Introduction to Programming

Week 1

Magnus Madsen

Week 1: Your first program

Monday

- Course formalities
- Software is everywhere

Thursday

- Getting started with Java
- Live programming

Prologue

Quote of the week

"To know a second language, is to have a second soul."

— Charlemagne

Epigram of the week

"To understand a program you must become both the machine and the program."

— Alan Perlis

Course formalities

A new course

Introduction to Programming (EN, 133285):

- A **brand new course** running for the first time in fall 2025.
- Focused on the fundamentals of programming (and less focused on Java).
- All materials and lectures in English.

(Will replace the Danish course in 2026.)

Staff

Lecturer: Assoc. Prof. **Magnus Madsen** (`magnusm@cs.au.dk`)

Class	Time	Room	Teaching Assistant (TA)
CS1	Tue and Thu 12–14	5125-423 and 5221-012	Maria S. Nielsen
CS2	Tue and Thu 12–14	5342-020 and 1531-019	Simon Olesen
CS3	Tue and Thu 12–14	5335-184 (both days)	Valdemar T.T. Knudsen
ITE	Wed and Fri 08–10	5125-120 and 5520-112	Tayo K. Krüger
ITE	Wed and Fri 08–10	5125-120 and 5520-112	William M. Vestergaard

Staff

Maria

Simon

Valdemar

Tayo

William

Heili

Schedule

Lecture every **Monday** at **14.15-16.00** in 1170–347

Lecture every **Thursday** at **08.15-10.00** in 1170–347

Deadline for mandatory hand-ins: decided by each class and TA.

Mandatory hand-ins

From the course description:

> **i** Students must submit a total of 10 weekly assignments which must be approved.

- The assignments are individual, but you may work together in small groups.

Mandatory hand-ins

The hand-ins are subject to the following requirements:

- The hand-ins must be submitted on Brightspace before the deadline.
- The TA will correct the hand-in, give feedback, and mark **PASS** or **FAIL**.
  - If a hand-in is **failed**, you must resubmit a hand-in to address the feedback.
- Hand-ins must be submitted on time.
  - Revised hand-ins must be submitted within a week of receiving the feedback.

If you cannot submit on time, you must **proactively** e-mail the TA with an explanation.

> **i** The hand-ins have no impact on the final grade. They simply have to be approved.

Exam

Exam

The exam consists of two parts:

i **Part A:** An individual take-home programming project. The exam is open-book, i.e. students may use all materials available except Generative AI. Students may discuss the project with each other, but may not share any source code.

i **Part B:** An individual written exam. The exam is closed-book, i.e. students may not use any materials. The scope of the written exam is the entire course syllabus plus the take-home programming project.

The two parts take place at different times and places.

- You have three days for the take-home programming project.
- You have two hours for the written exam.

Exam

You obtain **one grade** weighted approximately equally between **Part A** and **Part B**.

The Danish Grading System. The so-called “7-trins skalaen”:

Grade	Danish	English	ECTS
12	Fremragende	Outstanding	A
10	Fortrinligt	Excellent	B
7	Godt	Good	C
4	Jævnt	OK	D
02	Tilstrækkeligt	Adequate	E
00	Utilstrækkeligt	Inadequate	Fx
−3	Ringe	Poor	F

cf. *https://ufm.dk/en/education/the-danish-education-system/grading-system*

How to prepare for the exam

✓ I have read and understood the textbook and other material.

✓ I have solved all weekly exercises and understood them.

✓ I have completed all mandatory hand-ins and understood the feedback from the TA.

✓ I have attended lectures, TA classes, and the study cafe.

✓ Whenever I did not understand something, I asked for help.

If you do these things, you will be successful 🌈 🎉 🥳.

Generative AI

You are probably all familiar with LLMs, e.g. ChatGPT, CoPilot, Gemini, …

You are ***not*** permitted to use GenAI for hand-ins nor for the exam.

You are **permitted** to use GenAI for your **learning experience**.

- You can ask for explanations of concepts.
- You can ask for hints.
- You can ask for help.

If you use them, please use them responsibly.

Freedom, responsibility, and integrity (1/3)

University is not school.

- You have a lot of **freedom**, but you are **responsible for your own learning.**

Actions that are permitted, but *not* recommended:

- Skipping lectures
- Skipping classes
- Skipping exercises
- Not reading the book

Good luck with the exam, but other than that all is fine 👍

Freedom, responsibility, and integrity (2/3)

Actions with **consequences**:

- Not handing in the mandatory hand-ins
- Not handing in the mandatory hand-ins on time (*)

You fail the course ❌

- And you cannot attend the re-exam ❌
- But you can follow the course next year 👍

(*) Unless you proactively contact your TA with an acceptable reason.

Freedom, responsibility, and integrity (3/3)

Actions with **severe consequences**:

- Cheating
  - e.g. a friend does your hand-ins or exam project.
  - e.g. a friendly A.I. does your hand-ins or exam project.
  - ...

💀 **You will be kicked out of the university** 💀

Course description

Course content

Foundational: Understanding what a Java program is, how to compile it, and how to execute it. Reasoning about whether a program is well-formed and about its behavior.

Imperative Concepts: Writing simple methods using local variables, if-then-else, and for- and while loops. Writing simple data structures using plain objects and arrays.

Object-Oriented Concepts: Writing simple classes with encapsulated state, getters and setters, and using interfaces and inheritance.

Programming Techniques: Programming with common data structures, such as lists, sets, and maps. Applying basic debugging and testing techniques to understand how a program behaves. Manipulating the file system, including the creation, reading, and writing of files.

Learning objectives (1/2)

After the course, students will be able to:

- Explain how to write a Java computer program, compile it, and execute it.
- Use elementary imperative programming language constructs, including: primitive data types, local variables, assignment, arrays, if-then-else, and for- and while loops.
- Use elementary object-oriented programming language constructs, including: classes, interfaces, objects, and methods.
- Use common data structures such as lists, sets, and maps.

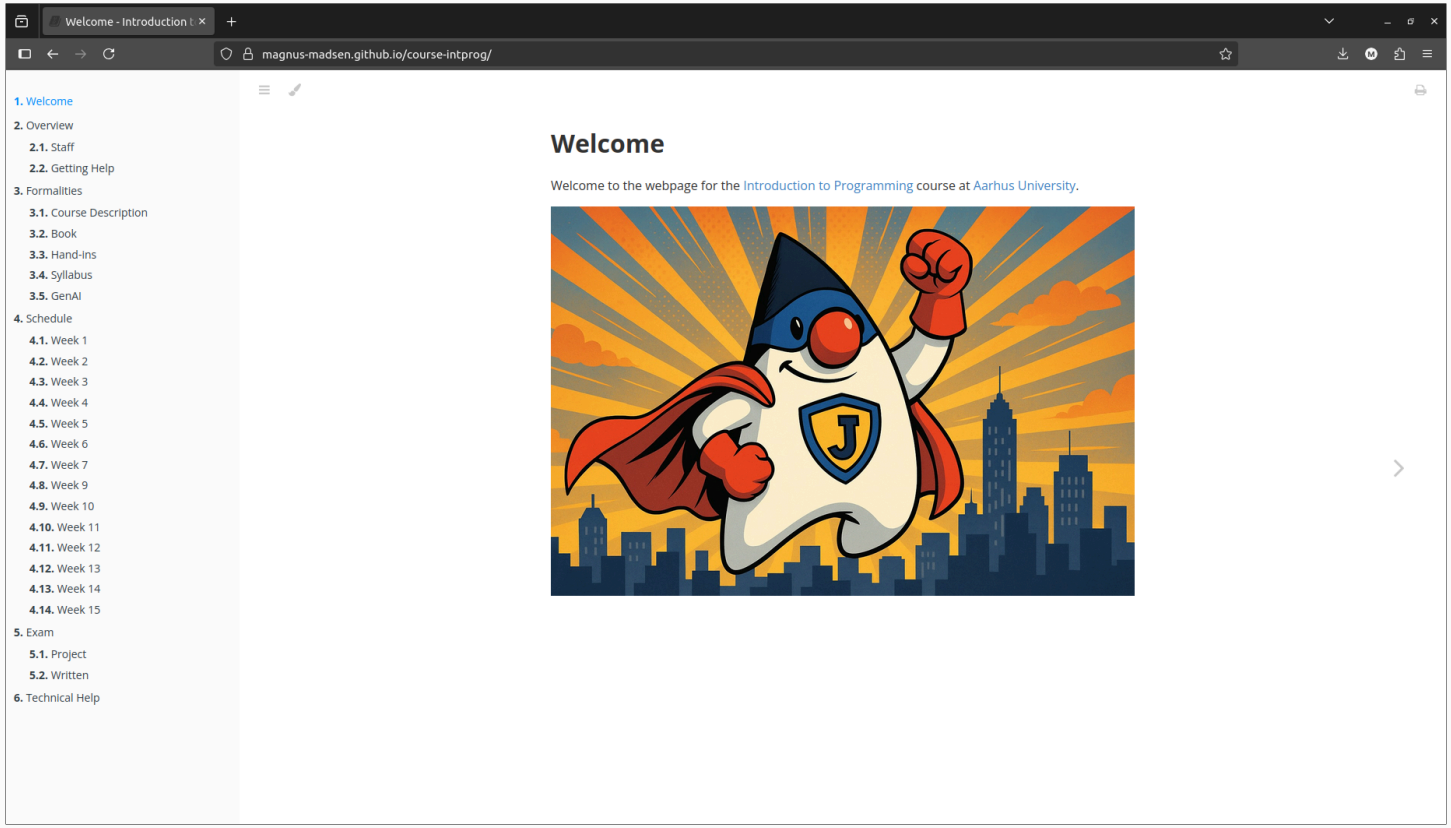
Learning objectives (2/2)

After the course, students will be able to:

- Identify, explain, and overcome compiler errors (e.g. syntax, semantic, or type errors).
- Apply programming techniques to write small programs in imperative or object-oriented style.
- Apply basic debugging and testing techniques to understand and correct program behavior.
- Apply advanced programming features such as inheritance and generics.

Learning materials

Course website



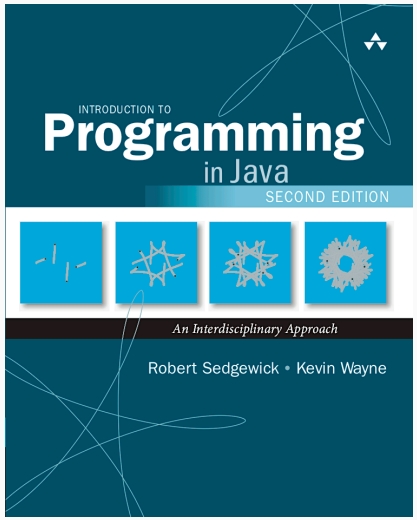
https://magnus-madsen.github.io/course-intprog/

Textbook

Introduction to Programming in Java (2nd edition).

In the words of the authors:

"... Programming is not difficult to learn and that harnessing the power of the computer is rewarding ..."



❗ You must obtain the book. The exercises and hand-ins are in it.

❗ Ensure you get the 2nd edition. The exercises have been renumbered.

Recorded lectures

Bad news: The lectures are – by choice – not recorded 😔

We want to motivate you to participate in the lectures! 🙋

Textbook lectures

The textbook has online lectures at:

- https://www.cubits.ai/collections/106

Each lecture costs 2$. You can buy all of them for 20$.

Q: Is it worth it? Probably not.

Q: Do I get paid for this commercial? No.

Q: Does the internet have an excessive amount of funny cat videos and Java tutorials? Yes.

How to spend your time

Total	**15 hours**
Lectures	4 hours
Reading	3 hours
Exercises	5 hours
Hand-in	3 hours

Getting help

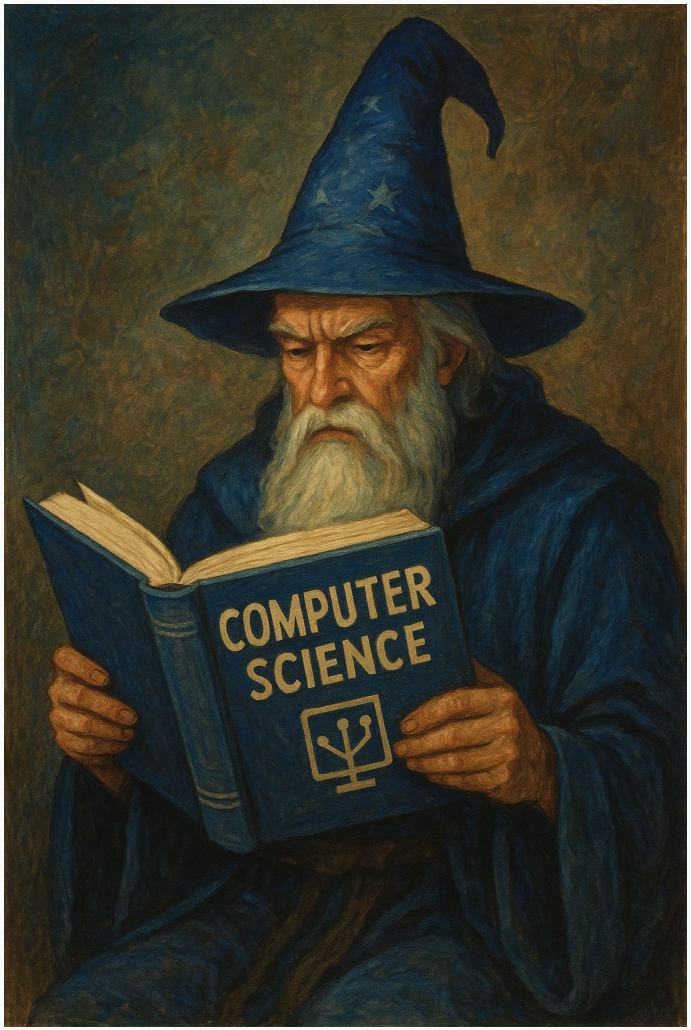
To get help:

- Use the Brightspace web-board for general programming questions.
- Use the Brightspace web-board for general questions about the course.
- Use the Brightspace web-board for questions about the exercises and hand-ins.
- Use the Brightspace web-board for tech support.
- E-mail the teaching assistant for specific questions about the hand-ins.
- E-mail the lecturer for specific questions about the course or the exam.

Tip: Brightspace supports anonymous posts 🐹

What if you already know programming?

- Read the textbook to ensure that you understand all the details.
- Enjoy the many fun exercises in the textbook.



Q: Do you have any questions about the course?

How to become a programmer

Programming is a contact sport

- You cannot become a cook by reading a book.
- You cannot become a pilot by reading a book.
- You cannot become a surgeon by reading a book.
- ...

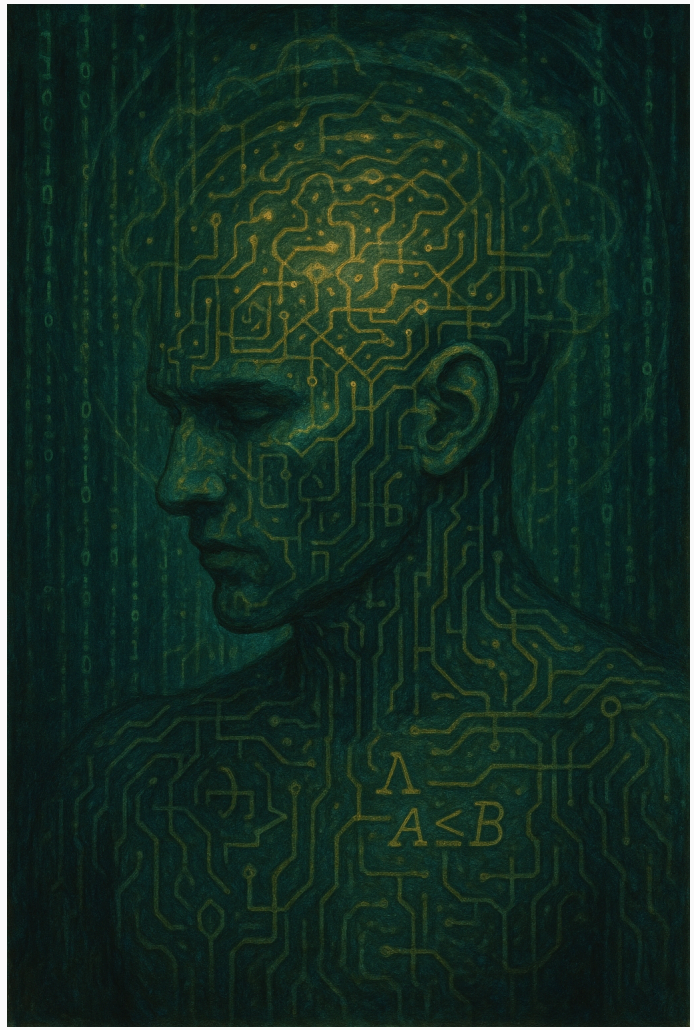
Learning by doing!

Programming is an intellectual pursuit

- Programs are mathematical objects.
- They are “mind materials” that you build with.

“To understand a program you must become both the machine and the program.”

— Alan Perlis



Programming is fun

Programming is an immensely rewarding experience.

You write something and then you can immediately run it.

- The short feedback loop is intoxicating.

And when your program finally works:

Software is everywhere

Software is everywhere

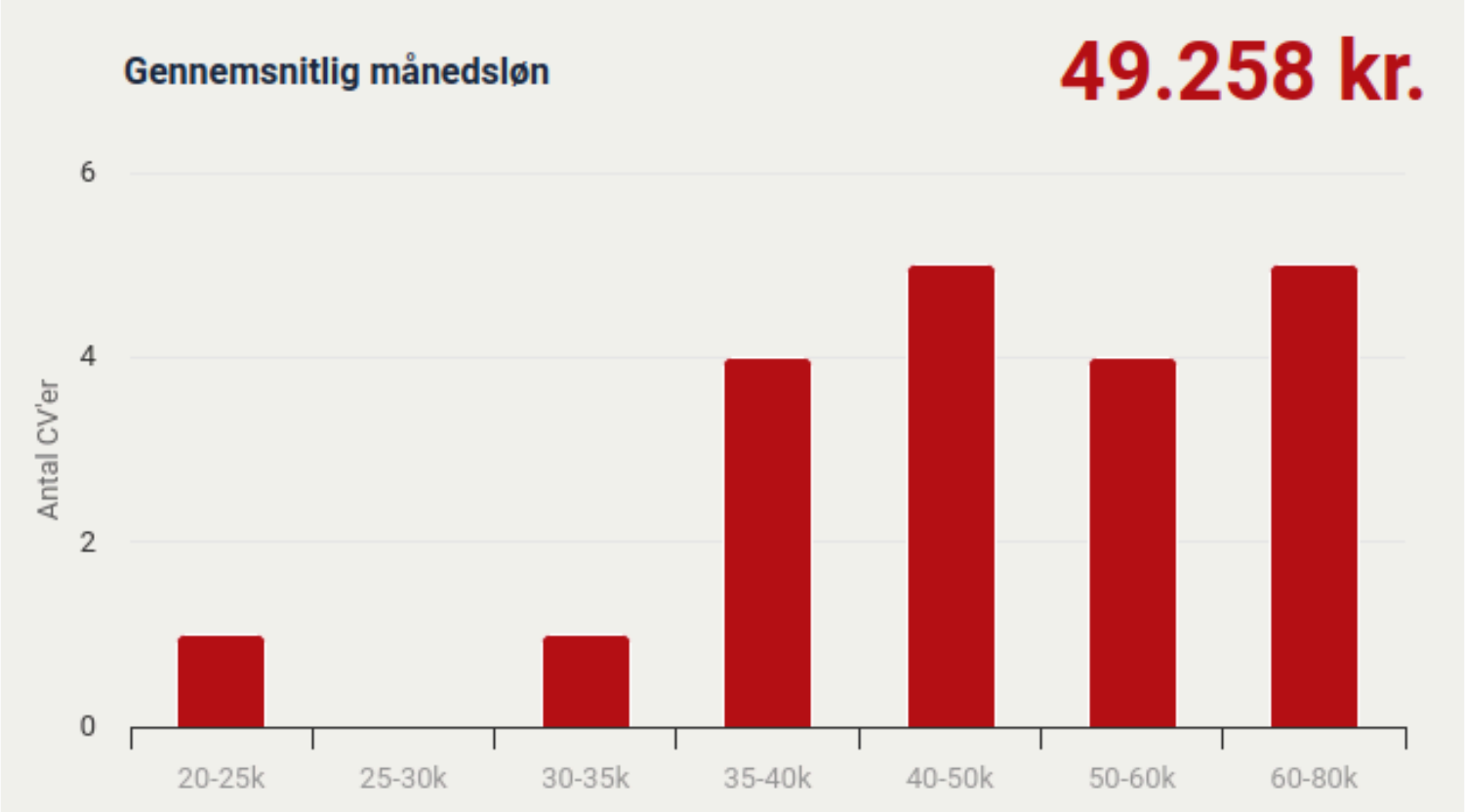
Q: How many computers are there in an Airbus A380?

Why study computer science?

Q: Why do you want to study computer science?

Q: What is the salary of a recently graduated computer science student?

Average salary



Monthly Salary

DKK	49,258
USD	7,386
EUR	6,603
JPY	1,076,233
GBP	5,562

Software engineer at Uber Aarhus with yearly bonus

Art	Specifikation	Antal	Sats	Beløb
1001	Månedsløn	160,33		96.250,00
3042	Sundhedsforsikring, beskatning	58,00		
3140	Transport Allowance			2.211,62
3191	Annual Bonus			330.000,00
8502	Løntillæg 0,45%			433,13
8750	ATP bidrag	160,33		-99,00
8861	Arbejdsmarkedsbidrag Grundlag: 428.853,75		8,00	-34.308,00
8906	A-indk: 394.545,75 Fradrag: 0,00 Skat af 394.545,75		27,00	-106.527,00
[redacted]				-500,92
9421	ESPP B - deduction	426.250,00	15,00	-63.937,50
9993	**Overført til konto** [redacted]			**223.522,33**

Årsoplysning	År til dato	Beskrivelse	Indv. måned	År til dato
(13) AM-indkomst	617.520,75	Ferieberet. løn - kalenderår	428.952,75	578.181,75
(14) Bidragsfri A-indkomst	0,00	Ferieberet. løn - ferieår	428.952,75	956.285,70
(15) A-skat	153.391,00	Fritvalgskonto	0,00	0,00
(16) AM-bidrag	49.400,00			

DKK	428,853
USD	35,117
EUR	31,300
JPY	5,099,344
GBP	26,340

Back to programming

Q: What is a program?

Programs vs. systems

We often talk about **“programming in the large”** vs. **“programming in small”**.

In this course, we focus on **programming in small**.

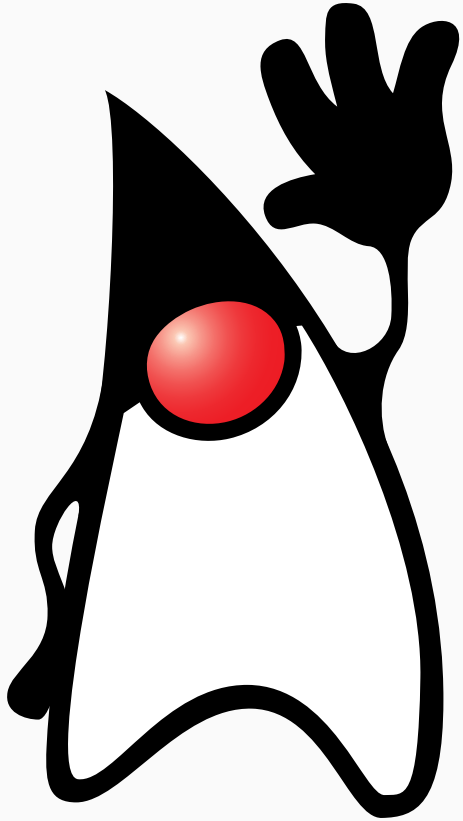
We will learn to write small programs as opposed to entire software systems.

- You must be able to walk before you can run 👶

Getting started with Java

What is Java?

- A programming language
- A compiler (`javac`)
- A runtime (`java`)
- A standard library
- ...



Why Java?

Java

- has a **simple** and **readable syntax** that is easy to learn.
- has a **standard library** with a lot of built-in functionality.
- has a **static type system** that catches errors before your program runs.
- has extensive **IDE** and **tooling** support.

Java ...

- is a **general purpose** language useful for any type of project.
- is **widely adopted** with over 16 million projects on GitHub.
- is open source and freely available.

Java ...

- programs run **reasonably fast** (faster than most languages).
- programs run on **any platform** including Windows, MacOS, and Linux.

Why not Java?

Java …

- is **verbose** and requires **more boilerplate code** compared to modern languages.
- was designed in the **computing era of the 1990s** for single-core desktop systems.

Java …

- lacks support for **functional programming** compared to modern languages.
- lacks **resource and memory safety** features found in languages such as **Rust**.
- faces competition from **modern alternatives** like **Kotlin**, **Rust**, and **Scala**.

Upshot: You will learn many programming languages over your career.

Your first program

We can write a program:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

and save it in the file `HelloWorld.java`.

Compiling a program (1/2)

We compile the program by opening a terminal and running the command:

```
$ javac HelloWorld.java
```

Surprisingly, if the program compiles successfully there is no output.

Running the `javac` command produces a `HelloWorld.class` file.

Compiling a program (2/2)

The `HelloWorld.java` file is a **text-file**, but the `HelloWorld.class` is a **binary-file**:

```
0000000 feca beba 0000 4100 1d00 000a 0002 0703
0000010 0400 000c 0005 0106 1000 616a 6176 6c2f
0000020 6e61 2f67 624f 656a 7463 0001 3c06 6e69
0000030 7469 013e 0300 2928 0956 0800 0900 0007
0000040 0c0a 0b00 0c00 0001 6a10 7661 2f61 616c
0000050 676e 532f 7379 6574 016d 0300 756f 0174
0000060 1500 6a4c 7661 2f61 6f69 502f 6972 746e
0000070 7453 6572 6d61 083b 0e00 0001 480c 6c65
0000080 6f6c 202c 6f57 6c72 0a64 1000 1100 0007
0000090 0c12 1300 1400 0001 6a13 7661 2f61 6f69
00000a0 502f 6972 746e 7453 6572 6d61 0001 7007
```

Decompiling

We can inspect the binary-file with `javap`:

```
$ javap -c HelloWorld.class
```

```
public class HelloWorld {
  /* ... */
  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #7  // Field java/lang/System.out
       3: ldc           #13 // String Hello, World
       5: invokevirtual #15 // Method java/io/PrintStream.println
       8: return
}
```

Run a program

After compiling a program, we can run it by invoking the `java` command.

```
$ java HelloWorld
```

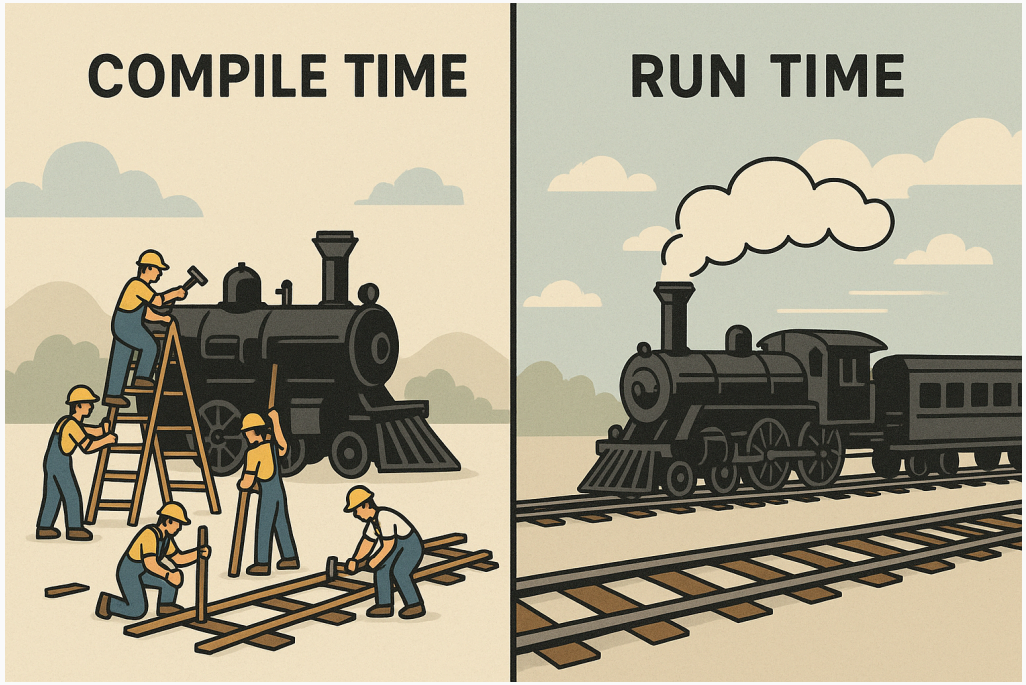
The terminal responds with:

```
Hello, World
```

Compile time vs. run time

Compile time is when your program is under construction. You can get a compile-time error when your program is not *valid Java code*.

Run time is when your program is *running*. You can get a runtime error when your program does not do what you want it to.



Command line arguments

Command line arguments allow the user to specify an *input* to the program.

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello " + args[0] + "!");
    }
}
```

Compile once	`$ javac Main.java`		
Execute with many inputs	`$ java Main Alice`	`$ java Main Bob`	`$ java Main World`
	`Hello Alice!`	`Hello Bob!`	`Hello World!`

Programming tools

A programmer's toolkit consists of many specialized tools that help development:

Essential Tools:

- **Command Line Interface (CLI)** – Direct system interaction
- **Compilers** – Translate source code to executable form
- **Interpreters** – Execute code directly without compilation
- **Text Editors** – Basic code writing and editing

Advanced Tools:

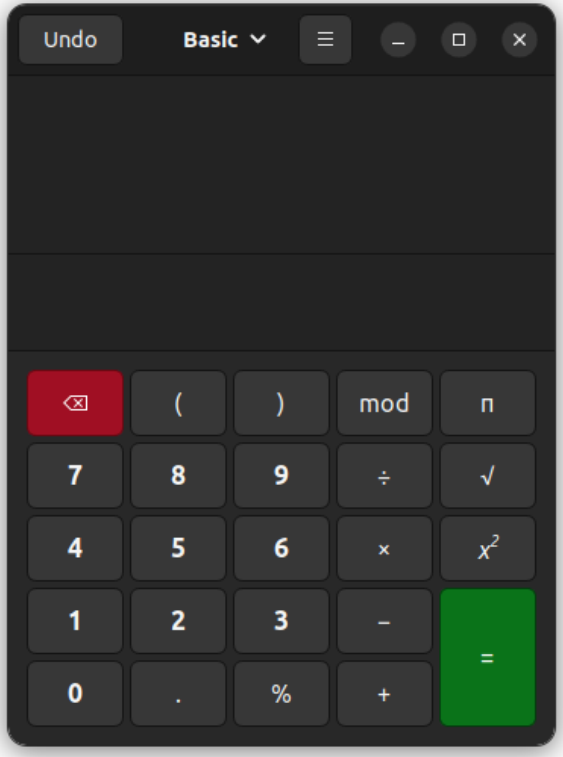
- **Integrated Development Environments (IDEs)** – All-in-one development suite
- **Version Control Systems (VCS)** – Track changes and collaboration
- **Debuggers** – Find and fix runtime errors
- **Profilers** – Analyze performance bottlenecks

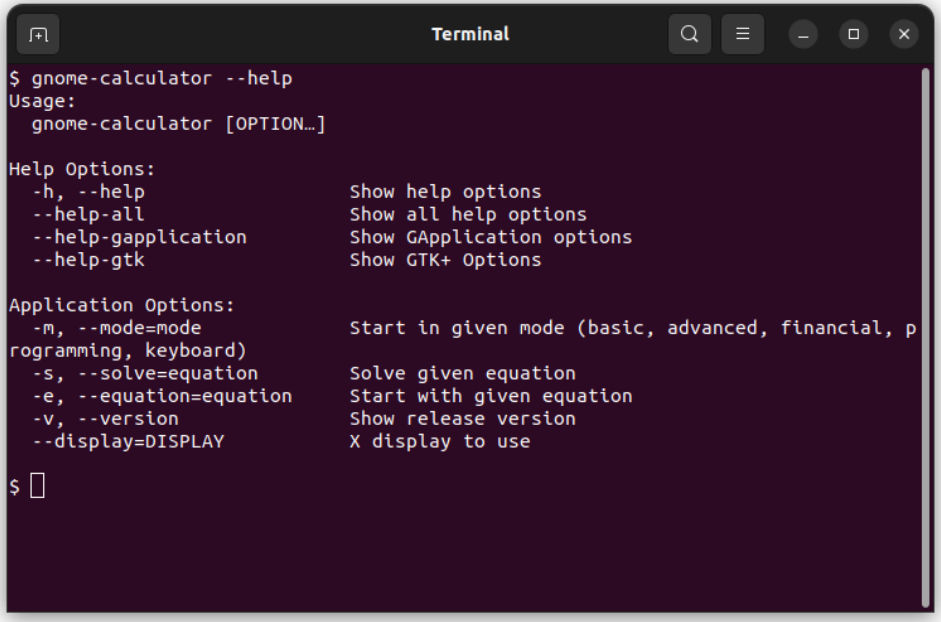
i Each tool serves a specific purpose in the software development lifecycle. We will explore these in detail on the following slides.

CLI

You may be used to opening and interacting with programs via a *Graphical User Interface*.

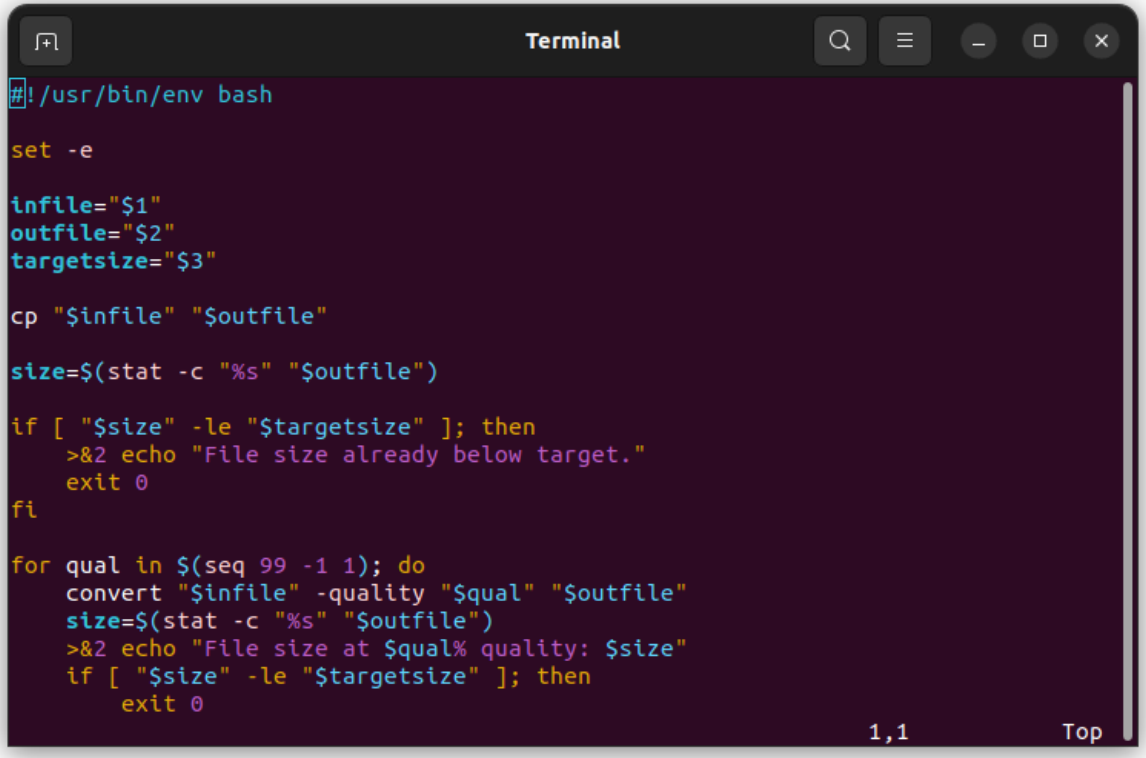
The *Command Line Interface* is more **powerful** and more **automatable**.





With the CLI we can combine commands into *scripts* to automate repetitive tasks.

This script repeatedly shrinks an image until it is the desired size.



Interpreters

Interpreters read source code and execute it **directly**, line by line, without creating a separate executable file.

How Interpreters Work:

- Read source code directly
- Parse and execute instructions immediately
- No separate compilation step required
- Code runs as soon as you type it

Examples:

- Python (`python script.py`)
- JavaScript (in web browsers)
- Ruby (`ruby script.rb`)
- Shell scripts (`bash script.sh`)

Advantages:

- ✓ Fast development cycle
- ✓ Interactive testing (REPL)
- ✓ Cross-platform portability
- ✓ Easy debugging

Disadvantages:

- ❌ Slower execution speed
- ❌ Runtime error discovery
- ❌ Requires interpreter

Compilers

Compilers translate code from one language to another.

Usually this is from a **high-level human-readable language** (such as Java) to a **low-level computer-readable language** (such as **bytecode** or **assembly**).

```
public class MyProgram {
    public static void main(String[] args)
}
```

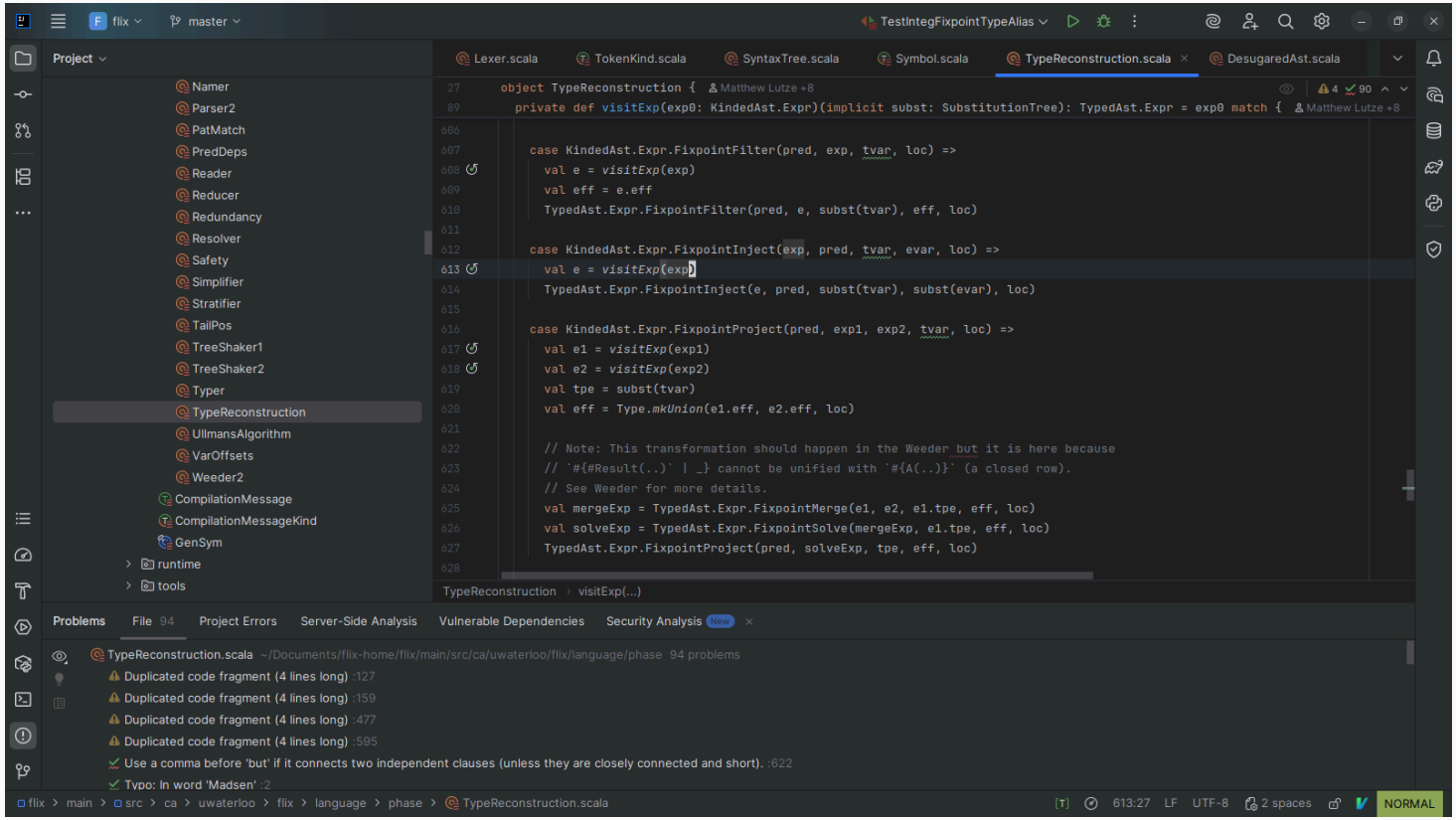
```
0000000 feca beba 0000
0000010 0400 000c 0005
0000020 6e61 2f67 624f
0000030 7469 013e 0300
0000040 0c0a 0b00 0c00
0000050 676e 532f 7379
0000060 1500 6a4c 7661
```

The low-level language is typically faster for a computer to execute.

Integrated Development Environments (IDEs)

An **Integrated Development Environment** is like Microsoft Word for programmers.

Programs are just text, but IDEs help to write, organize, test, and run your code.



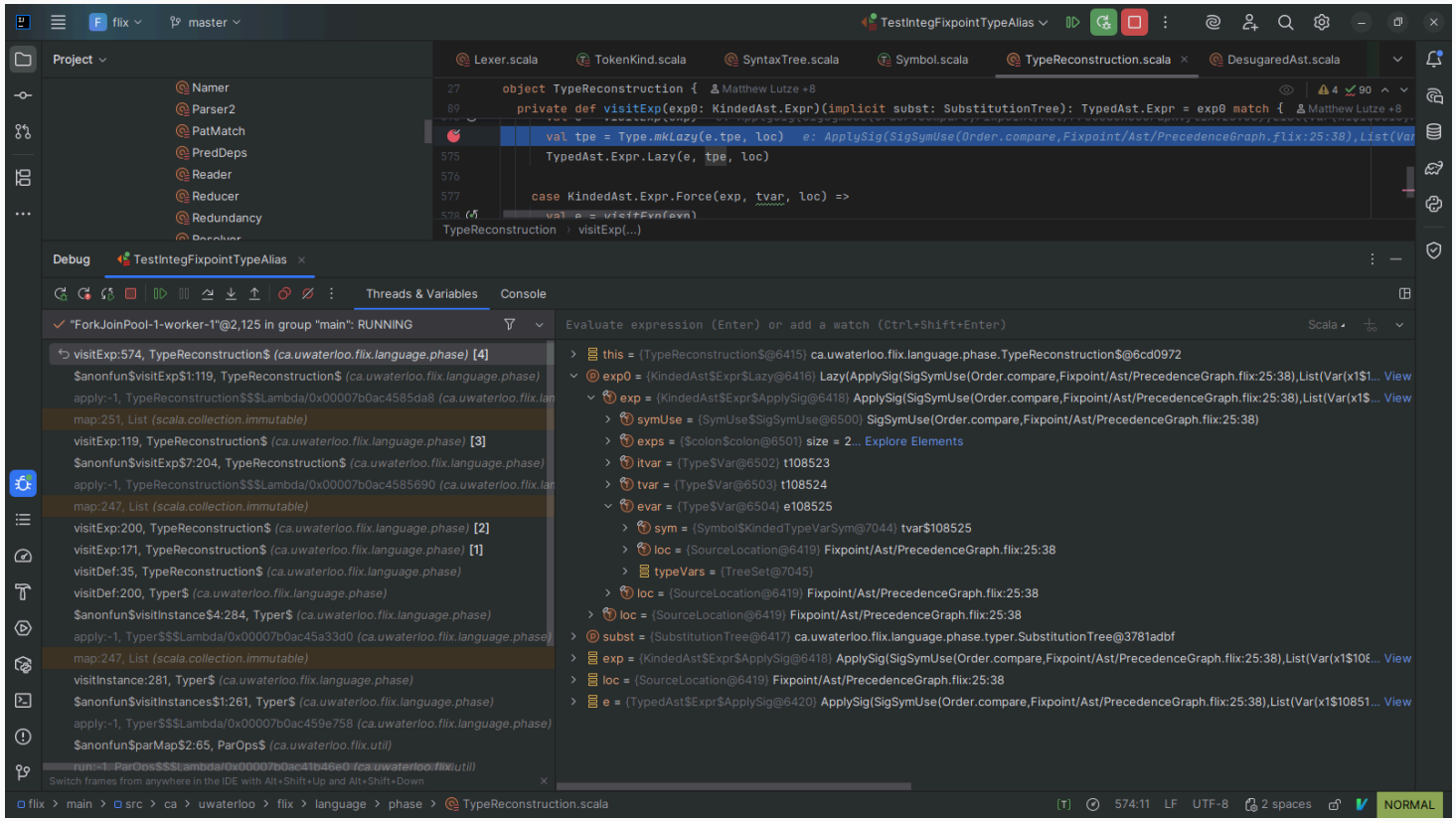
Version Control Systems (VSCs)

Version Control Systems, such as **Git**, allow the programmer to manage multiple *versions* of a project.

- They track the complete history of your code changes over time.
- They make it possible to collaborate with others on the same project.

Debuggers

Debuggers allow us to inspect the **state** of a program while it is running.



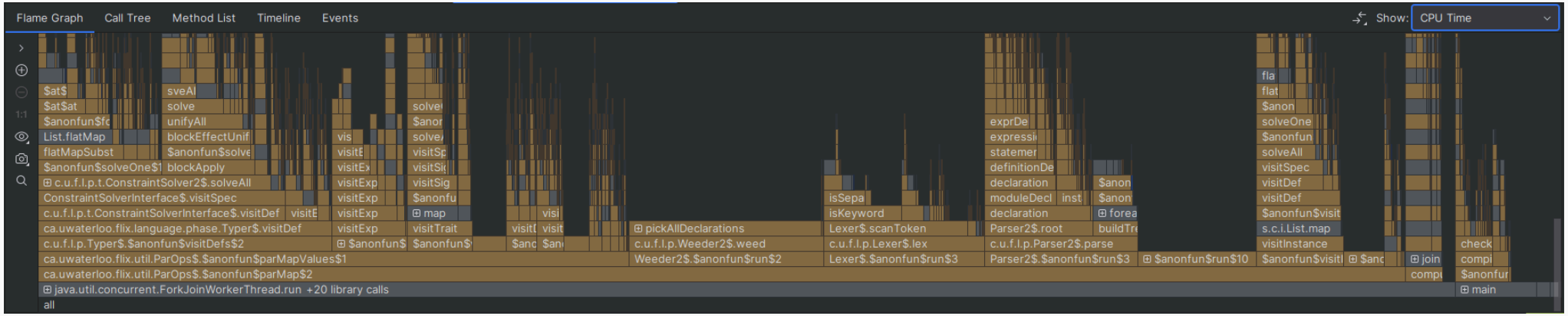
They are often integrated into an IDE.

Profilers

Profilers tracks the **time** and **memory** usage of our code.

We can use them to identify and fix **bottlenecks** – places in our code that run slowly.

For example, a **flame graph** shows the time spent in each part of a program.



Epilogue

Compiler error of the week (1/2)

What is the problem here?

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
```

Compiler error of the week (1/2)

What is the problem here?

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
```

The Java compiler reports:

```
Main.java:5: error: reached end of file while parsing
```

Do you understand the issue now?

Compiler error of the week (2/2)

What is the problem here?

```
$ javac MyProgram.java
$ java MyProgram.class
```

Compiler error of the week (2/2)

What is the problem here?

```
$ javac MyProgram.java
$ java MyProgram.class
```

The terminal reports:

```
Error: Could not find or load main class MyProgram.class
Caused by: java.lang.ClassNotFoundException: MyProgram.class
```

Do you understand the issue now?

Live programming

Live programming

Today, we shall see how to:

- compile and run a program with nothing but a text editor, `javac`, and `java`.
  - how to open the terminal.
  - how to check if Java is correctly installed.
- use IntelliJ IDEA to write a program and run it from the terminal.
- use IntelliJ IDEA to write a program and run it from within IDEA.
- how to have multiple programs in the same IDEA project.

Sources for images and slides

- https://introcs.cs.princeton.edu/java/lectures/
- https://commons.wikimedia.org/wiki/File:HelpDeskHappyBryce.jpg
- https://commons.wikimedia.org/wiki/File:NASA_Mars_Rover.jpg
- https://commons.wikimedia.org/wiki/File:Surgeon_performing_robot-assisted_surgery.jpg
- https://commons.wikimedia.org/wiki/File:20210216-ARS-LSC-0506_(51020911508).jpg
- https://commons.wikimedia.org/wiki/File:%E7%96%AF%E7%8B%82%E7%9C%BC%E9%95%9C%E8%9B%87%E8%BF%87%E5%B1%B1%E8%BD%A6.JPG
- https://commons.wikimedia.org/wiki/File:Daft_Punk_in_2013_2.jpg
- https://commons.wikimedia.org/wiki/File:Claas_Lexion_480-20080806-RM-133439.jpg
- https://www.reddit.com/r/dkloenseddel/comments/1k0l761/softwareingeni%C3%B8r_aarhus/
- https://commons.wikimedia.org/wiki/File:Duke_(Java_mascot)_waving.svg
- https://phdcomics.com/comics/archive.php?comicid=1531