

Principles of Software Construction: Objects, Design, and Concurrency

IDEs, Build system, Continuous Integration, Libraries

Bogdan Vasilescu

Jonathan Aldrich



Administrivia

Small updates to the homework schedule

- Removed one assignment, adjusted others
- Responds to workload variations in last semester's edition of the class

Quiz Time

Under “Quizzes → Lecture 3 Quiz” on Canvas.

What did we talk about Thursday?

Productivity Requires Automation Requires Abstraction



Productivity Requires Automation Requires Abstraction

```
C++ source #1
1 // Type your code here, or load an example.
2 int square(int num) {
3     return num * num;
4 }

#1 with MSP430 gcc 4.5.3 x
MSP430 gcc 4.5.3 compiler options...
11010 .LX0: .text // Intel
1
2 /*****
3  * Function `square(int)'
4  *****/
5 square(int):
6     push    r10
7     push    r4
8     mov     r1, r4
9     add     #4, r4
10    sub     #2, r1
11    mov     r15, -6(r4)
12    mov     -6(r4), r10
13    mov     -6(r4), r12
14    call    #__mulhi3
15    mov     r14, r15
16    add     #2, r1
17    pop     r4
18    pop     r10
19    ret
20 ;; End of function
```

Automation Requires Abstraction

We all treat familiar levels of abstraction as normal/natural

- That's fine if you only do the laundry
 - Not so much if you are an appliance mechanic
 - How to debug a broken washing machine?
- Also slow to evolve
 - e.g. the shift to concentrated laundry detergent
- Abstractions simplify engineering work, and allow engineers to focus on the hard parts.
 - But, effective engineers also know what is beneath the abstractions

Automation Requires Abstraction

Today's "normal":

- Integrated-development environments (IDEs) galore
 - Web-based too! Press "." on a GitHub (file) page 🤖
- Frequent build, test, release
 - In some companies, every commit is a "release"
- Never write code for which there is a useful library
 - Define "useful"?
- All of the above, entangled

Abstraction, Reuse, and Programming Tools

- For each in {Build systems, IDE, libraries, CI}:
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- What is next?

Abstraction, Reuse, and Programming Tools

- For each in {**Build systems**, IDE, libraries, CI}:
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- What is next?

Quick overview of today's toolchain: Build Systems

How does this happen?

The image shows a side-by-side comparison of C++ source code and its assembly output. On the left, a window titled 'C++ source #1' contains the following code:

```
1 // Type your code here, or load an example.
2 int square(int num) {
3     return num * num;
4 }
```

On the right, a window titled '#1 with MSP430 gcc 4.5.3' shows the assembly output for the same code. The assembly is for the MSP430 architecture, version 4.5.3. The output is as follows:

```
11010 .LXD: .text // Intel
1
2 /*****
3  * Function `square(int)'
4  *****/
5 square(int):
6     push    r10
7     push    r4
8     mov     r1, r4
9     add     #4, r4
10    sub     #2, r1
11    mov     r15, -6(r4)
12    mov     -6(r4), r10
13    mov     -6(r4), r12
14    call   #__mulhi3
15    mov     r14, r15
16    add     #2, r1
```

This is Java code!

Starting a program: Java

All Java code is in classes, so how to create an object and call a method?

Special syntax for *main* method in class (`java X` calls *main* in *X*)

```
// start with: java Printer
```

```
class Printer {
```

```
void print() {
```

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

```
}
```

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

```
    Printer obj = new Printer();
```

```
    obj.print();
```

```
}
```

```
}
```

in Java,
everything is
a class

main must be
public and
static

Main method to be
executed, here used to
create object and invoke
method

Static methods belong to
class not the object,
generally avoid them

This is Typescript code!

Starting a program: Javascript

Objects do not do anything on their own, they wait for method calls

Every program needs a starting point, or waits for events

```
// start with: node file.js
function createPrinter() {
  return {
    print: function() { console.log("hi"); }
  }
}
const printer = createPrinter();
printer.print()
// hi
```

Defining interfaces,
functions, classes

Starting:
Creating objects and
calling methods

SimpleHello.java/SimpleHello.ts

- Java:
 - javac (or your IDE) compiles .java files to .class files
 - java runs the .class files
 - Command line execution vs. in VSCode
 - What's the -cp doing?

SimpleHello.java/SimpleHello.ts

- Java:
 - javac (or your IDE) compiles .java files to .class files
 - java runs the .class files
 - Command line execution vs. in VSCode
 - -cp gives a path to the classes directory tree
 - Organized by package just like Java source
 - -cp also includes .jar files for libraries collected by maven
 - A .jar file is a zipped up tree of .class files, plus some metadata

SimpleHello.java/SimpleHello.ts

- Java:
 - javac (or your IDE) compiles .java files to .class files
 - java runs the .class files
 - Command line execution vs. in VSCode
 - -cp gives a path to the classes directory tree & .jar files for libraries
- TypeScript:
 - Is transpiled to JavaScript, which is then executed inside a runtime environment.
 - Outside the browser: we use node.js
 - Node looks for modules in the node_modules directory
 - How do modules get there? from npm
 - npm uses package.json to track modules we depend on & their versions
- Why do we encourage you to run everything in an IDE/use Maven/npm?
 - Hints: what are the roles of -cp in Java? Or node_modules in typescript?

Quick overview of today's toolchain: Build Systems

Compiling is “easy” when all your source code is here

Nowadays, your code is not “here”

- Even libraries that you use in the IDE!
- Interfaces make that possible

```
217 [INFO] -----< org.example:FlashCards >-----
218 [INFO] Building FlashCards 1.0-SNAPSHOT
219 [INFO] -----[ jar ]-----
220 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-
221 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-
222 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/23/maven-
223 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/23/maven-
224 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/22/maven-parent-
225 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/22/maven-parent-
226 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/11/apache-11.pom
227 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/11/apache-11.pom (15 kB at 1.2
228 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-
229 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-
230 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-compiler-
231 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-compiler-
232 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/24/maven-
233 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/24/maven-
234 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/23/maven-parent-
235 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/23/maven-parent-
236 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/13/apache-13.pom
237 [INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/13/apache-13.pom (14 kB at 688
238 [INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-compiler-
```

Quick overview of today's toolchain: Build Systems

- Building a few basic tasks:
 - Compiling & linking, to produce an executable
 - Creating secondary *artifacts*, e.g. documentation-pages, linter reports, test suite reports
 - Different levels of “depth” may be appropriate, for large code bases (e.g. Google)
- Popular options:
 - For Java: Maven and Gradle -- historically Ant.
 - You could do any homework in either, in principle; we just standardize to make helping you and grading easier.
 - For JS/TS: Node(JS)
 - Generally coupled with the Node Package Manager (NPM)
- Often built into IDEs, as plugins

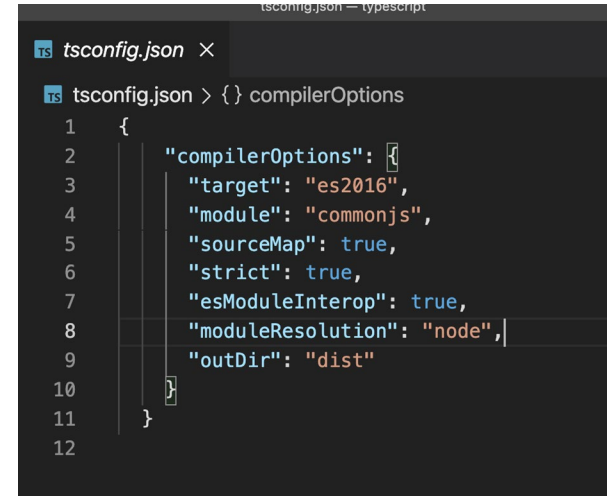
Under the Hood: Build Systems

- These days: intricately tied with IDEs, package managers
- Projects often come with a build config file or two
 - 'pom.xml' for Maven
 - 'tsconfig.json' + 'package.json' for TypeScript+NPM -- the second deals with packages
 - These can be nested, one per (sub-)directory, to compose larger systems
 - On GitHub, you can create links across repositories

Under the Hood: Build Systems

Projects typically require build config files

- Checked into source control
 - 'pom.xml' for Maven
 - 'tsconfig.json' + 'package.json' for TypeScript+NPM -- the second deals with packages
- Relevant components:
 - Compilation source and target version
 - High-level configuration options
 - Targets for various development phases
 - Relevant plugins
 - Dependencies with versions



```
tsconfig.json — typescript
tsconfig.json x
tsconfig.json > {} compilerOptions
1  {
2    "compilerOptions": {
3      "target": "es2016",
4      "module": "commonjs",
5      "sourceMap": true,
6      "strict": true,
7      "esModuleInterop": true,
8      "moduleResolution": "node",
9      "outDir": "dist"
10   }
11 }
12
```



```
m pom.xml (FlashCards)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <modelVersion>4.0.0</modelVersion>
```

Maven Phases

Although hardly a comprehensive list, these are the most common *default* lifecycle phases executed.

- **validate**: validate the project is correct and all necessary information is available
- **compile**: compile the source code of the project
- **test**: test the compiled source code using a suitable unit testing framework. These tests should not require the code
- **package**: take the compiled code and package it in its distributable format, such as a JAR.
- **integration-test**: process and deploy the package if necessary into an environment where integration tests can be
- **verify**: run any checks to verify the package is valid and meets quality criteria
- **install**: install the package into the local repository, for use as a dependency in other projects locally
- **deploy**: done in an integration or release environment, copies the final package to the remote repository for sharing

There are two other Maven lifecycles of note beyond the *default* list above. They are

- **clean**: cleans up artifacts created by prior builds
- **site**: generates site documentation for this project

<https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>

```
31 <version>RELEASE</version>
32 <scope>test</scope>
33 </dependency>
project > dependencies > dependency
Build Dependencies
```



- Node.js is a JS runtime. npm is its package manager.

```
package.json 1, M ×
{} package.json > {} dependencies
1  {
2    "name": "hw1-flashcards",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    > Debug
7    "scripts": {
8      "compile": "tsc",
9      "lint": "ts-standard",
10     "start": "node dist/index.js"
11   },
12   "author": "",
13   "license": "ISC",
14   "devDependencies": {
15     "@types/node": "^17.0.8",
16     "@types/readline-sync": "^1.4.4",
17     "ts-standard": "^10.0.0",
18     "typescript": "^4.4.2"
19   },
20   "dependencies": {
21     "readline-sync": "^1.4.10",
22   }
23 }
```

EXPLORER

▼ TYPESCRIPT

- ▼ .vscode
 - { launch.json
- > cards
- > dist
- > node_modules
- ▼ src
 - > cards
 - > data
 - > ordering
- TS index.ts
- TS ui.ts
- ◆ .gitignore
- { package-lock.json
- { package.json
- i README.md
- ts tsconfig.json

> OUTLINE

> TIMELINE

- Start Debugging F5
- Run Without Debugging ^ F5
- Stop Debugging ⌘ F5
- Restart Debugging ⌘ ⇧ F5
- Open Configurations
- Add Configuration...**
- Step Over F10
- Step Into F11
- Step Out ⇧ F11
- Continue F5
- Toggle Breakpoint F9
- New Breakpoint >
- Enable All Breakpoints
- Disable All Breakpoints
- Remove All Breakpoints
- Install Additional Debuggers...

```
launch.json — typescript
package.json tsconfig.json launch.json U X
Launch Targets > {} Launch Program
IntelliSense to learn about possible attributes.
ver to view descriptions of existing attributes.
r more information, visit: https://go.microsoft.com/fwlink/?linkid=
ion": "0.2.0",
igurations": [
  {
    "type": "pwa-node",
    "request": "launch",
    "name": "Launch Program",
    "skipFiles": [
      "<node_internals>/**"
    ],
    "program": "${file}",
    "console": "integratedTerminal",
    "preLaunchTask": "tsc: build - tsconfig.json",
  }
]
```

Add Configuration...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
5t8ng5pd3cdtm0000gn/T/node-cdp.27252-4.sock", "deferredMode": false, "waitForDebugger": "", "execPath": "/Users/clegoues/.nvm/versions/node/v17.3.0/bin/node", "onlyEntrypoint": false, "autoAttachMode": "always", "fileCallback": "/var/folders/6y/3yk0l2ms1yn5t8ng5pd3cdtm0000gn/T/node-debug-callback-b1bc858d1e8e06c9"}' /Users/clegoues/.nvm/versions/node/v17.3.0/bin/node ./dist/index.js
Debugger attached.
7 cards to go...

Next cue: Describes a group of objects that are treated the same way as a single instance of the same type of object.
answer> []
```

npm
build - t... ✓
Launch Pro...



.vscode > { } launch.json > Launch Targets > { } Launch Program

```
1 {
2   // Use IntelliSense to learn about
3   // Hover to view descriptions of e
4   // For more information, visit: ht
5   "version": "0.2.0",
6   "configurations": [
7     {
8       "type": "pwa-node",
9       "request": "launch",
10      "name": "Launch Program",
11      "skipFiles": [
12        "<node_internals>/**"
13      ],
14      "program": "${file}",
15      "console": "integratedTerminal",
16      "preLaunchTask": "tsc: build - tsconfig.json",
17      "args": [ "--help" ],
18      "outFiles": [
19        "${workspaceFolder}/dist/**/*.js"
20      ]
21    }
22  ]
23 }
24 }
```

```
node/v17.3.0/bin/node ./dist/index.js --help
Debugger attached.
zsh: killed /usr/bin/env /Users/clegoues/.nvm/versions/node/v17.3.0/bin/n
ode --help
(base) clegoues@clegoues-macbook-air typescript % npm run start -- --help[]
```


RUN AND DEBUG

RUN

Run and Debug

To customize Run and Debug
create a launch.json file.

Show an automatic debug
configurations.

Main.java

ClassLoader.class

ClassLoaders.class x

```

171     }
172
173     @Override
174     protected Class<?> loadClass(String cn, boolean resolve)
175         throws ClassNotFoundException
176     {
177         // for compatibility reasons, say where restricted package list has
178         // been updated to list API packages in the unnamed module.
179         @SuppressWarnings("removal")
180         SecurityManager sm = System.getSecurityManager();
181         if (sm != null) {
182             int i = cn.lastIndexOf(ch: '.');
183             if (i != -1) {
184                 sm.checkPackageAccess(cn.substring(beginIndex: 0, i));
185             }
186         }
187
188         return super.loadClass(cn, resolve);
189     }

```

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

JUPYTER

Debug: Main + - □ □ ×

```

/usr/bin/env /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:58115 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/clegoues/courses/17-214/lecture-stuff/hw1-flashcards-clegoues/java/target/classes edu.cmu.cs214.hw1.Main
(base) clegoues@clegoues-macbook-air java % /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:58115 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/clegoues/courses/17-214/lecture-stuff/hw1-flashcards-clegoues/java/target/classes edu.cmu.cs214.hw1.Main
Next cue: Enables selecting an algorithm at runtime by providing a corresponding object implementing the algorithm.
(base) clegoues@clegoues-macbook-air java % cd /Users/clegoues/courses/17-214/lecture-stuff/hw1-flashcards-clegoues/java ; /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:58136 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/clegoues/courses/17-214/lecture-stuff/hw1-flashcards-clegoues/java/target/classes edu.cmu.cs214.hw1.Main
(base) clegoues@clegoues-macbook-air java %

```

BREAKPOINTS

 Uncaught Exceptions

 Caught Exceptions

 Main.java src/main/java/e... 21

What's going on in HW1?

Abstraction, Reuse, and Programming Tools

- For each in {Build systems, IDE, libraries, CI}:
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- What is next?

Abstraction, Reuse, and Programming Tools

- For each in {Build systems, **IDE**, libraries, CI}:
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- What is next?

Quick overview of today's toolchain: IDEs

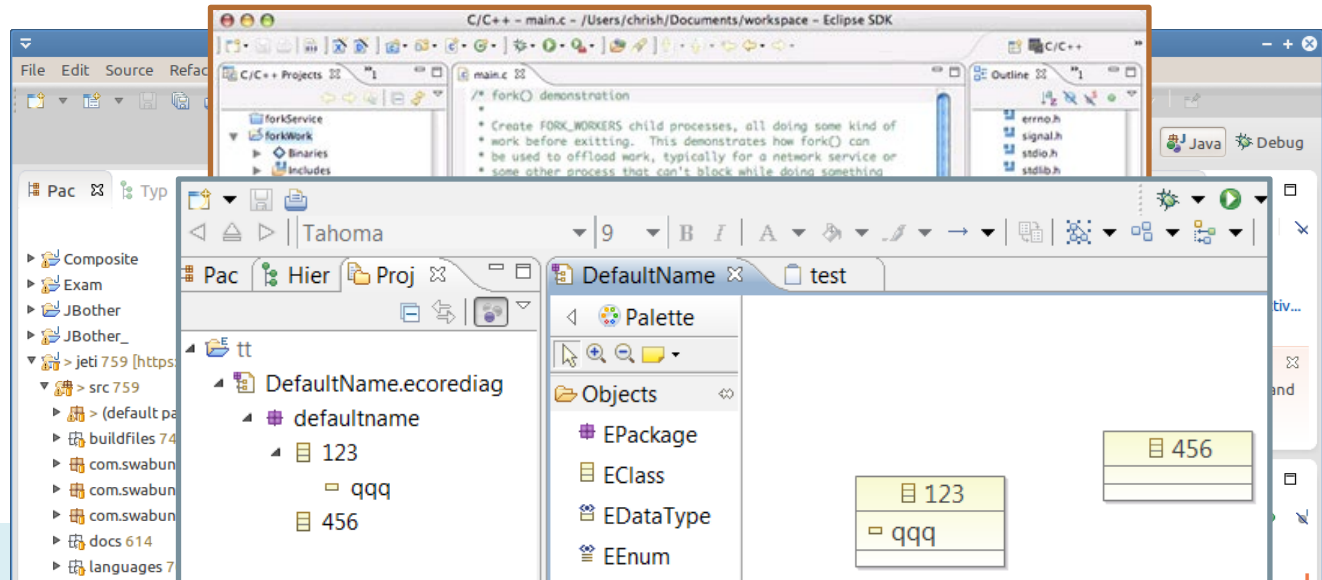
Integrated Development Environments bundle development workflows in a single UI

- Editing, refactoring, running & debugging, adding dependencies, compiling, deploying, plugins, you name it
- They often try to be everything, with mixed results
- Leverage them to the fullest extent, to automate and check your work

IDEs are key to *managing complexity*.

Quick overview of today's toolchain: IDEs

Eclipse was the dominant player in Java for 20-odd years, owing to its powerful backbone and plugin architecture



Quick overview of today's toolchain: IDEs

Recently, IntelliJ has been more dominant

- Packs a lot of “recipes” to create certain types of projects (e.g., web-app with Spring & Maven)

VSCode is surging in popularity

- Local & web, lightweight but with a massive plugin ecosystem

In general: choose based on need!

- You can relearn key-bindings; “killer features” are rare and temporary
- E.g., Android: might want Android Studio (itself built on IntelliJ) since Google supports it

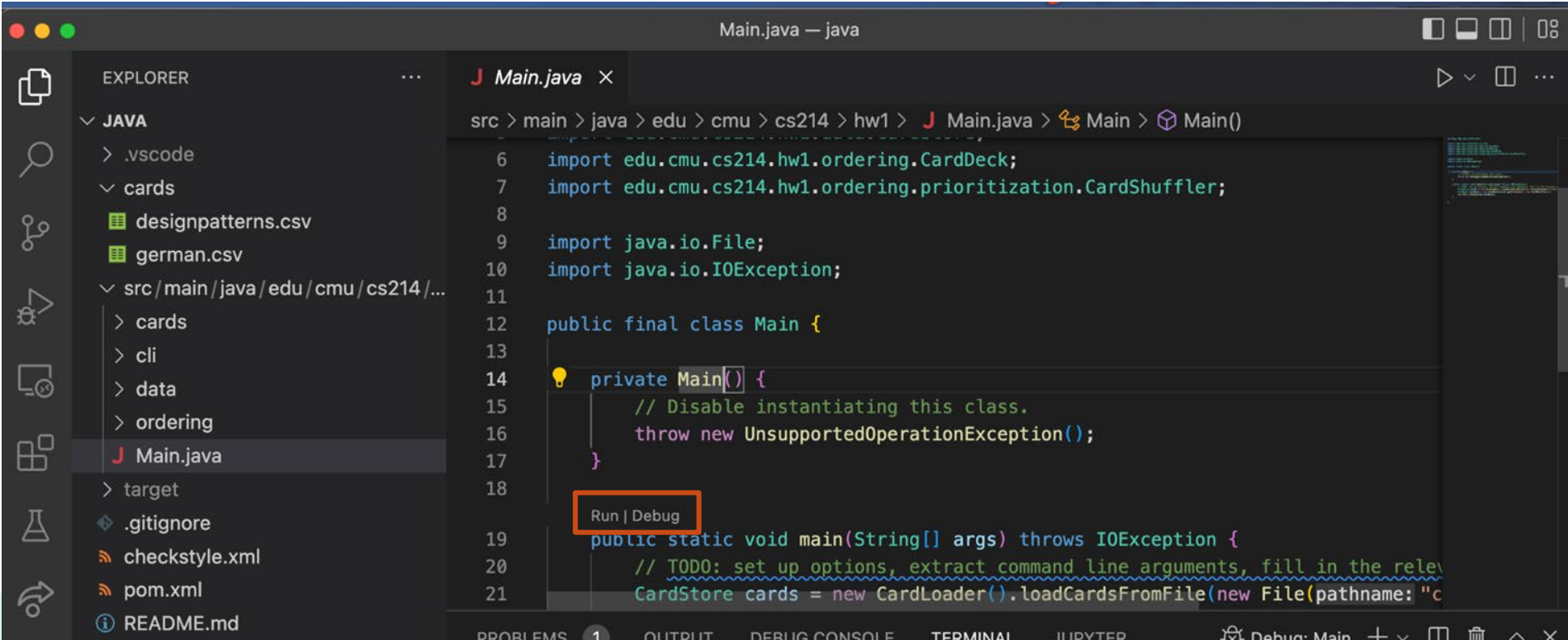
We recommended VSCode because you're programming in two languages, it's very current, and then it's easier for us to help you.

Under the Hood: IDEs

- The engine: continuous parsing, building
 - Key feature: most partial programs don't parse, but IDEs make sense of them
 - That allows quickly relaying compile warnings/errors and useful suggestions
 - Same with API resolution
- Powered by rapid incremental compilation
 - Only build what has been updated
 - Virtually every edit you make triggers a compilation, re-linking
 - Of just the changed code and its dependencies
 - Works because *very little* of the code changes most of the time
 - But no free lunch: tends to drop optimizations (mostly fine), may struggle with big projects
 - Just try it: call an API with the wrong parameters & see how fast it triggers an alert; contrast with running a full Maven build (e.g., with ``mvn install``)

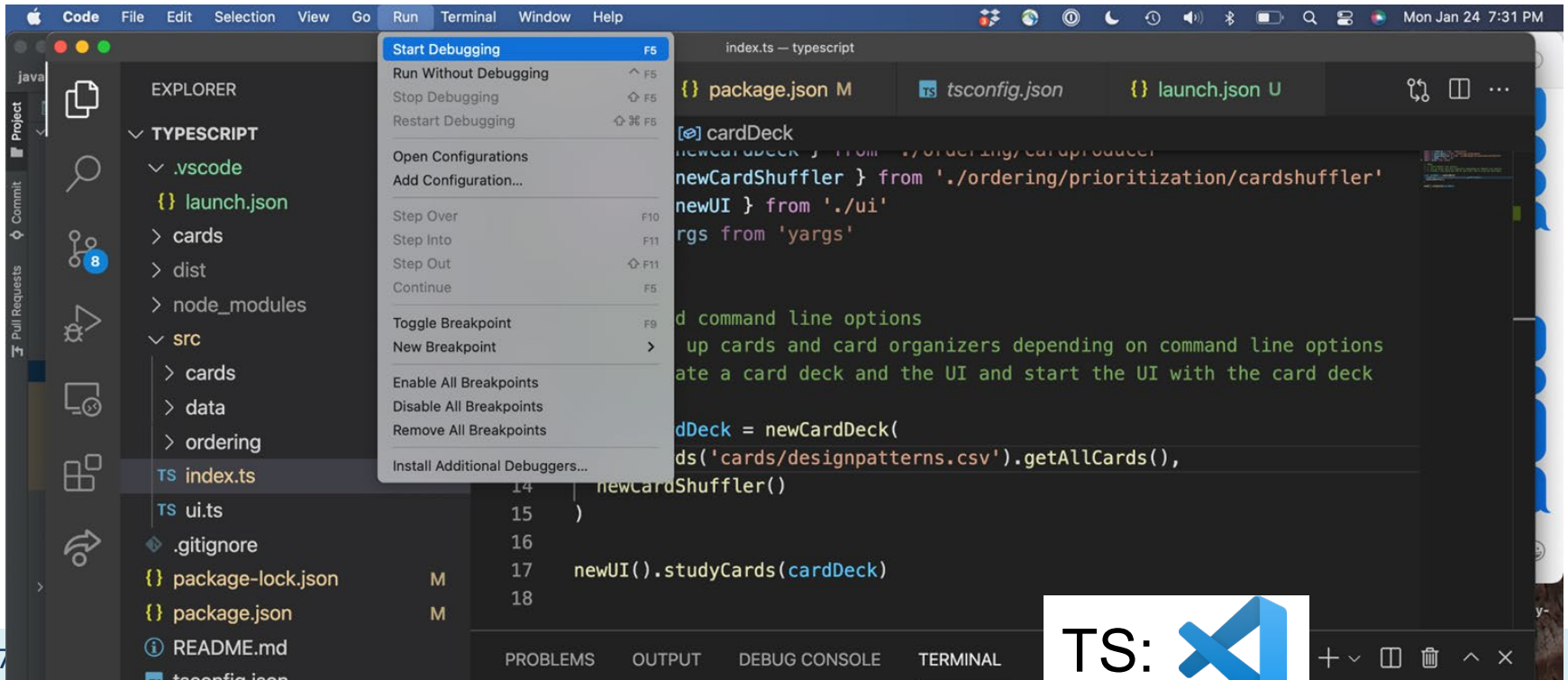
Under the Hood: IDEs

Automate common programming actions, like debugging, which is often the default mode when you run in the IDE (like in VSCode)



Under the Hood: IDEs

Debugging allows setting breakpoints in the GUI, access to rich execution info.



Code File Edit Selection View Go Run Terminal Window Help

index.ts — typescript

Launch Program

index.ts M x package.json M tsconfig.json launch.json U

VARIABLES

- Local
 - cardDeck: undefined
 - cardproducer_1: {newCardDec...
 - cardshuffler_1: {newCardShu...
 - exports: {__esModule: true}
 - module: Module {file: /...
- WATCH
- CALL STACK
 - La... PAUSED ON BREAKPOINT
 - <anonymous> src/index.ts
 - Show 6 More: Skipped by skipFi
 - LOADED SCRIPTS
 - BREAKPOINTS
 - Caught Exceptions
 - Uncaught Exceptions

index.ts > cardDeck

```
import { newCardDeck } from './ordering/cardproducer'
import { newCardShuffler } from './ordering/prioritization/cardshuffler'
import { newUI } from './ui'
import yargs from 'yargs'

// TODOs
// 1. load command line options
// 2. set up cards and card organizers depending on command line options
// 3. create a card deck and the UI and start the UI with the card deck

const cardDeck = newCardDeck(
  loadCards('cards/designpatterns.csv').getAllCards(),
  newCardShuffler()
)

newUI().studyCards(cardDeck)
```

Debugger attached.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
S=—require "/Applications/Visual Studio Code.app/Contents/Resources/app/ex
tensions/ms-vscode.js-debug/src/bootloader.bundle.js" --inspect-publish-uid
=http' 'VSCODE_INSPECTOR_OPTIONS={"inspectorIpc":"/var/folders/6y/3yk0l2ms1
yn5t8ng5pd3cdtm0000gn/T/node-cdp.29317-1.sock", "deferredMode": false, "waitFo
rDebugger": "", "execPath": "/Users/clegoues/.nvm/versions/node/v17.3.0/bin/no
de", "onlyEntrypoint": false, "autoAttachMode": "always", "fileCallback": "/var/f
olders/6y/3yk0l2ms1yn5t8ng5pd3cdtm0000gn/T/node-debug-callback-deb1bd35d68a
bbf0"}' /Users/clegoues/.nvm/versions/node/v17.3.0/bin/node ./dist/index.js
```

npm

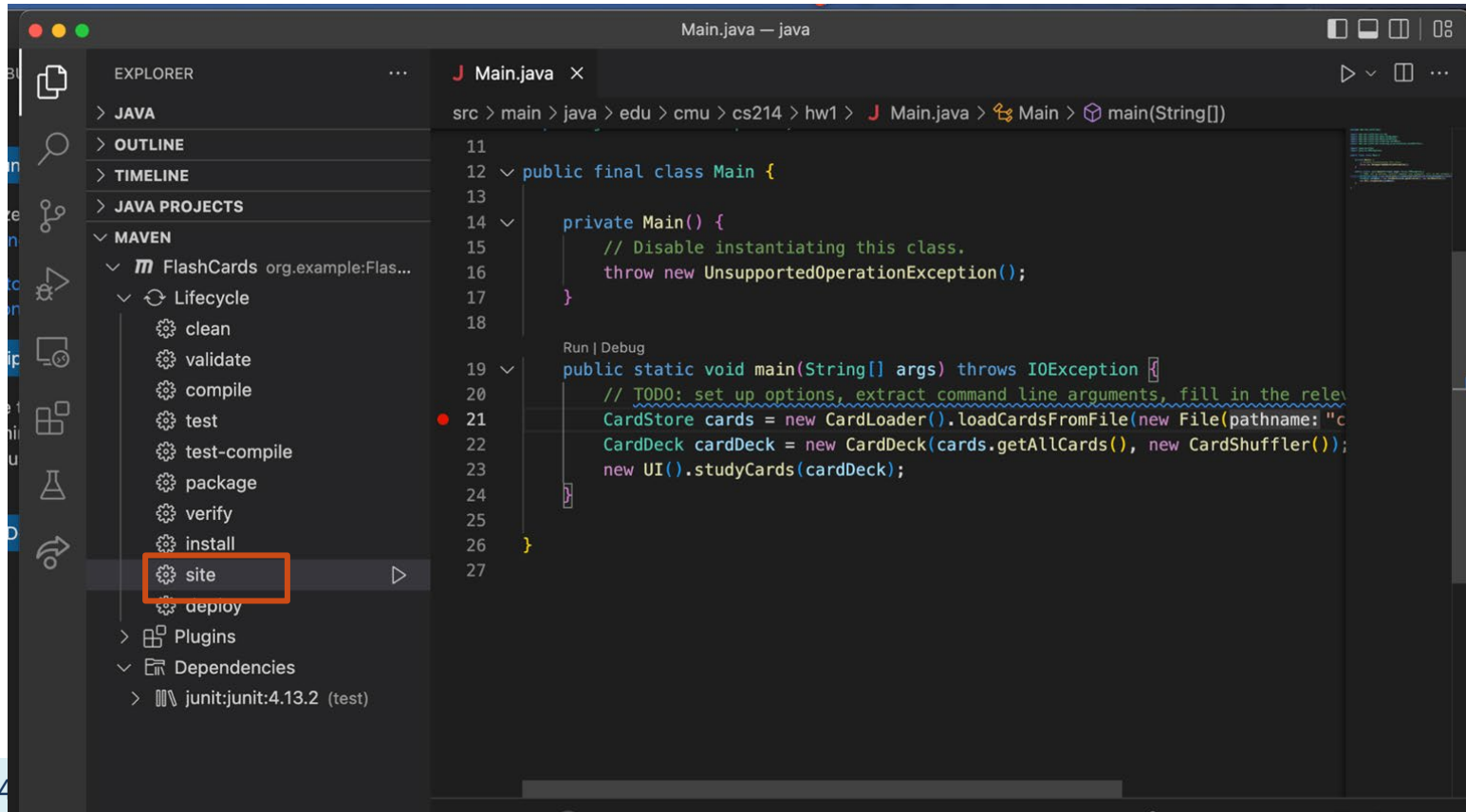
build - t...

Launch Pro...

main* 0 0 0 Launch Program (typescript) Claire Live Share Ln 13, Col 12 Spaces: 2 UTF-8 LF {} TypeScript

Under the Hood: IDEs

Combine build systems + IDEs + plugins (checkstyle example/demo!)

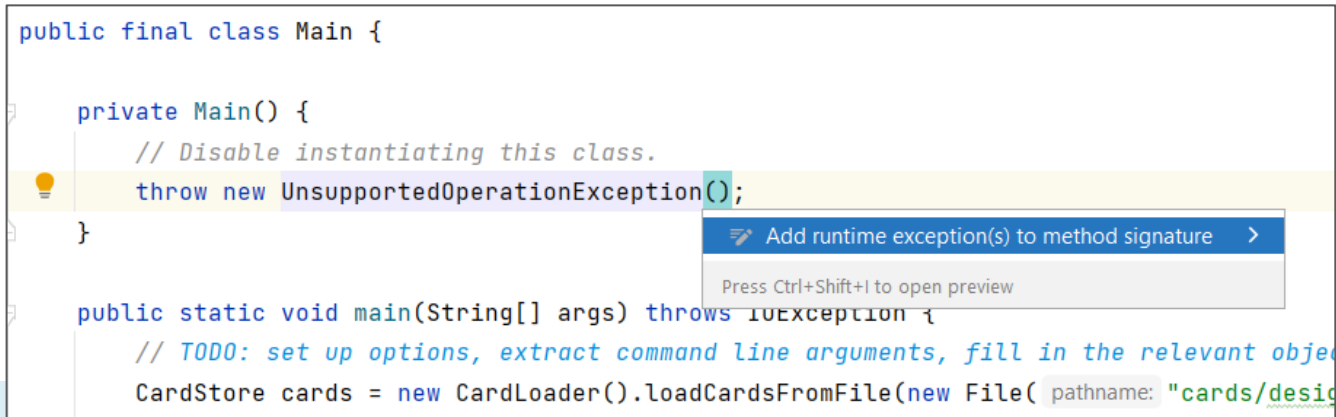


Under the Hood: IDEs

Automate common programming actions:

- Handy refactorings, suggestions
 - E.g., just press `alt+enter` in IntelliJ while highlighting nearly any code
 - Keyboard shortcuts are super useful: explore your IDE!
 - These can make you a better programmer: encode a lot of best-practices
 - Though, don't read into them too much

```
public final class Main {  
  
    private Main() {  
        // Disable instantiating this class.  
        throw new UnsupportedOperationException();  
    }  
  
    public static void main(String[] args) throws IOException {  
        // TODO: set up options, extract command line arguments, fill in the relevant objects  
        CardStore cards = new CardLoader().loadCardsFromFile(new File("cards/design"))  
    }  
}
```

A screenshot of the IntelliJ IDEA IDE showing a code completion popup. The code is in a Java class named 'Main'. The method 'private Main()' is highlighted, and the line 'throw new UnsupportedOperationException();' is selected. A yellow tooltip is visible over the line, containing a lightbulb icon and the text 'Add runtime exception(s) to method signature >'. Below the tooltip, there is a smaller grey box with the text 'Press Ctrl+Shift+I to open preview'. The rest of the code is visible in the background, including a 'main' method and a 'CardStore' object.

Under the Hood: IDEs

- IDE designers spend a lot of time automating common development tasks
 - Sometimes they get a little too helpful (modifying pom's)
 - Many plugins provide customized experience
 - Mostly evolve with new tools, prioritizing emerging routines
- Useful to know how these actions work
 - Often not much more than invoking commands for you
 - VSCode is very explicit about this in the terminal -- great for customization

```
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ ID  
Process finished with exit code 0
```

Abstraction, Reuse, and Programming Tools

- For each in {Build systems, IDE, **libraries**, CI}:
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- What is next?

Quick overview of today's toolchain: Libraries




Reusable packages of code.



Publicly hosted on various *package managers*

- Often tied, but not inextricably linked, to build tools, and languages
- Maven/Gradle for Java, NPM for JS/TS, Nuget for C#, ...
- Cross-language package registries, e.g., GitHub Packages

Easy publishing

Use industry and community-standard package managers with native tooling commands. Then authenticate and publish directly to GitHub.

The right side of the slide shows a screenshot of the GitHub Packages interface. It displays a search for 'github-packages' with a version of 2.0.0. Below the search results, it shows the repository 'GitHub/github-packages' with 0 stars and 612 forks.

EXPLORER

e.g. spring azure storage

Input keywords to search artifacts from Maven Central Repository. (Press 'Enter' to confirm or 'Escape' to cancel)

✓ JAVA

> .vscode

✓ cards

designpatterns.csv

german.csv

✓ src/main/java/edu/cmu/cs214/...

> cards

> cli

> data

> OUTLINE

> TIMELINE

> JAVA PROJECTS

✓ MAVEN

✓ FlashCards org.example:Flas...

> Lifecycle

> Plugins

✓ Dependencies + ↻

> junit:junit:4.13.2 (test)

```
175  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
@Override  
protected Class<?> loadClass(String cn, boolean resolve)  
    throws ClassNotFoundException  
{  
    // for compatibility reasons, say where restricted package list has  
    // been updated to list API packages in the unnamed module.  
    @SuppressWarnings("removal")  
    SecurityManager sm = System.getSecurityManager();  
    if (sm != null) {  
        int i = cn.lastIndexOf(ch: '.');  
        if (i != -1) {  
            sm.checkPackageAccess(cn.substring(beginIndex: 0, i));  
        }  
    }  
    return super.loadClass(cn, resolve);  
}
```

PROBLEMS 1

OUTPUT

DEBUG CONSOLE

TERMINAL

JUPYTER

Debug: Main +

v

□

🗑

^

x

```
/usr/bin/env /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java -agentlib:j  
dwp=transport=dt_socket,server=n,suspend=y,address=localhost:58115 -XX:+ShowCodeDetailsInExceptionM  
essages -cp /Users/clegoues/courses/17-214/lecture-stuff/hw1-flashcards-clegoues/java/target/classe  
s edu.cmu.cs214.hw1.Main
```

```
(base) clegoues@clegoues-macbook-air java % /usr/bin/env /Library/Java/JavaVirtualMachines/jdk-18.  
0.2.1.jdk/Contents/Home/bin/java -agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=loca  
lhost:58115 -XX:+ShowCodeDetailsInExceptionMessages -cp /Users/clegoues/courses/17-214/lecture-stuf  
f/hw1-flashcards-clegoues/java/target/classes edu.cmu.cs214.hw1.Main
```

```
Next cue: Enables selecting an algorithm at runtime by providing a corresponding object implementin
```

Code File Edit Selection View Go Run Terminal Window Help

index.ts — typescript

EXPLORER

- ▼ TYPESCRIPT
 - ▼ .vscode
 - { } launch.json U
 - > cards
 - > dist
 - > node_modules
 - ▼ src
 - > cards
 - > data
 - > ordering
 - TS index.ts M
 - TS ui.ts
 - ◆ .gitignore
 - { } package-lock.json M
 - { } package.json M
 - 📄 README.md
 - ts tsconfig.json
- ▼ OUTLINE
- ▼ TIMELINE

src > TS index.ts > [] cardDeck

```
2 import { newCardDeck } from './ordering/cardproducer'
3 import { newCardShuffler } from './ordering/prioritization/cardshuffler'
4 import { newUI } from './ui'
5 import yargs from 'yargs'
6
7 // TODOs
8 // 1. load command line options
9 // 2. set up cards and card organizers depending on command line options
10 // 3. create a card deck and the UI and start the UI with the card deck
11
12 const cardDeck = newCardDeck(
13   loadCards('cards/designpatterns.csv').getAllCards(),
14   newCardShuffler()
15 )
16
17 newUI().studyCards(cardDeck)
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
npm
Ignore insecure directories and continue [y] or abort compinit [n]? y
(base) clegoues@clegoues-macbook-air typescript %
(base) clegoues@clegoues-macbook-air typescript % npm i --save-dev @types/yargs
added 2 packages, and audited 281 packages in 800ms
68 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
(base) clegoues@clegoues-macbook-air typescript % npm i --save-dev @types/yargs
```

main* 0 0 Claire Live Share Ln 13, Col 55 Spaces: 2 UTF-8 LF {} TypeScript

Quick overview of today's toolchain: Libraries

- Myriad. Publicly hosted on various *managers*
 - Often tied, but not inextricably linked, to build tools, and languages
 - Maven, Gradle, NPM, Nuget, Docker, ...
 - Cross-language package registries, e.g., GitHub Packages
- Releases are generally fast-paced or frigid
 - Almost all volunteer-based, so support waivers, as does documentation quality
 - Often open-source, so you can check out the status & details on GitHub
 - Beware of vulnerabilities and bugs, esp. with minor-releases and nightly's, old packages

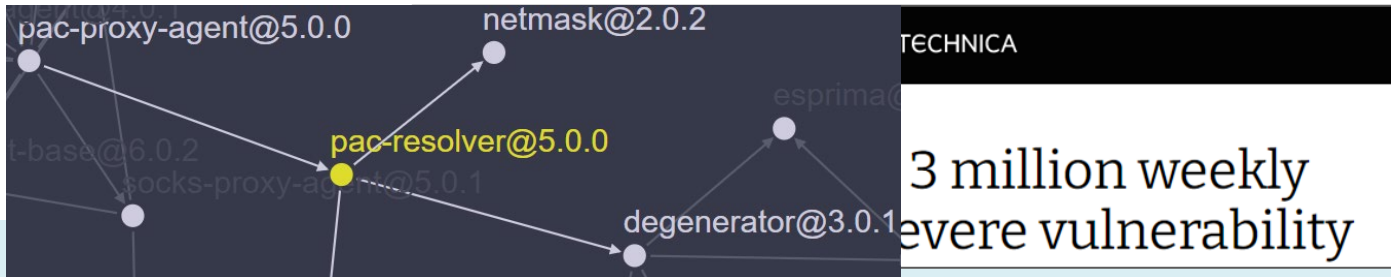


NPM package with 3 million weekly downloads had a severe vulnerability

Quick overview of today's toolchain: Libraries

- A Case-Study:

- 'pac-resolver' (3M weekly downloads) has a major security vulnerability
 - Uses 'degenerator' (same author), which misuses a Node module
 - [“The vm module is not a security mechanism. Do not use it to run untrusted code.”](#)
 - (a mistake that's been made before: people rarely read disclaimers)
- 'pac-proxy-agent' (**2M weekly** downloads, same author) uses the above
 - Is widely popular, the main reason people use 'degenerator'
 - Most people using this package have never heard of the latter -- many never will



Log4j software bug: What you need to know

Casual computer users have probably used logging software, but it's used across



Bree Fowler ✓

Dec. 21, 2021 9:01 a.m. PT



Getty Images

<https://threatpost.com> › Vulnerabilities

Third Log4J Bug Can Trigger DoS; Apache Issues Patch

Dec 20, 2021 — The latest **bug** isn't a variant of the Log4Shell remote-code execution (RCE) **bug** that's plagued IT teams since Dec. 10, coming under active ...

<https://www.scmagazine.com> › Application security

Log4j, again, needs patching as new bug is found and ...

Dec 28, 2021 — Researchers at Checkmarx discovered a way to use **Log4j** to launch malicious code, forcing yet another round of patching for affected users.

Under the Hood: Libraries & Frameworks

Packages can be either:

- Libraries:
 - A set of classes and methods that provide reusable functionality
 - Typically: programmer calls, library returns data, that's it.

Under the Hood: Libraries & Frameworks

Packages can be either:

- Libraries:
 - A set of classes and methods that provide reusable functionality
 - Typically: programmer calls, library returns data, that's it.
- Frameworks:
 - Reusable skeleton code that can be customized into an application
 - Framework calls back into client code
 - The Hollywood principle: "Don't call us. We'll call you."
 - E.g., GUI development: you declare your UI elements, how they drawn; framework invokes them
 - Principle: [inversion of control](#)

Under the Hood: Libraries & Frameworks

Packages can be either:

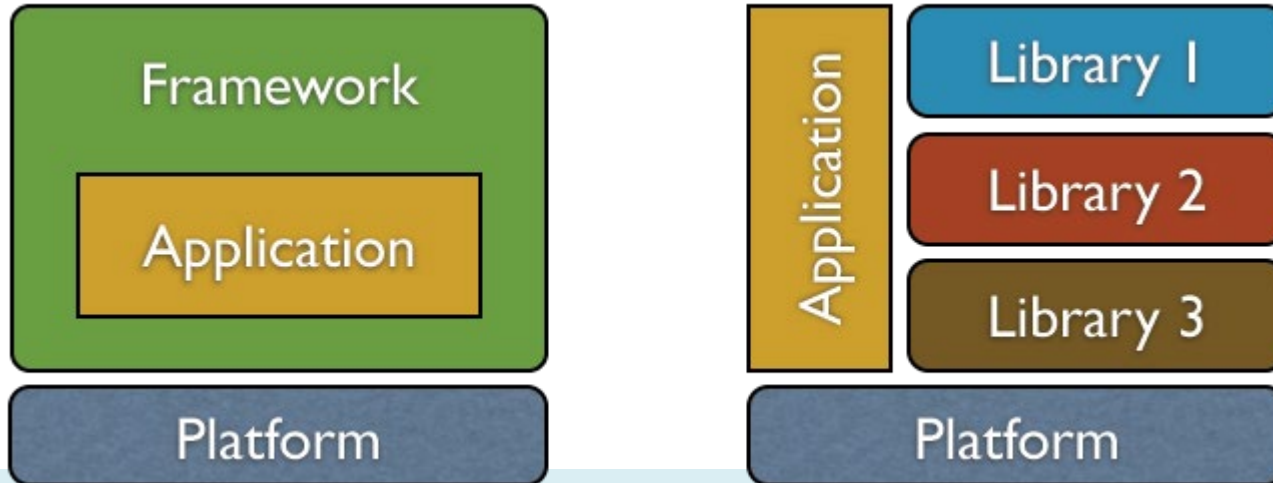
- Libraries:
 - A set of classes and methods that provide reusable functionality
 - Typically: programmer calls, library returns data, that's it.
- Frameworks:
 - Reusable skeleton code that can be customized into an application
 - Framework calls back into client code
 - The Hollywood principle: "Don't call us. We'll call you."
 - E.g., GUI development: you declare your UI elements, how they drawn; framework invokes them
 - Principle: [inversion of control](#)
- You typically use zero/one framework and many libraries
 - Frameworks might be especially constraining, but in exchange you can reuse more functionality
 - Some tools are a bit of both, and not all frameworks quite invert control

Under the Hood: Libraries & Frameworks

Which kind is a command-line parsing package?

Which kind is Android?

How about a tool that runs tests based on annotations you add in your code?



Under the Hood: Libraries & Frameworks

Look into:

- **Stated Goal:**
 - A simple interface (“get started in one line!”) also means lots of abstraction
 - That’s neither good nor bad; know what you need
 - Docs with “advanced use cases” are always neat
- **Maintenance:**
 - Active release cycle, recent updates to documentation
 - GitHub build status, issue tracker (filled with unmerged ‘dependabot’ PRs?)
 - Lots of companies deliberately lag by one minor (or even major) version
- **Transitive dependencies**
 - Myriad, beyond inspection. Using OSS in corporate environments is a headache

Abstraction, Reuse, and Programming Tools

- For each in {Build systems, IDE, libraries, **CI**):
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- What is next?

Quick overview of today's toolchain: Continuous Integration

CI: Automates standard build, test, deploy pipelines

(Technically, the latter is “CD”)

Typically builds from scratch in a clean *container*

Often tied to code-review; triggers on new commits, pull requests

- Ideally, official releases pass the build

Produces (long) logs with debugging outputs

Under the Hood: Continuous Integration

Defines a series of actions to be run in a clean build:

- Actions start from the very top:
 - Clone repository, checkout branch
 - Download & install Java/Node
 - Invoke commands with timeouts
- Travis allocates a new (Docker) container for each build
 - Think of this like a fresh, temporary computer
 - Usually with a few default libraries present (i.e., based on an *image*)
- That means: **fully replicable builds**

Continuous integration – GitHub Actions

You can see the results of builds over time

clegoues / clegoues.github.io Public
forked from academicpages/academicpages.github.io

<> Code Pull requests Actions Projects Wiki Security Insights Settings

Workflows [New workflow](#)

All workflows

Build and Deploy

All workflows

Showing runs from all workflows

Filter workflow runs

56 workflow runs

- correct ESE ref**
Build and Deploy #56: Commit e4402b9 pushed by clegoues
- fix up nav bar**
Build and Deploy #55: Commit f1653ba pushed by clegoues
- footer fixes**
Build and Deploy #54: Commit db51fd5 pushed by clegoues
- minor layout tweak, cite myself properly**
Build and Deploy #53: Commit edfd3e0 pushed by clegoues
- adding github etc links**
Build and Deploy #52: Commit 67f9bf9 pushed by clegoues

```
▶ 163 Installing SSH key from: default repository key
164 Using /home/travis/.netrc to clone repository.
165
166
▼ 167 $ git clone --depth=50 --branch=TypeScript https://github.com/CMU-17-214/template-21f-hw1.git CMU-17-214/template-21f-hw1
168 Cloning into 'CMU-17-214/template-21f-hw1'...
169 remote: Enumerating objects: 117, done.
170 remote: Counting objects: 100% (117/117), done.
171 remote: Compressing objects: 100% (73/73), done.
172 remote: Total 117 (delta 50), reused 104 (delta 37), pack-reused 0
173 Receiving objects: 100% (117/117), 69.89 KiB | 2.25 MiB/s, done.
174 Resolving deltas: 100% (50/50), done.
175 $ cd CMU-17-214/template-21f-hw1
176 $ git checkout -qf 0d657225c8cbdd52751c2f88527f93f4099b041e
177
178
▶ 179 $ nvm install 16
180 Downloading and installing node v16.8.0...
181 Downloading https://nodejs.org/dist/v16.8.0/node-v16.8.0-linux-x64.tar.xz...
182 Computing checksum with sha256sum
183 Checksums matched!
184 Now using node v16.8.0 (npm v7.21.0)
185
▶ 186 Setting up build cache
192
▶ 193
194
▶ 195 $ node --version
196 v16.8.0
197 $ npm --version
198 7.21.0
199 $ nvm --version
200 0.38.0
201
▶ 202 $ npm ci
210
211 $ timeout 5m npm run compile
212
213 > hw1-flashcards@1.0.0 compile
214 > tsc
```

Under the Hood: Continuous Integration

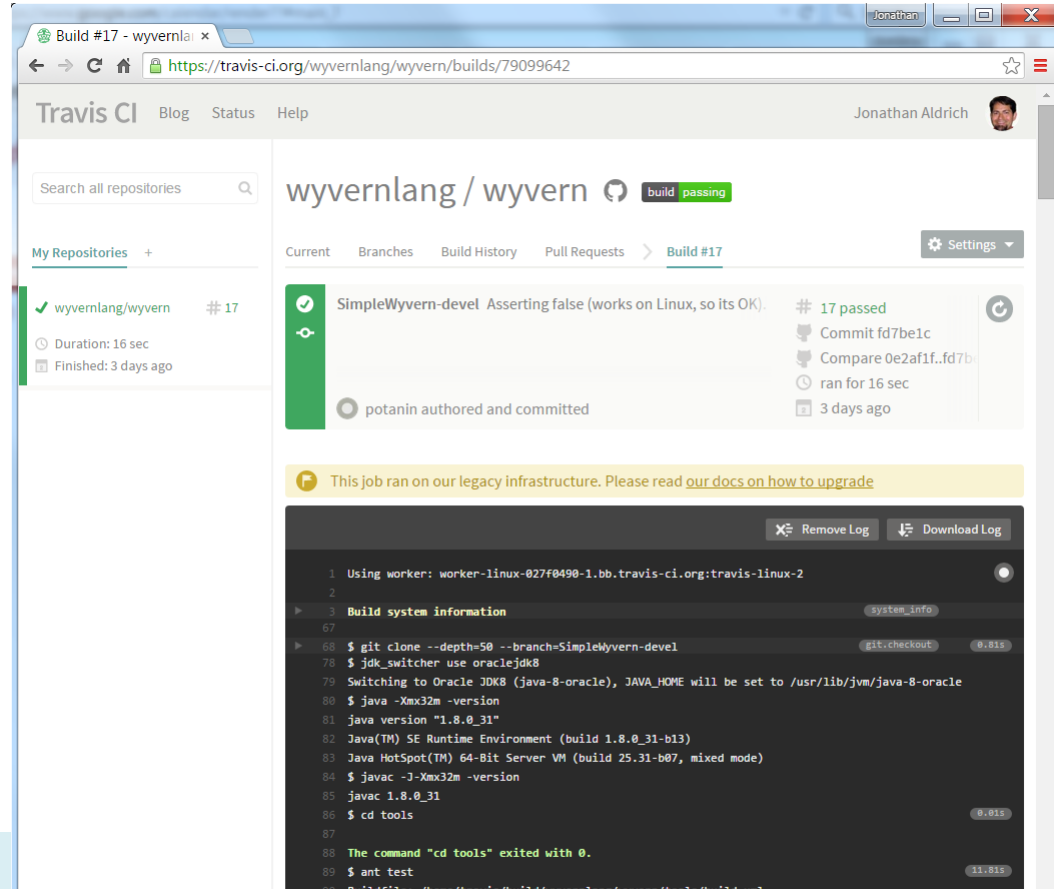
Automatically builds, tests, and displays the result

We – and everyone else – used to use Travis CI.

- Until they randomly stopped supporting OSS.

GitHub has native CI support, and it's pretty good: GitHub Actions.

- Sidebar on how our GH Actions are configured for HW1



The screenshot displays a Travis CI build page for the repository 'wyvernlang / wyvern'. The build status is 'passing'. The build name is 'SimpleWyvern-devel' and it asserts 'false (works on Linux, so its OK)'. The build duration is 16 seconds and it finished 3 days ago. The build log shows the following steps:

```
1 Using worker: worker-linux-027f0490-1.bb.travis-ci.org:travis-linux-2
2
3 Build system information
67
68 $ git clone --depth=50 --branch=SimpleWyvern-devel
69 $ jdk_switcher use oraclejdk8
70 Switching to Oracle JDK8 (java-8-oracle), JAVA_HOME will be set to /usr/lib/jvm/java-8-oracle
71 $ java -Xmx32m -version
72 java version "1.8.0_31"
73 Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
74 Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
75 $ javac -J-Xmx32m -version
76 javac 1.8.0_31
77 $ cd tools
78
79 The command "cd tools" exited with 0.
80 $ ant test
```


Quick overview of today's toolchain: not mentioned

Docker: containerize applications for coarse-grained reuse

Cloud: deploy and scale rapidly, release seamlessly

Bug/Issue trackers, often integrated with reviews

Behind the Abstraction: Some Nuance

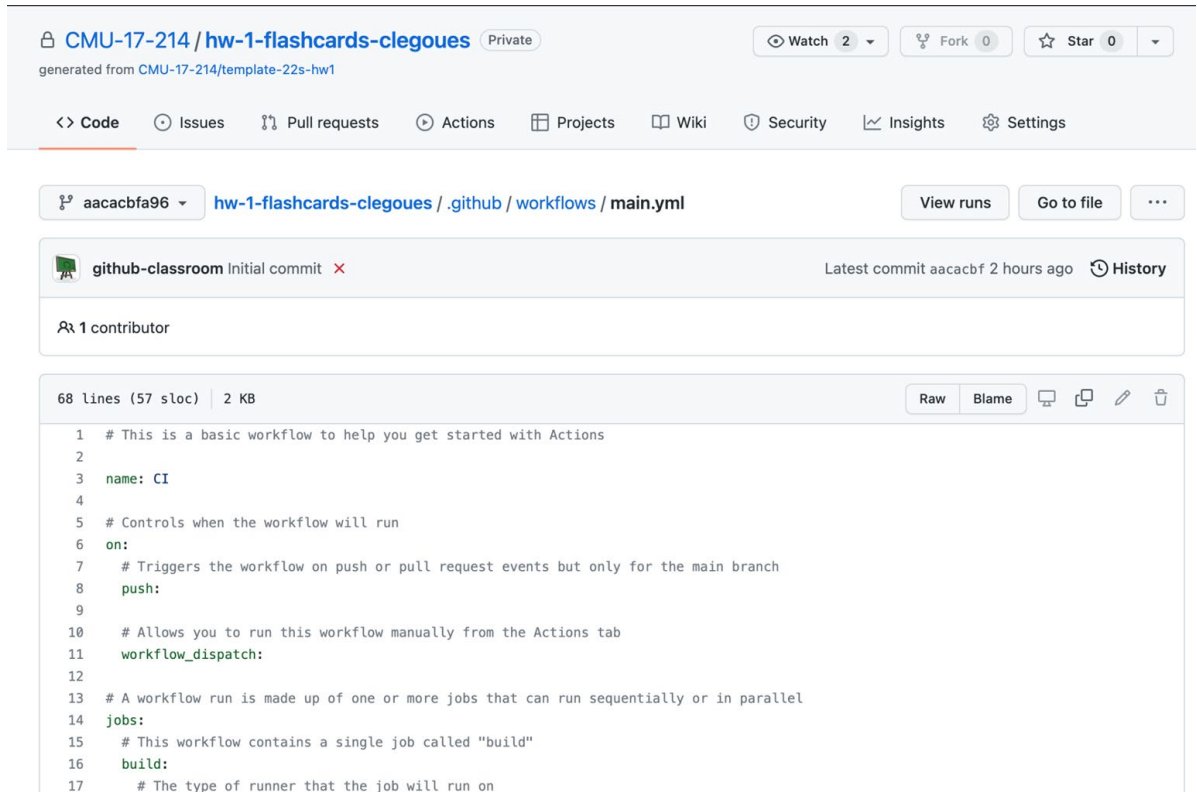
- Automation vs. Reuse
 - We tend to automate common chains of actions
 - Gear-up := {Press clutch, switch gear, release clutch while accelerating}
 - To facilitate reusing such “subroutines”, we introduce abstractions
 - Accelerate in ‘D’ => Gear-up when needed
- Reuse vs. Interfaces
 - Interfaces facilitate reuse through abstraction
 - Allow upgrading implementation without breaking things
 - Provide explicit & transparent contract



Behind the Abstraction, Some Nuance

Most tools are abstractions of common commands

- Typically operated via GUI and/or a DSL
- Obvious for GitHub Actions: just read the Yaml
 - Script-like languages are common
 - Involving a vocabulary of “targets”
 - E.g., `mvn site`



The screenshot shows a GitHub repository page for 'CMU-17-214 / hw-1-flashcards-clegoues'. The repository is private and generated from a template. The page displays the workflow file 'main.yml' in the '.github / workflows' directory. The workflow is an initial commit by 'github-classroom' 2 hours ago. The workflow file content is as follows:

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the workflow will run
6 on:
7   # Triggers the workflow on push or pull request events but only for the main branch
8   push:
9
10  # Allows you to run this workflow manually from the Actions tab
11  workflow_dispatch:
12
13 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
14 jobs:
15   # This workflow contains a single job called "build"
16   build:
17     # The type of runner that the job will run on
```

Behind the Abstraction, Some Nuance

Most tools are abstractions of common commands

- Typically operated via GUI and/or a DSL
- Obvious for GitHub Actions: just read the Yaml
 - Script-like languages are common
 - Involving a vocabulary of “targets”
 - E.g., `mvn site`

Abstraction can also “trap” us

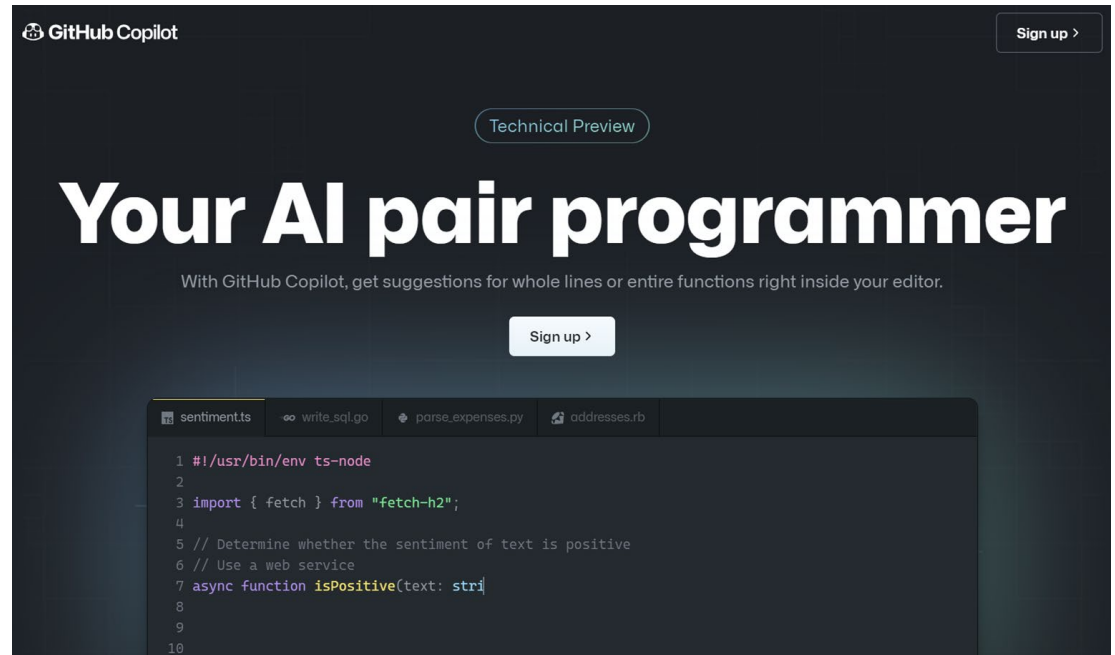
- When/how do we leave the abstraction?
- Command-line comes built into IDEs for a reason
- Non-trivial in general! May require switching/“patching” libraries
 - E.g., Maven → Gradle for more unusual build routines

Abstraction, Reuse, and Programming Tools

- For each in {Build systems, IDE, libraries, **CI**):
 - What is it?
 - What happens under the hood?
 - What are some practical ways to use it?
- **What is next?**

What's Next: AI Powered Programming

- Easier in Web IDEs
 - Which are themselves “next”

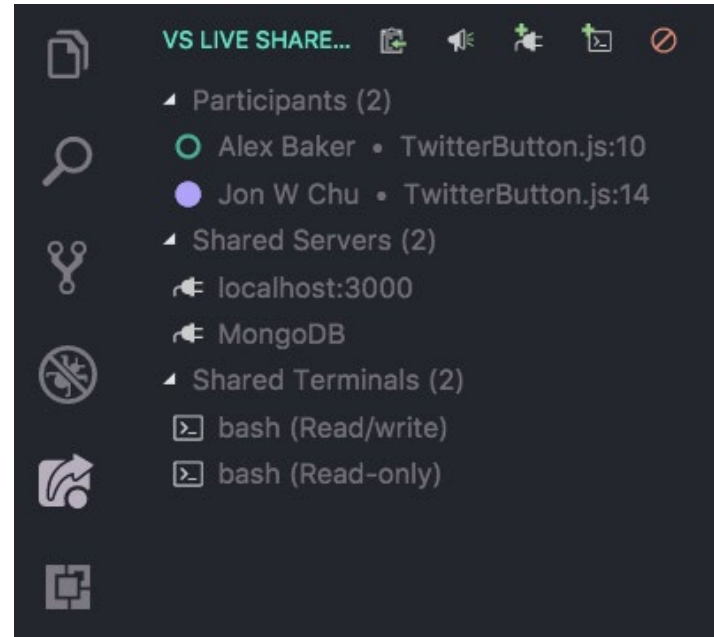


The image shows the GitHub Copilot website landing page. At the top left is the GitHub Copilot logo, and at the top right is a "Sign up >" button. Below the logo is a "Technical Preview" badge. The main heading is "Your AI pair programmer" in large white text. Underneath is the tagline: "With GitHub Copilot, get suggestions for whole lines or entire functions right inside your editor." Below this is another "Sign up >" button. At the bottom, there is a screenshot of a code editor with several tabs: "sentiment.ts", "write.sql.go", "parse_expenses.py", and "addresses.rb". The active tab "sentiment.ts" shows the following code:

```
1 #!/usr/bin/env ts-node
2
3 import { fetch } from "fetch-h2";
4
5 // Determine whether the sentiment of text is positive
6 // Use a web service
7 async function isPositive(text: stri
8
9
10
```

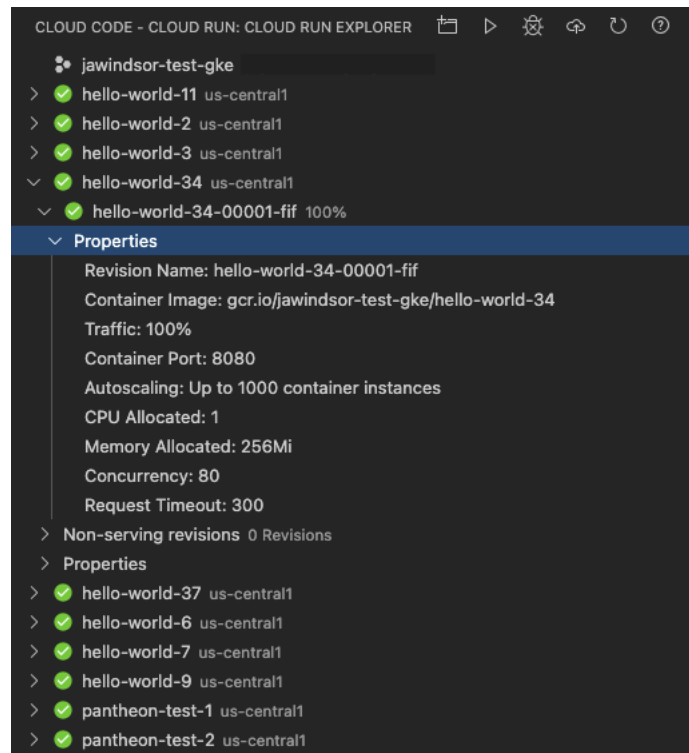
What's Next: Collaborative online coding

- Think: Google Docs for code
- E.g. VS Live Share
- How will this change “commits”?



What's Next: Tighter IDE-to-cloud integration

- Google Cloud is pushing on this with VSCode
- We will (lightly) touch on Containers & Clouds in this course



Review of the Quiz

Bloch generally recommends that fields are not made directly accessible from outside a class. EITHER

- (1) Give an example of a circumstance in which it's OK to bend that rule, OR
- (2) give a specific example of what can go wrong if you violate it.

Summary

- Programming Tools are abundant, and rapidly evolving
 - Learn multiple; you will have to inevitably
- They rely on abstractions through interfaces to facilitate reuse
 - Which come in many shapes: GUI, API, DSL
 - And can be a limitation -- choose wisely
- Your HW1 toolchain sets you up for all homeworks
 - With modest variations (frameworks, new build targets)
 - Self-discovery is a big asset
 - Recitation should be helpful!