Agentic Programming -

A Compiler Case Study

A Brief Intro to Agentic Programming

Jian Weng

CEMSE, KAUST

Week-1 Session-1

Agentic Programming for Compilation

~~(class renamed for 40-char limit in system)~~

A Brief Intro to Agentic Programming

Jian Weng

CEMSE, KAUST

Week-1 Session-1

Agenda

- **Course goals & expectations**
- Why compilers
- Tooling history
- Agent workflow + reliability
- Context setup habits
- Wrap-up & Q&A

Course Policy

- No attendance enforcement
- It is **my duty** to keep the instruction important enough so that you all come
- It is **your duty** to come to the class for learning
- It is **your duty** to catch up after absence

AI / LLM Usage

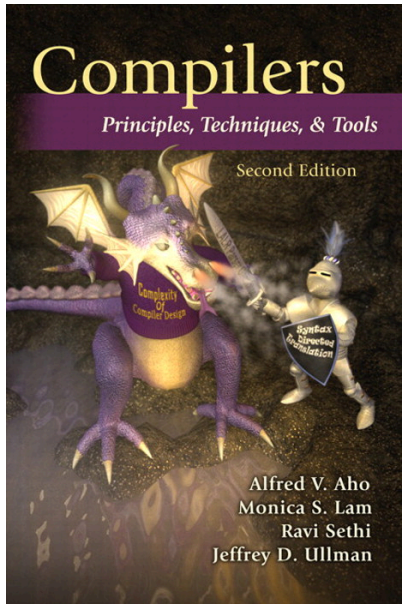
- You **SHOULD** use them as much as possible — reliably
- This class will teach you **how**
- The load of this class is **UNACHIEVABLE** without their help

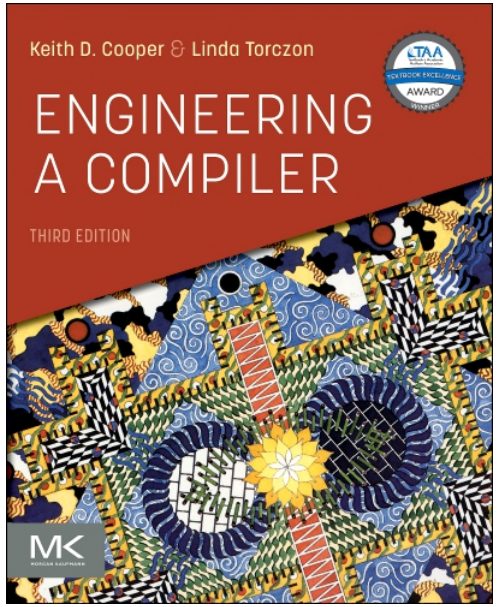
Grading Policy

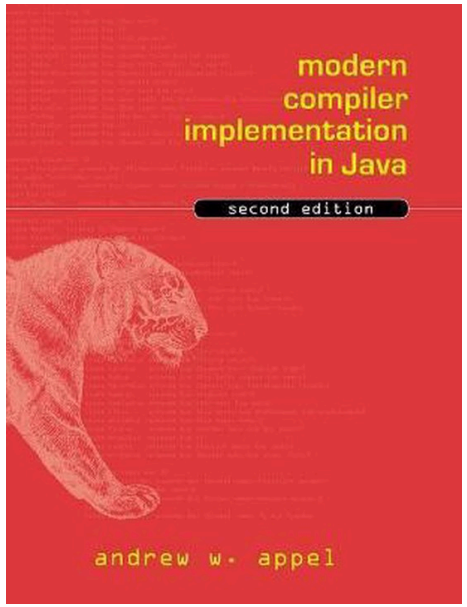
- We do not have final exams
- Grading is purely based on the project checkpoints
- Checkpoint 1 (10%): Setting up the agent structure
- Checkpoint 2 (20%): Use agent to build the parser
- Checkpoint 3 (30%): Generate LLVM intermediate representation
    - Here to get a non-fail score
- Final submission (40%): Based on your passing test cases
- I will curve the scores based on the highest score

Goal: Writing a Compiler using AI Agents

- You will be an expert in:
  - **AI agent development**
  - **Compiler engineering**







Agenda

- Course goals & expectations
- **Why compilers**
- Tooling history
- Agent workflow + reliability
- Context setup habits
- Wrap-up & Q&A

Why Compilers?

- Well-defined, well-studied project
- Strong, intensive engineering effort
- My hardest undergraduate class
- My first experience managing:
  - Thousands of lines of code
  - A real system

Agenda

- Course goals & expectations
- Why compilers
- **Tooling history**
- Agent workflow + reliability
- Context setup habits
- Wrap-up & Q&A

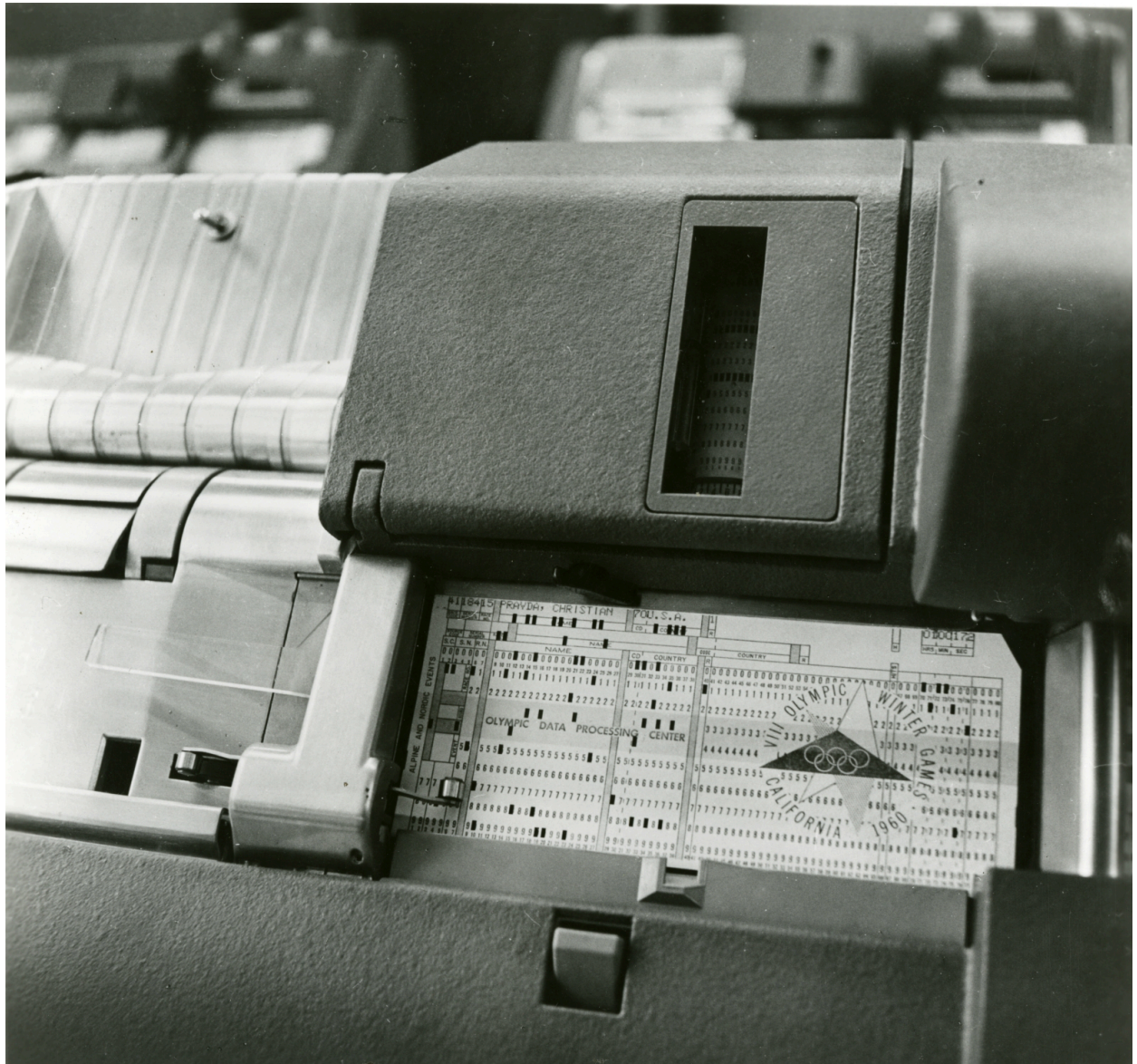
A History of Ergonomic Programming

- What do you use today?
  - IDE: VSCode? Cursor?
  - Command-line: Emacs? Vim?
- How do you write code?
  - Auto-completion?
  - Copilot? *tab tab tab*
- How do you run code?

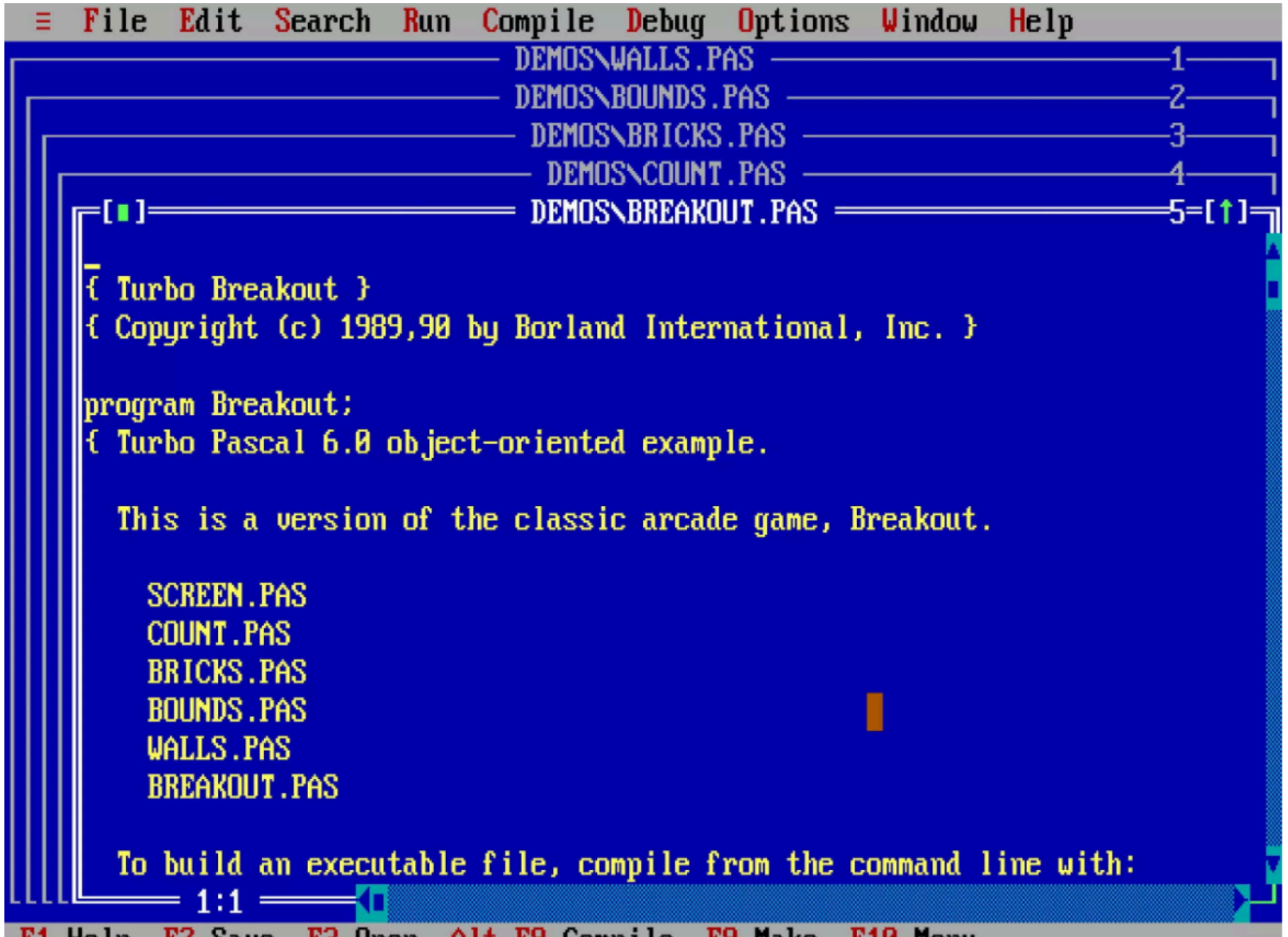
> I loved command-line editors — seamless development & execution.

Did you ever use one of these before?

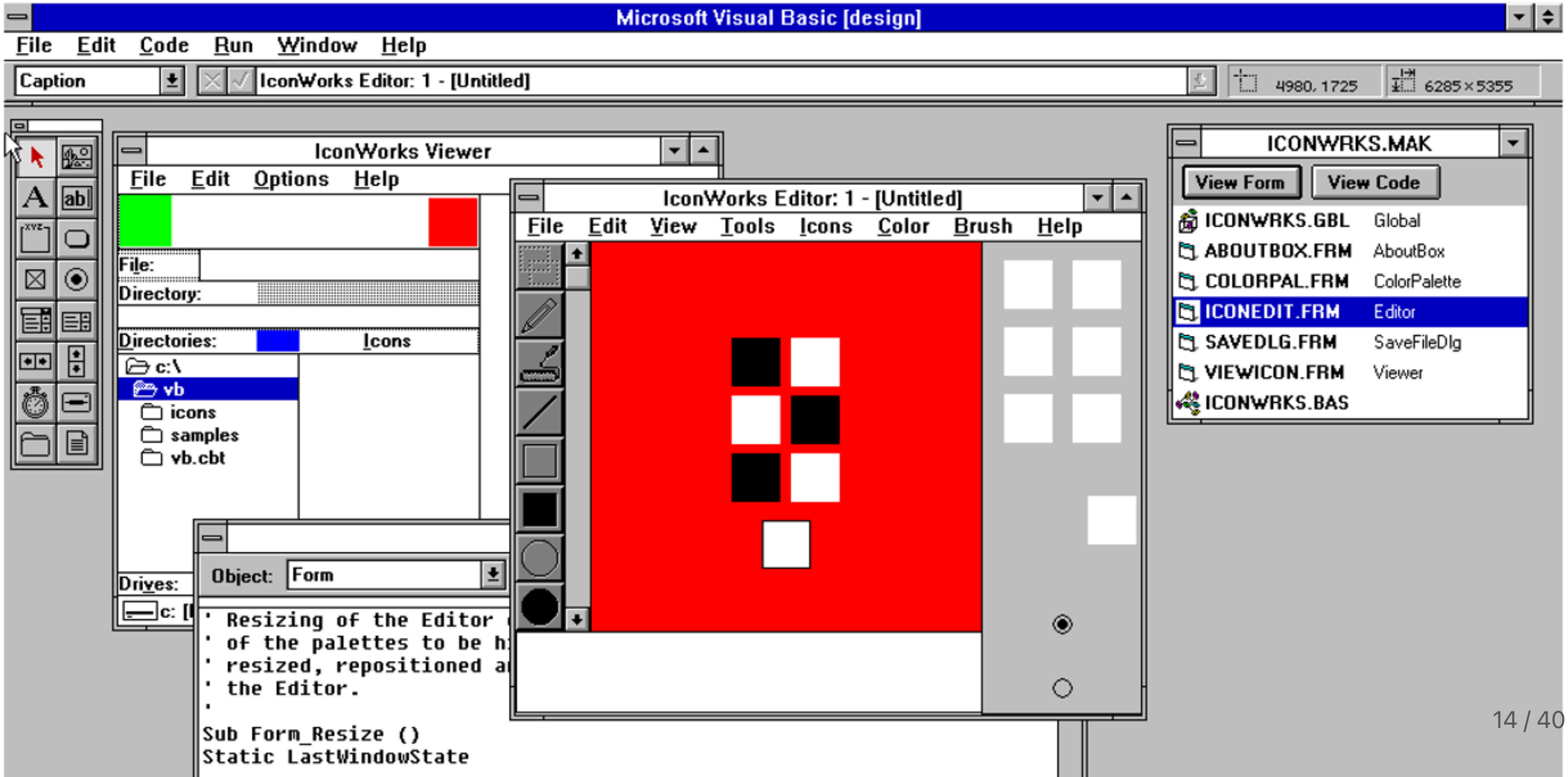
Punch the Tapes



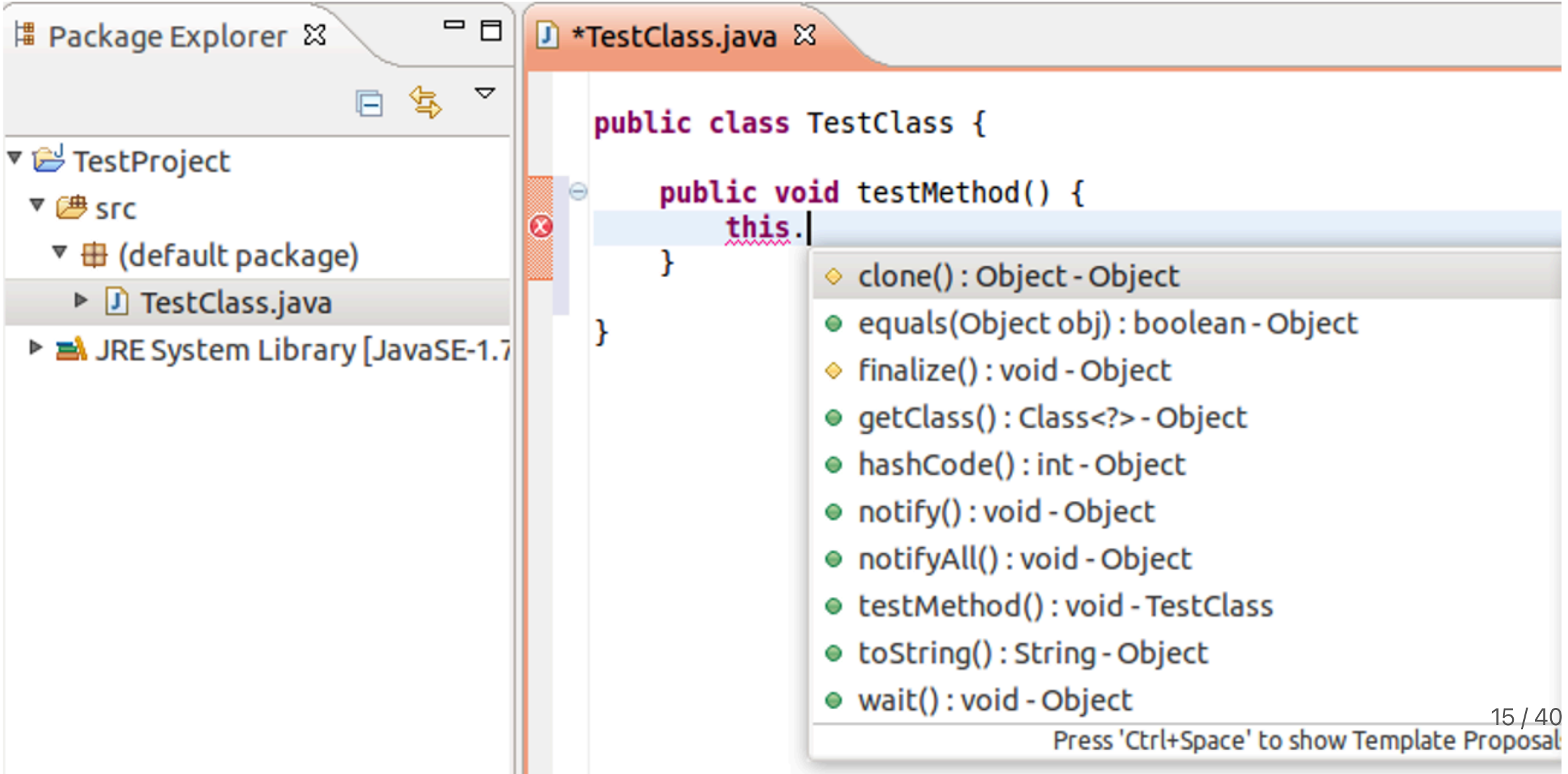
Turbo Pascal (1983)



Visual Basic (1991)



Eclipse (2001)



Copilot (2021)

```
from __future__ import print_function
import argparse
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

class Sequence(nn.Module):
    def __init__(self):
        super(Sequence, self).__init__()
        self.lstm1 = nn.LSTMCell(1, 51)
        self.lstm2 = nn.LSTMCell(51, 51)
        self.linear = nn.Linear(51, 1)
```

Syllabus

This class teaches how to:

- Use **AI agents** to efficiently develop & manage a project
- Engineer a **compiler**

> First time teaching — content may evolve
>
> Core focus remains unchanged

This is NOT a class for vibe coding

vibe 1 of 3 noun (1)

ˈvīb

plural vibes ˈvībz

Synonyms of *vibe* >

informal

: a distinctive feeling or quality capable of being sensed

> This place has a good/bad *vibe*.

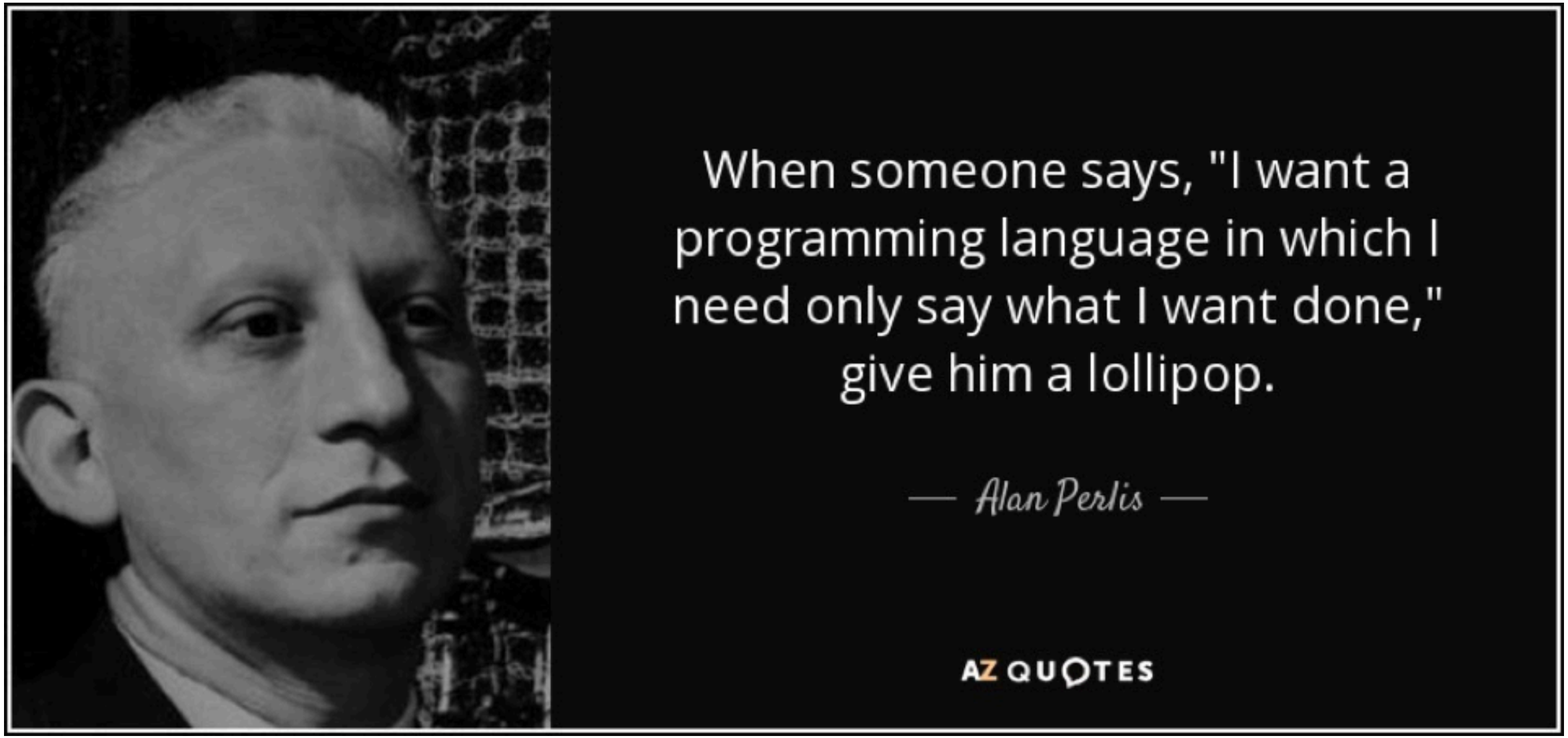
> She gave me a weird *vibe*. = She gave off a weird *vibe*. = I got a weird *vibe* from her.

> Though zebra print looks positively of the moment, it also conveys a groovy retro *vibe*.
> – Liana Satenstein and Madeline Fass

> He seems to be in on every conversation, every deal, every *vibe* that is winging through the room.
> – Albert Goldman

This is NOT a Class for “Vibe Coding”

- Do **not** abuse your feelings
- If you do not know what you want:
  - Neither does AI
- AI is only as good as **you**
- AI agents are:
  - A *team of you*
  - For coding, documentation, and testing
- Heavy engineering can be done **all-in-one**



- We are approaching this, but not 100% there yet.

Agenda

- Course goals & expectations
- Why compilers
- Tooling history
- **Agent workflow + reliability**
- Context setup habits
- Wrap-up & Q&A

How is agentic programming architected?

[Prompt]

> Write a loop in C that sums up the given array and return the value.

[CLI / Tools]

- **Claude Code/Cursor/Codex**

[Models]

- Haiku (low; cheap); Sonnet (medium; capable); Opus (clever; expensive)
- Note: Check latest version numbers before class (e.g., Opus 4.1, Sonnet 4, Haiku 3.5)

How do we use it? (CONT'D)

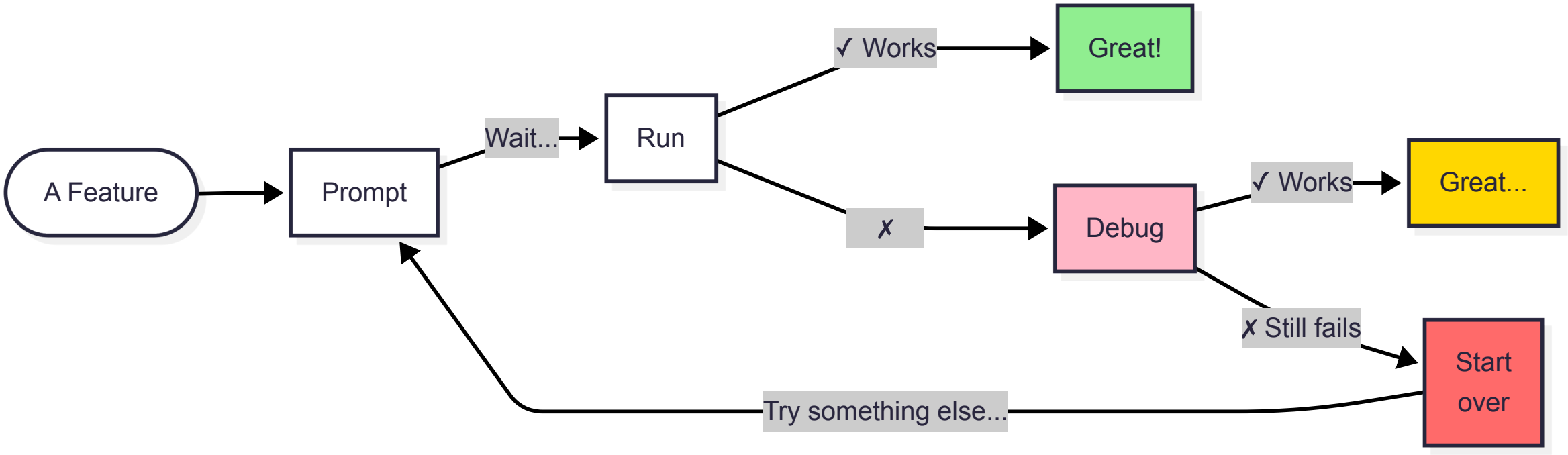
I believe this is what you do with AI agents every day:

- Type in the prompt:

> I want something, please implement it for me.

- Wait for it to finish...
- Run the code
    - It works! Great!
    - It fails! Start an infinite battle with prompts to debug
        - It works finally... It is still great?
        - It still fails... Start over with a new round of battles

How do we use it? (CONT'D)



- Is it really a good way of using AI agents?
- What is your key pain of using AI agents?

Pain of using AI Agents - 1

- Do you remember slide 3?
  - Use AI agents as much as possible **in a reliable way**
- An honest question:
  - No matter who wrote it (either you or AI agent), how much courage do you need to run the code after writing?
- **Key pain**: Unreliable results
  - Hallucinated code?
  - Buggy code?

Lemma 1: Code is wrong until tested

- Agent is actually proactive, it will try to fix the code until it is correct.
- But how does it know the code is correct?
  - Write test cases!
  - Run test cases!
- Leave the pain of fixing bugs to the agent!
  - No worries, it won't complain!
  - It is your wallet to complain (maybe)
- What if the bug cannot be fixed? (Later on this)
  - Maybe as late as weeks later...

Pain of using AI Agents - 2

- Pre-AI era: I want to start a new project
  - but where are the project templates?
- AI-agent era: I need to start from somewhere,
  - but from where?
- **Key pain**: Setting up the context
  - Which file should we hack to implement it?
  - How to run the test cases to verify the change?
  - Further, project-wise, does the implementation make sense?

Wrong Solution: Let the AI agent figure it out

Prompt:

> Read the codebase to understand the project and find where to implement the new feature X and have tests implemented and run to verify the implementation.

- Too long context to read
- Too general to understand the goal
- 👉 Bad effects of achieving the goal
  - Hard to understand the goal
  - Fails to find the right files

Diverge a little bit...

How do LLMs work? (Simplified)

- LLMs predict the next token based on the context they've seen
- Context is stored in a memory (KV-cache) and used for predictions
- As context grows, predicting accurately becomes harder and harder

Key takeaway:

- **Don't make the AI guess the context from scratch**
- **Set up the context properly** to get reliable results

But I do not want to do it repeatedly 😭

Backup: How LLM works (Technical Details)

Do you know how an LLM works?

- Transformers, attention mechanism, tokens, embeddings, etc.
- T: An input token.
- $Q := W_Q T, K := W_K T, V := W_V T$: QKV linear projection.
  - Append the new K, V to a stateful memory
- Attention: $\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$
- FFN Layer: Projections + Non-linear activation to generate new tokens to next transformer blocks.

Backup: How LLM works (CONT'D)

We can see that

1. K, V are the key to remember the context;
2. swapping model weights is expensive;
3. KV-cache to synthesize a single new token to reflect a new context.

Predicting the next token is unreliable when context grows.

- Do not guess the context from scratch!
- Set up the context properly!

Agenda

- Course goals & expectations
- Why compilers
- Tooling history
- Agent workflow + reliability
- **Context setup habits**
- Wrap-up & Q&A

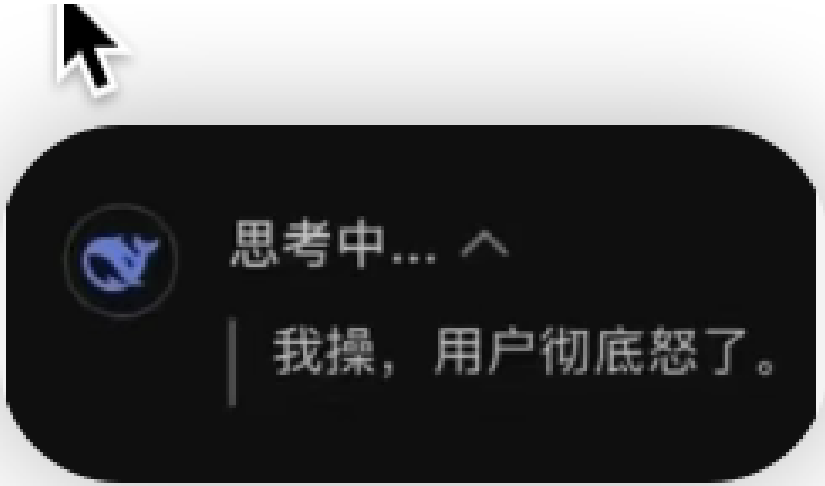
Setting up the context

- `CLAUDE.md` or `AGENTS.md` file
  - Project overview
  - How to setup the project
  - How to run the test cases
  - General design decisions and architecture of this project

> NOTE: These two files are the MOST precious files for your context
>
> Keep as much important information there as possible!

Chain of Thoughts (CoT)

- What stood out about DeepSeek?
  - Low-cost training?
  - **An exposed chain of thought**
    - Damn it! The user is angry with me!



Chain of Thoughts (CoT) (CONT'D)

- CoT helps answer quality by
  - Breaking down a complex problem into smaller steps
  - Letting the model reason through each step sequentially
- Setting up the context properly is like
  - A manually expanded CoT for the AI agent
  - **Key idea:** When you need help, be more specific to what you want!
    - If you do not know what you want, neither does the AI agent!

Agenda (Backup)

- Course goals & expectations
- Why compilers
- Tooling history
- Agent workflow + reliability
- Context setup habits
- **Backup: More on context setup**

Lemma 2: LLMs are few-shot learners

- How do kids learn recognizing things?
  - This was an example when I was learning ML nearly 10 years ago.
  - John Hopcroft told me this story:
    - His grandson had a book called *The Best Word Book Ever*
    - His grandson never saw a firetruck before
    - But one day on the road, he called out "A firetruck!"
    - He learned this by seeing even one example in the book!
- **Key idea:** Provide few-shot examples in the context to help the AI agent understand what you want!

But, do not overdo it!

When are few-shot examples required?

- ✅ Write a test case
- ✅ Following project coding style
- ❌ Implement a new algorithm
- ❓ Implement a new feature
  - It depends on how "new" the feature is.
  - If it has a similar and unified interface design, yes.
  - If it is a breaking change, no.

A New Programming Paradigm

- **DO NOT** tell AI what to do
- **Document** the purpose, design, and architecture of your project well
  - Then, ask AI to read the documents to understand what you want
  - Optionally, provide few-shot examples to help AI understand better
- **LET AI FIGURE OUT** how to implement it

Wrap-up

- Have your LLM agents installed and ready
  - `claude` is preferred
    - It is the 1st coding CLI and defined the most standards
    - All the following class will stick to `claude` CLI
  - `codex` is strong but not that transparent
  - I personally do not like `cursor`

References

- Stanford CS146
- Claude Doc Collection