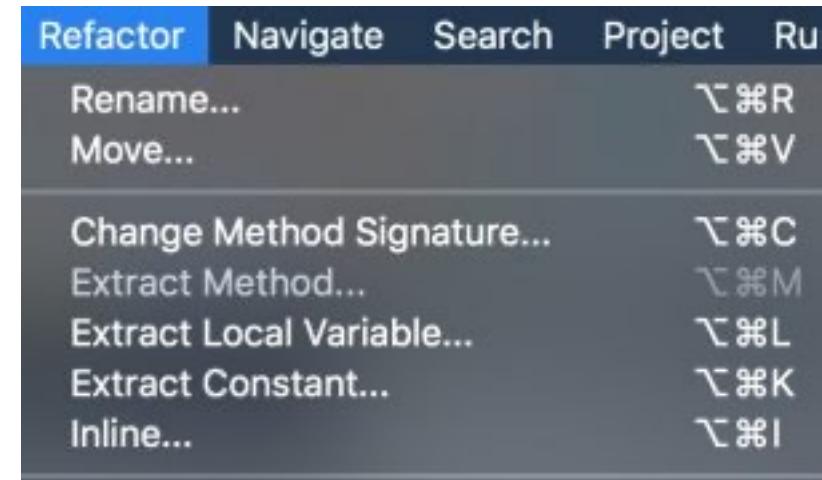
In 2000, I created the first open-source refactoring tool



In 2004, I joined the team



In 2023-24 doing a Sabbatical at JetBrains



Research



To go fast, go alone. To go far, go with others.

ExtractMethod-Assist

LLM + IDE: Together We Go Further

Dorin Pomian, Malinda Dilhara, Zarina Kurbatova, Abhiram Bellur, Egor Bogomolov, Andrey Sokolov, Timofey Bryksin, Danny Dig



ExtractMethod in IntelliJ today

- Top-5 most used refactorings, IntelliJ was the first Java IDE to support
- Solution to remove Long Method, it improves:
readability, reuse, testing, reduce technical debt

ExtractMethod in IntelliJ today

- ✖ Semi-automated
- ✖ No recommendations

```
// Write myMethods
int methodCount = 0;
for (JvmMethod jvmMethod : jvmClass.getMethods()) methodCount++;
DataInputOutputUtil.writeInt(out, methodCount);
for (JvmMethod jvmMethod : jvmClass.getMethods()) {
    writeJvmMethod(jvmMethod, out);
}

// Write AnnotationTargets
int elemTypeCount = 0;
for (ElemType elemType : jvmClass.getAnnotationTargets()) elemTypeCount++;
DataInputOutputUtil.writeInt(out, elemTypeCount);
for (ElemType elemType : jvmClass.getAnnotationTargets()) {
    writeElemType(elemType, out);
}

if (jvmClass.getRetentionPolicy() != null) {
    out.writeUTF(jvmClass.getRetentionPolicy().name());
}
else {
    out.writeUTF(s: "");
}
```

Extract Method Research



Many research tools

- JDeodorant
- JExtract
- LiveREF
- REMS
- GEMS
- SEMI



Optimize software quality metrics (complexity, coupling, cohesion)



Generate refactorings that do not align with developers' preferences

Large Language Models (LLMs) and Refactoring

Corpus of 2,849 real-world extract methods:

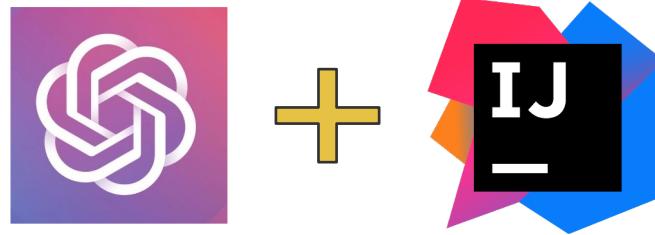
-  LLMs are creative and prolific: average 20 suggestions per method
-  63% of suggestions are hallucinations:
 - 46% are invalid: non-compiling or semantically incorrect
 - 17% are not useful (e.g. one liners, or entire method body)



Key Idea



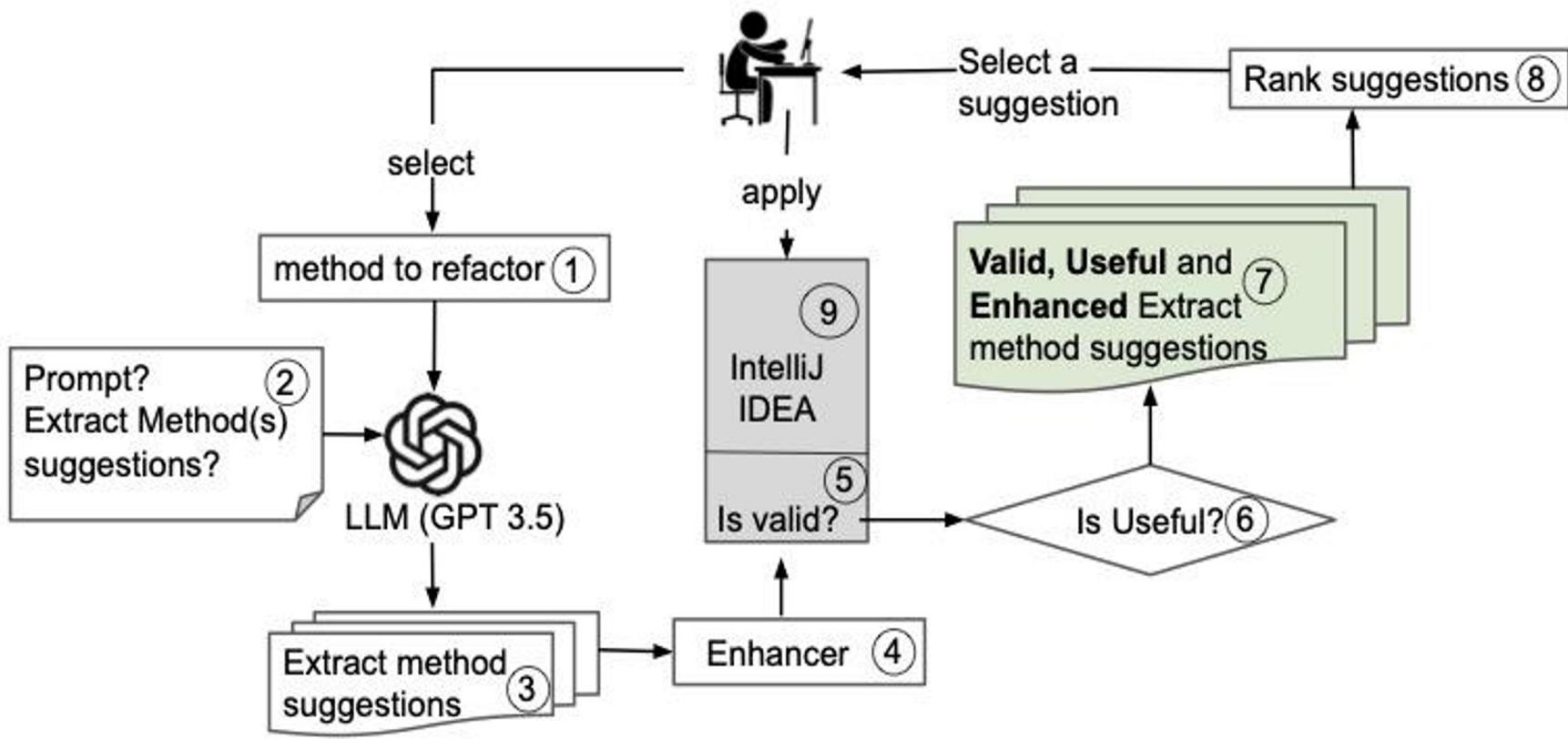
Combine creative capabilities of LLMs + full power of static analysis in IntelliJ to apply refactorings safely



Use static analysis techniques to filter, further enhance, and rank LLM-provided suggestions



Under the Hood



Empirical Evaluation

RQ1: How effective are LLMs for refactoring?

A: WOW effect, but with a high rate of hallucinations

RQ2: What are best practices for using LLMs for refactoring?

A: sensitivity analysis of LLM hyper-parameters, ablation study for our components

RQ3: Effectiveness of EM-Assist

1. Synthetic corpus: 122 Extract Methods

- Used by other tools (JDeodorant, JExtract, LiveREF, REMS, GEMS, SEMI)
- EM-Assist has 61% recall
- Next best tools 54% recall (ML-based), and 52% (static analysis)

2. Extended corpus: 2,840 Extract Methods from real world

- EM-Assist has 42% recall
- Compared to 6.5% recall rate reported by its peers (i.e., *LiveREF*)

RQ4: Usefulness and Usability

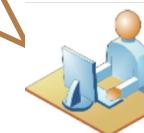
Usefulness of suggestions: fire-house survey with 16 experienced developers
- 81% positive rating for recommendations

Usability study with 18 developers: 94% gave a positive rating

"Thank you for interesting suggestions! Hope to see this in production."



"These suggestions made me look at this code with new eyes, and I will refactor it."



Lessons Learned

Prompt engineering: do what LLM suggests, not what they do

- remove hallucinations automatically, execute with the IDE
- precise prompt for higher quality suggestions
- few-shot learning worked best

Taming LLM non-determinism:

- re-prompting not a waste
- newly-designed ranking to match LLM workflow

Sweet spot: temperatures and number of iterations

Effectiveness is slightly better on synthetic benchmarks, but dramatic improvements (6.4x) on real-world large datasets

Research Preview: Programming By Example



Learn coding best practices from open-source ML Python repos and transplant these into other code



Cannot apply these to new sites unless the code is exactly the same



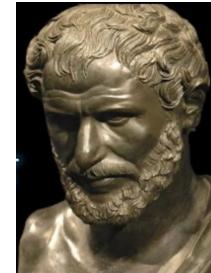
Use LLM to generate many examples, validate and apply suggestions to new locations

25x improvements over previous state of the art

We submitted hundreds of patches to famous open-source projects (e.g., TensorFlow), they accepted 83%

Executive Summary

Change is the only guaranteed constant



Generative AI + Programming Environments + Human

“ ... spot on with what I hate to do as a programmer, happy if LLMs could help with that” – NEC software engineer

“...Game changer! Enter the code assistant, the great equalizer. It's the ace up the sleeve for OEMs” --- Lenovo CTO

Our responsibility as a research community for trustworthy GenAI

PPI Center: industry-university-government partnership

- iterative process, present progress, get feedback, improve, reenter

PPICenter.org

danny.dig@colorado.edu

