

Programming – Lecture 15

Going Beyond ACM

- Lectures Beyond Inf-ProgOO
- IDEs Beyond Eclipse
- Projects Beyond Homework
- Java Beyond ACM
- Programming Beyond Inf-ProgOO

Programming – Lecture 15

Going Beyond ACM

- Lectures Beyond Inf-ProgOO
- IDEs Beyond Eclipse
- Projects Beyond Homework
- Java Beyond ACM
- Programming Beyond Inf-ProgOO

Implementing Languages

Context-free grammars

Parsing source code

Building compilers

Prof. Dr. Hanus

Übersetzerbau

Wahlpflicht

Efficiency and Data Structures

Defining “performance”

Efficiency of algorithms

Efficiency of data structures

Prof. Dr. Jansen

ADS

2. Semester

Prof. Dr. Jansen

Effiziente Algorithmen

Wahlpflicht

Graphics Programming

Image data streams

Feature extraction

Modeling 3D objects

Rendering

Prof. Dr. Koch

**Einführung in die
Bildverarbeitung**

Wahlpflicht

Prof. Dr. Koch

**Computer
Graphik**

Wahlpflicht

Memory Management

Stack and Heap

Virtual memory

Paging and memory hierarchy

Prof. Dr. Landsiedel

Betriebs- und Kommunikationssysteme

2. Semester

Concurrency

Race conditions and synchronization

Distributed systems

Patterns for parallel and distributed systems

PD Dr. Huch

**Concurrent and
Distributed Progr.**

Wahlpflicht

Prof. Dr. Hasselbring

**[...] parallele und
verteilte Systeme**

Wahlpflicht

Real-Time Programming

Control flow and data flow languages

Model-based design

Concurrency and scheduling

Worst-case execution times

Prof. Dr. von. Hanxleden

**Synchrone
Sprachen**

Wahlpflicht

Prof. Dr. von Hanxleden

**Embedded real-
time systems**

Wahlpflicht

Programming Patterns

Requirements analysis

Specification tools

Design patterns

Prof. Dr. Hasselbring
Softwaretechnik

4. Semester

Prof. Dr. Hasselbring
Softwareprojekt

4. Semester

Programming Languages

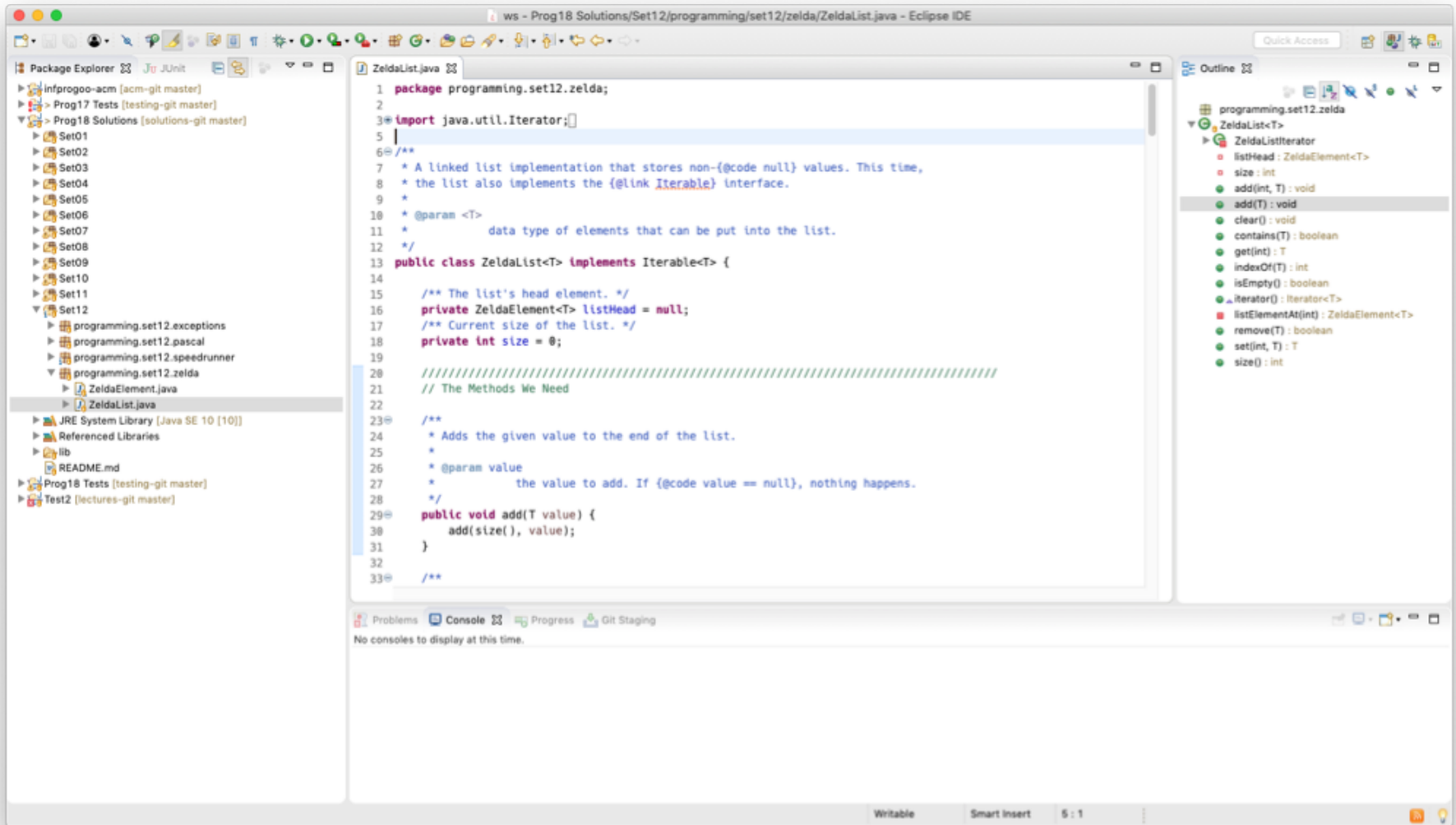
Well...

Programming – Lecture 15

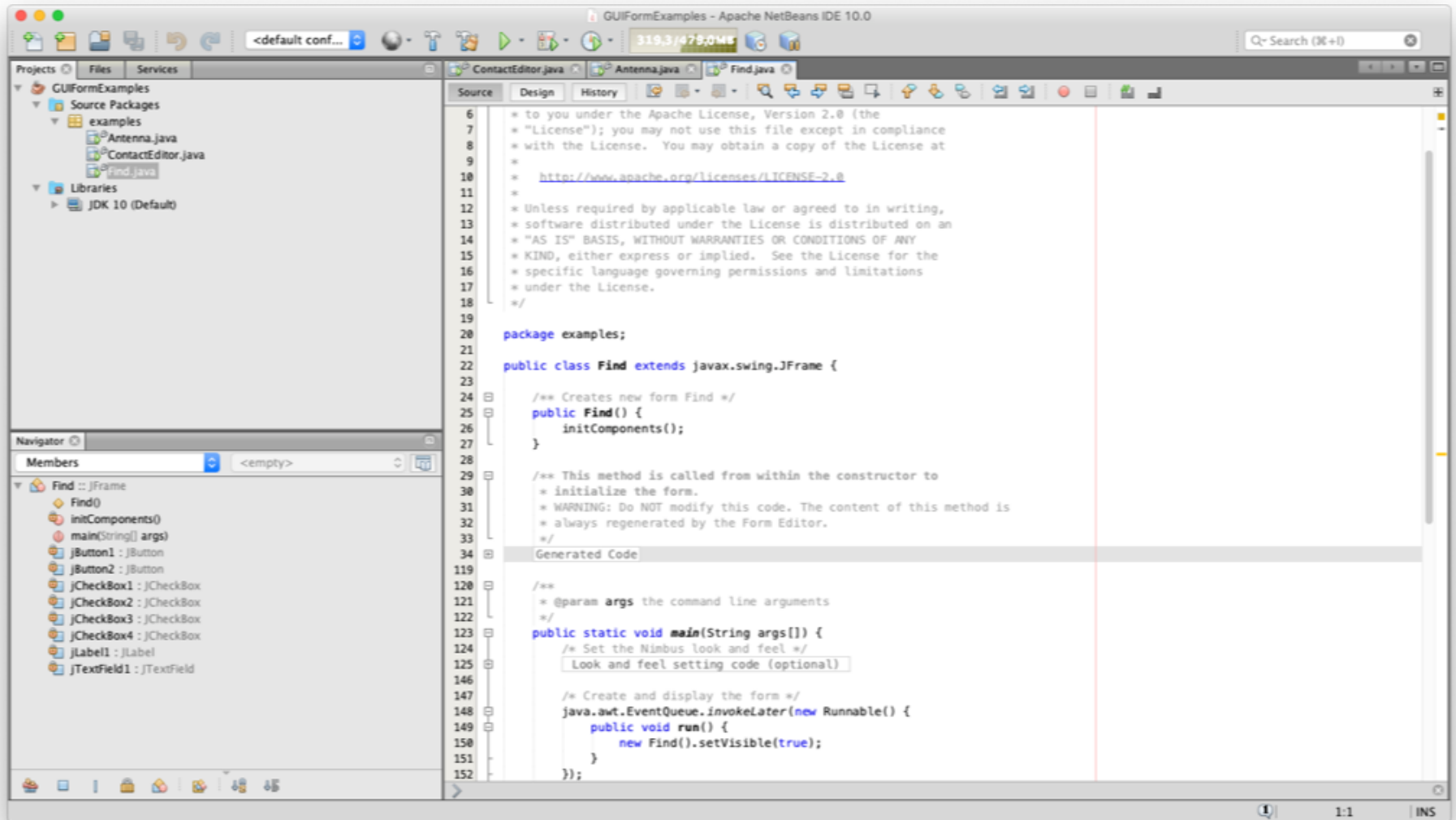
Going Beyond ACM

- Lectures Beyond Inf-ProgOO
- IDEs Beyond Eclipse
- Projects Beyond Homework
- Java Beyond ACM
- Programming Beyond Inf-ProgOO

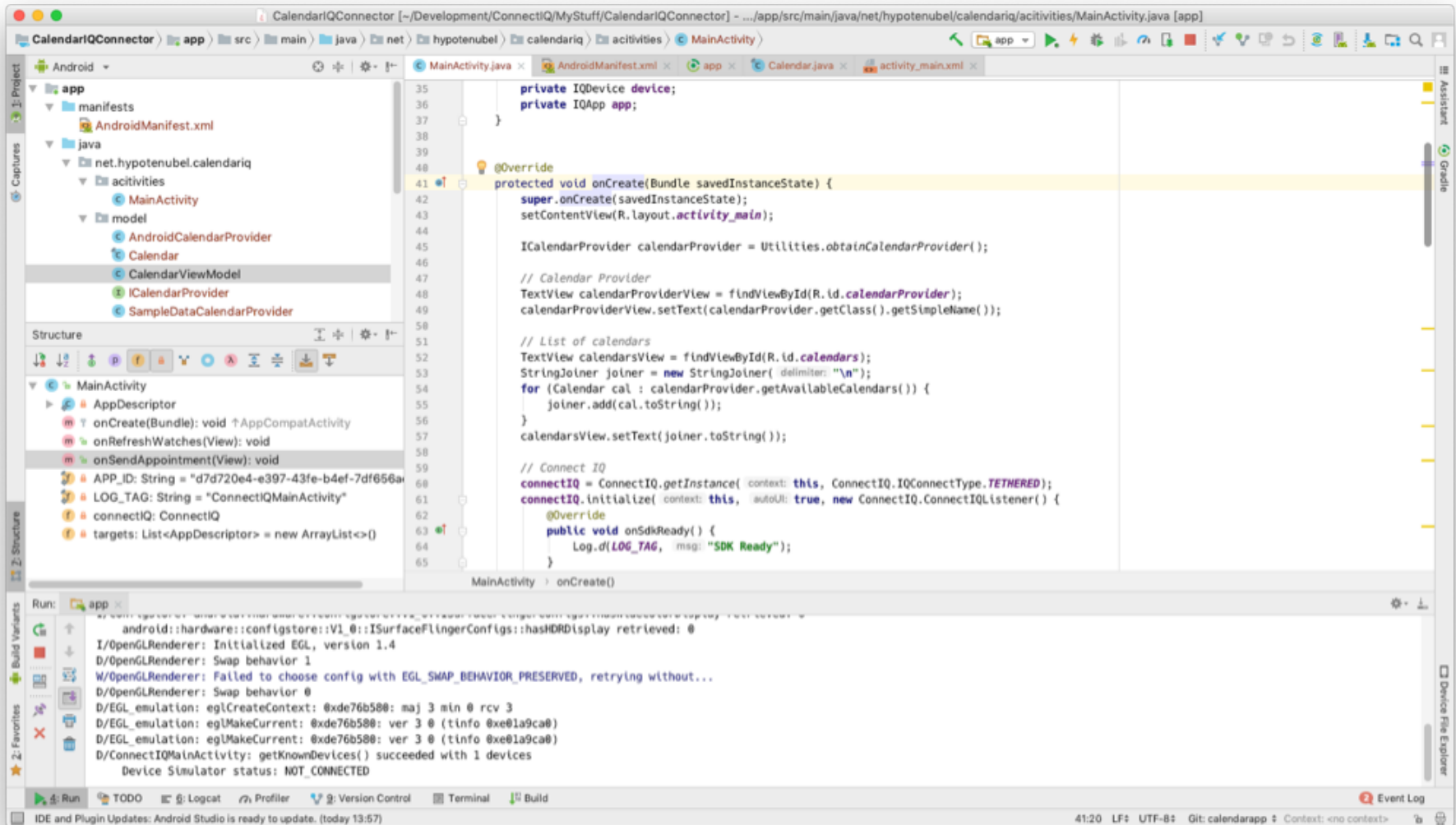
More Eclipse



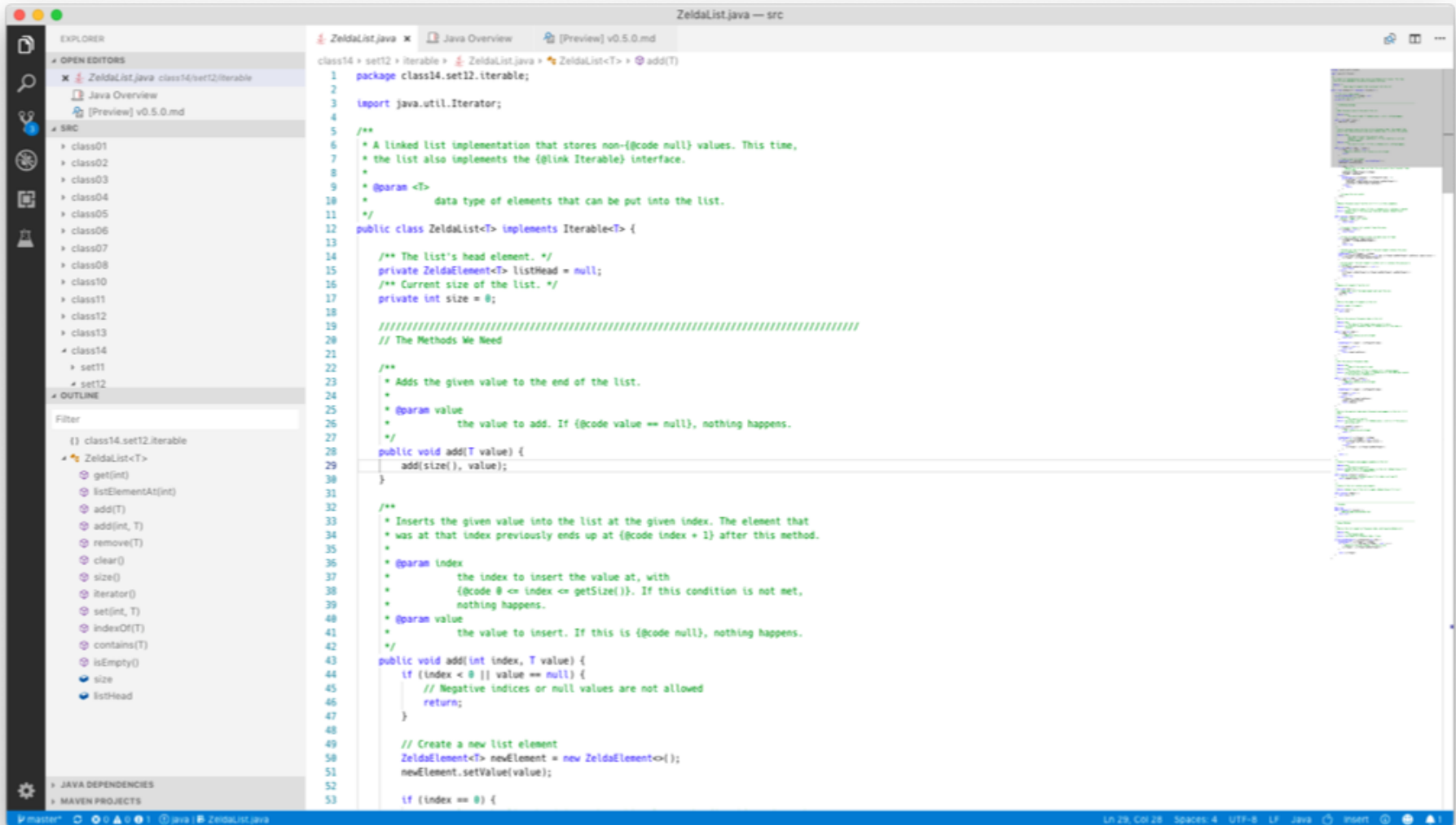
NetBeans



IntelliJ IDEA



Visual Studio Code



Remember

IDEs are monsters
and each works differently,
so give them time
before forming an opinion!

Programming – Lecture 15

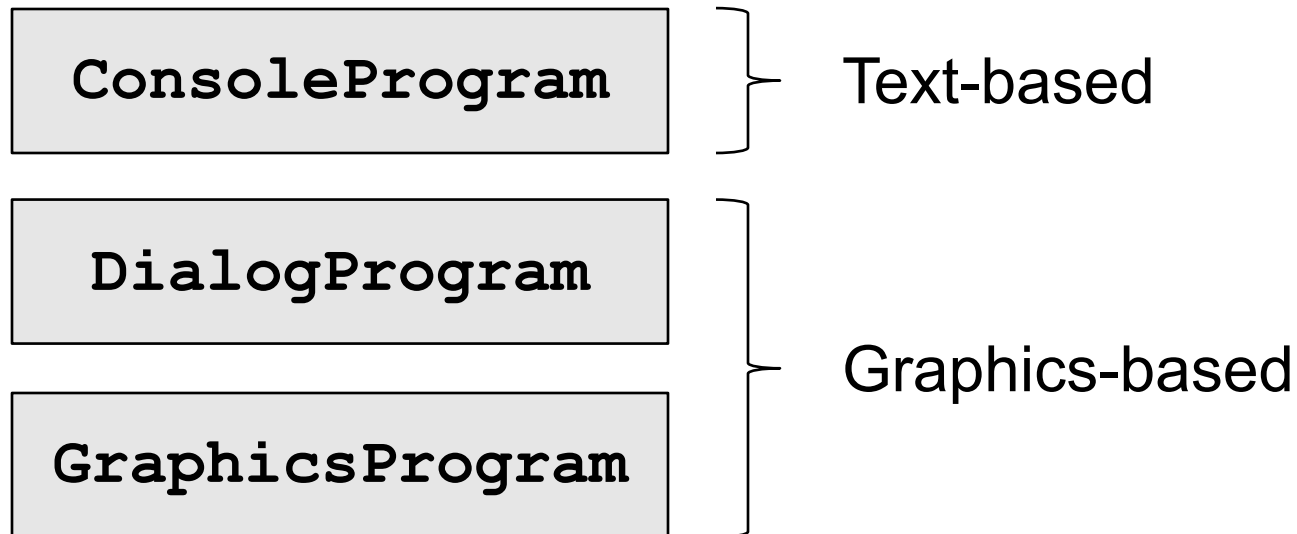
Going Beyond ACM

- Lectures Beyond Inf-ProgOO
- IDEs Beyond Eclipse
- Projects Beyond Homework
- **Java Beyond ACM**
- Programming Beyond Inf-ProgOO

The ACM Library

So far, we have used the ACM library to write our programs.

Most of our programs were instances of one of these classes:



Alternatives

ConsoleProgram

DialogProgram

GraphicsProgram

Alternatives: Text-Based

Simple Java programs are text-based by default.
Remember Lecture 2:

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

Class Definition

ACM:

```
public class Wubbel
    extends ConsoleProgram {
    //...
}
```

Java:

No restrictions. Every class can have a **main** method.

Program Entry Point

ACM:

```
public void run() {  
    //...  
}
```

Java:

```
public static void main(String[] args) {  
    //...  
}
```

Console Output

ACM:

```
println(...);
```

Java:

```
System.out.println(...);
```

In Eclipse, just type "syso<ctrl><space>"

Console Input

ACM:

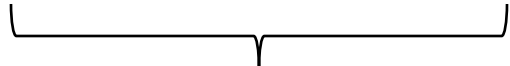
```
String s = readLine();  
int i = readInt();
```

Java:

```
Scanner scanner = new Scanner(System.in);  
String s = scanner.next();  
int i = scanner.nextInt();
```


Command-Line Arguments

```
public static void main(String[] args) {  
    //...  
}
```

Array of Arguments

Command-Line Arguments

Entry Point:

```
public static void main(String[] args) {  
    for (String arg : args)  
        System.out.println(arg);  
}
```

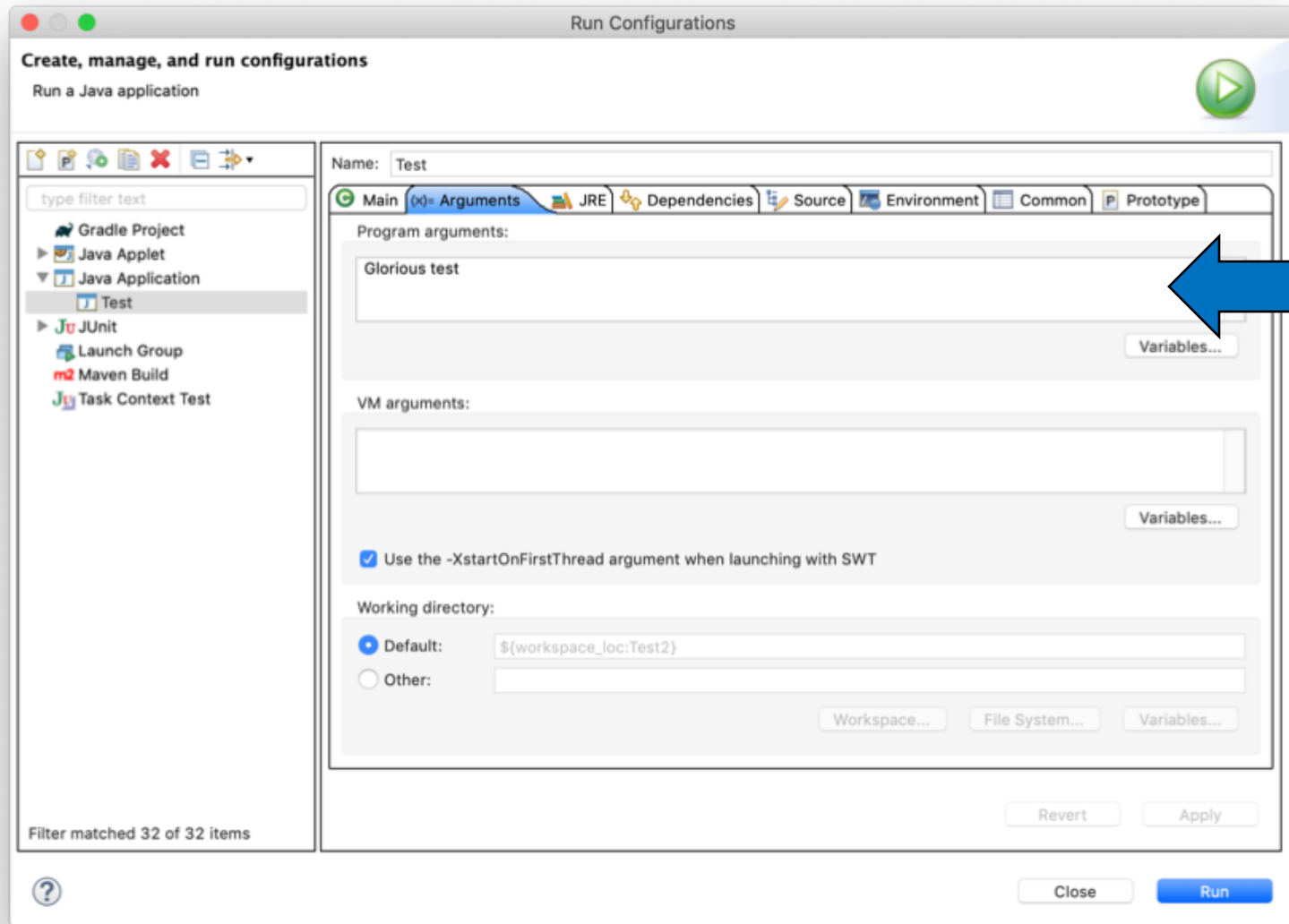
Invocation:

```
java MyClass Glorious test
```

Output:

```
Glorious  
test
```

Command-Line Arguments



Common Code Patterns

Static Code:

```
public static void main(String[] args) {  
    // Call only other static methods  
}
```

Only use for small, single-class programs.

Common Code Patterns

Slim Main Method:

```
public static void main(String[] args) {  
    MyClass mc = new MyClass(args);  
    mc.startDoingStuff();  
}
```

Standard pattern. Only uses main to instantiate your main class and have it start doing stuff.

Alternatives

`ConsoleProgram`

`DialogProgram`

`GraphicsProgram`

Alternatives: Graphics-Based

Java GUI land
has become a sad place...

Java AWT



Wikipedia Example

Java AWT

The original Java GUI library.

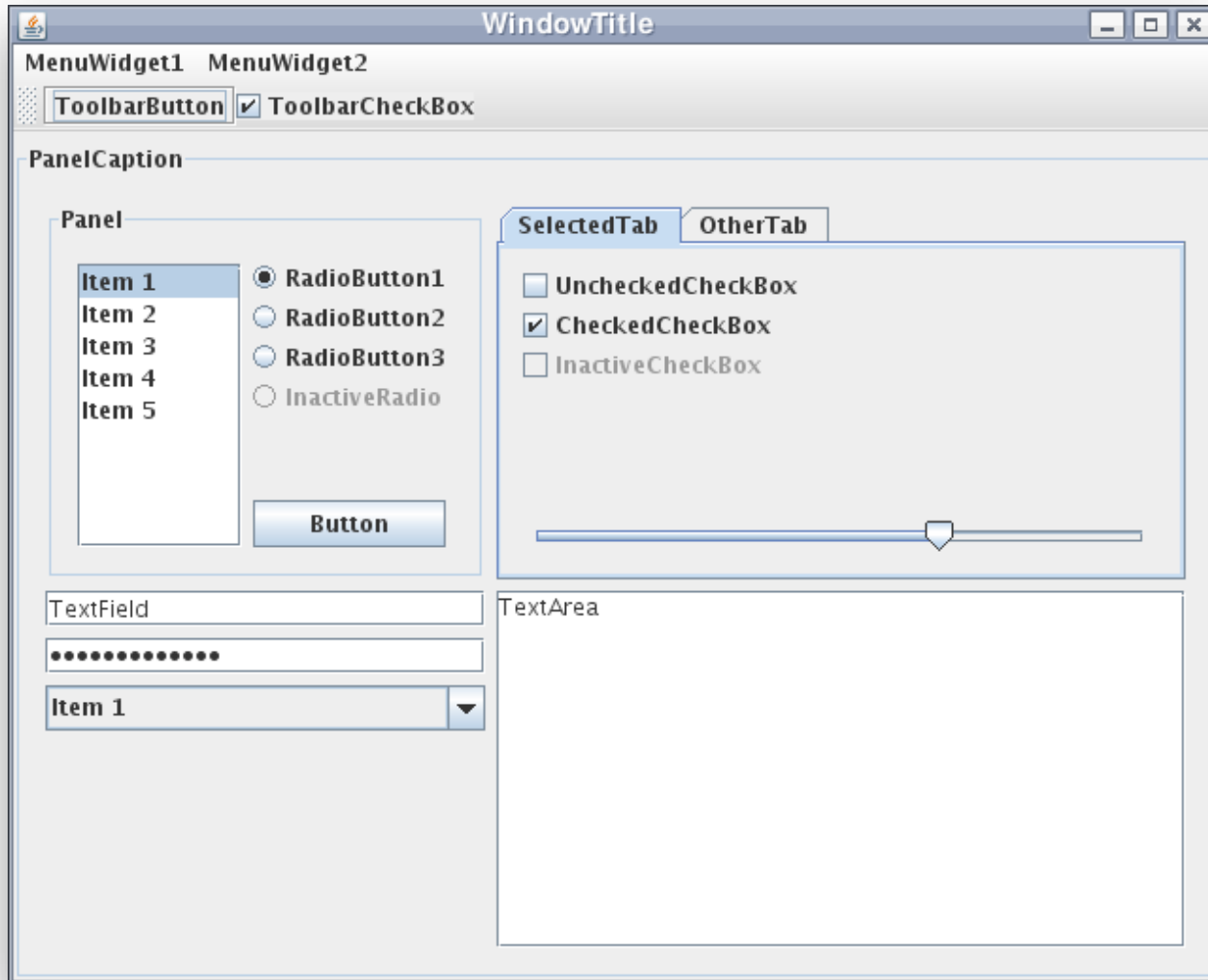
Can look plain and outdated.

Not rich in features.

Verdict

Don't use directly.

Java Swing



Wikipedia Example

Java Swing

The screenshot displays the TuneQ software interface. The title bar reads "TuneQ: Anlagen > Antrieb HW61, AVV_ZVV". The main window is divided into several sections:

- Navigation:** A tree view on the left showing the hierarchy of "Strecken" (Strecke 1 to 5) and "Datensammler" (GUV Albertplatz, Btf. Gorbitz, etc.).
- Übersicht:** A central panel showing details for "Antrieb EW 7 - Strecke 1".
- Table:** A table listing components under "Baugruppen".
- Buttons:** "Neu...", "Bearbeiten...", "Löschen", "Hoch", "Runter" at the bottom.
- Footer:** "3 Ungelesene Nachrichten" in the bottom right corner.

Antrieb EW 7 - Strecke 1
Friedrichstr. - Maxstr./Könneritzstr.

Status: In Betrieb
Montage: Gleismitte
Ausführung: Flachbettweiche

Gerätenr.: 320 513
EDV-Nr.: 109 244 318
Einbau: 12.03.2005
Inspektion: 21.02.2007

[Eigenschaften bearbeiten](#)

Produkt

Baureihe: HW 61 Stellkraft: 5000 N
Antriebsform: Elektromagnetisch Verschluss: Ja
Variante: AVV-ZVV Feder: Ja
Art: Umstellweiche Dämpfung: Ja
Zungenaufschlag: 30 - 70 mm Richtungsschalter: nein
Höhe mit Kasten: 300 mm Zungenprüfer: Ja
Höhe ohne Kasten: 205 mm Verschlussen: Ja
Betriebsspannung: 600/750 V DC Auffahrbar: Ja

[Produktdetails zeigen](#)

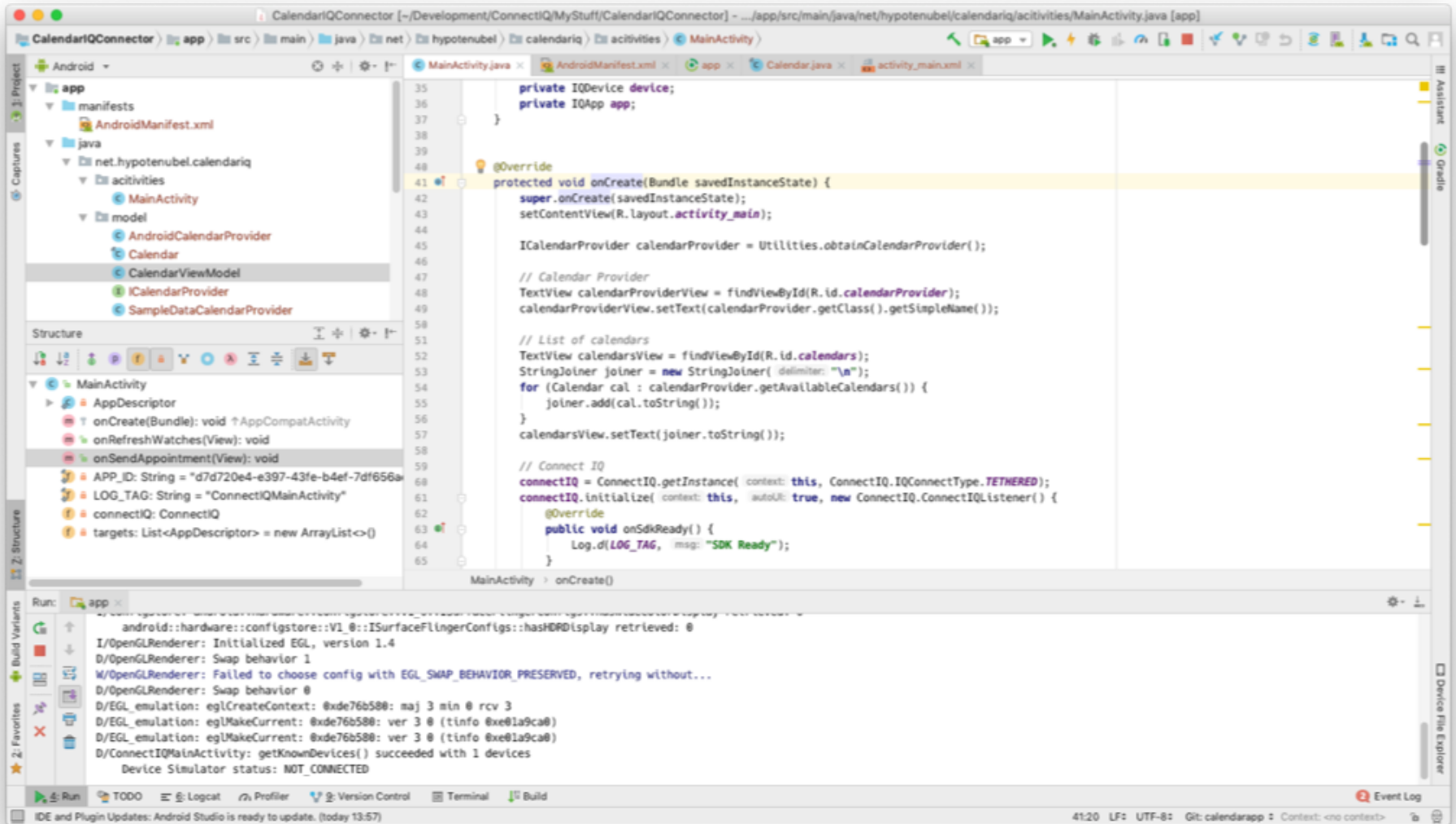
Baugruppen:

Bezeichnung	EDV-Nummer	Gerätenr.	Einbau	Bemerkung
Erdkasten		320 514	12.03.2005	
Doppelzugmagnet	30032002		12.03.2005	
Antriebswelle	31051008		30.11.2006	

Neu... Bearbeiten... Löschen Hoch Runter

! 3 Ungelesene Nachrichten

Java Swing



Java Swing

Successor to AWT.

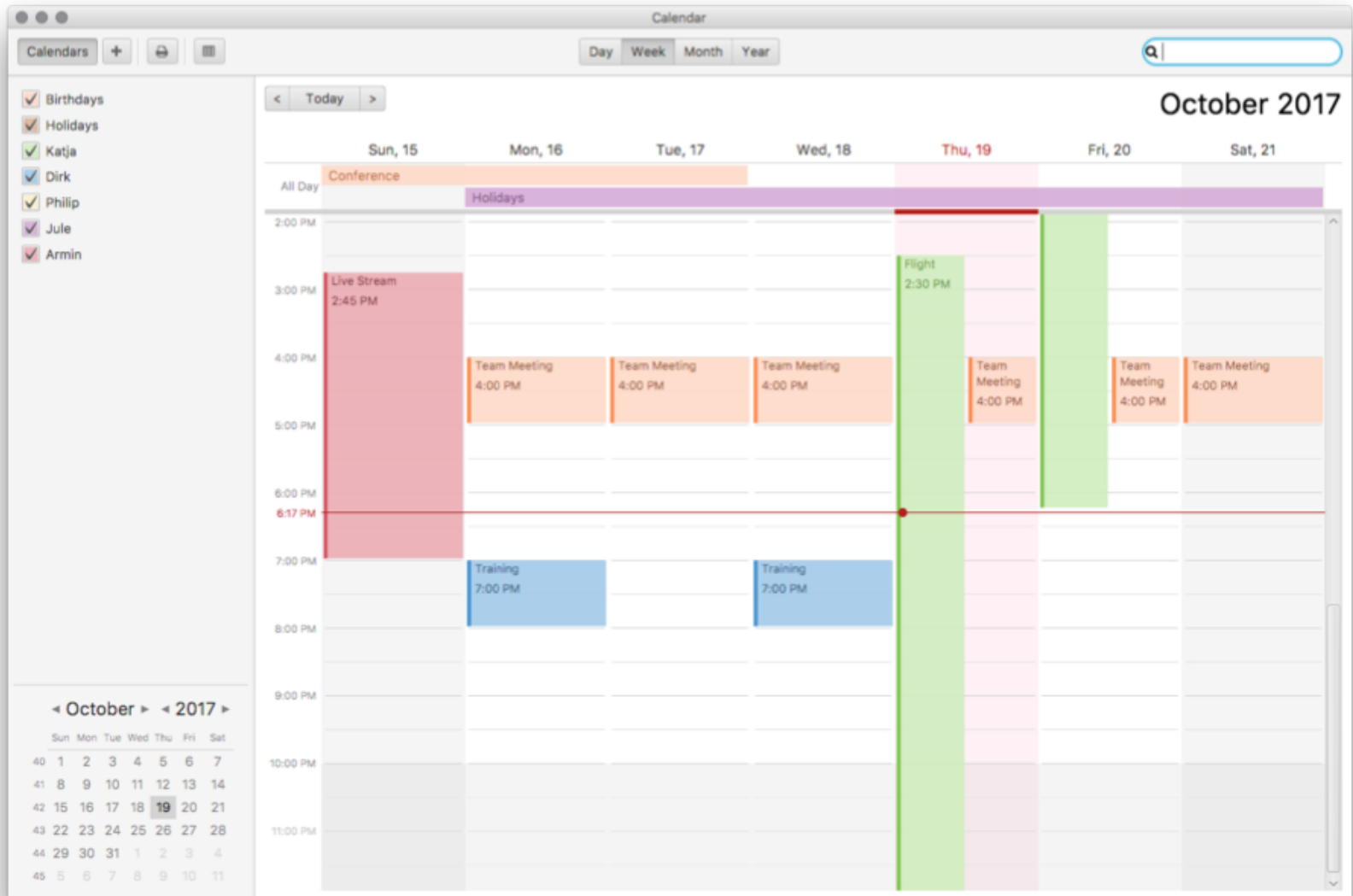
Rich in features and 3rd party support.

Work required to make it look really good.

Verdict

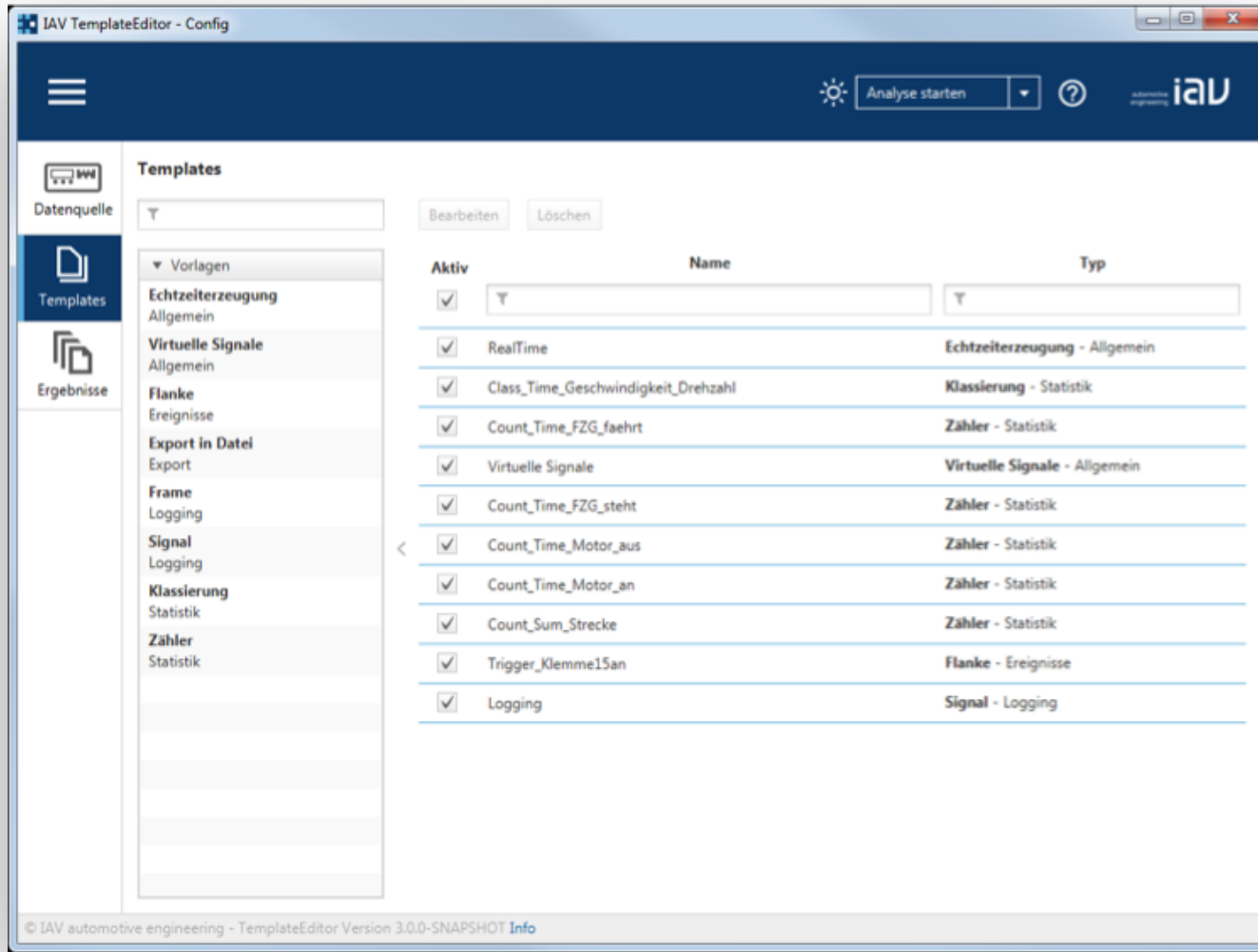
The simplest choice for Java GUIs.

JavaFX



CalendarFX

JavaFX



JavaFX

Modern, beautiful GUIs.

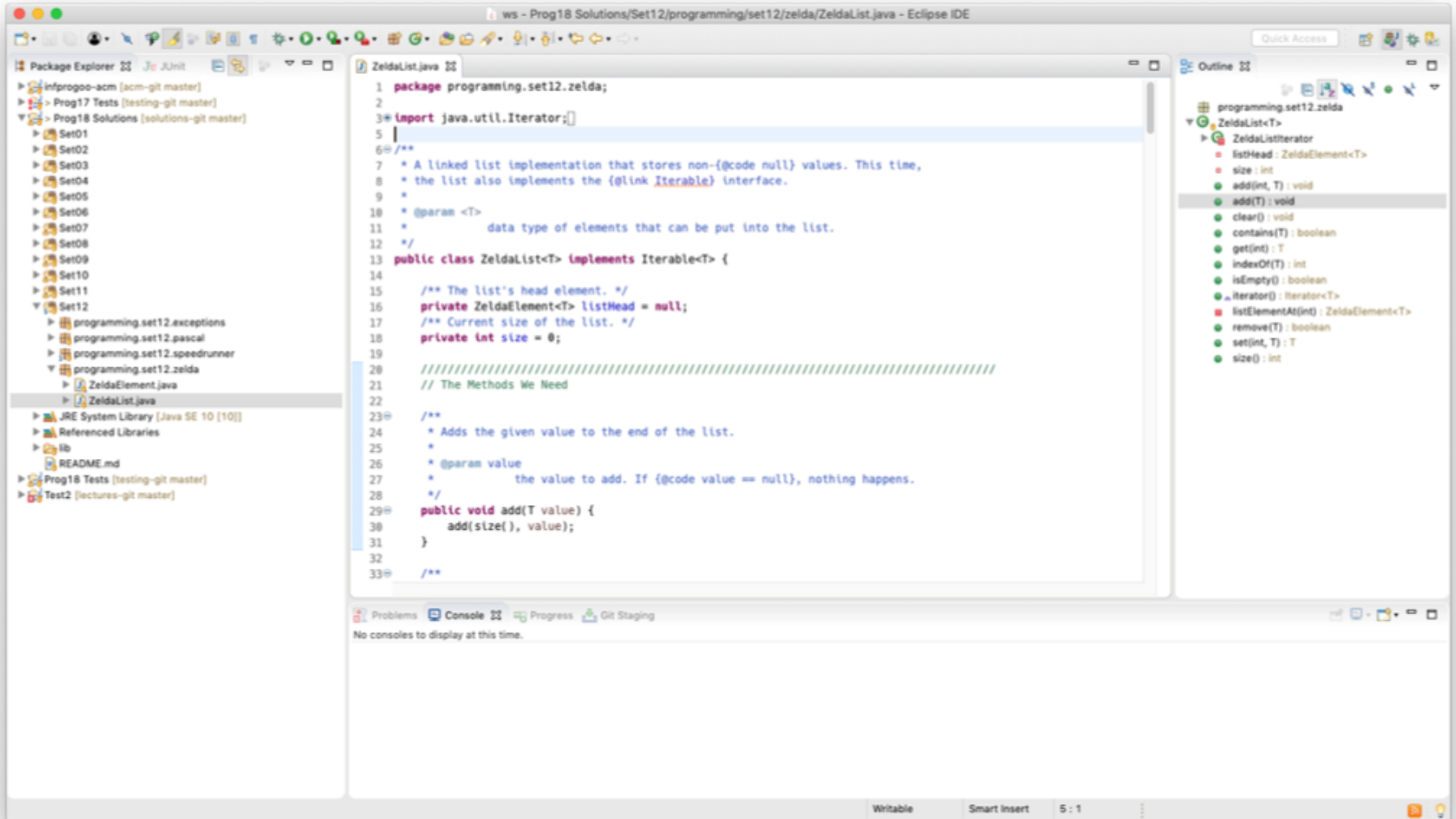
Easier to work with than Swing.

Since removed from JDK,
but still exists as OpenJFX.

Verdict

Nicer GUIs, harder to setup.

SWT



Eclipse

SWT

Platform-specific controls.

Feels fast and integrates well with the OS.

Perhaps a bit more difficult to use.

Verdict

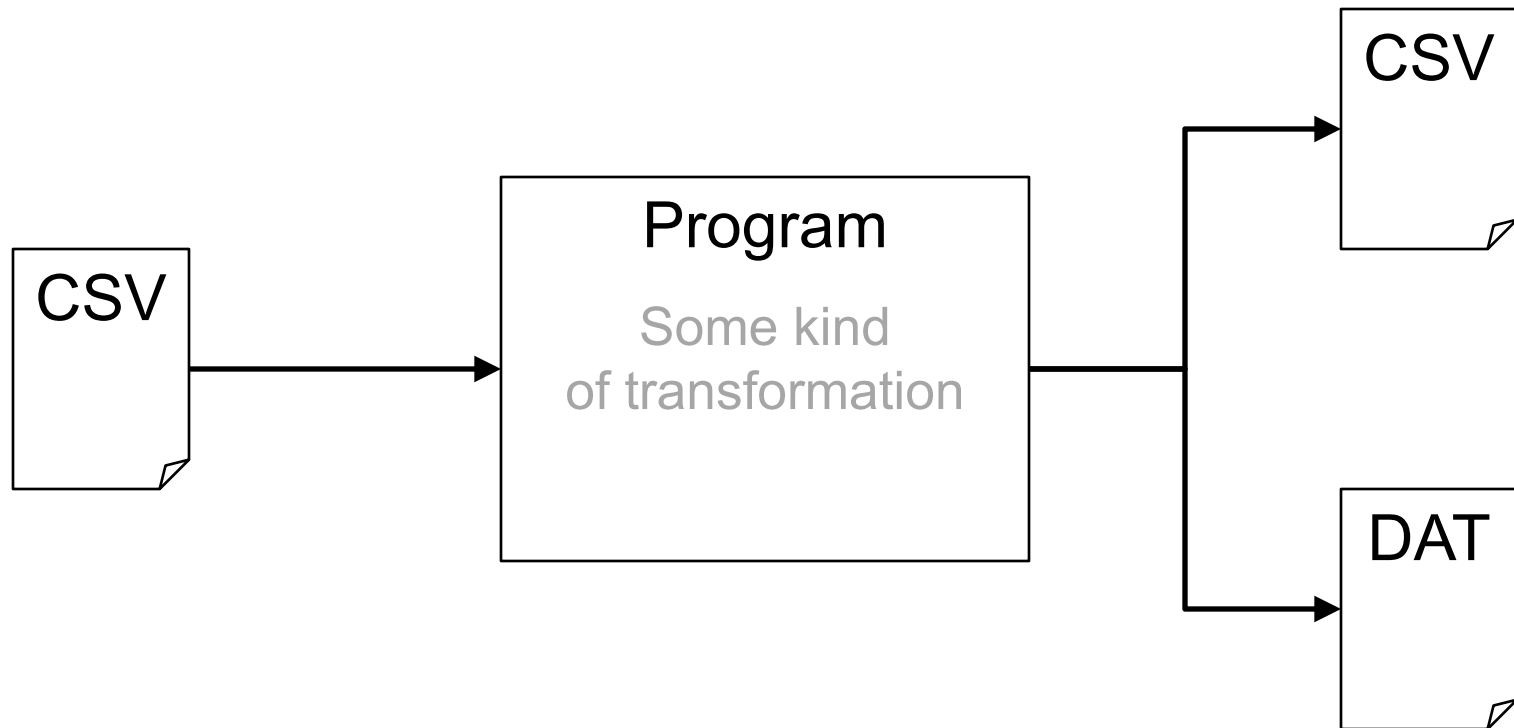
Native feeling, harder to setup.

Programming – Lecture 15

Going Beyond ACM

- Lectures Beyond Inf-ProgOO
- IDEs Beyond Eclipse
- **Projects Beyond Homework**
- Java Beyond ACM
- Programming Beyond Inf-ProgOO

Small Console Programs



Something You're Missing

The screenshot displays the C.R.A.P. (Completely Respectable Algorithmic Program) software interface. The main window, titled "Untitled - C.R.A.P. (Completely Respectable Algorithmic Program)", features a menu bar with "File", "Edit", "View", "Format", "Algorithms", and "Help". Below the menu is a toolbar with icons for file operations and a zoom level set to 100%. The central workspace, labeled "Graph 1", shows a directed graph with 10 nodes (numbered 1-10) and 12 edges. The nodes are arranged on a grid, with node 7 at the bottom center, node 10 in the middle, and nodes 1-6, 8, and 9 scattered around. The edges form a complex network of connections between these nodes.

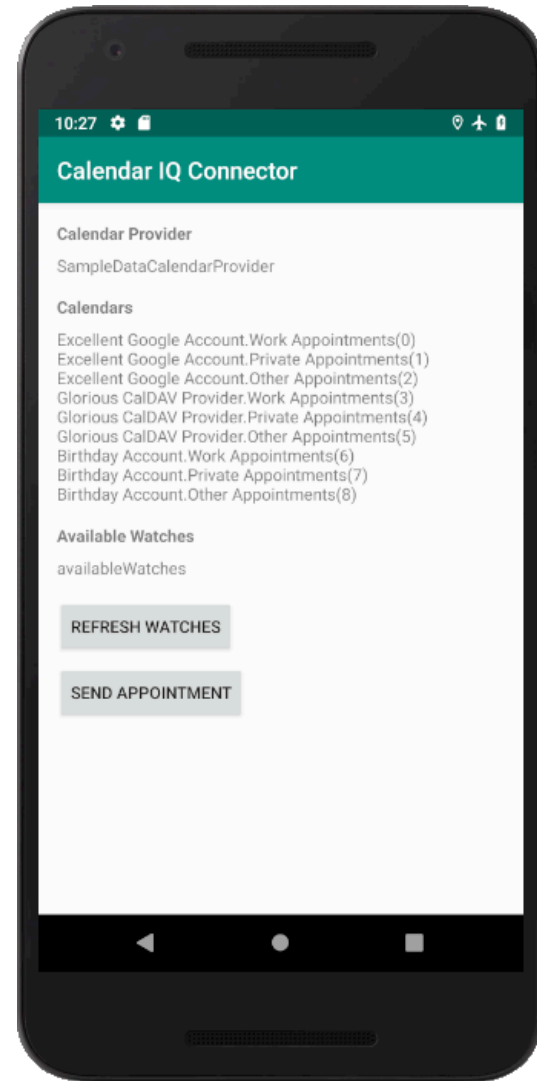
On the right side, an "Online Help" window is open, displaying "Help Contents". The text reads: "Welcome to the online help of C.R.A.P. To make you feel right at home as fast as possible, it is divided into two sections: The first section contains a little hands-on guide that will help getting you started. The second section provides an in-depth explanation of C.R.A.P.'s interface and features." Below this, there are two sections: "Getting Started" with links for "Tutorial 1: Drawing a Graph" and "Tutorial 2: Visualizing an Algorithm", and "The C.R.A.P. Guide" with links for "The Main Window", "Working with Files", and "Loading Files".

At the bottom of the main window, a status bar indicates "Graph Graph 1 properly aligned.", "10 Nodes, 12 Edges", "(432, 238)", and an "Edit" button.

An Android App



Android
Studio



I'm Fresh Out Of Ideas 😞

Thou shalt be helped!

Open Campus

Coworking space

Fablab

Waterkant

Toppoint

Mit Leuten und Werkzeugen
Dinge tun!

Raceyard



Programming – Lecture 15

Going Beyond ACM

- Lectures Beyond Inf-ProgOO
- IDEs Beyond Eclipse
- Projects Beyond Homework
- Java Beyond ACM
- **Programming Beyond Inf-ProgOO**

Don't Be Afraid

We have tried to teach you concepts using Java as our example language.

Now go forth
and learn new languages!

Which Language to Learn?



The Two Main Questions

1

What do I want to build?

2

Where should it run?

Android Development



Kotlin



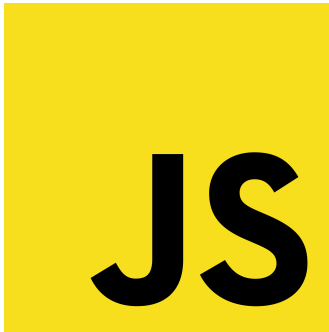
Java

iOS and Mac Development



Swift

Web Development



JavaScript

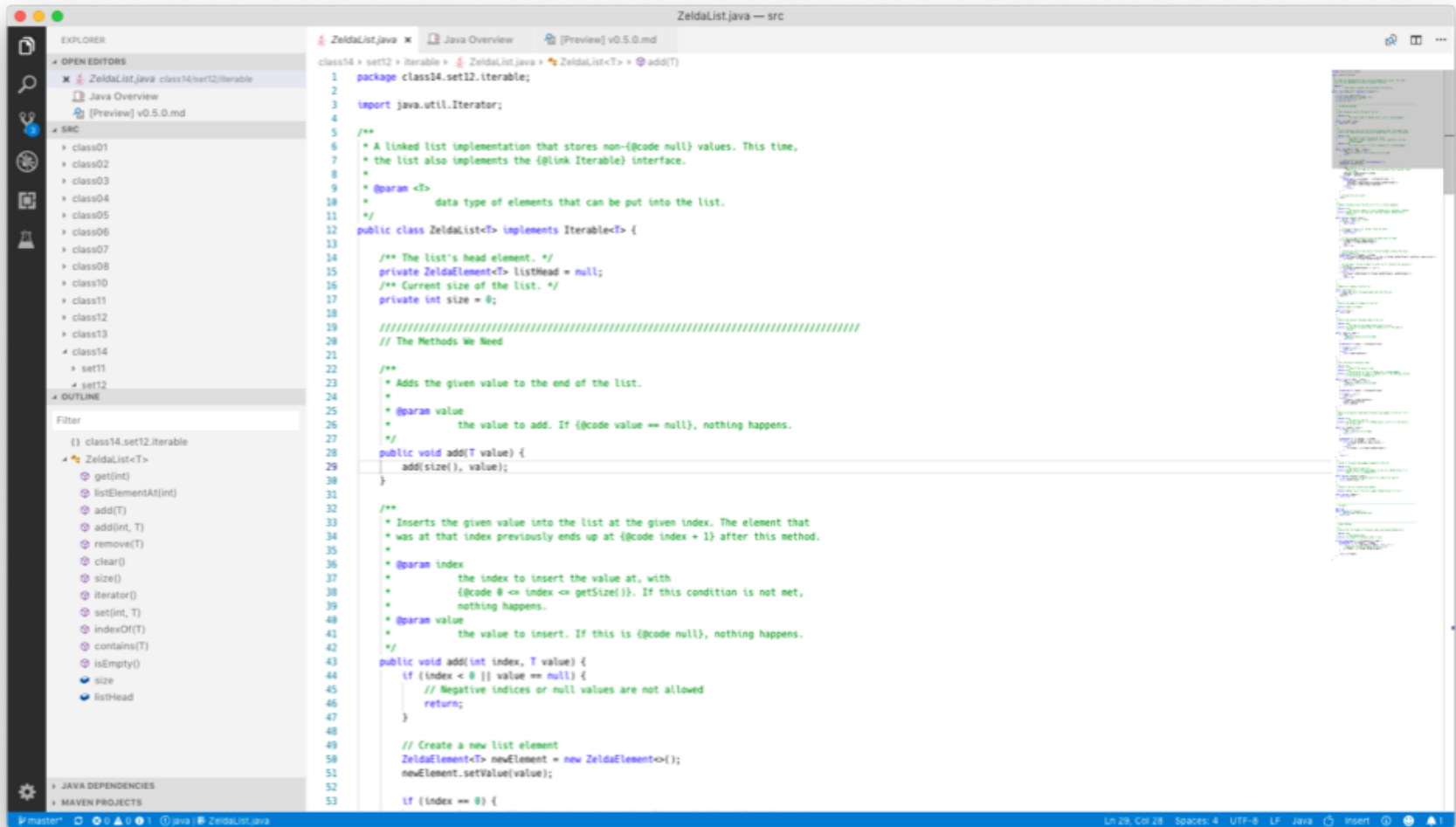


Ruby



PHP

Web Development



Visual Studio Code

Web Development

The screenshot displays the Theia IDE interface. On the left is a file explorer showing a workspace with various files and folders. The central code editor shows the following SCChart code for a test chart:

```
1 scchart test {  
2   signal I  
3   int cnt = 0  
4  
5   region in:  
6  
7   initial state a  
8   go to b if I  
9  
10  state b  
11  go to a do I  
12  
13  region count:  
14  initial state count  
15  go to count if I do cnt++  
16 }
```

The diagram view on the right visualizes this code. It features a main container labeled 'test' with two regions: 'in' and 'count'. The 'in' region contains two states, 'a' and 'b', with a transition from 'a' to 'b' labeled 'I' and a return transition from 'b' to 'a' labeled 'I'. The 'count' region contains a state 'count' with a self-loop transition labeled 'I / cnt++'. The settings panel on the right is expanded to show 'Appearance', 'Dataflow', 'Navigation', and 'Analysis / Debugging' options.

Command-Line Tools



Python



Perl



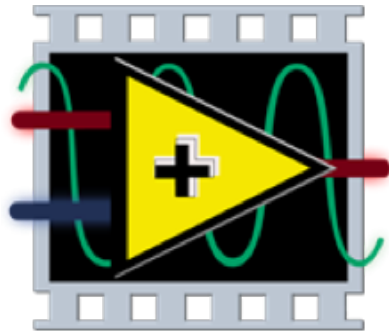
Ruby

Tinkering With Hardware

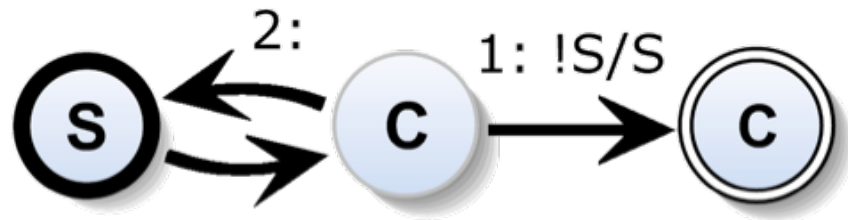


C / C++

Visual Languages

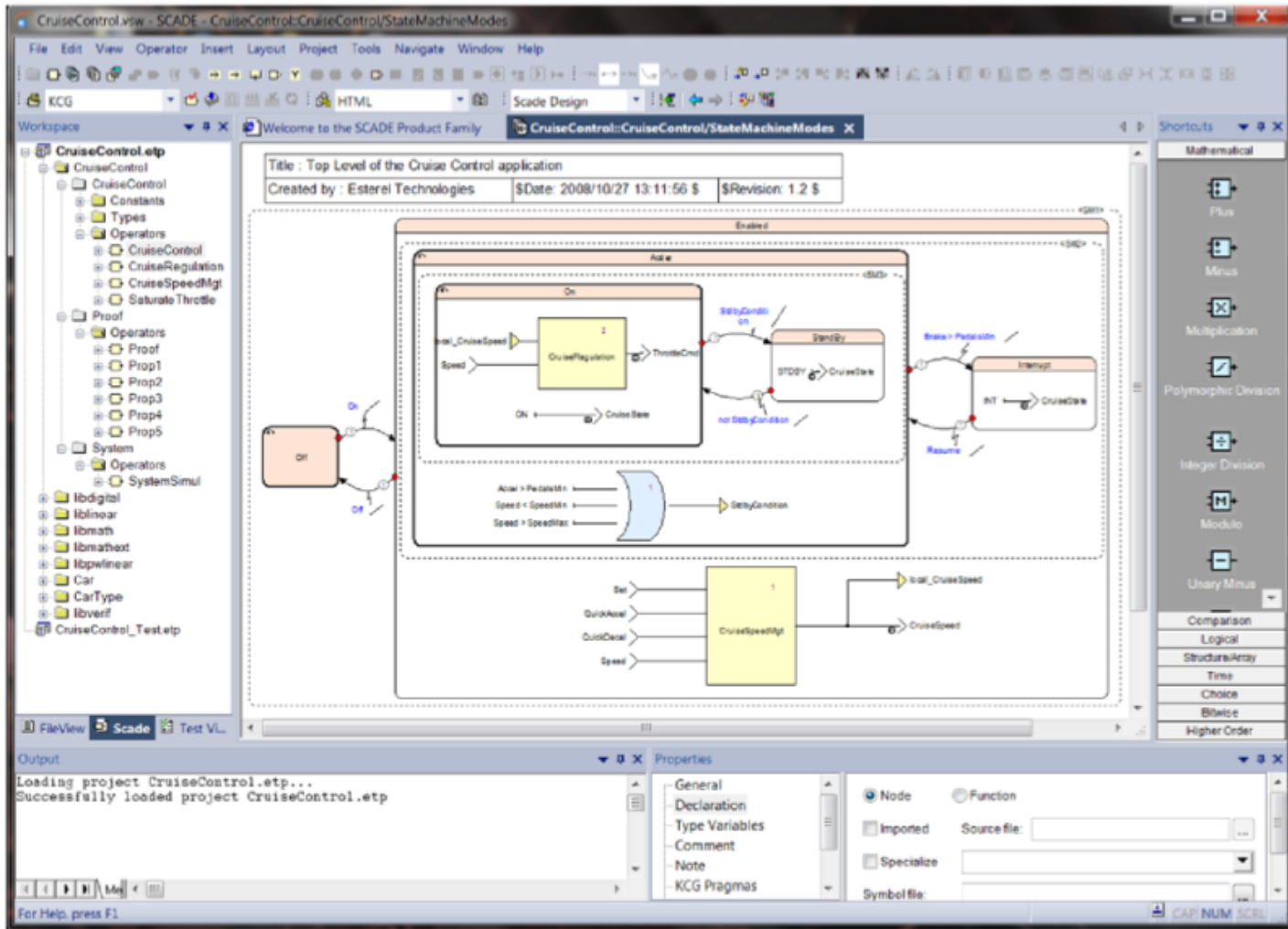


LabVIEW



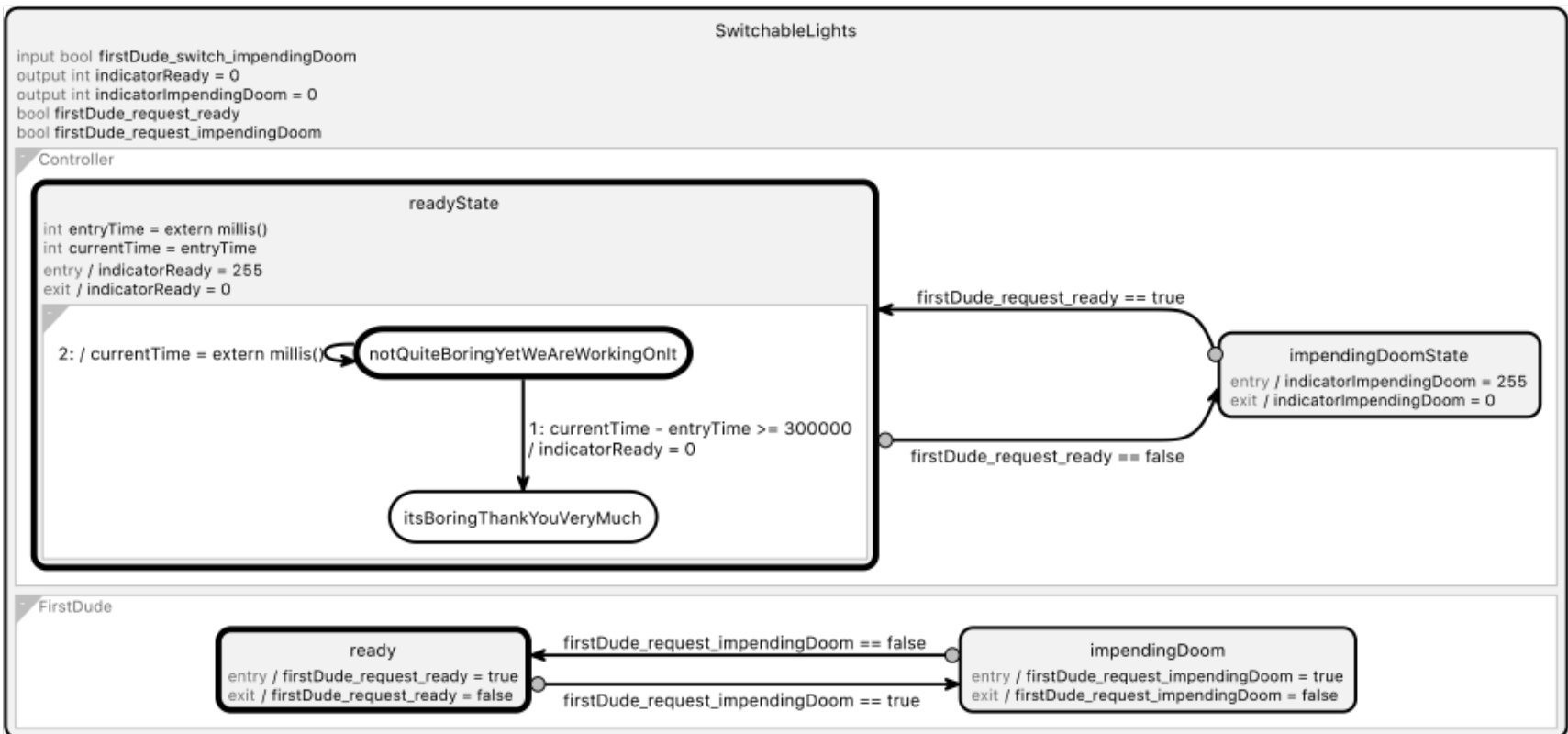
SCCharts

Visual Languages

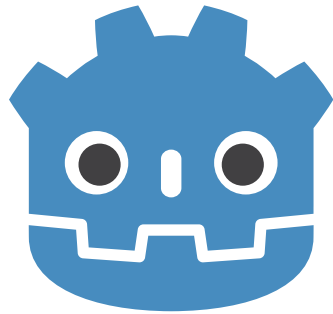


SCADE

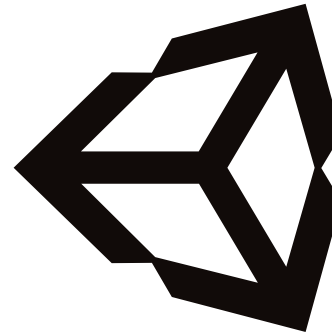
Visual Languages



Games



Godot



Unity

You're just at the start
of your journey.

Now go and enjoy
being programmers!