Principles of Software Construction: Objects, Design, and Concurrency

IDEs, Build system, Continuous Integration, Libraries

Vincent Hellendoorn **Claire Le Goues**

Carnegie Mellon University
School of Computer Science

institute for SOFTWARE RESEARCH

Administrivia

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

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 drive your car
  - Not so much if you are a mechanic
  - How to debug a broken transmission?
- Also slow to evolve
  - *Conf.* people adamantly refusing to use an automatic
- 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?

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

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
```

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:
  - Is compiled to bytecode, which is then run in a runtime environment (JVM + GC, etc).
  - Command line execution vs. in VSCode
    - What's the -cp doing?
- Typescript:
  - Is transpiled to javascript, which is then executed inside a runtime environment.
  - Outside the browser: we use node.js
- Why do we encourage you to run everything in an IDE/use Maven/npm?
  - Hints: what are the roles of -cp in Java? Or package.json 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

```
[INFO] ----------------------< org.example:FlashCards >----------------------
[INFO] Building FlashCards 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.pom (8.1 kB at 30 kB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/23/maven-plugins-23.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/23/maven-plugins-23.pom (9.2 kB at 708 kB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/22/maven-parent-22.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/22/maven-parent-22.pom (30 kB at 1.5 MB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/11/apache-11.pom
[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 MB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.jar
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-resources-plugin/2.6/maven-resources-plugin-2.6.jar (30 kB at 1.6 MB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-compiler-plugin/3.1/maven-compiler-plugin-3.1.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-compiler-plugin/3.1/maven-compiler-plugin-3.1.pom (10 kB at 928 kB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/24/maven-plugins-24.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/24/maven-plugins-24.pom (11 kB at 982 kB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/23/maven-parent-23.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/maven-parent/23/maven-parent-23.pom (33 kB at 1.4 MB/s)
[INFO] Downloading from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/13/apache-13.pom
[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/13/apache-13.pom (14 kB at 608 kB/s)
[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

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
```

Maven™

pom.xml (FlashCards)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <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

```
        <version>RELEASE</version>
        <scope>test</scope>
    </dependency>
```

project › dependencies › dependency

Build Dependencies

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

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

package.json — claire-hw1-js

package.json 1, M

package.json > { } dependencies

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

Code File Edit Selection View Go Run Terminal Window Help

Mon Jan 24 5:45 PM

launch.json — typescript

EXPLORER

TYPESCRIPT
- .vscode
  - launch.json
- cards
- dist
- node_modules
- src
  - cards
  - data
  - ordering
  - index.ts
  - ui.ts
- .gitignore
- package-lock.json
- package.json
- README.md
- tsconfig.json

OUTLINE

TIMELINE

Run menu:

- 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...

{} package.json | tsconfig.json | {} launch.json U

.json > Launch Targets > {} Launch Program

```
e 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,"waitForD
ebugger":"","execPath":"/Users/clegoues/.nvm/versions/node/v17.3.0/bin/node
","onlyEntrypoint":false,"autoAttachMode":"always","fileCallback":"/var/fol
ders/6y/3yk0l2ms1yn5t8ng5pd3cdtm0000gn/T/node-debug-callback-b1bc858d1e8e06
c9"}' /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...

main* Python 3.9.6 64-bit 0 0 Launch Program (typescript) Claire Live Share LF JSON with Comments

{} launch.json U ×

.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           {
9               "type": "pwa-node",
10              "request": "launch",
11              "name": "Launch Program",
12              "skipFiles": [
13                  "<node_internals>/**"
14              ],
15              "program": "${file}",
16              "console":"integratedTerminal",
17              "preLaunchTask": "tsc: build - tsconfig.json",
18              "args": [ "--help" ],
19              "outFiles": [
20                  "${workspaceFolder}/dist/**/*.js"
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
```

ram (typescript) Claire Live Share Ln 13, Col 12 Spaces: 2 UTF-8 LF {} TypeScript

ClassLoaders.class — java

RUN AND DEBUG

RUN

Run and Debug

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

Show all automatic debug configurations.

Main.java | ClassLoader.class | ClassLoaders.class

```
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
- [x] Main.java src/main/java/e... 21

main | 0 0 1 | Live Share | Ln 180, Col 1 | Spaces: 4 | Java

What’s going on in HW1?

Also: quick Maven setup demo.

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

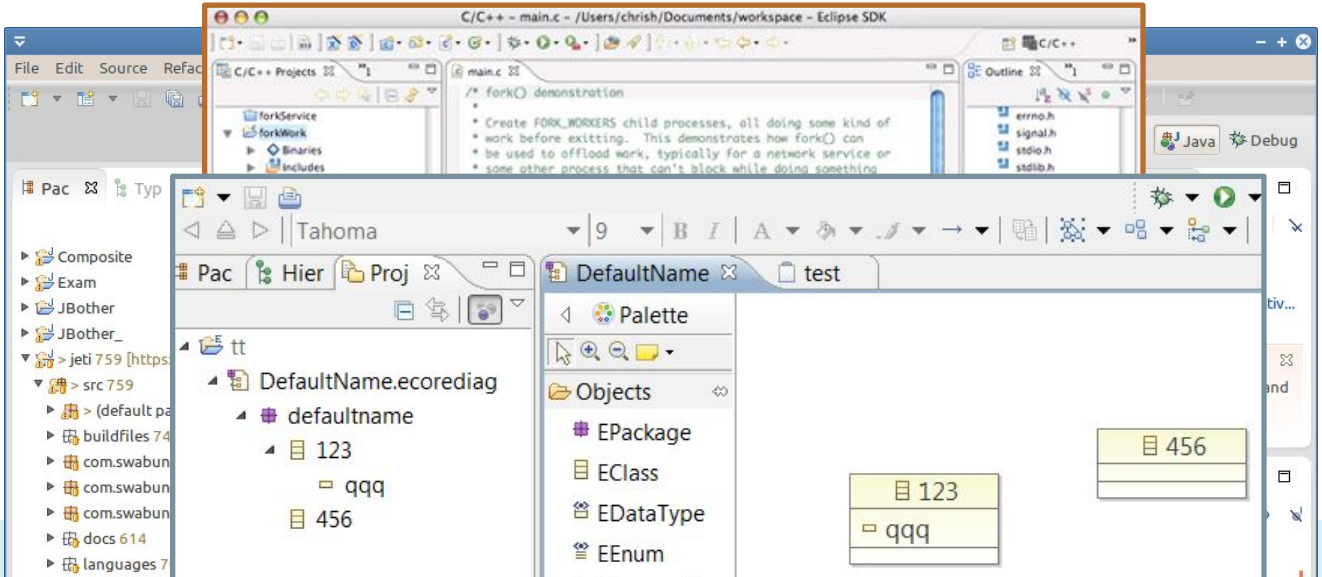
Integrated Development Environments bundle development workflows in a single UI

- 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 <u>incremental compilation</u>
  - 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)

Main.java — java

EXPLORER

JAVA
- .vscode
- cards
  - designpatterns.csv
  - german.csv
- src/main/java/edu/cmu/cs214/...
  - cards
  - cli
  - data
  - ordering
  - Main.java
- target
- .gitignore
- checkstyle.xml
- pom.xml
- README.md

Main.java

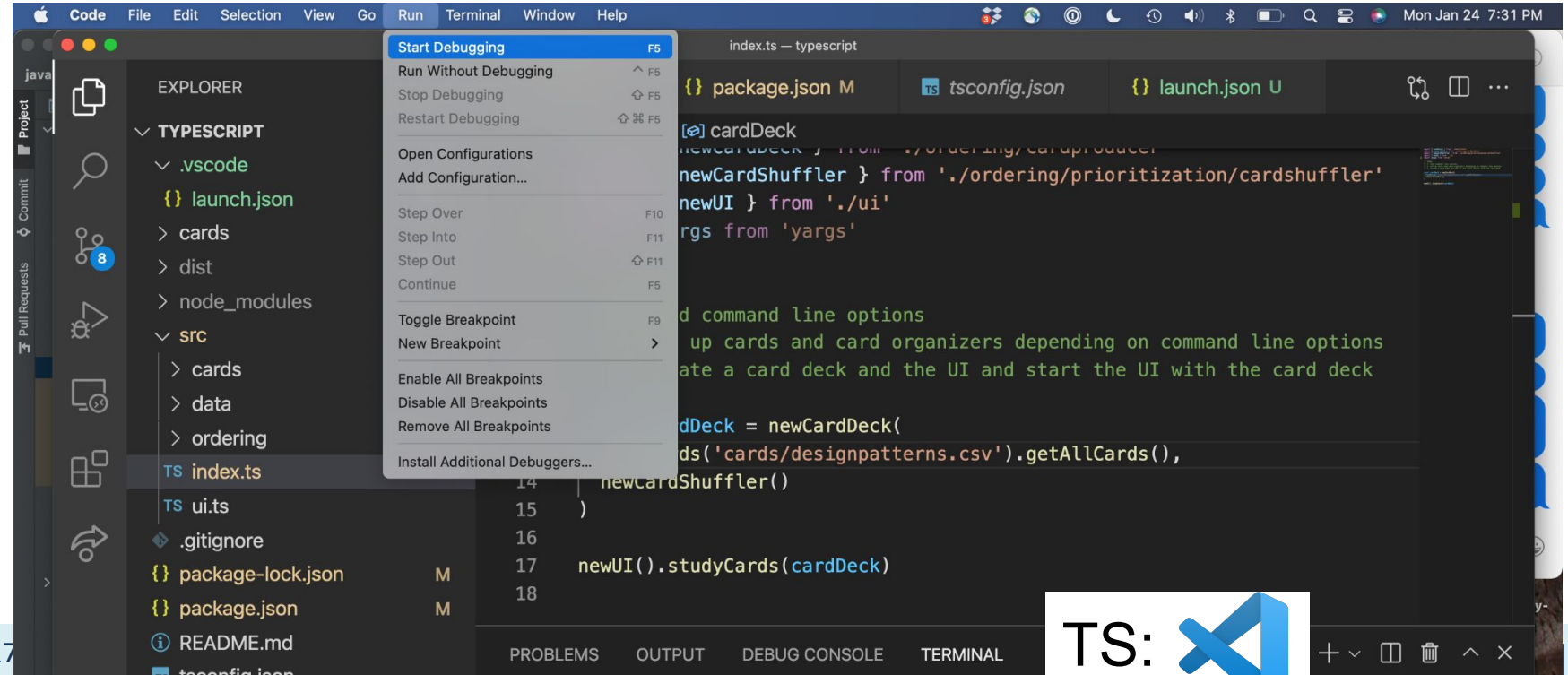
src > main > java > edu > cmu > cs214 > hw1 > Main.java > Main > Main()

```
import edu.cmu.cs214.hw1.ordering.CardDeck;
import edu.cmu.cs214.hw1.ordering.prioritization.CardShuffler;

import java.io.File;
import java.io.IOException;

public final class Main {

    private Main() {
        // Disable instantiating this class.
        throw new UnsupportedOperationException();
    }

    Run | Debug
    public static void main(String[] args) throws IOException {
        // TODO: set up options, extract command line arguments, fill in the relev
        CardStore cards = new CardLoader().loadCardsFromFile(new File(pathname: "c
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Debug: Main

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

TS index.ts M | {} package.json M | tsconfig.json | {} launch.json U

index.ts > cardDeck

VARIABLES

Local

cardDeck: undefined

cardproducer_1: {newCardDec…

cardshuffler_1: {newCardShu…

exports: {__esModule: true}

WATCH

CALL STACK

La… PAUSED ON BREAKPOINT

<anonymous> src/index.ts

Show 6 More: Skipped by skipFil

LOADED SCRIPTS

BREAKPOINTS

Caught Exceptions

Uncaught Exceptions

index.ts src 13

```
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)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
S=--require "/Applications/Visual Studio Code.app/Contents/Resources/app/extensions/ms-vscode.js-debug/src/bootloader.bundle.js" --inspect-publish-uid=http' 'VSCODE_INSPECTOR_OPTIONS={"inspectorIpc":"/var/folders/6y/3yk0l2ms1yn5t8ng5pd3cdtm0000gn/T/node-cdp.29317-1.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-deb1bd35d68abbf0"}' /Users/clegoues/.nvm/versions/node/v17.3.0/bin/node ./dist/index.js

Debugger attached.
```

npm

build - t…

Launch Pro…

main* 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!)

Main.java — java

EXPLORER

JAVA
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
FlashCards org.example:Flas...
Lifecycle
clean
validate
compile
test
test-compile
package
verify
install
site
deploy
Plugins
Dependencies
junit:junit:4.13.2 (test)

Main.java

src > main > java > edu > cmu > cs214 > hw1 > Main.java > Main > main(String[])

```
11
12  public final class Main {
13
14      private Main() {
15          // Disable instantiating this class.
16          throw new UnsupportedOperationException();
17      }
18
    Run | Debug
19      public static void main(String[] args) throws IOException {
20          // TODO: set up options, extract command line arguments, fill in the relev
21          CardStore cards = new CardLoader().loadCardsFromFile(new File(pathname: "c
22          CardDeck cardDeck = new CardDeck(cards.getAllCards(), new CardShuffler());
23          new UI().studyCards(cardDeck);
24      }
25
26  }
27
```

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 obje
        CardStore cards = new CardLoader().loadCardsFromFile(new File( pathname: "cards/desig
```

Add runtime exception(s) to method signature

Press Ctrl+Shift+I to open preview

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

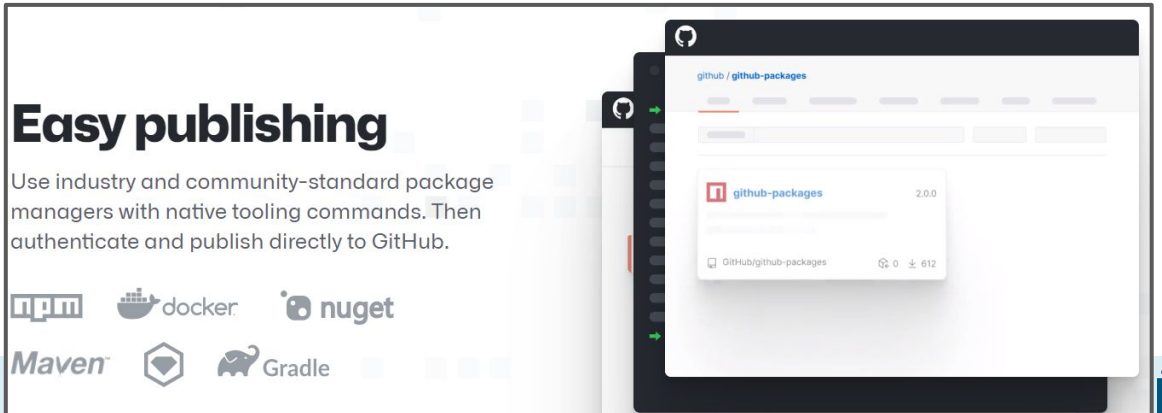
- For each in {Build systems, IDE, **libraries**, CI}:
  - What is it?
  - What happens under the hood?
  - What are some practical ways to use it?
- What is next?

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#, ...
- Registries of managers, e.g., GitHub Packages



ClassLoaders.class — java

EXPLORER

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)

e.g. spring azure storage

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

```
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 implementin
```

Code File Edit Selection View Go Run Terminal Window Help Mon Jan 24 6:08 PM

index.ts — typescript

EXPLORER

TYPESCRIPT
- .vscode
  - launch.json U
- cards
- dist
- node_modules
- src
  - cards
  - data
  - ordering
  - index.ts M
  - ui.ts
- .gitignore
- package-lock.json M
- package.json M
- README.md
- tsconfig.json

OUTLINE

TIMELINE

TS index.ts M | {} package.json M | tsconfig.json | {} launch.json U

src > TS index.ts > cardDeck

```
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)
```

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
```

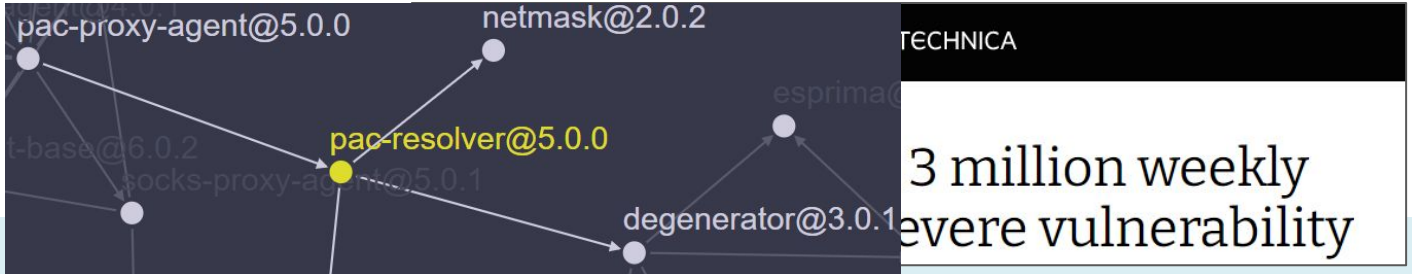
main* 0 0 Claire Live Share Ln 13, Col 55 Spaces: 2 UTF-8 LF {} TypeScript

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, …
  - Registries of managers, 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

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



Gaming Home Entertainment Services & Software

Log4j software bug: What you need to know

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

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

Getty Images

With Christmas just days away, federal officials are warning those who

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., Android development: you declare your UI elements, activities to be composed
  - Principle: inversion of control

Under the Hood: Libraries & Frameworks

Packages can be either:

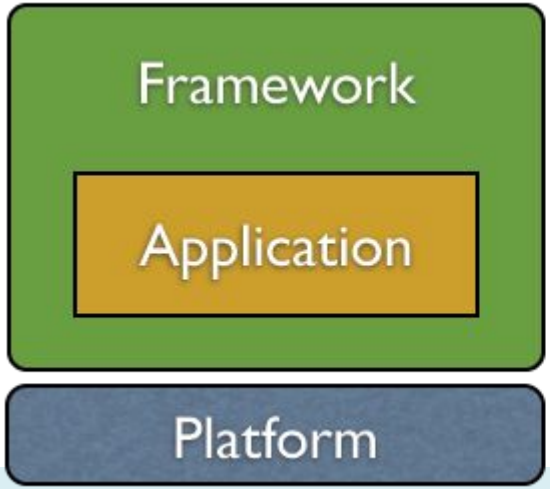
- Libraries:
  - A set of classes and methods that provide reusable functionality
  - Typically: programmer calls, library returns data, that's it.
- Frameworks:
  - Reusable skeleton code that can be customized into an application
  - Framework calls back into client code
    - The Hollywood principle: "Don't call us. We'll call you."
  - E.g., Android development: you declare your UI elements, activities to be composed
  - Principle: inversion of control
- You typically use zero/one framework and many libraries
  - Frameworks might be especially constraining, but for good reason.
  - 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?



http://tom.lokhorst.eu/2010/09/why-libraries-are-better-than-frameworks`

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
- Recursive dependencies
  - Myriad, beyond inspection. Using OSS in corporate environments is a headache

Frameworks

Whitebox:

- Extension via subclassing and overriding methods
- Common design pattern(s):
  - Template method
- Subclass has main method but gives control to framework

Blackbox:

- Extension via implementing a plugin interface
- Common design pattern(s):
  - Command
  - Observer
- Plugin-loading mechanism loads plugins and gives control to the framework

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 | Settin

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                                                    ssh_key
165 Using /home/travis/.netrc to clone repository.
166
167 $ git clone --depth=50 --branch=TypeScript https://github.com/CMU-17-214/template-21f-hw1.git CMU-17-214/template-21f-hw1   git.checkout   1.75s
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                                                                                                    0.01s
179 $ nvm install 16                                                                   nvm.install   3.65s
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                                                                          cache.1
192
193                                                                                                cache.npm
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                                                                           install.npm   3.04s
210
211 $ timeout 5m npm run compile                                                                     4.93s
212
213 > hw1-flashcards@1.0.0 compile
214 > tsc
```

Under the Hood: Continuous Integration

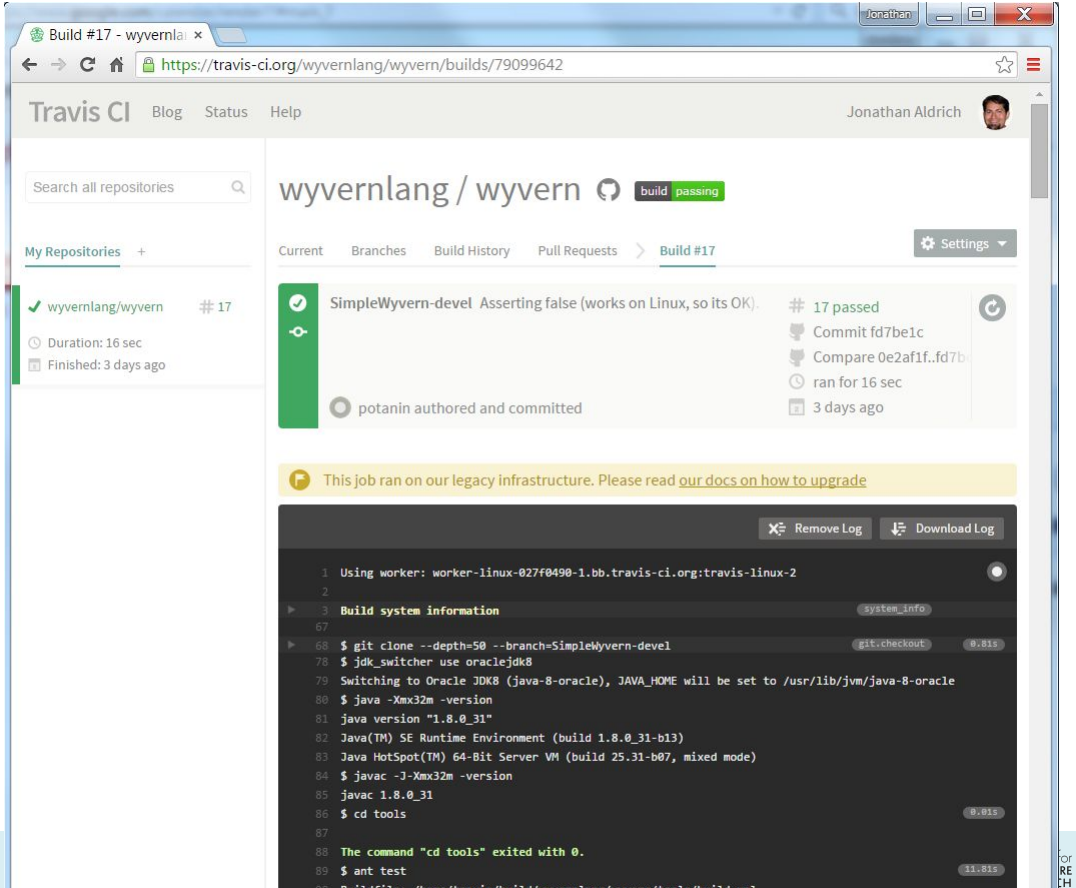
Automatically builds, tests, and displays the result

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



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`

CMU-17-214 / hw-1-flashcards-clegoues Private

generated from CMU-17-214/template-22s-hw1

Watch 2 | Fork 0 | Star 0

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

aacacbfa96 | hw-1-flashcards-clegoues / .github / workflows / main.yml | View runs | Go to file | ···

github-classroom Initial commit ✕ | Latest commit aacacbf 2 hours ago | History

1 contributor

68 lines (57 sloc) | 2 KB | Raw | Blame

```yaml
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "build"
  build:
    # 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

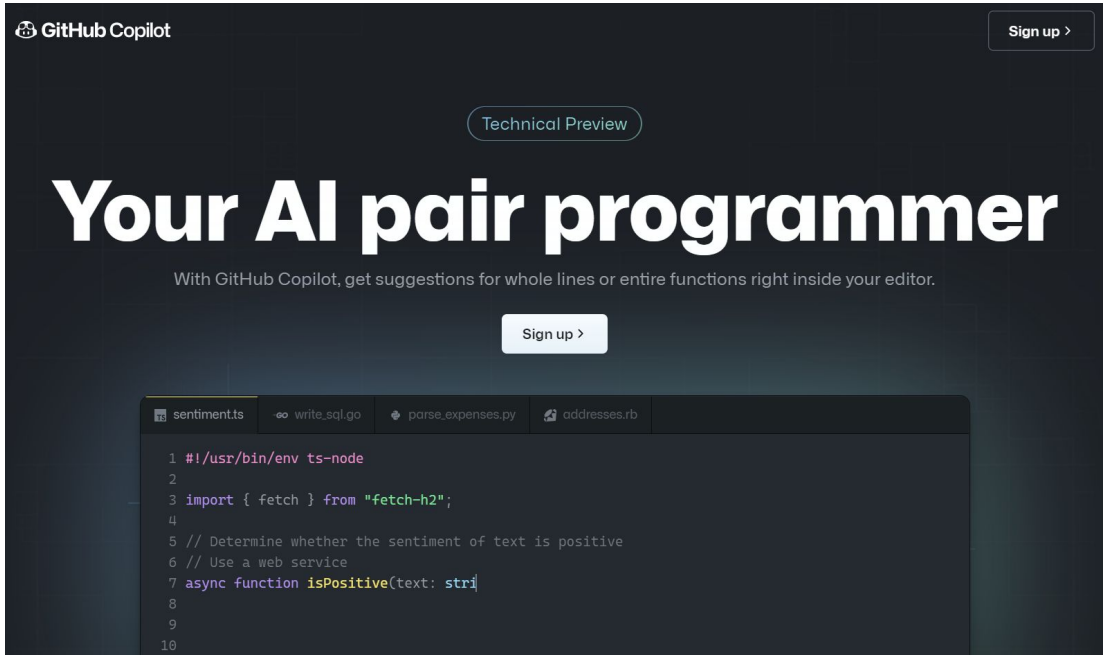
- When/how do we leave the abstraction?
- Command-line comes built into IDEs for a reason
- Non-trivial in general! May require switching/“patching” libraries
  - E.g., Maven → Gradle for more unusual build routines

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?**

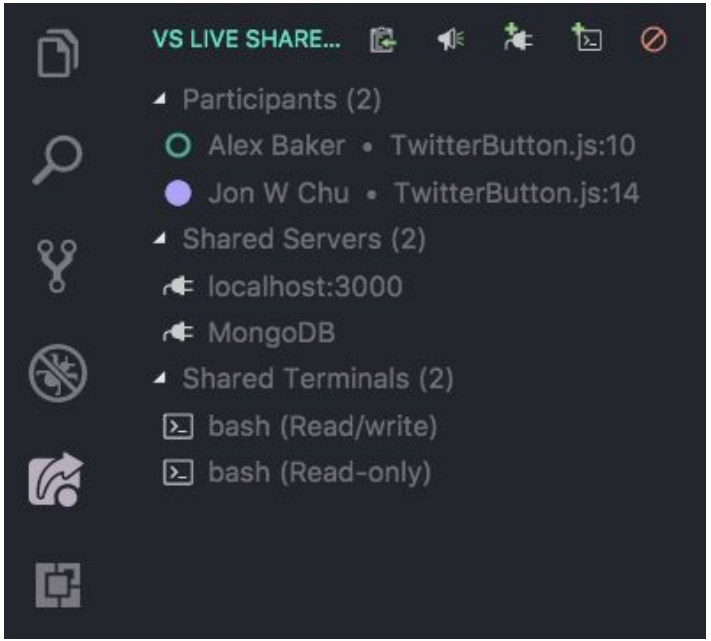
What’s Next: AI Powered Programming

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



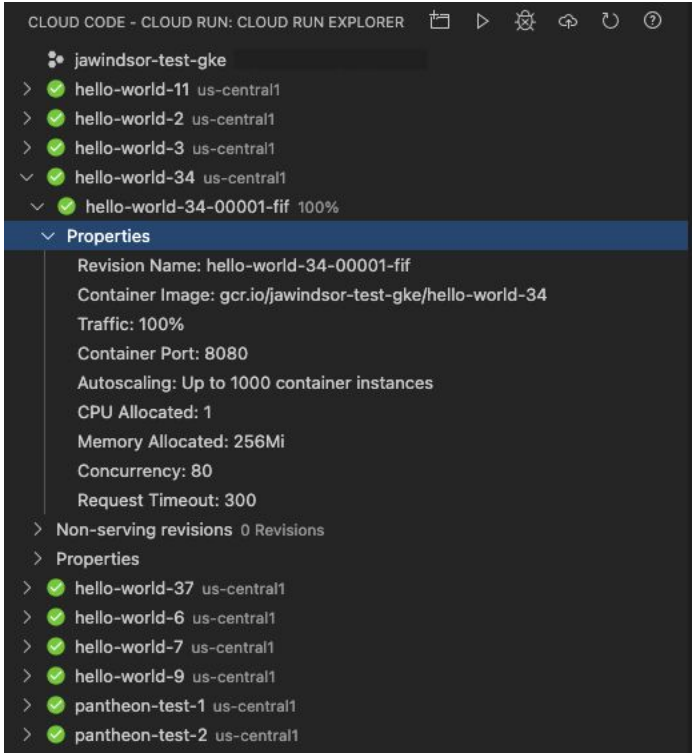
What’s Next: Collaborative online coding

- 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



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!