Principles of Software Construction: Objects, Design, and Concurrency

IDEs, Build system, Continuous Integration, Libraries

Bogdan Vasilescu **Claire Le Goues**

Carnegie Mellon University
School of Computer Science

institute for SOFTWARE RESEARCH

Administrivia

Wait list update: should be able to basically clear it. We'll give updated deadlines for students who enroll late when they're officially enrolled.

Please don't email just one of us unless it's very sensitive.

- Only email us about big picture stuff.
- Waitlist questions: include Jenni Cooper on the CC

Please check Piazza periodically both for technical and non-technical questions!

- We encourage you to help each other out.

We'll be in a classroom next week.

New staff, new OH, **new recitation.**

We have hired three new TAs! Welcome to Li Guo, Deyuan Chen, and Yuwei Li.

There is a new recitation section: W 7-7:50.

- It is remote only.
- AFAIK it doesn't exist yet but the registrar is working on it
- So we will have it this week.
- If you are on the waitlist and can attend this one, do try to.

We now have a* lot *more office hours!

Links/info on course calendar on website. Click on "more info" if necessary.

Homework 1: Welcome to the deep end?

The milestone has set you up for the rest of the semester! Hooray!

All of the code necessary for the options we're asking you to support with the command line library exist in the code.

We're here to help! Come to recitation and/or Office Hours.

- We're using OHQueue (see Piazza).

Mini-quiz

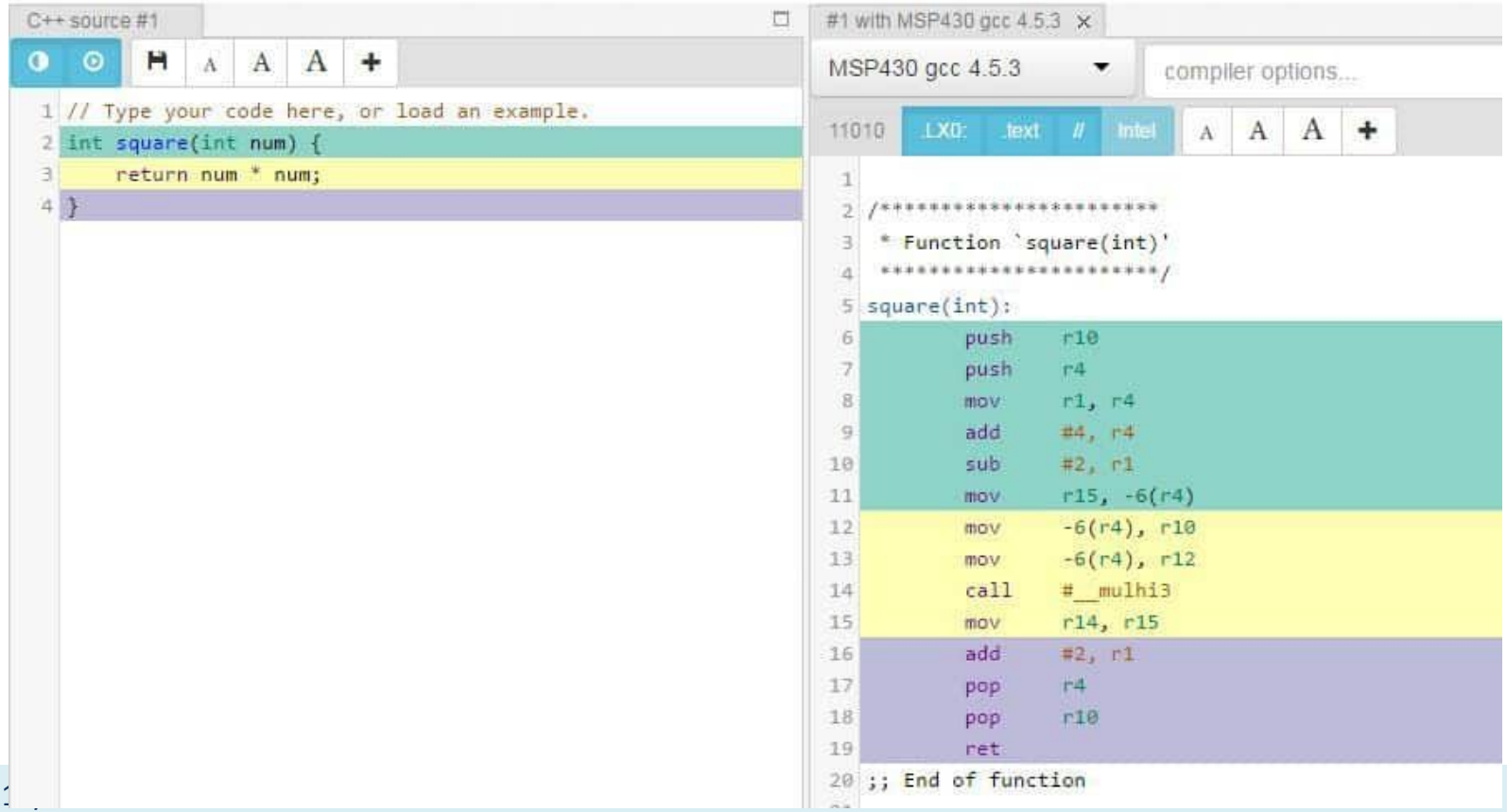
https://forms.gle/gaCB7xkPFijaUYWy6

Abstraction, Reuse, and Programming Tools

- For each in {IDE, Build systems, libraries, CI}:
  - What is it today?
  - What is under the hood?
- What is next?

Automation Requires Abstraction

Automation Requires Abstraction



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
- Engineers seek out abstractions that simplify their work, help focus on the hard parts
  - They 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 {IDE, Build systems, libraries, CI}:
  - What is it today?
  - What is under the hood?
- What is next?

Abstraction, Reuse, and Programming Tools

- For each in {**IDE**, Build systems, libraries, CI}:
  - **What is it today?**
  - What is under the hood?
- What is next?

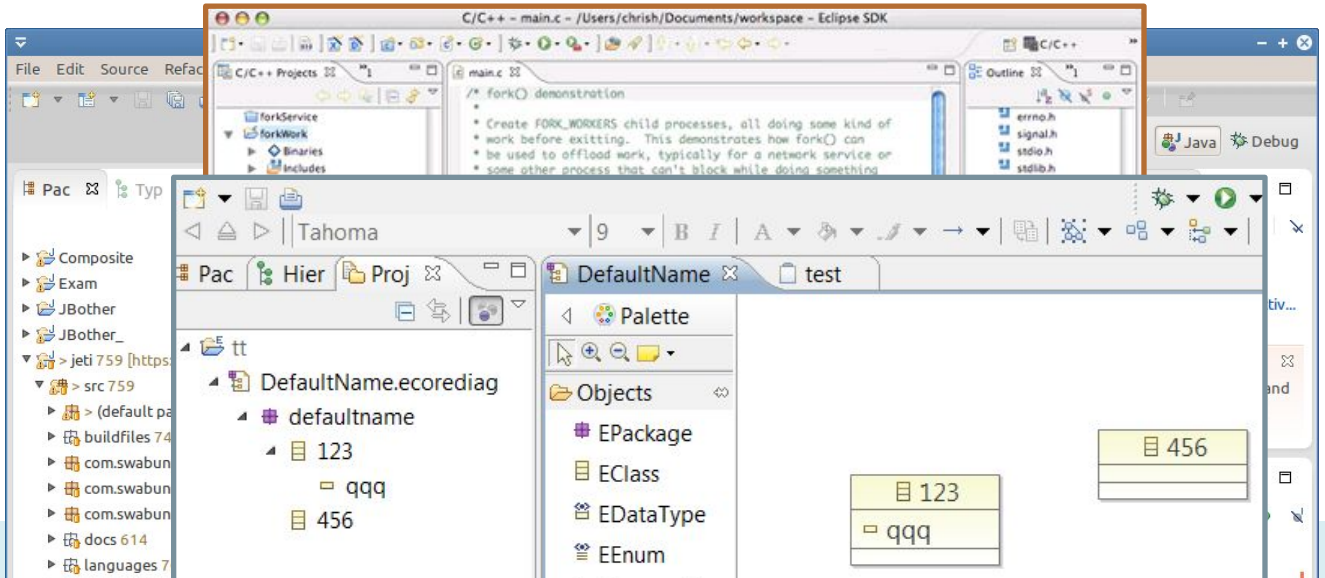
Quick overview of today's toolchain: IDEs

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

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
  - Quick tangent: if you can build either a large product or a platform, build a platform

But 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 for TS and IntelliJ for Java, but you can actually use either for both (and we don't care).

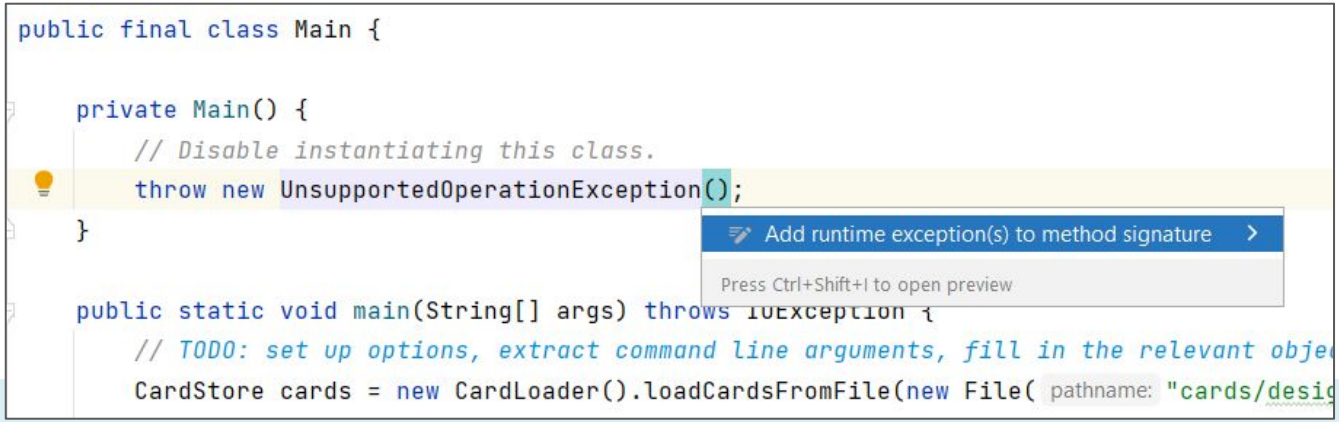
Abstraction, Reuse, and Programming Tools

- For each in {**IDE**, Build systems, libraries, CI}:
  - What is it today?
  - **What is under the hood?**
- What is next?

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

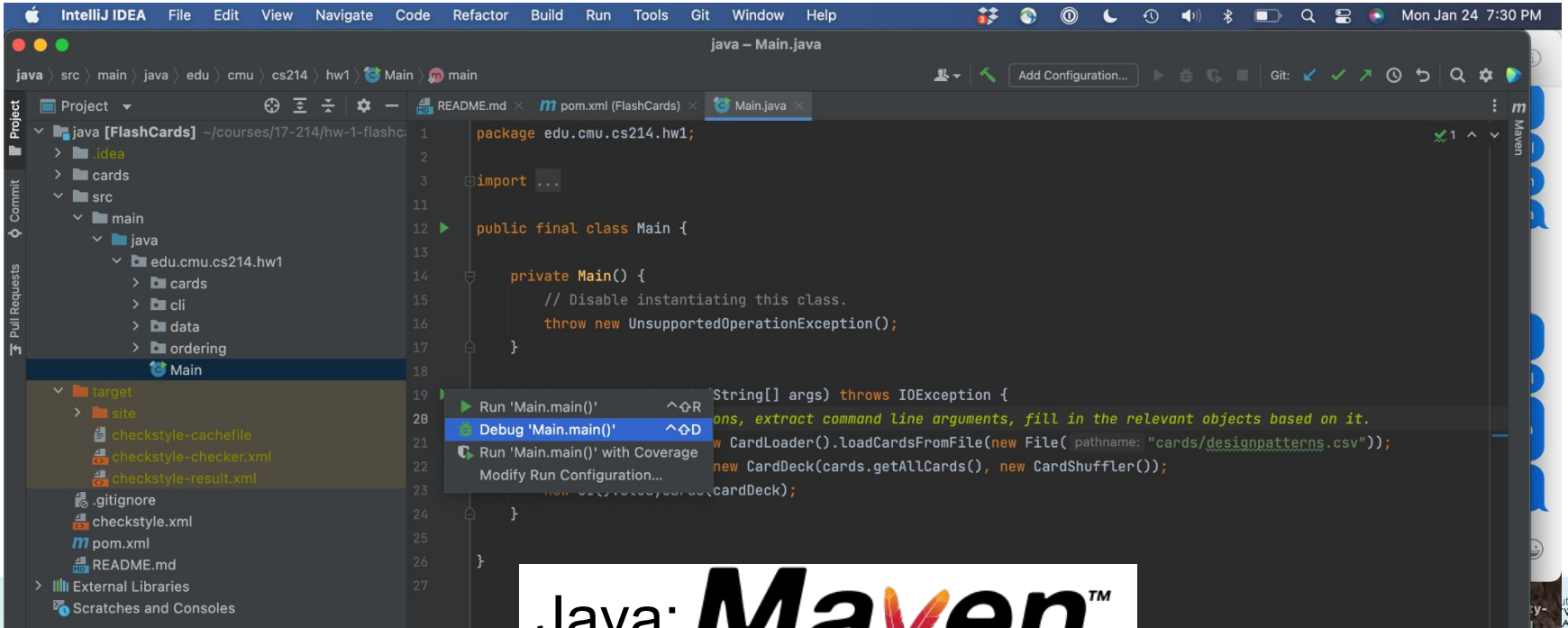


Under the Hood: IDEs

- 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`)

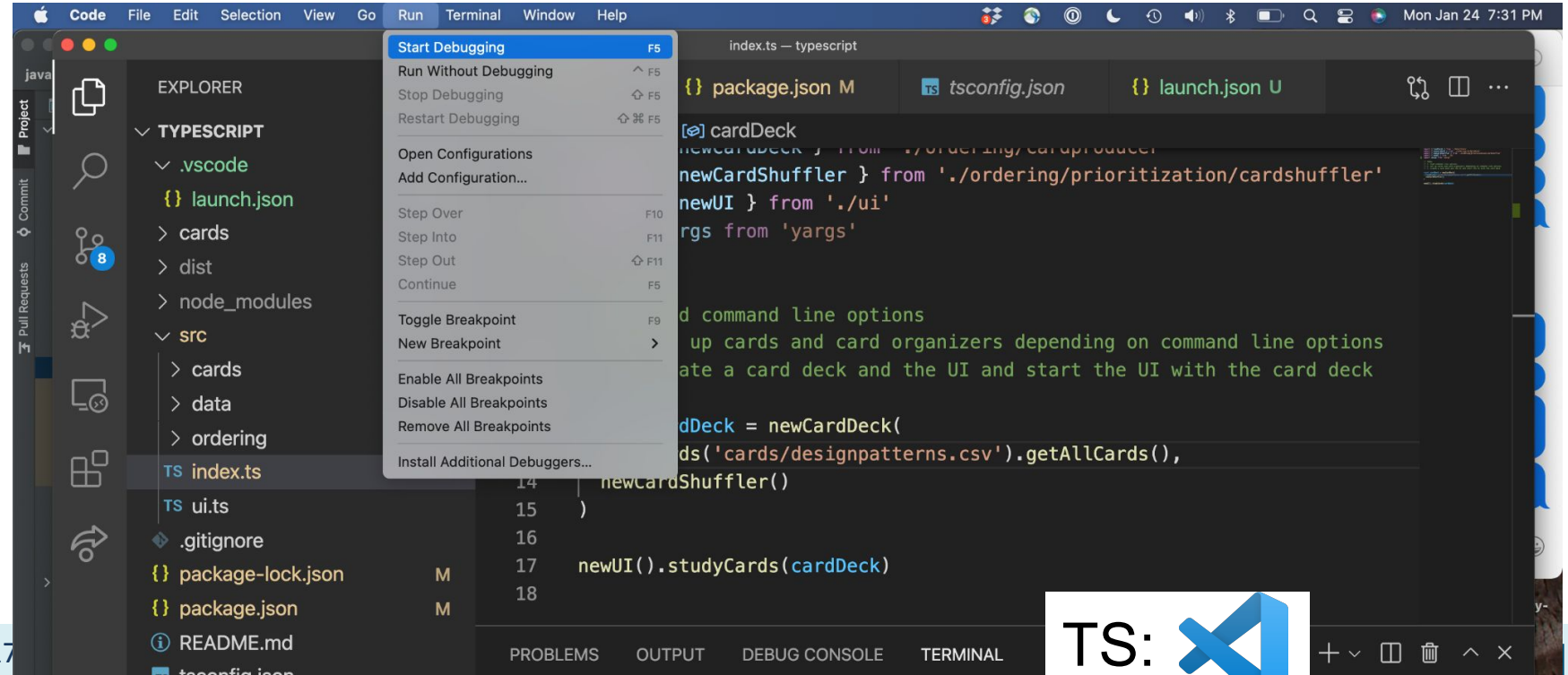
Under the Hood: IDEs

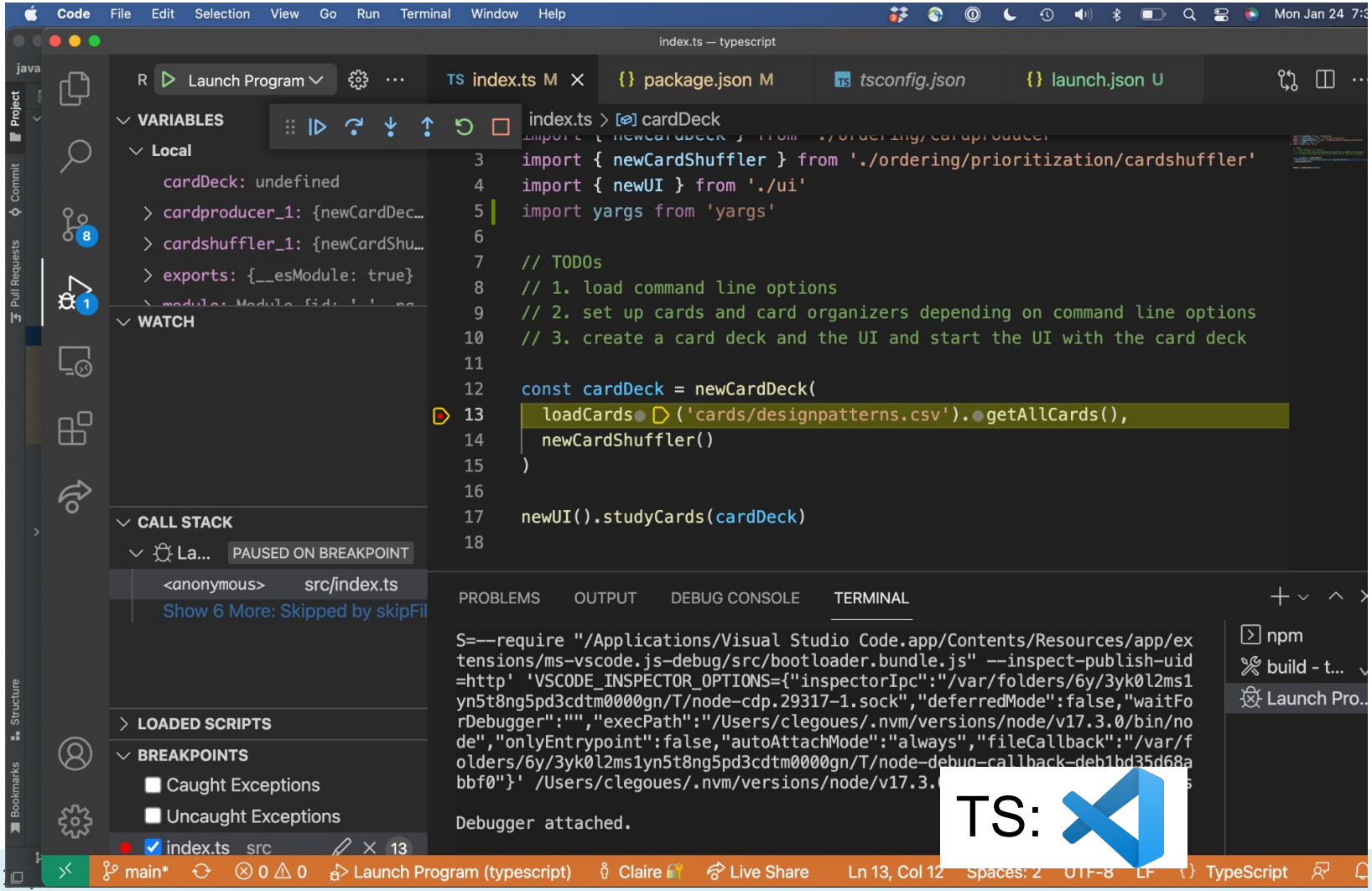
Automate common programming actions, like debugging, which is often the default mode when you run in the IDE (like in VSCode)



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
Mon Jan 24
index.ts — typescript
Launch Program
index.ts M
package.json M
tsconfig.json
launch.json U
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
index.ts > cardDeck
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
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.
Debugger attached.
npm
build - t...
Launch Pro...
main*
Launch Program (typescript)
Claire
Live Share
Ln 13, Col 12
Spaces: 2
UTF-8
LF
TypeScript
TS:

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, IntelliJ are very explicit about this in the terminal -- great for customization

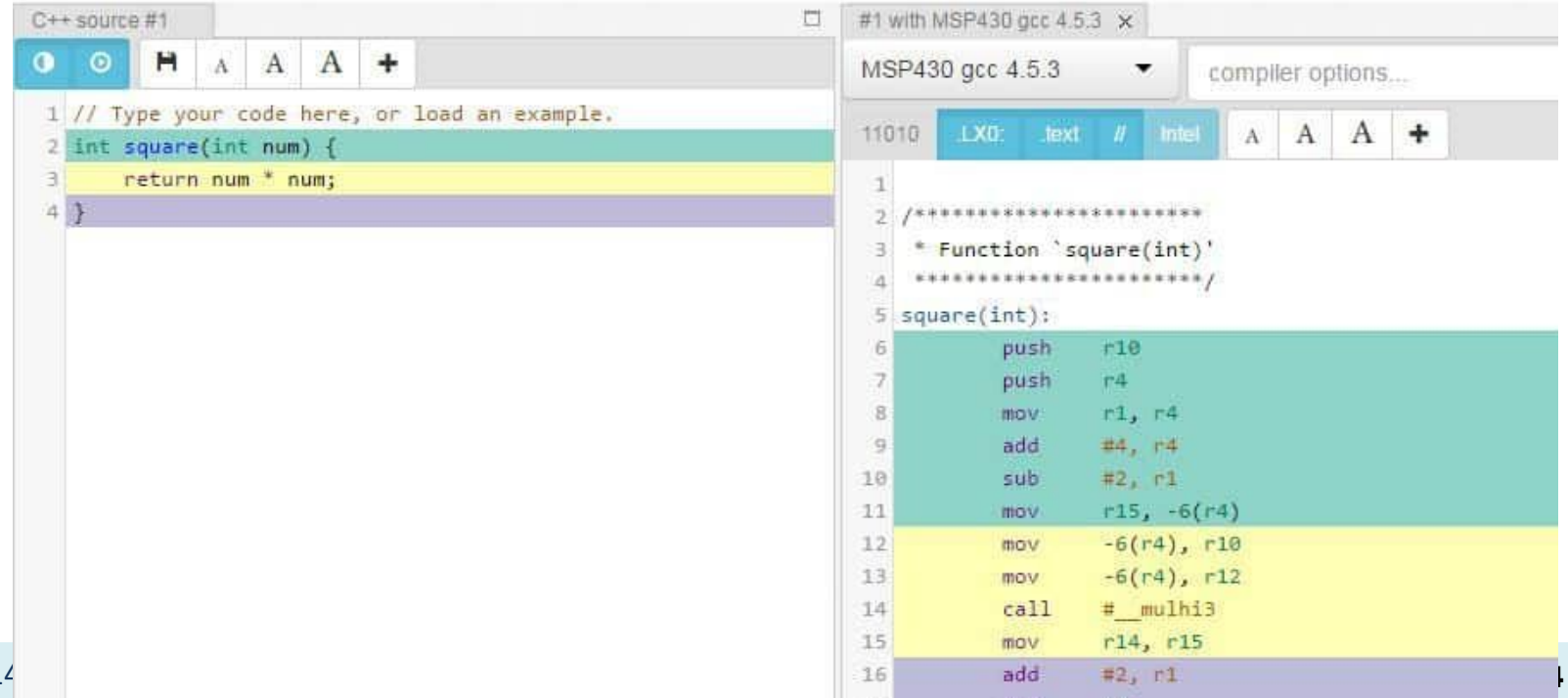
```
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ ID

Process finished with exit code 0
```

Abstraction, Reuse, and Programming Tools

- For each in {IDE, **Build systems**, libraries, CI}:
  - **What is it today?**
  - What is under the hood?
- What is next?

Quick overview of today's toolchain: Build Systems

How does this happen?



Quick overview of today's toolchain: Build Systems

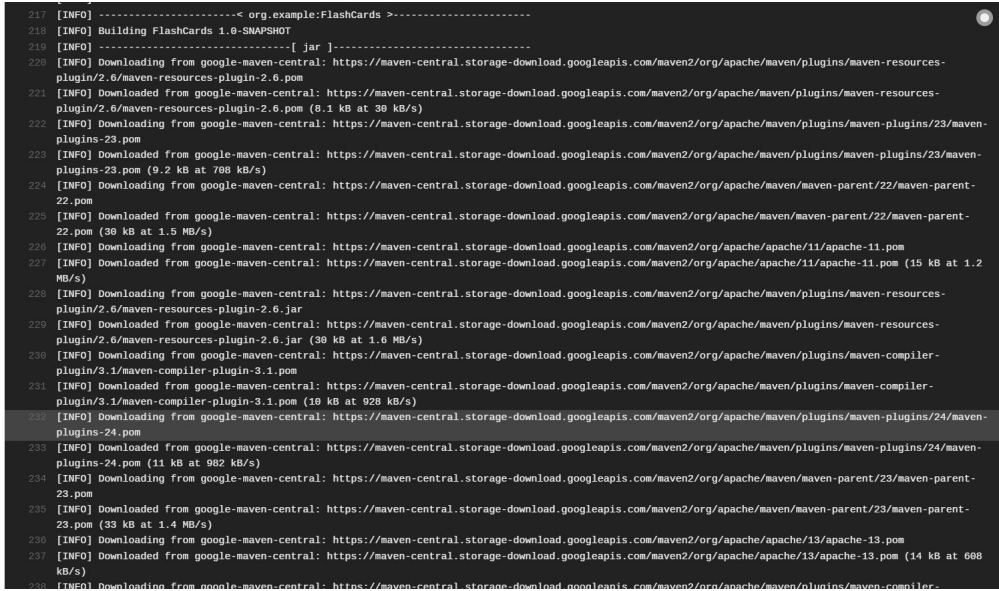
Compiling is "easy" when all your source code is here

Nowadays, your code is not "here"

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

Study the CI log:

- What is it doing?
- Downloading, compiling, running checks
- Most of this is "building", using Maven
- More on CI later



Quick overview of today's toolchain: Build Systems

- Has 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; we're not attached to one
  - For JS/TS: Node(JS)
    - Generally coupled with the Node Package Manager (NPM)
- Often built into IDEs, as plugins

Abstraction, Reuse, and Programming Tools

- For each in {IDE, **Build systems**, libraries, CI}:
  - What is it today?
  - **What is under the hood?**
- What is next?

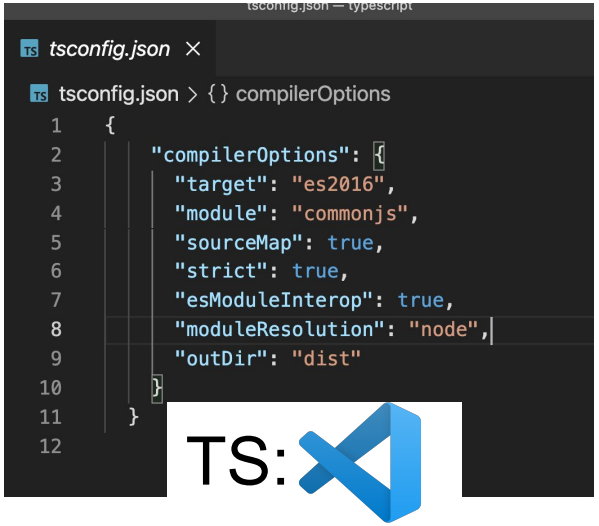
Under the Hood: Build Systems

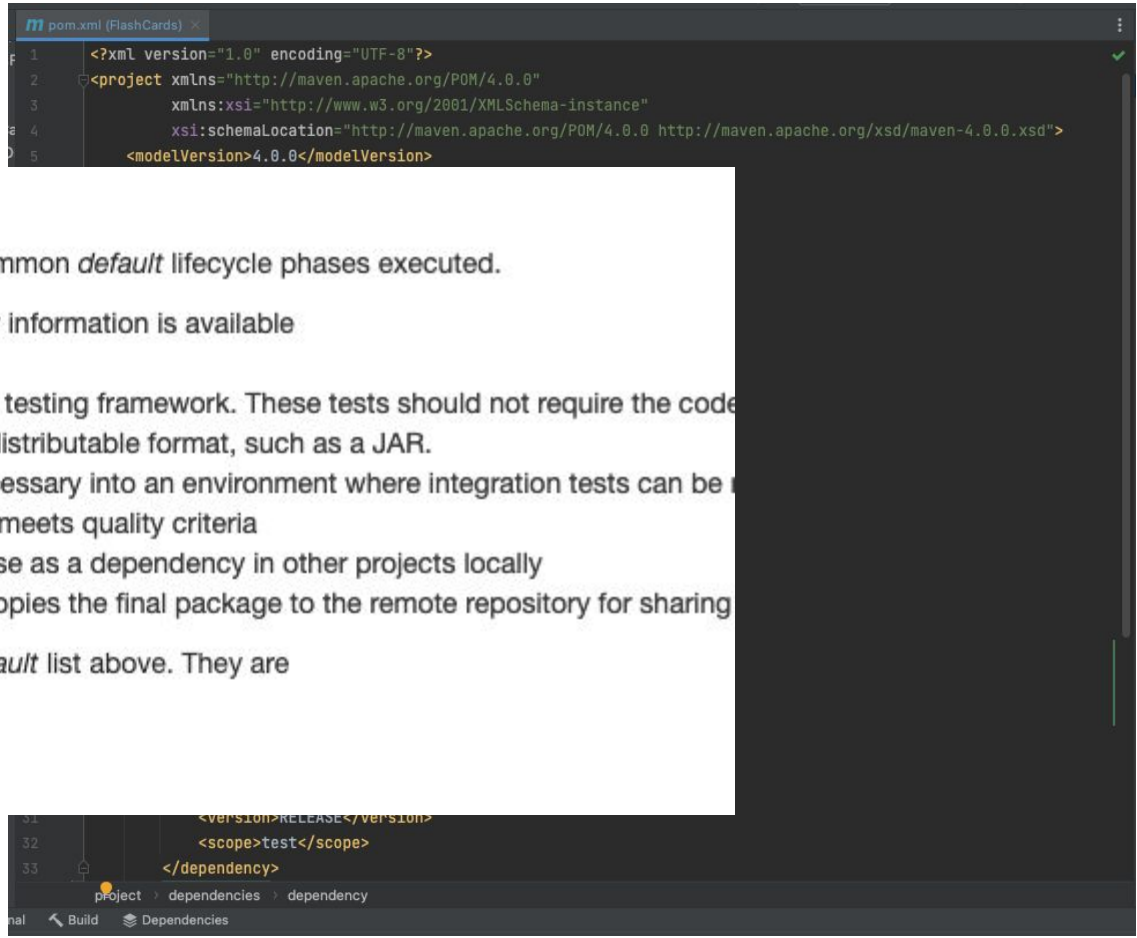
- These days: intricately tied with IDEs, package managers
- Projects often come with a build config file or two
  - ‘pom.xml’ for Maven
  - ‘tsconfig.json’ + ‘package.json’ for TypeScript+NPM -- the second deals with packages
  - These can be nested, one per (sub-)directory, to compose larger systems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

Under the Hood: Build Systems

Projects 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
- Specifies:
  - Compilation source and target version
  - High-level configuration options
  - Targets for various phases in development
    - “lifecycle” in Maven; e.g. ‘compile’, ‘test’, ‘deploy’
  - Often involving plugins
  - Dependencies with versions





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

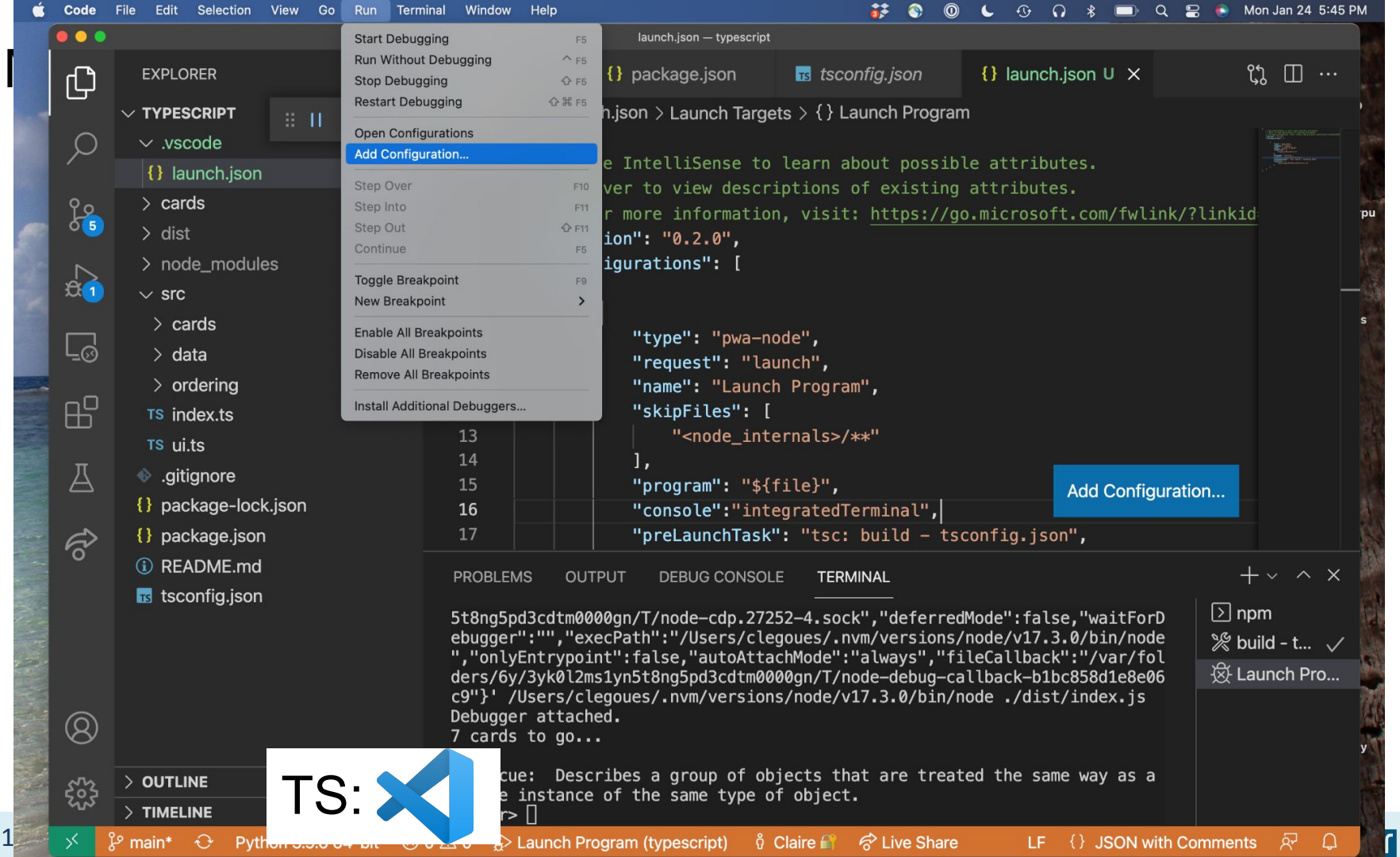
- Node.js is a JS runtime. npm is its package manager.

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
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...
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
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,"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...
cue: Describes a group of objects that are treated the same way as a
e instance of the same type of object.
npm
build - t...
Launch Pro...
TS:
main*
Launch Program (typescript)
Claire
Live Share
LF
JSON with Comments

{} launch.json U ×

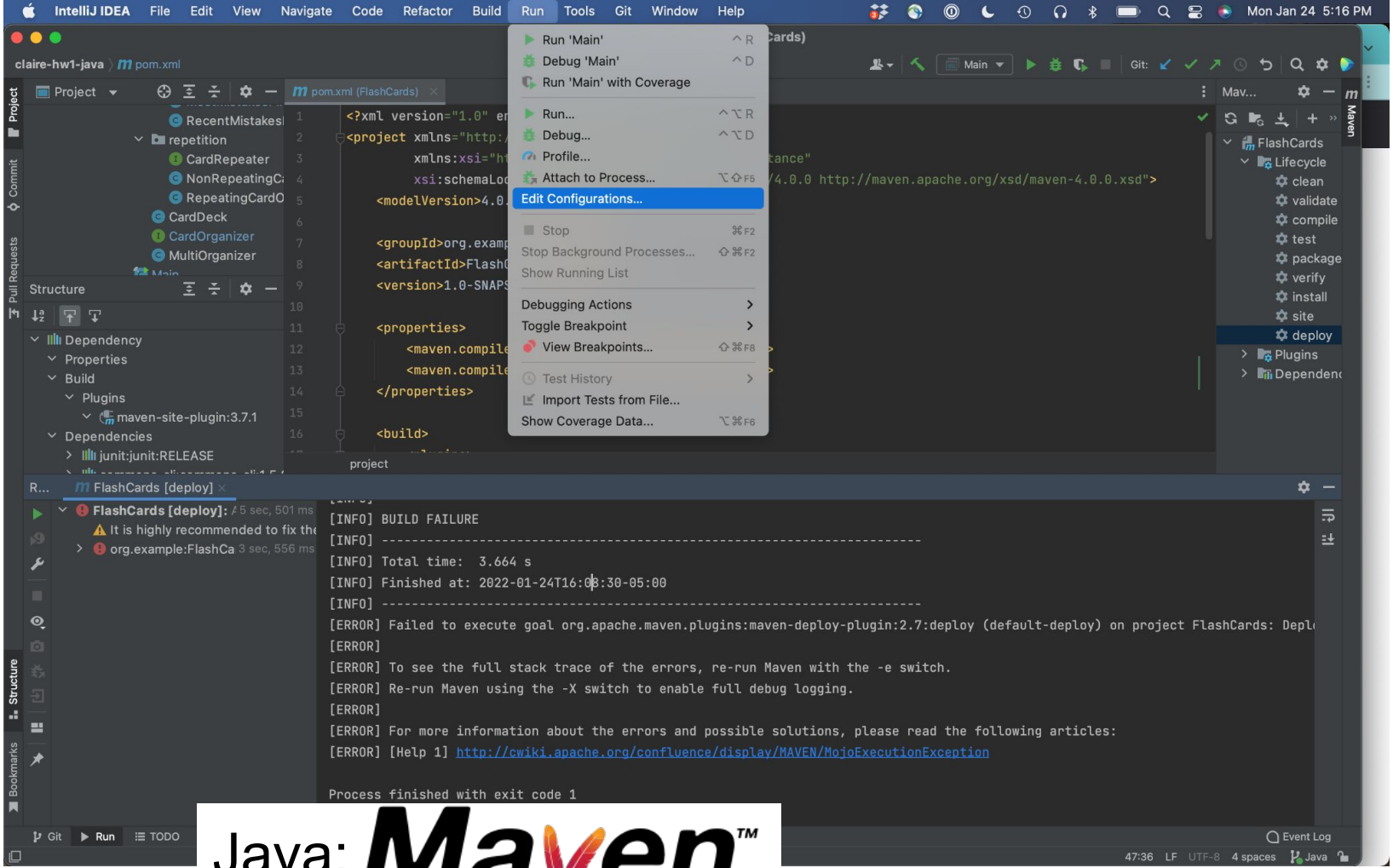
.vscode > {} launch.json > Launch Targets > {} Launch Program

```
{
    // Use IntelliSense to learn about
    // Hover to view descriptions of e
    // For more information, visit: ht
    "version": "0.2.0",
    "configurations": [

        {
            "type": "pwa-node",
            "request": "launch",
            "name": "Launch Program",
            "skipFiles": [
                "<node_internals>/**"
            ],
            "program": "${file}",
            "console":"integratedTerminal",
            "preLaunchTask": "tsc: build - tsconfig.json",
            "args": [ "--help" ],
            "outFiles": [
                "${workspaceFolder}/dist/**/*.js"
            ]
        }
    ]
}
```

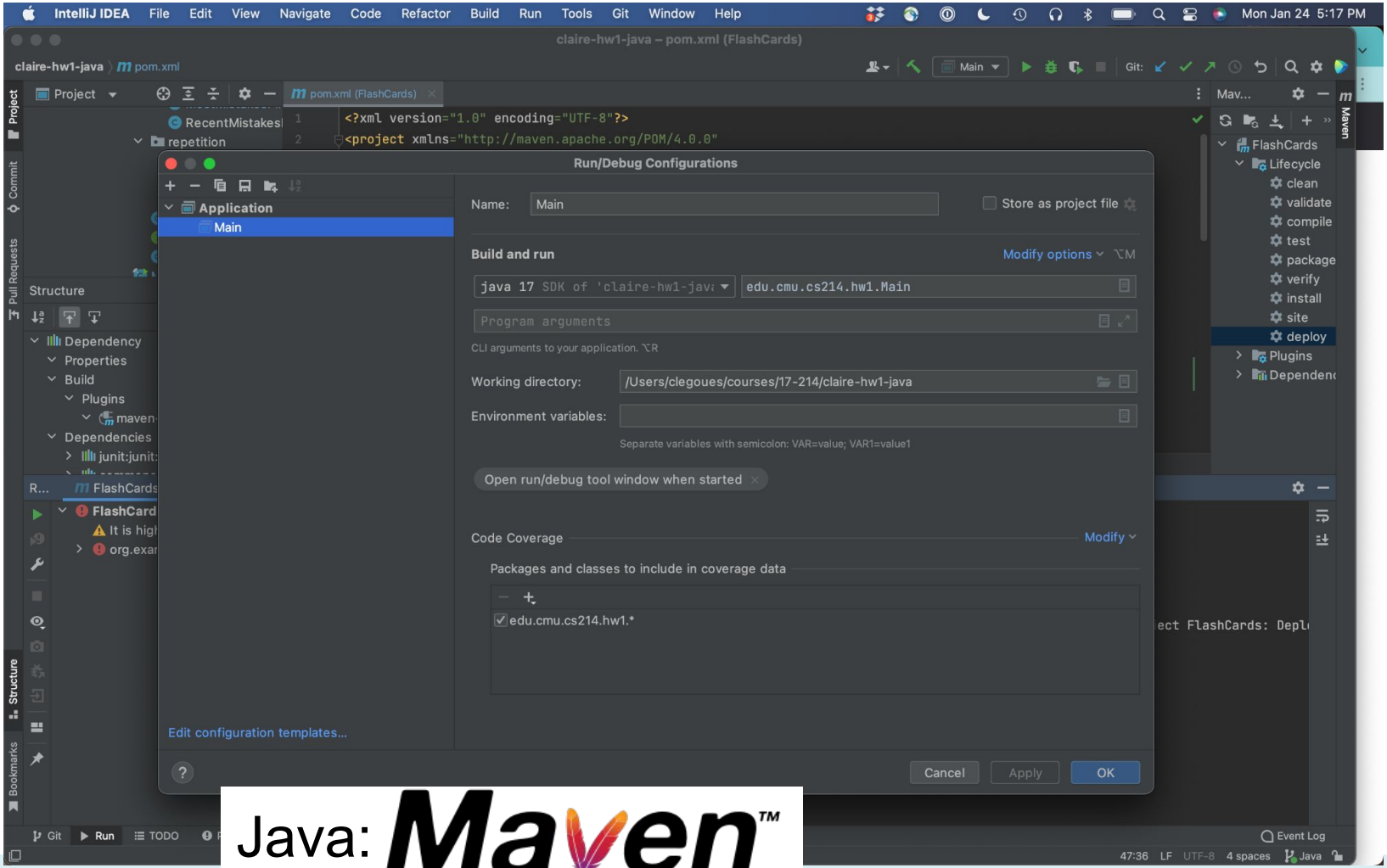
```
node/v17.3.0/bin/node ./dist/index.js --help
Debugger attached.
zsh: killed     /usr/bin/env   /Users/clegoues/.nvm/versions/node/v17.3.0/bin/n
ode  --help
(base) clegoues@clegoues-macbook-air typescript % npm run start -- --help
```

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

TS:


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Mon Jan 24 5:16 PM
claire-hw1-java pom.xml
Run 'Main'
Debug 'Main'
Run 'Main' with Coverage
Run...
Debug...
Profile...
Attach to Process...
Edit Configurations...
Stop
Stop Background Processes...
Show Running List
Debugging Actions
Toggle Breakpoint
View Breakpoints...
Test History
Import Tests from File...
Show Coverage Data...
Project
pom.xml (FlashCards)
RecentMistakes
repetition
CardRepeater
NonRepeatingCa
RepeatingCardO
CardDeck
CardOrganizer
MultiOrganizer
Structure
Dependency
Properties
Build
Plugins
maven-site-plugin:3.7.1
Dependencies
junit:junit:RELEASE
<groupId>org.examp
<artifactId>FlashC
<version>1.0-SNAPS
<properties>
</properties>
<build>
project
Maven
FlashCards
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
FlashCards [deploy]
FlashCards [deploy]: 5 sec, 501 ms
It is highly recommended to fix the
org.example:FlashCa 3 sec, 556 ms
[INFO] BUILD FAILURE
[INFO] Total time: 3.664 s
[INFO] Finished at: 2022-01-24T16:03:30-05:00
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-deploy-plugin:2.7:deploy (default-deploy) on project FlashCards: Deplo
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException
Process finished with exit code 1
Git Run TODO
Event Log
47:36 LF UTF-8 4 spaces Java

Java: Maven™

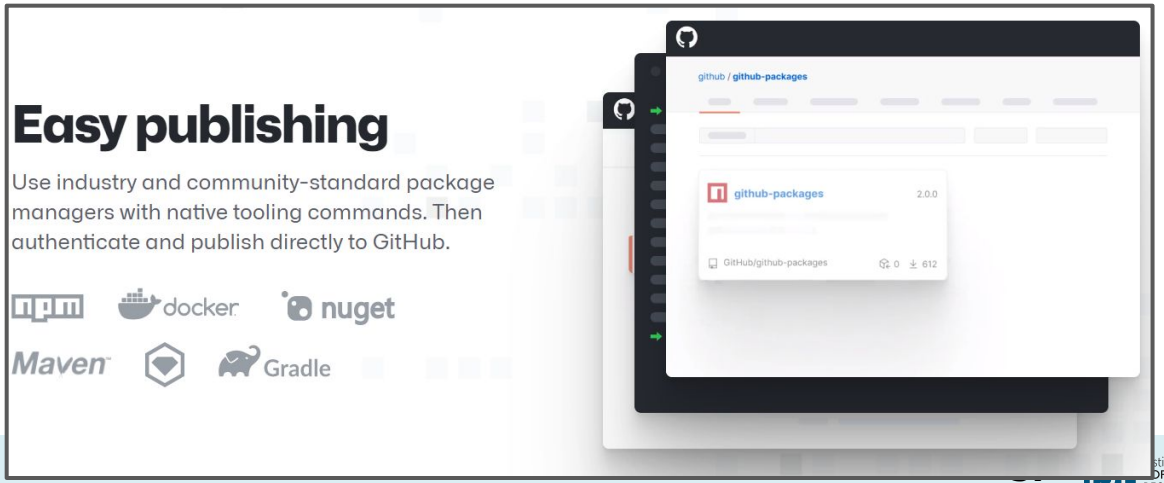

IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Mon Jan 24 5:17 PM
claire-hw1-java – pom.xml (FlashCards)
Run/Debug Configurations
Application
Main
Name: Main
Store as project file
Build and run
Modify options
java 17 SDK of 'claire-hw1-java'
edu.cmu.cs214.hw1.Main
Program arguments
CLI arguments to your application.
Working directory: /Users/clegoues/courses/17-214/claire-hw1-java
Environment variables:
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Code Coverage
Modify
Packages and classes to include in coverage data
edu.cmu.cs214.hw1.*
Edit configuration templates...
Cancel
Apply
OK
Java: Maven™

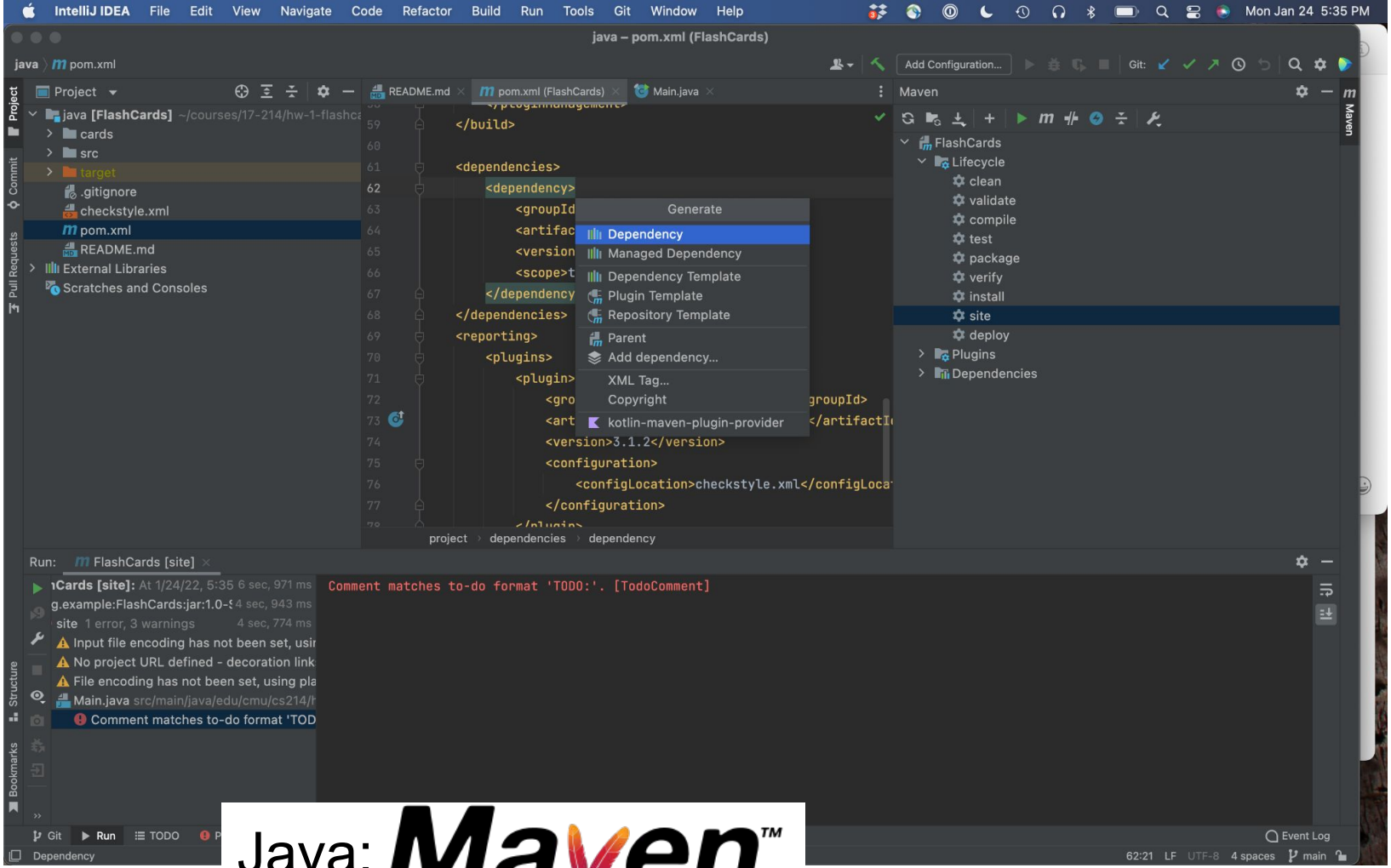
Abstraction, Reuse, and Programming Tools

- For each in {IDE, Build systems, **libraries**, CI}:
  - **What is it today?**
  - What is under the hood?
- What is next?

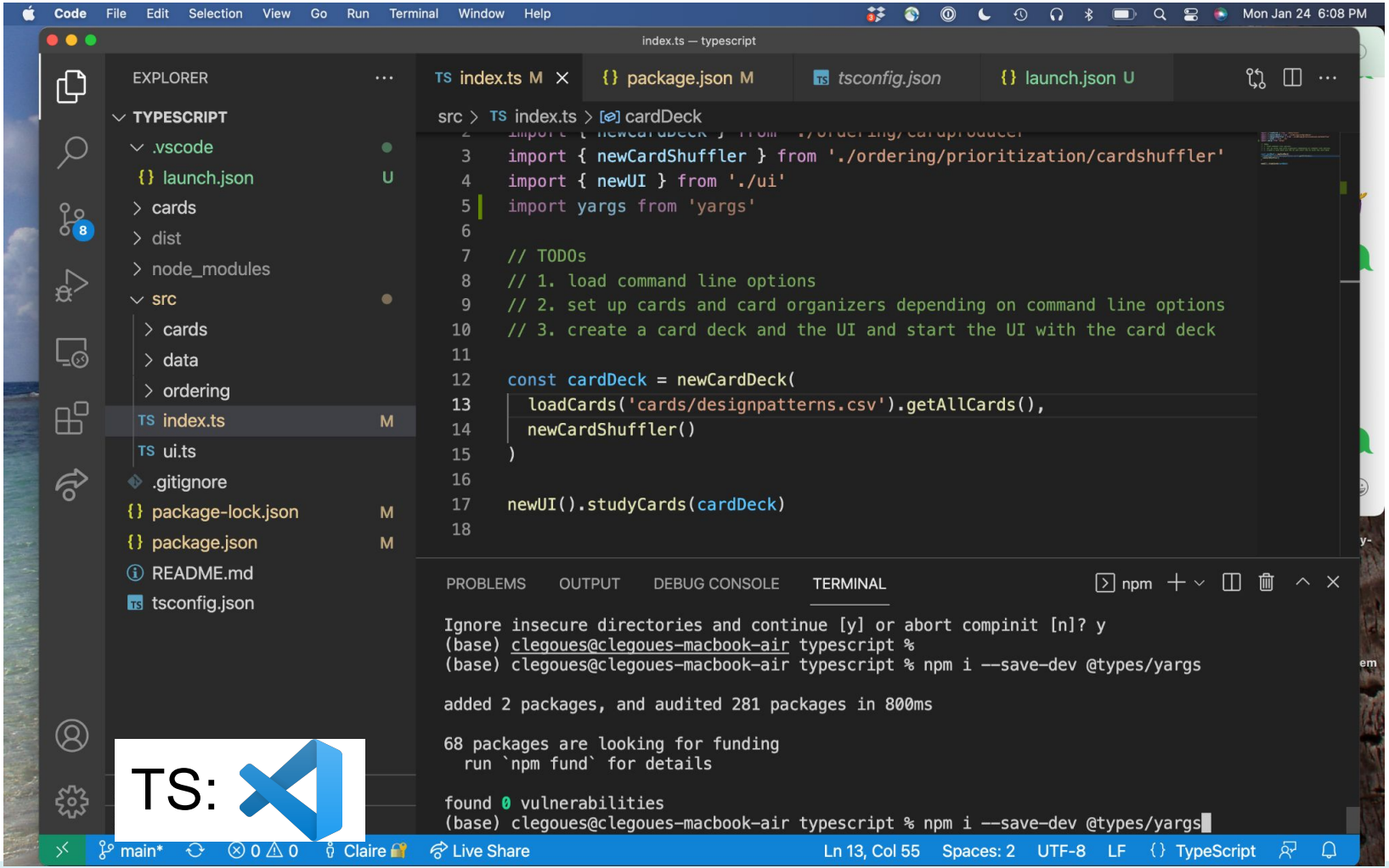
Quick overview of today's toolchain: Libraries

- Myriad. 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




IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Mon Jan 24 5:35 PM
java – pom.xml (FlashCards)
java pom.xml
Add Configuration...
Git:
Project
java [FlashCards] ~/courses/17-214/hw-1-flashca
cards
src
target
.gitignore
checkstyle.xml
pom.xml
README.md
External Libraries
Scratches and Consoles
README.md
pom.xml (FlashCards)
Main.java
59 </build>
61 <dependencies>
62 <dependency>
63 <groupId
64 <artifac
65 <version
66 <scope>t
67 </dependency
68 </dependencies>
69 <reporting>
70 <plugins>
71 <plugin>
72 <gro groupId>
73 <art </artifactI
74 <version>3.1.2</version>
75 <configuration>
76 <configLocation>checkstyle.xml</configLoca
77 </configuration>
Generate
Dependency
Managed Dependency
Dependency Template
Plugin Template
Repository Template
Parent
Add dependency...
XML Tag...
Copyright
kotlin-maven-plugin-provider
project dependencies dependency
Maven
FlashCards
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
Dependencies
Run: FlashCards [site]
Cards [site]: At 1/24/22, 5:35 6 sec, 971 ms
g.example:FlashCards:jar:1.0-S 4 sec, 943 ms
site 1 error, 3 warnings 4 sec, 774 ms
Input file encoding has not been set, usir
No project URL defined - decoration link
File encoding has not been set, using pla
Main.java src/main/java/edu/cmu/cs214/h
Comment matches to-do format 'TOD
Comment matches to-do format 'TODO:'. [TodoComment]
Git Run TODO
Event Log
Dependency
62:21 LF UTF-8 4 spaces main

Java: Maven™

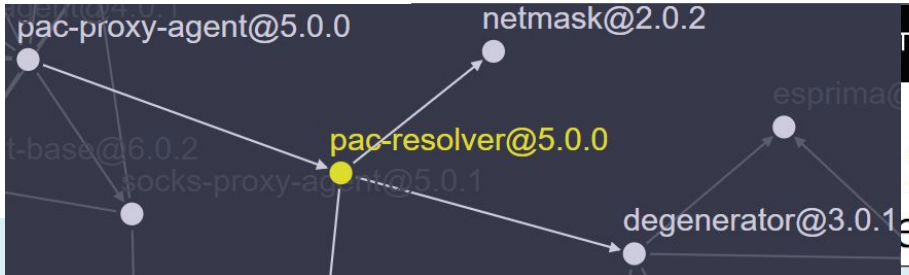

Code File Edit Selection View Go Run Terminal Window Help
Mon Jan 24 6:08 PM
index.ts — typescript
EXPLORER
TYPESCRIPT
.vscode
launch.json U
cards
dist
node_modules
src
cards
data
ordering
index.ts M
ui.ts
.gitignore
package-lock.json M
package.json M
README.md
tsconfig.json
index.ts M
package.json M
tsconfig.json
launch.json U
src > 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
TS:

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

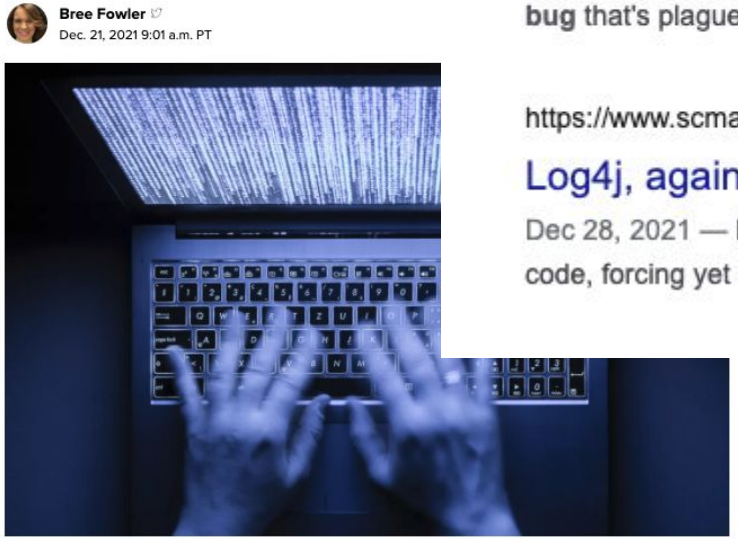
- A Case-Study:
  - 'pac-resolver' (3M weekly downloads) has a major security vulnerability
    - Uses 'degenerator' (same author), which misuses a Node module
    - "The vm module is not a security mechanism. Do not use it to run untrusted code."
    - (a mistake that's been made before: people rarely 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.

Abstraction, Reuse, and Programming Tools

- For each in {IDE, Build systems, **libraries**, CI}:
  - What is it today?
  - **What is under the hood?**
- What is next?

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:

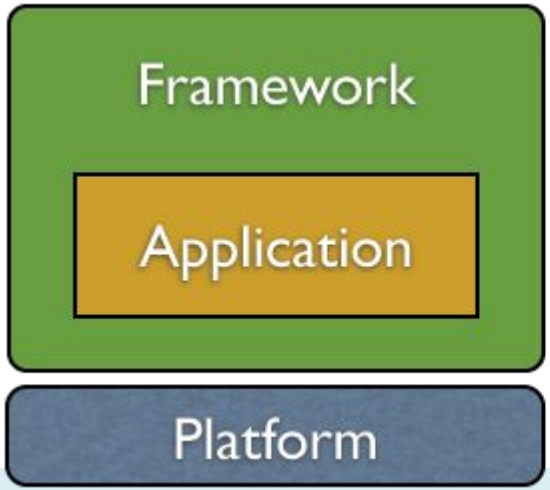
- Libraries:
  - A set of classes and methods that provide reusable functionality
  - Typically: programmer calls, library returns data, that's it.
- Frameworks:
  - Reusable skeleton code that can be customized into an application
  - Framework calls back into client code
    - The Hollywood principle: "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 {IDE, Build systems, libraries, **CI**}:
  - **What is it today?**
  - What is under the hood?
- 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

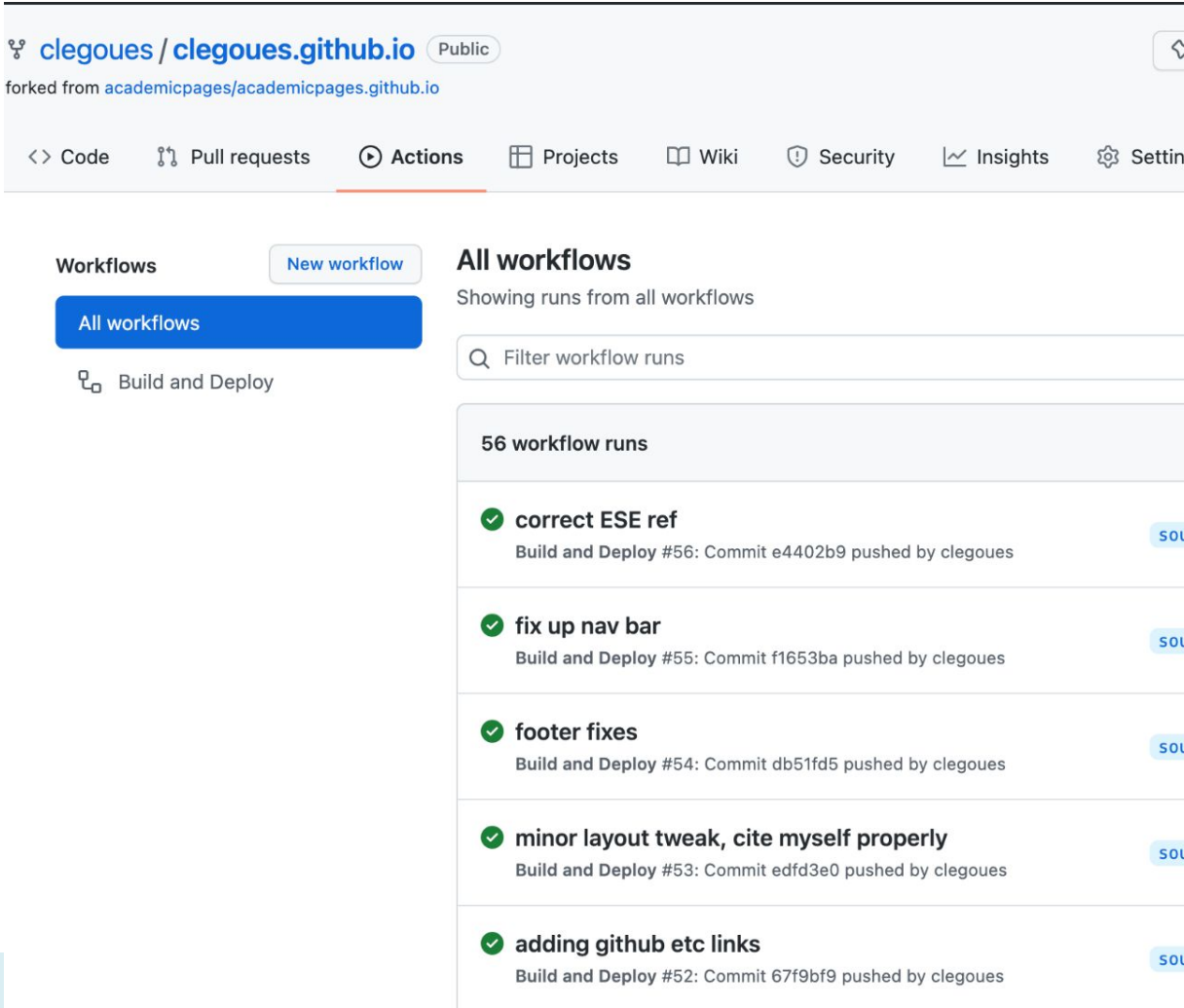
Under the Hood: Continuous Integration

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



```
Installing SSH key from: default repository key                                  ssh_key
Using /home/travis/.netrc to clone repository.

$ 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
Cloning into 'CMU-17-214/template-21f-hw1'...
remote: Enumerating objects: 117, done.
remote: Counting objects: 100% (117/117), done.
remote: Compressing objects: 100% (73/73), done.
remote: Total 117 (delta 50), reused 104 (delta 37), pack-reused 0
Receiving objects: 100% (117/117), 69.89 KiB | 2.25 MiB/s, done.
Resolving deltas: 100% (50/50), done.
$ cd CMU-17-214/template-21f-hw1
$ git checkout -qf 0d657225c8cbdd52751c2f88527f93f4099b041e

                                                                                 0.01s
$ nvm install 16                                                                 nvm.install   3.65s
Downloading and installing node v16.8.0...
Downloading https://nodejs.org/dist/v16.8.0/node-v16.8.0-linux-x64.tar.xz...
Computing checksum with sha256sum
Checksums matched!
Now using node v16.8.0 (npm v7.21.0)

Setting up build cache                                                           cache.1

                                                                                 cache.npm
$ node --version
v16.8.0
$ npm --version
7.21.0
$ nvm --version
0.38.0

$ npm ci                                                                         install.npm   3.04s

$ timeout 5m npm run compile                                                     4.93s

> hw1-flashcards@1.0.0 compile
> tsc
```

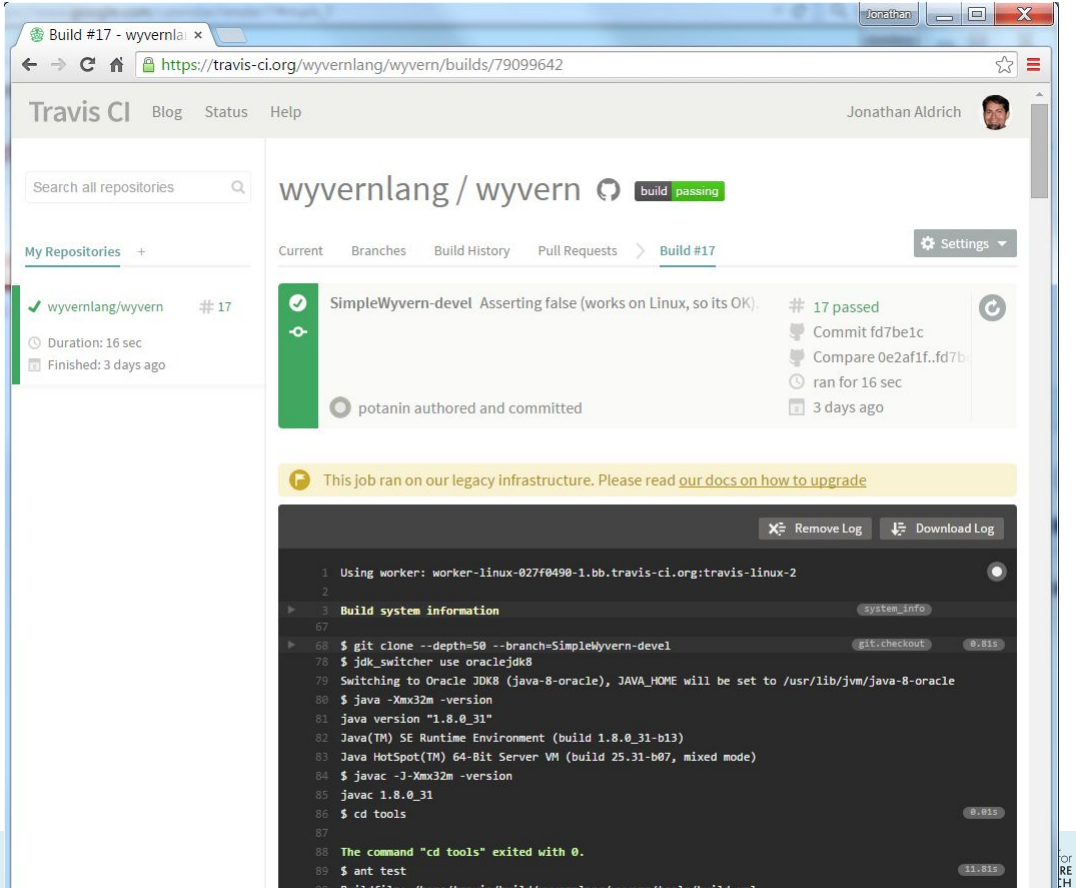
Under the Hood: Continuous Integration

Automatically builds, tests, and displays the result

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

- Until they randomly stopped supporting OSS.

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



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

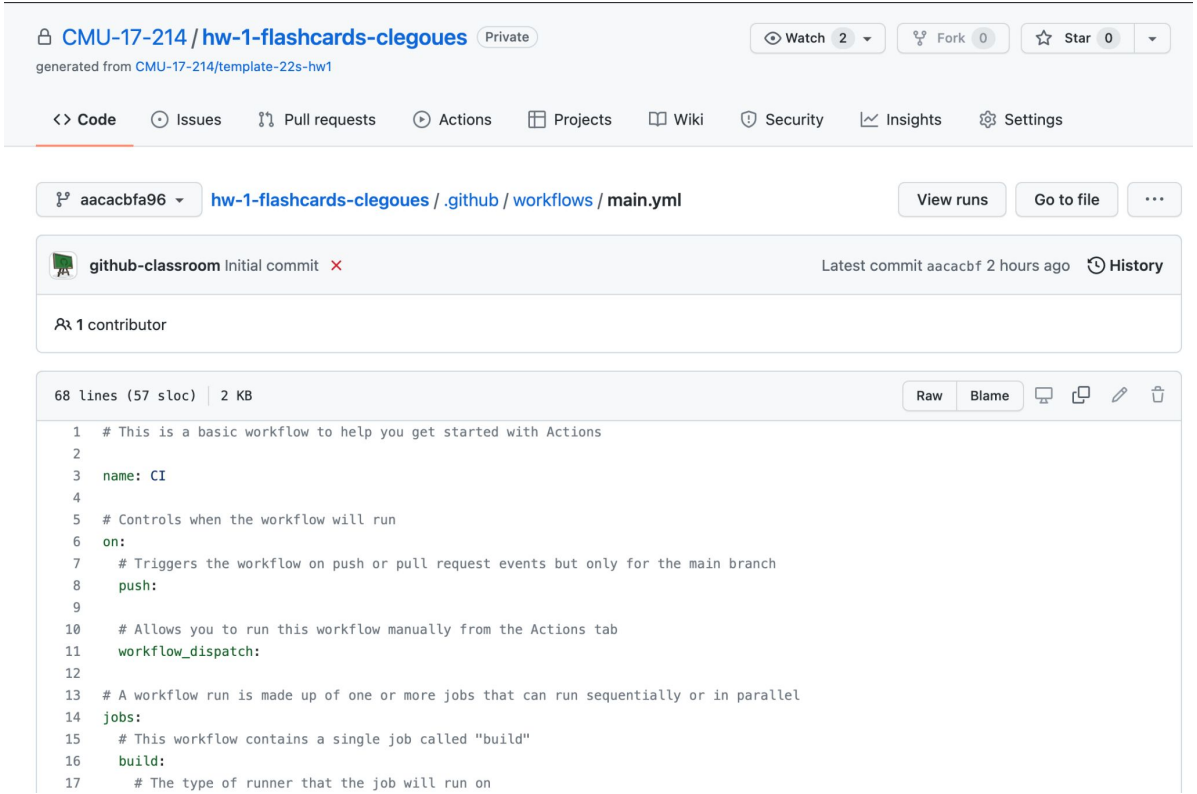
Behind the Abstraction: Some Nuance

- 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`



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

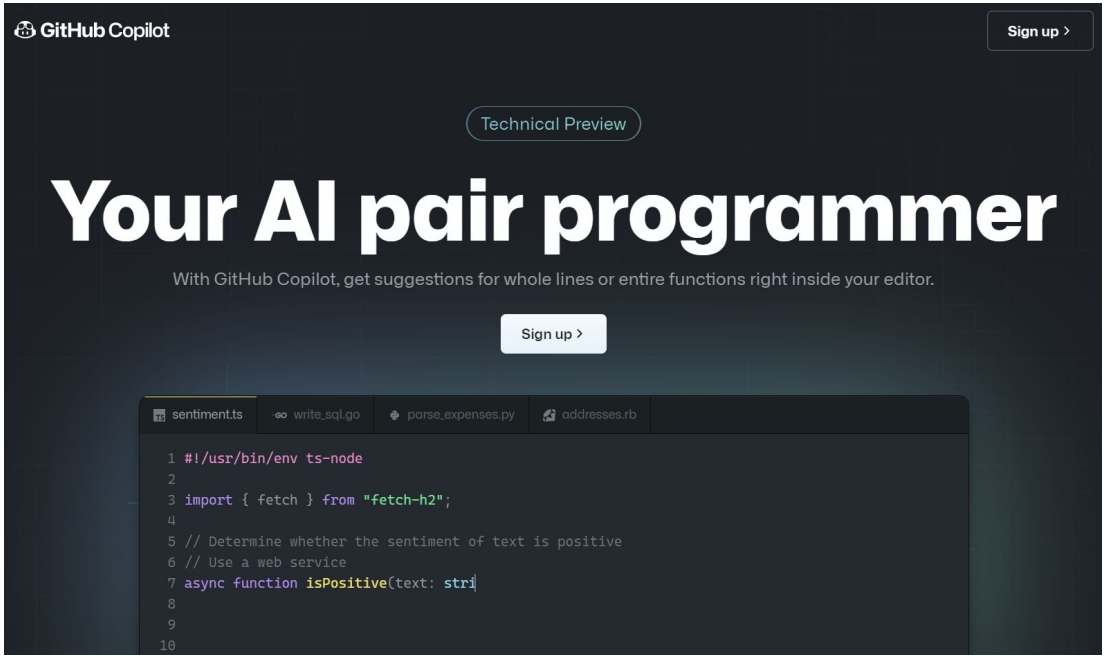
- When/how do we leave the abstraction?
- Command-line comes built into IDEs for a reason
- Non-trivial in general! May require switching/“patching” libraries
  - E.g., Maven → Gradle for more unusual build routines

Abstraction, Reuse, and Programming Tools

- For each in {IDE, Build systems, libraries, CI}:
  - What is it today?
  - What is under the hood?
- **What is next?** *(any guesses?)*

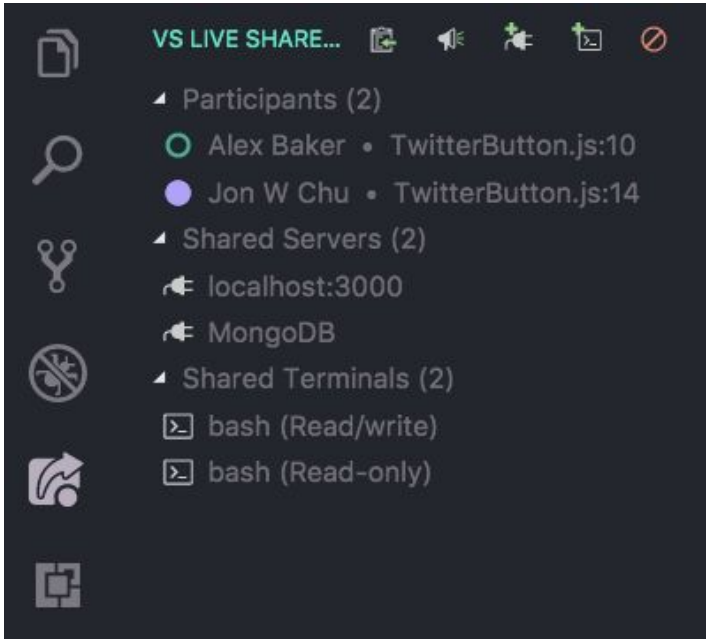
What's Next: AI Powered Programming

- Easier in Web IDEs
  - Which are themselves "next"



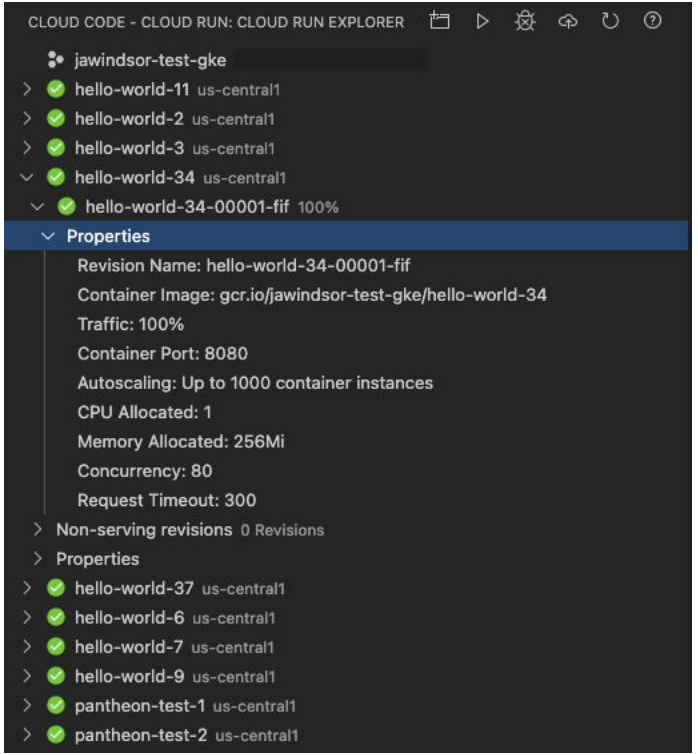
What's Next: Collaborative online coding

- 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!