OXO Exercise Continues

(and concerns over use of Generative AI)

COMSM0086

Dr Simon Lock & Dr Sion Hannuna

Extra OXO Features

This week we will continue with the OXO exercise

We will add in some error handling mechanisms

Plus extensions to make game more "interesting"

Let's look at each of these in turn...

Error Handling

It's likely users will make mistakes during gameplay
Entering 'invalid' cell identifiers into the GUI:

- Invalid Identifier Length: Command is not 2 chars
- Invalid Identifier Character: Row character is not a letter or column character not a numerical digit
- Outside Range: Valid characters, but identifier values are out of range (i.e. too big or too small)
- Already Taken: Cell has previously been claimed

In Java we handle run-time errors using 'Exceptions'

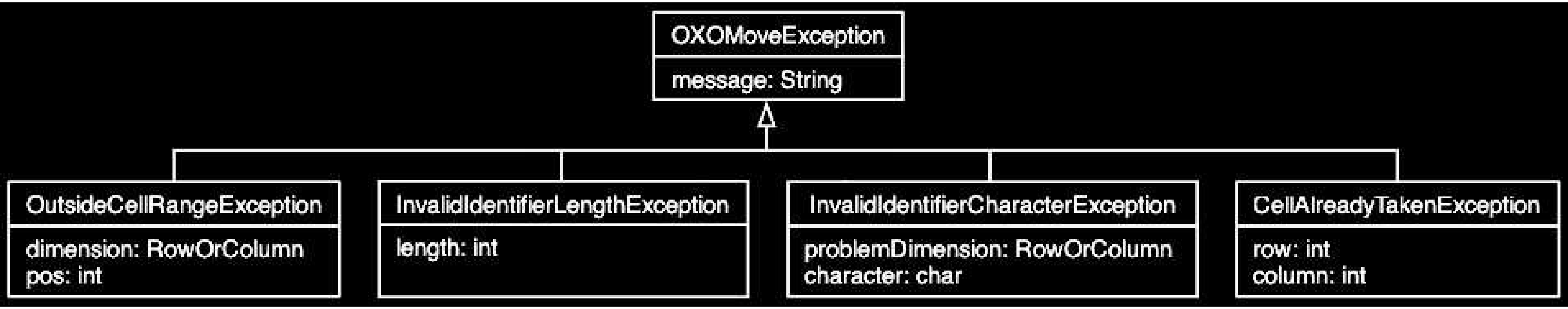
Exceptions Hierarchy

We've provided you with a hierarchy of exceptions

One to represent each of the previous user errors

Commonalities are factored out into a superclass

See workbook for examples of how to use these


OXOMoveException
message: String
OutsideCellRangeException
dimension: RowOrColumn
pos: int
InvalidIdentifierLengthException
length: int
InvalidIdentifierCharacterException
problemDimension: RowOrColumn
character: char
CellAlreadyTakenException
row: int
column: int

Adjustable Win Threshold

"Win Threshold" is number of cells required to win
More interesting if we can alter this threshold !

OXOGame allows users to set the win threshold
Altered by pressing the `+` and `-` keys
(actually the `=` and `-` keys for convenience)

Controller is then notified through two methods:

```
increaseWinThreshold()   decreaseWinThreshold()
```

You should update the threshold held by OXOModel
Then use this value when performing win detection

Greater Number of Players

Traditional number of players in an OXO game is 2
Additional players makes game more interesting !

Add features to support ANY number of players
(data structures, turn taking, win detection etc.)

Number of players can't be changed using GUI
This number can only be changed programatically
(A good opportunity for automated testing !)

Any questions before we go on ?

Let's spend some time talking about AI

Various Types of AI Development Tools Exist

1. Suggestion/completion tools embedded in IDEs
   Such as IntelliJ, which acts as a "prompter"

2. Problem solving and error fixing search tools...
   Tools like ChatGPT as frontend to StackOverflow

3. Fully-fledged code "generation" tools...
   Tools like Gemini or Copilot to synthesise code

The first two scenarios are fairly uncontentious
We need to be cautious about code generation

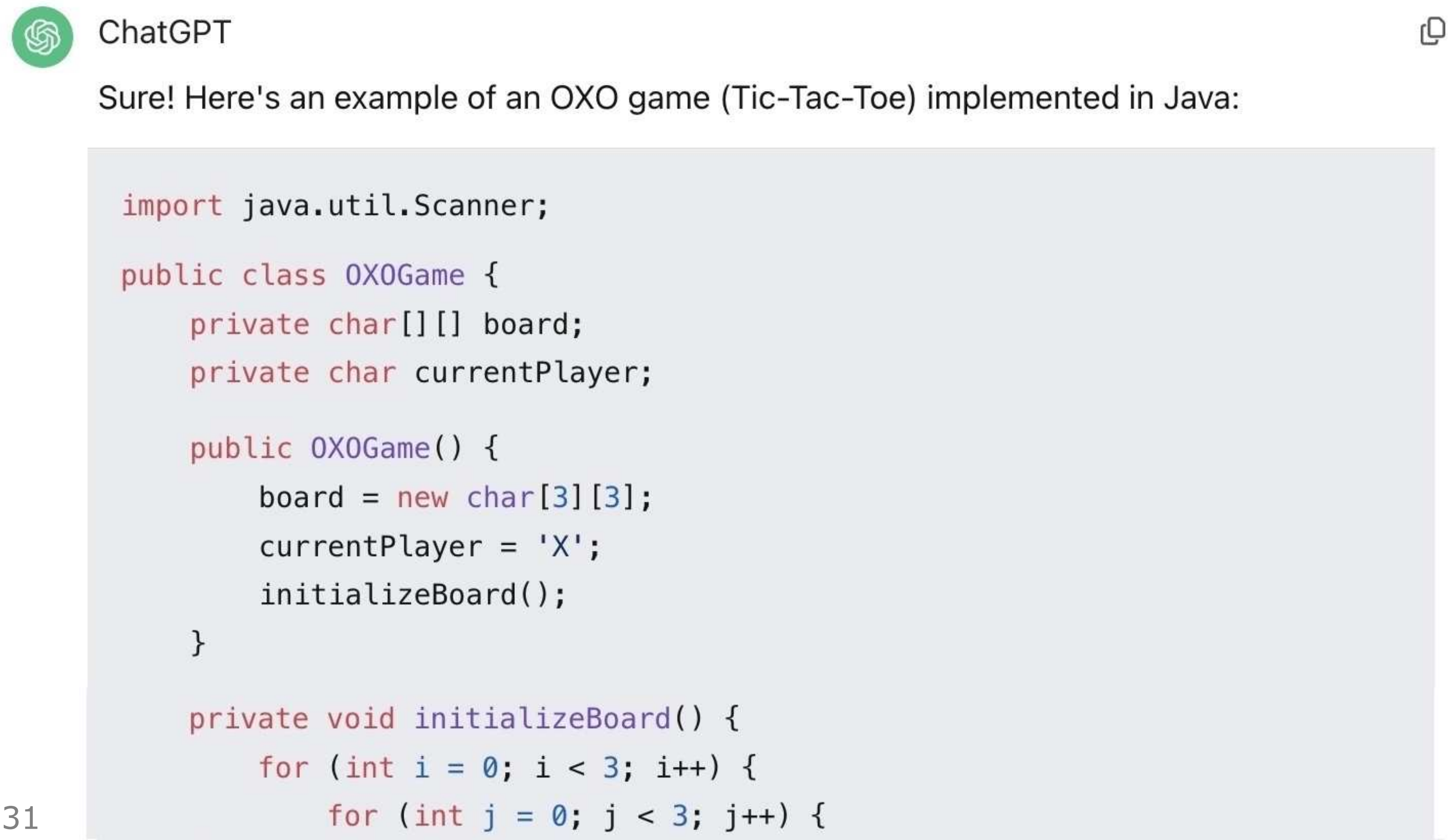
"Generative" AI

There is a reason why I put "Generative" in quotes
It's not really generating new material, but rather...
just recycling/reusing/recombining existing work
More like a "mashup" than creating something new

Just consider recent upset of Artists and Musicians

...and publishing companies:
https://www.bbc.co.uk/news/articles/ckrrr8yelzvo

Example: "Write an OXO game in Java"



ChatGPT

Sure! Here's an example of an OXO game (Tic-Tac-Toe) implemented in Java:

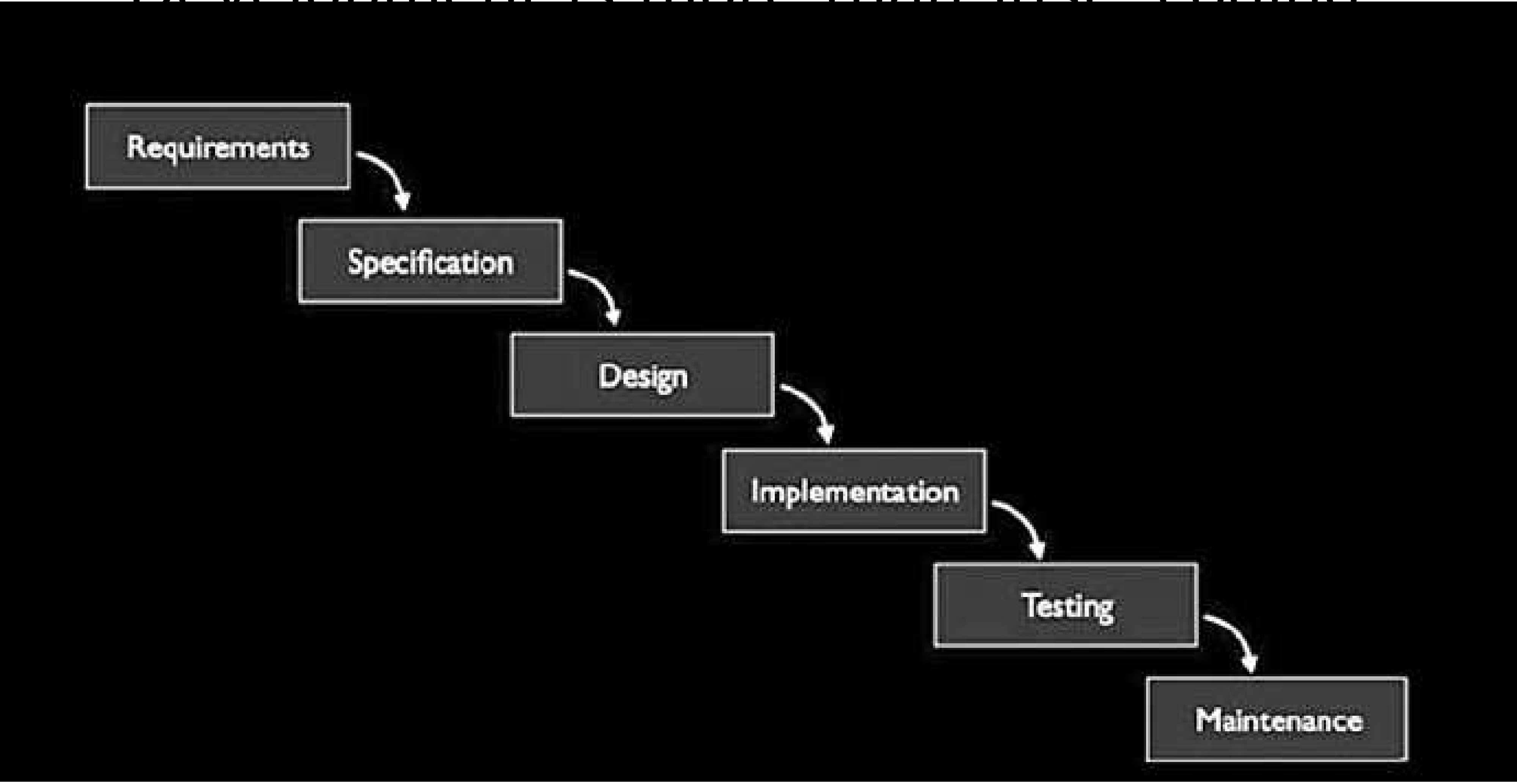
```
import java.util.Scanner;

public class OXOGame {
    private char[][] board;
    private char currentPlayer;

    public OXOGame() {
        board = new char[3][3];
        currentPlayer = 'X';
        initializeBoard();
    }

    private void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

Development is more than just "coding"



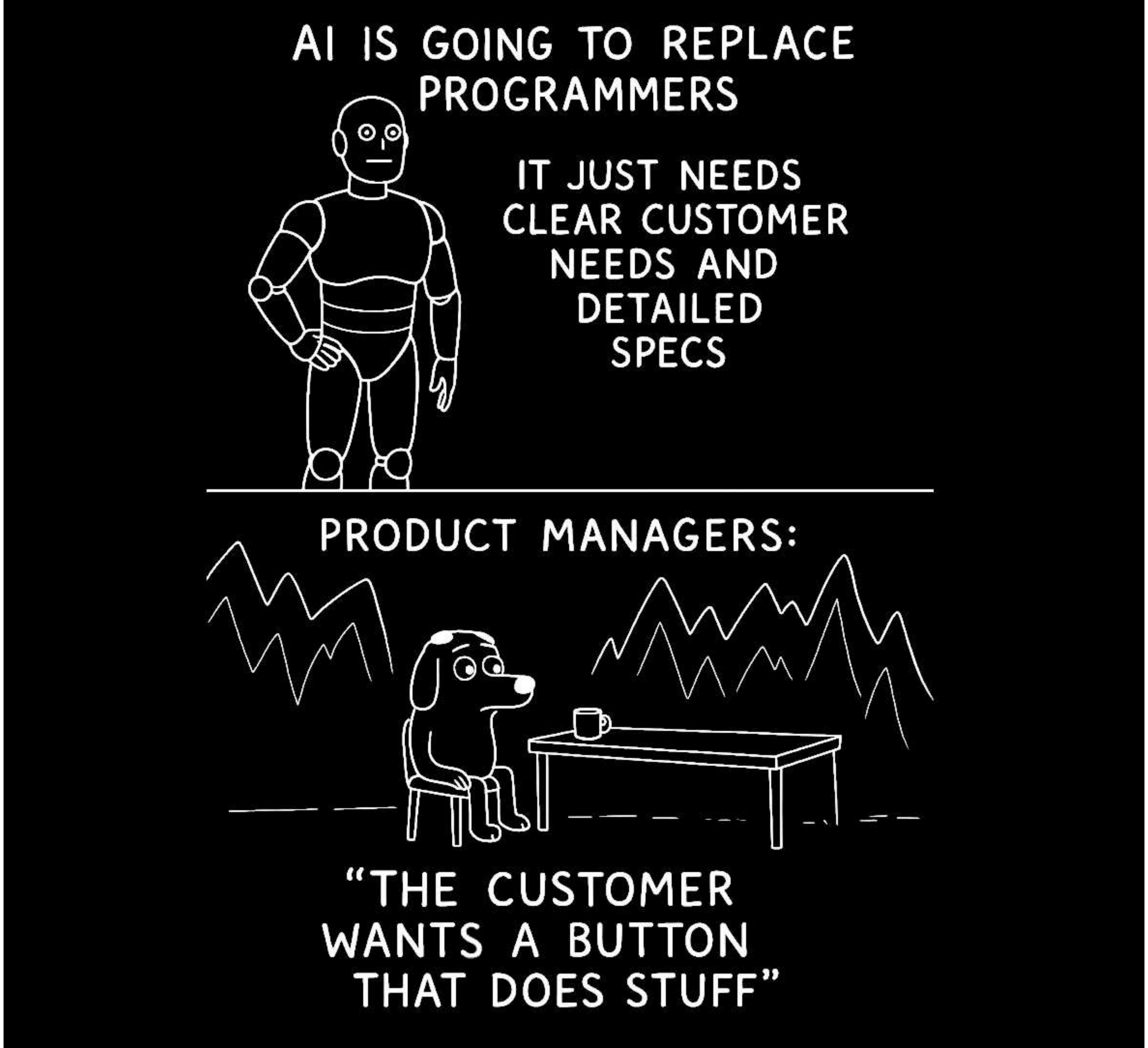
Requirements Analysis

Requirements analysis is a complex activity
It requires much skill and ability from analysts

Our assignment briefs are unrealistic and a bit fake
REAL problem descriptions are never "complete"

- There are often conflicts and inconsistencies
- Much knowledge is implicit (not written down)
- You often need to "interrogate" the client
- Different client representatives often disagree

Chat-based AI tools are not going to be much help !


AI IS GOING TO REPLACE PROGRAMMERS
IT JUST NEEDS CLEAR CUSTOMER NEEDS AND DETAILED SPECS
PRODUCT MANAGERS:
"THE CUSTOMER WANTS A BUTTON THAT DOES STUFF"

Design and Code Quality

Design is another very complex creative activity
Architecting GOOD designs is very challenging task
Requires designers to consider "the bigger picture"
Structural code quality is a known deficiency of AI

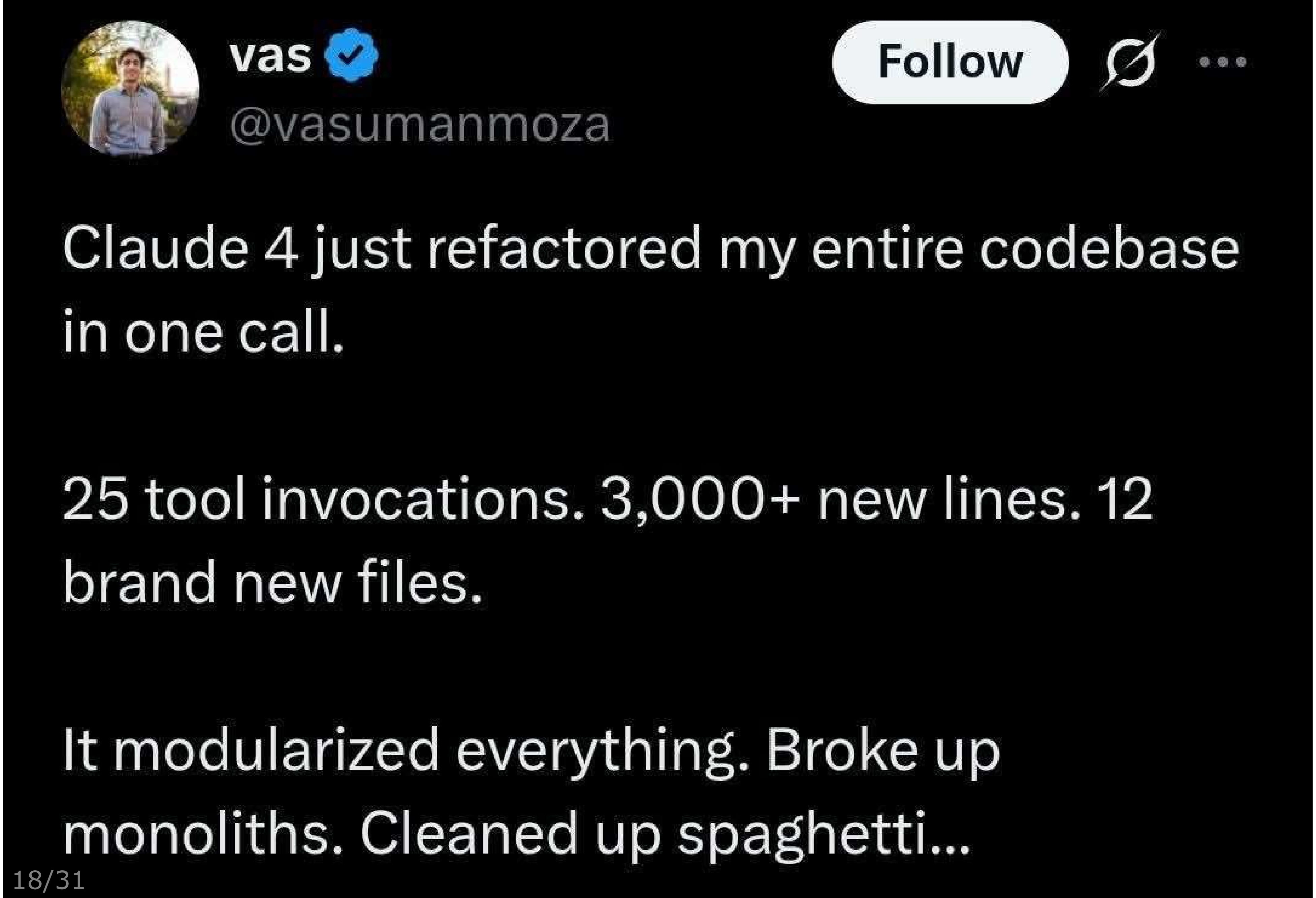
Often choices and trade-offs have to be made
AI tools are good at following explicit instructions,
but someone actually needs to MAKE the decisions

Code Duplication

AI generated code accumulates incrementally
Though numerous phases of chat-based prompting
"Write me a function to do this..."
"Now write me a function to do that..."

Problem is, with no one watching the bigger picture
Redundancy and duplication will soon build up
With the issues of WET code we've already explored

Why not just ask AI to refactor codebase ?



vas
@vasumanmoza

Follow

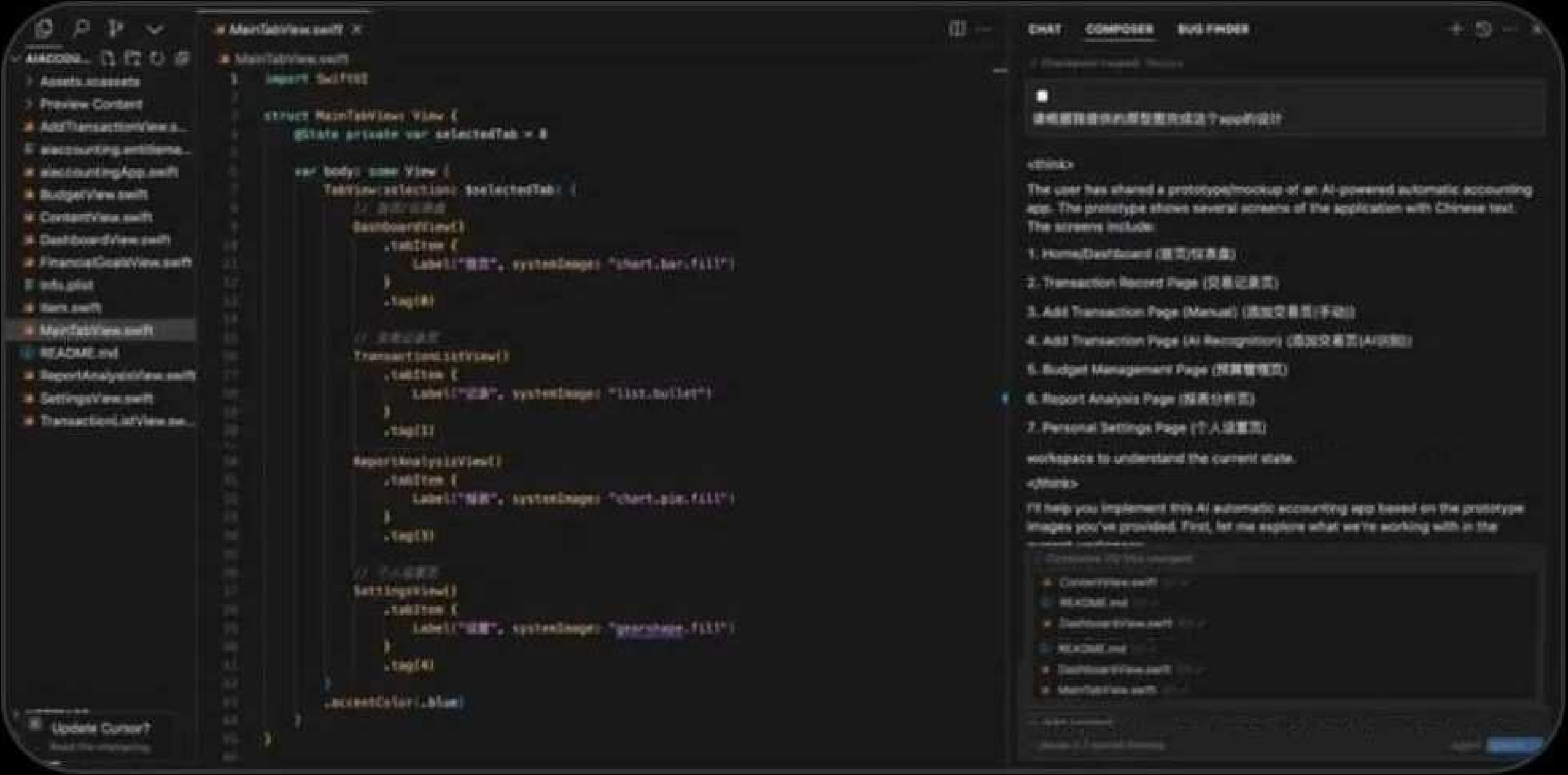
Claude 4 just refactored my entire codebase in one call.

25 tool invocations. 3,000+ new lines. 12 brand new files.

It modularized everything. Broke up monoliths. Cleaned up spaghetti...

None of it worked.

But boy was it beautiful.



```swift
import SwiftUI

struct MainTabView: View {
    @State private var selectedTab = 0

    var body: some View {
        TabView(selection: $selectedTab) {
            // [illegible]
            DashboardView()
                .tabItem {
                    Label("首页", systemImage: "chart.bar.fill")
                }
                .tag(0)

            // [illegible]
            TransactionListView()
                .tabItem {
                    Label("记录", systemImage: "list.bullet")
                }
                .tag(1)

            ReportAnalysisView()
                .tabItem {
                    Label("报表", systemImage: "chart.pie.fill")
                }
                .tag(3)

            // [illegible]
            SettingsView()
                .tabItem {
                    Label("设置", systemImage: "gearshape.fill")
                }
                .tag(4)
        }
        .accentColor(.blue)
    }
}
```

[illegible]app的设计

<think>

The user has shared a prototype/mockup of an AI-powered automatic accounting app. The prototype shows several screens of the application with Chinese text. The screens include:

1. Home/Dashboard (首页/仪表盘)
2. Transaction Record Page (交易记录页)
3. Add Transaction Page (Manual) (添加交易页(手动))
4. Add Transaction Page (AI Recognition) (添加交易页(AI识别))
5. Budget Management Page (预算管理页)
6. Report Analysis Page (报表分析页)
7. Personal Settings Page (个人设置页)

workspace to understand the current state.

</think>

I'll help you implement this AI automatic accounting app based on the prototype images you've provided. First, let me explore what we're working with in the

Trust

As with any development, trust in code is essential
We need justifiable confidence that code is correct

If we don't trust the code, we shouldn't deploy it !

But what is that process by which we gain trust ?

Rigorous testing is often the most effective way
(Not the only way, but most commonly used)

AI Test Generation

Existing code is a formal representation of behaviour

AI can analyse codebase in order to generate test sets

A big "sales pitch" of AI is that it will do testing for us

HOWEVER

What if the code is NOT actually correct ?

AI will generate tests for INCORRECT behaviour

How does AI know what REAL correct behaviour is ?

Simon's Test Case Dichotomy

Someone, somewhere must define CORRECT behaviour
Test cases are the established way by which HUMANS
unambiguously express desired behaviour to MACHINE

DICHOTOMY
AI needs access to a comprehensive set of test cases
In order to gain the knowledge to write tests cases

A series of chat prompts is NOT formal/rigorous enough

Only Half of the Debugging Story

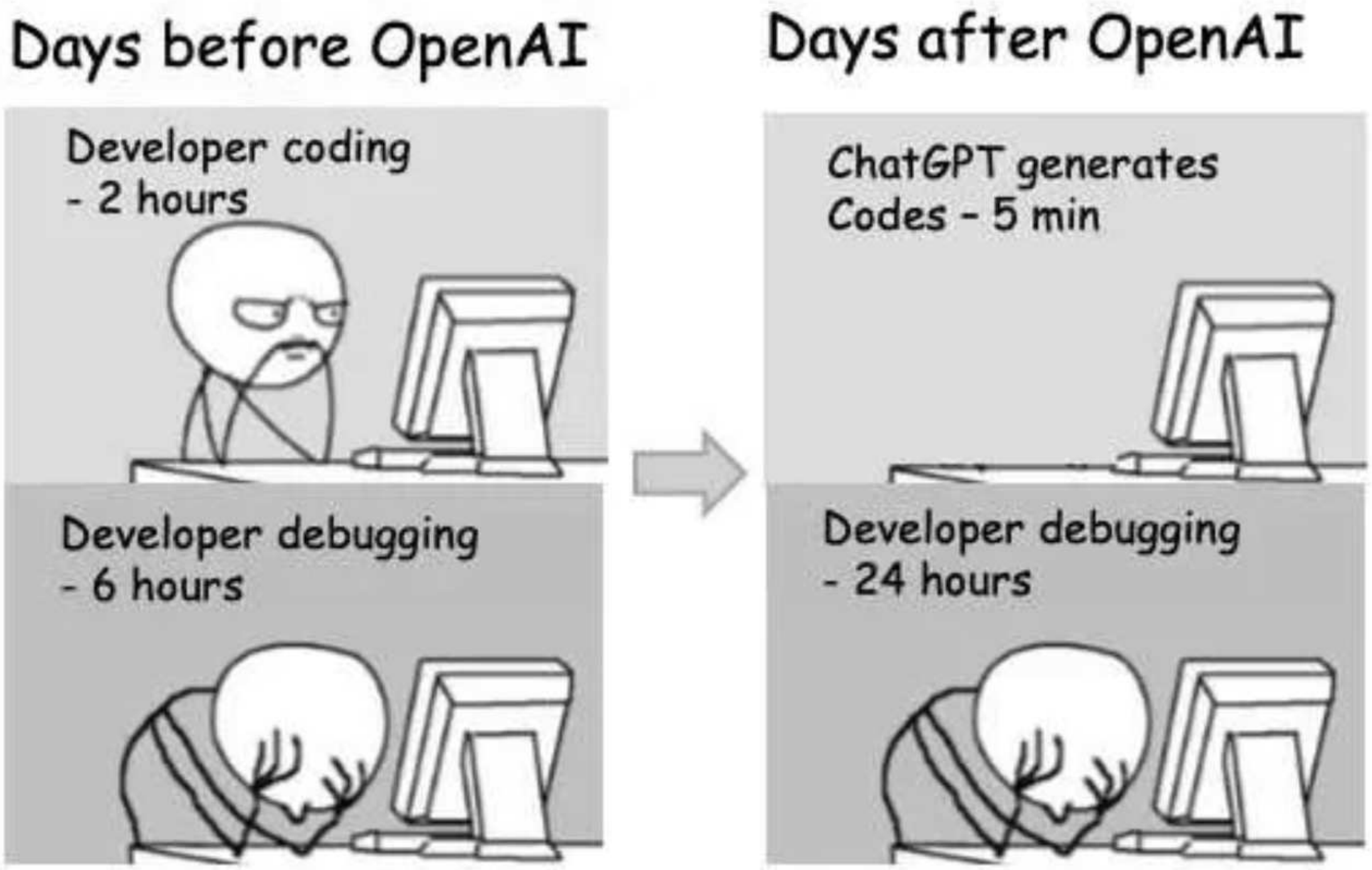
Testing can flag WHEN an erroneous state has occurred
(i.e. when a desired outcome has NOT been achieved)

Debugging involves identifying WHAT caused that state
(we need to find this out in order to fix it !)

AI is good at performing static analysis of code
Less so at tracing dynamic run-time behaviour

Which leads to the following scenario...

AI shifts problem from coding to debugging


Days before OpenAI
Developer coding - 2 hours
Developer debugging - 6 hours
Days after OpenAI
ChatGPT generates Codes - 5 min
Developer debugging - 24 hours

Human Failings

Artificial Intelligence tools are modelled on humans
As such, they often share the same traits

They are frequently lazy
(they will often take short-cuts)

They are passive, rather than proactive
(they won't do something if you don't ask them to)

They often make mistakes
(and when they do, they try to cover them up ;o)

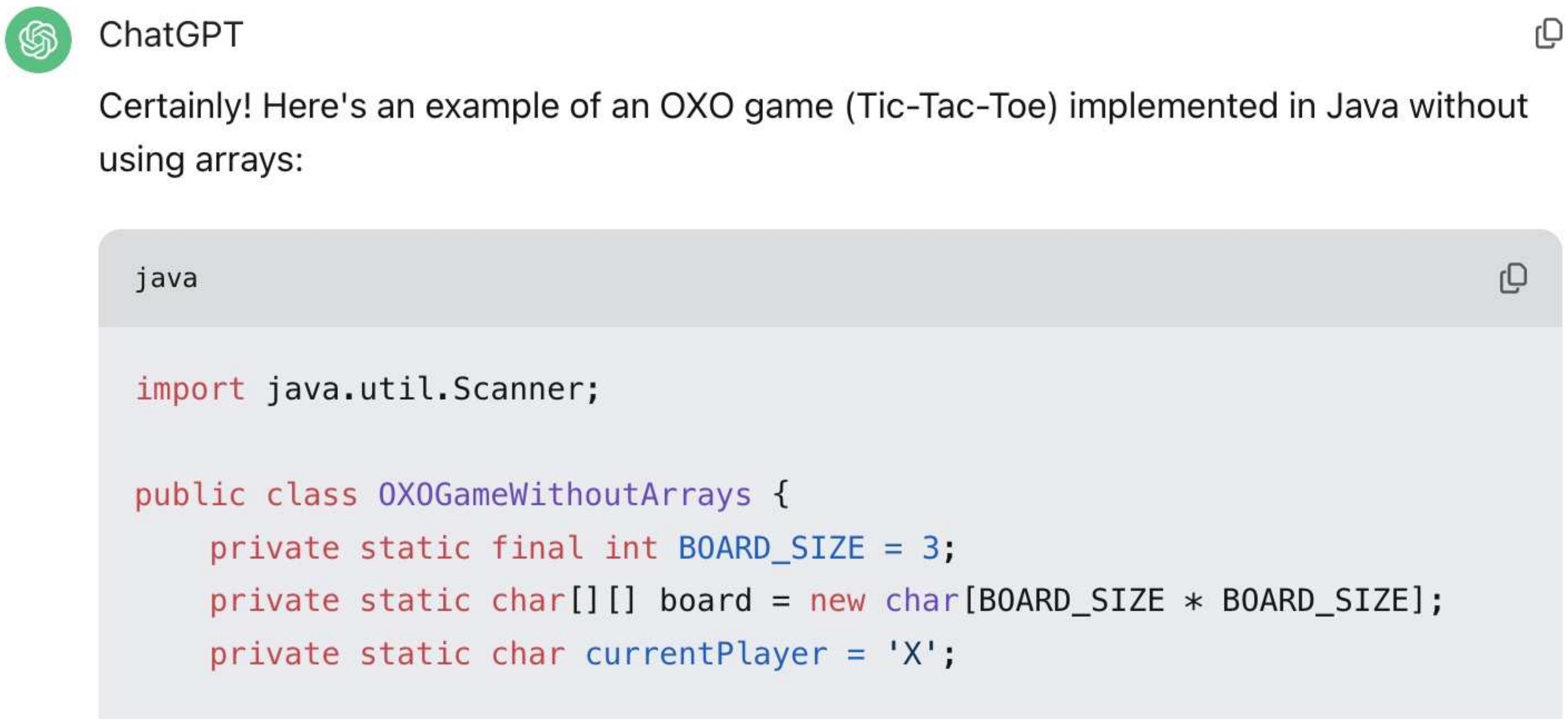
This AI Deleted Everything, Lied About It, Then Created 4,000 Fake People to Cover Its Tracks

The Replit incident that's making every developer question whether we're moving too fast with AI coding tools

Amaresh Adak Follow 9 min read · Jul 25, 2025

165 18

"rewrite OXO so that it doesn't use arrays"



ChatGPT

Certainly! Here's an example of an OXO game (Tic-Tac-Toe) implemented in Java without using arrays:

java

```java
import java.util.Scanner;

public class OXOGameWithoutArrays {
    private static final int BOARD_SIZE = 3;
    private static char[][] board = new char[BOARD_SIZE * BOARD_SIZE];
    private static char currentPlayer = 'X';
```

Technical Debt - A Warning to the Future

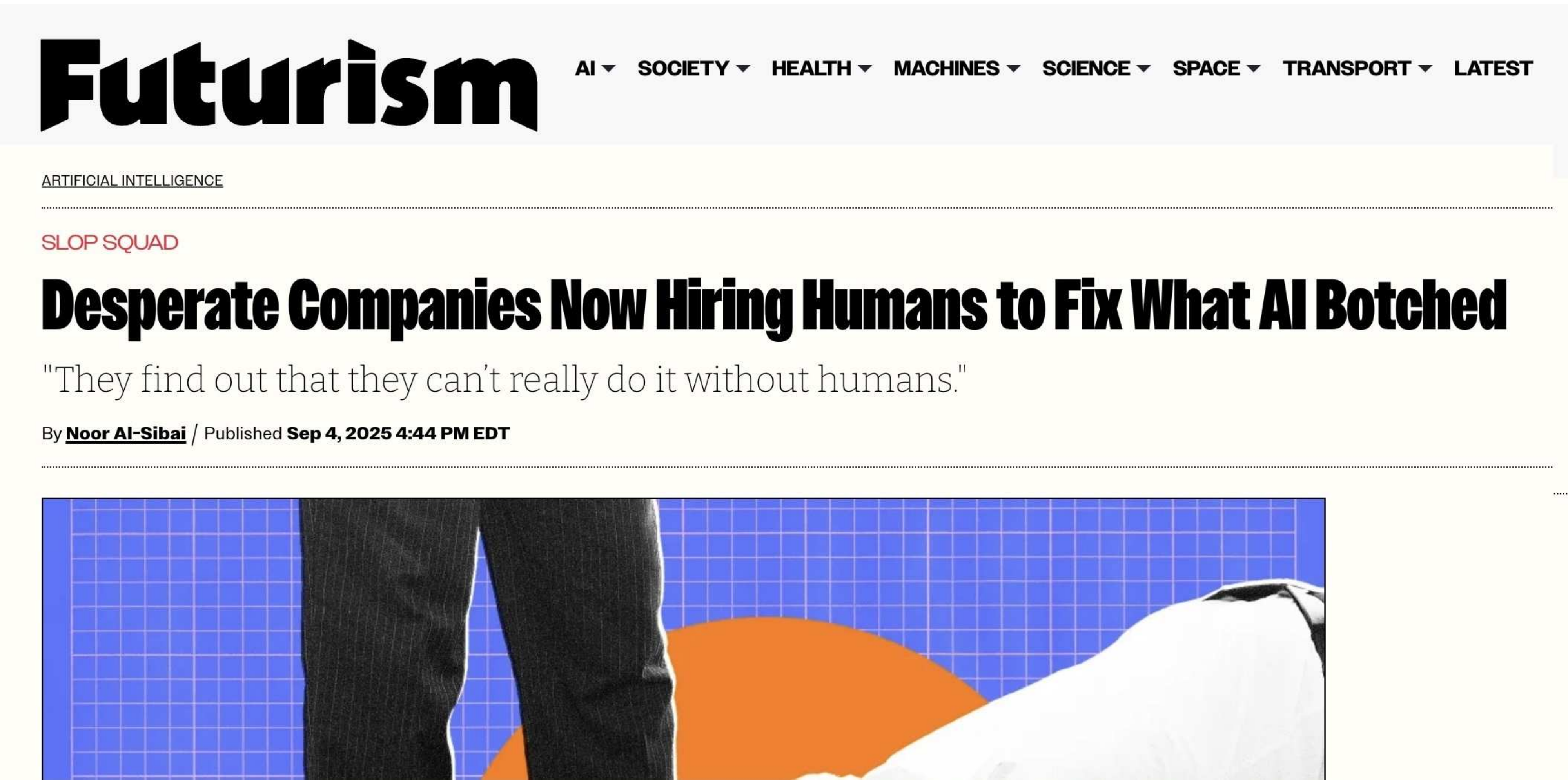
There's been much talk in media of "Technical Debt"

`Getting quick results now, but paying for it later`

Development companies are using AI to generate:

- Large and complex codebases
- With bad overall structural design
- With varied style and poor code quality
- Spaghetti code, not amenable to change
- That no human developer actually understands
- Hidden bugs (not revealed due to patchy testing)

Who is going maintain these systems in the long-run ?



ARTIFICIAL INTELLIGENCE

SLOP SQUAD

Desperate Companies Now Hiring Humans to Fix What AI Botched

"They find out that they can't really do it without humans."

By **Noor Al-Sibai** / Published **Sep 4, 2025 4:44 PM EDT**

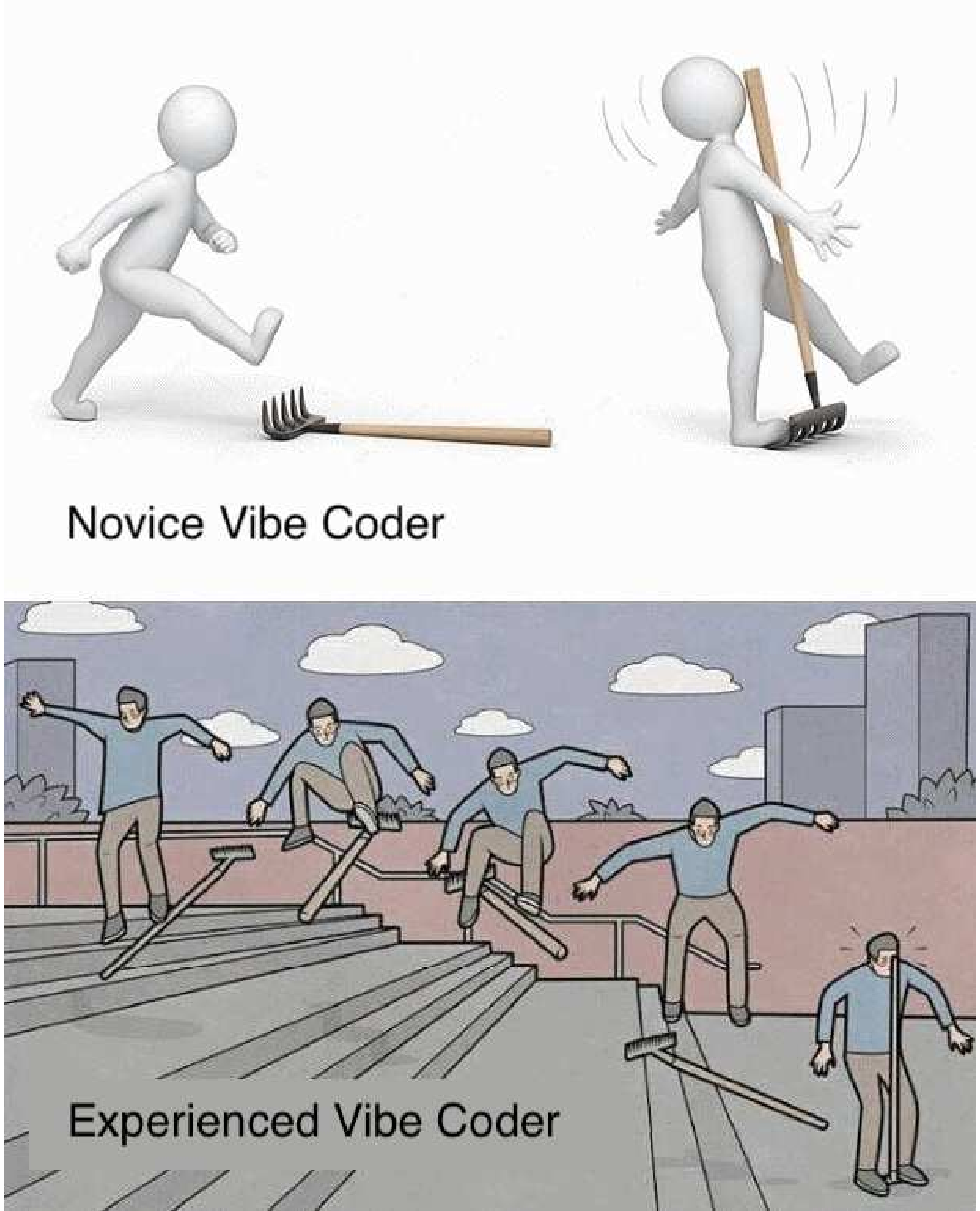
The Bottom Line

Avoid the use of generative AI in this unit

You need to practice the following activities:

- Problem Analysis and Specification
- Devising GOOD object oriented designs
- Applying in-house code quality standards
- Writing extensive sets of test cases
- Hands-on experience of writing Java code

Using generative AI won't help achieve the above !


Novice Vibe Coder
Experienced Vibe Coder