Principles of Software Construction: Objects, Design, and Concurrency

IDEs, Build system, Continuous Integration, Libraries

Christian Kästner Vincent Hellendoorn





Outline

- On Homework 1
- 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?



Homework 1

Welcome to the deep end!

• Java/TS + IDE + Maven/Npm + GitHub + Travis + linter!?



Homework 1

Welcome to the deep end!

- Java/TS + IDE + Maven/Npm + GitHub + Travis + linter!?
- We're here to help:
 - Recitation tomorrow, walks through all this setup
 - Find some clarifications on Piazza
 - E.g., only implement what is asked for; all other functionality (repeating, flipping question/answer) is already there.
 - Use office hours (see course calendar)





Homework 1

Welcome to the deep end!

- Java/TS + IDE + Maven/Npm + GitHub + Travis + linter!?
- We're here to help:
 - Recitation tomorrow, walks through all this setup
 - Find some clarifications on Piazza
 - E.g., only implement what is asked for; all other functionality (repeating, flipping question/answer) is already there.
 - Use office hours (see course calendar)
- Actual coding effort is small -- reading & setting up is the point
- Small typo detected on Piazza in `mostmistakes.ts`; fixed now. Not essential to your HW.





Mini-quiz

https://forms.gle/9tnB5BszVz9KTY7r5





Outline

- On Homework 1
- 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?







17-214/514



0 0 H A A A +	MSD120 mag 4 E 2
	complier options
<pre>1 // Type your code here, or load an example. 2 int square(int num) {</pre>	11010 .LX0: .text // Intel A A A +
3 return num * num;	1
4 }	2 /*******************
	<pre>3 * Function `square(int)'</pre>
	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
	1/ pop r4
	10 pop F10
	19 FEL

institute for

IS

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



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" (we will)
- All of the above, entangled



Outline

- On Homework 1
- 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?



- 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:

17-214/514

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



institute fo

- Recently, IntelliJ has been more dominant
 - Packs a lot of "recipes" to create certain types of projects (e.g., web-app with Spring & Maven)



- 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



- 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
 - For this homework, choose what you'd like. We suggest IntelliJ for Java, VSCode for TS



Build Systems:

• How does this happen?

C++ source #1	#1 with MSP430 gcc 4.5.3 ×
• • H ∧ A A +	MSP430 gcc 4.5.3 Compiler options
<pre>1 // Type your code here, or load an example. 2 int square(int num) {</pre>	11010 LX0: .text // Intel A A A +
3 return num * num; 4 }	1 2 /************************************
	6 push r10 7 push r4
	8 mov r1, r4 9 add #4, r4
	11 mov r15, -6(r4) 12 mov -6(r4), r10
	13 mov -6(r4), r12 14 call #_mulhi3

institute fo

IS

Build Systems:

- Compiling is "easy" when all your source code is here
 - (Please don't tell a compiler expert I said that)
- Nowadays, your code is not "here"
 - Even libraries that you use in the IDE!
 - Interfaces make that possible



Build Systems:

- Compiling is "easy" when all your source code is here
 - (Please don't tell a compiler expert I said that)
- Nowadays, your code is not "here"
 - Even libraries that you use in the IDE!
 - Interfaces make that possible
- Study the Travis log:
 - What is it doing?
 - Downloading, compiling, running checks
 - Most of this is "building", using Maven
 - More on Travis later

		,
	217	[INFO]
		[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
		[INF0] 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/23/maven-plugins/23/ma
		plugins-23.pom
		[INF0] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/23/mav
		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
		[IWF0] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/apache/11/apache-11.pom (15 kB at un (2) un (2
_		MD/S) [TNFO] Demlanding from gangle much parts], https://pougn.control.stange.domland.ganglenic.ar/pougn//arg/anghe/much/alumins/much-sonurses
s I		[INFO] DUMINGALING FOR GOODER MARENCENTERT. HTtps://maren-tentral.storage-uumingad.googleapis.tom/marenz/org/apathe/maren/plugins/maren-tesourtes-
~		pingin/2.0/maydm=resumres-pingin/2.0.jai TIMEN Downloadad fram annah_mayan_cantral: https://mayan_cantral.etorana_download_annaha/mayan/anacha/mayan/alunins/mayan_resource-
		[In o] comitouted from google-meter-central: nets://mater-central:storage-domitout.googleapis.com/mater.com/graphenet/program.com/program.com/mater.com/ces- nlumin/2 (mayon-resources-nlumin.2 & far (38 kB at 1 & MR/c))
		program i om meter resources program i or parte i on the i on or of the second s
		In a point and the provided and the prov
		jungan tianan any any pangan tiang ang pangan tiang
		pluain/3.1/maven-compiler-pluain-3.1.pom (19 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/mm
		plugins-24.pom
		[INFO] Downloaded from google-maven-central: https://maven-central.storage-download.googleapis.com/maven2/org/apache/maven/plugins/maven-plugins/24/mav
		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

ven/nlugins/maven-

17-214/514

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)



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



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



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



24



Libraries:

- A Case-Study:
 - 'pac-resolver' (3M weekly downloads) has a major security vulnerability
 - Uses 'degenerator' (same author), which misuses a Node module
 - <u>"The vm module is not a security mechanism. Do not use it to run untrusted code."</u>
 - (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

25



Continuous Integration:

- 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



Not mentioned:

- Docker: containerize applications for coarse-grained reuse
- Cloud: deploy and scale rapidly, release seamlessly
- Bug/Issue trackers, often integrated with reviews



Outline

- On Homework 1
- 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?



First, a bit of 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





First, a bit of 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





First, a bit of nuance:

- Most tools are abstractions of common commands
 - Typically operated via GUI and/or a DSL
 - Obvious for Travis: just read the Yaml
 - Script-like languages are common
 - Involving a vocabulary of "targets"
 - E.g., `mvn site`

ဦး Ja	va 👻 hw1-flash-cards-VHellendoorn / .travis.yml
陳 g	github-classroom Initialize Java 🗙
ጸኣ 1 co	ontributor
4 line	es (4 sloc) 72 Bytes
1	language: java
2	jdk: oraclejdk16
3	script:
4	- timeout 5m mvn site

17-214/514

First, a bit of nuance:

- Most tools are abstractions of common commands
 - Typically operated via GUI and/or a DSL
 - Obvious for Travis: 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
 - $\blacksquare \quad E.g., Maven \rightarrow Gradle \text{ for more unusual build routines}$



Outline

- On Homework 1
- 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?



- 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





• 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`)



- Debugging
 - \circ $\,$ Often the default mode when you run in the IDE







- Debugging
 - \circ $\,$ Often the default mode when you run in the IDE
 - Allows setting breakpoints
 - Which give you rich insight into execution



- Debugging
 - \circ $\,$ Often the default mode when you run in the IDE
 - Allows setting breakpoints
 - Which give you rich insight into execution

RUN ▷ No Configurat > 😚 …	≣ en_de.txt	🕏 translate.py 🗙	8	10 ?	• •	•	C	
<pre> VARIABLES V Locals Special variables de: 'ich\n' en: 'I' </pre>	<pre>translate.py 1 index 2 with o 3 fo 4 5</pre>	<pre>/ > = {} pen('en_de.txt') as f: r l in f: en, de = l.split(' ') index[en] = de</pre>						
<pre>> f: <_io.TextIOWrapper name='e > index: {'I': 'ich\n'} 1: 'I ich\n'</pre>	6							
17-214/514							38	isr

• 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





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



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
 - 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
 - Not shown: in package.json

```
{
    "compilerOptions": {
        "target": "es2016",
        "module": "commonjs",
        "sourceMap": true,
        "strict": true,
        "esModuleInterop": true,
        "moduleResolution": "node",
        "outDir": "dist"
    }
```

RESEARCH

1

2

3

4

6

8

9

0

17-214/514

Packages can be either:

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



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



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



Which kind is a command-line parsing package?



17-214/514

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



Which kind is a command-line parsing package?

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

• More on Thursday



17-214/514

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



Libraries

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





Continuous Integration

Defines a series of <u>actions</u> to be run in a <u>clean build</u>:

- 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



		Tesotycom	
	Installing SSH key from: default repository key	ssh_key	
	Using /home/travis/.netrc to clone repository.		
	<pre>\$ git clonedepth=50branch=TypeScript https://github.com/CMU-17-214/template-21f-hw1.git CMU-17-214/template-21f-hw1</pre>	git.checkout	1.75s
	Cloning into 'CMU-1/-214/template-21T-hw1'		
	remote: Enumerating objects: 117, done.		
	remote: Compressing objects: 100% (11//11/), 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
180	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 Shazəosum Checksums matchedi		
184	Now using node v16 8 A (nom v7 21 A)		
	Setting up build cache	cache.1	
		cache.npm	
	\$ nodeversion		
	v16.8.0		
	\$ npmversion		
	7.21.0		
	\$ nvmversion		
	0.38.0		
	\$ nnm ci	install nom	3 04s
210		anotarrahii	3.043
	\$ timeout 5m npm run compile		4.93s
	> hw1-flashcards@1.0.0 compile		
	> tsc		

Continuous integration – Travis CI

Automatically builds, tests, and displays the result

17-214/514

	Build #17 - wyvernla ×		- Jonathan	
Travis Cl Big Stats Hep Scarch all repositories (wyceniang/wyceni) (wyceniang/wyceniang/wycenian	C A A https://travis	-ci.org/wyvernlang/wyvern/builds/79099642		\$
Search all repositories Ayrepositories Ayrepositories Ayrepositories Ayrepositories Area Ayrepositories Area Are	Travis Cl Blog Status	Help	Jonathan Aldrich	
Ay Repositories + Current Branches Build History Pull Requests Build #17 Wysenlang/wysern #17 Duration: 16 sec Finished: 3 days ago SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Duration: 16 sec Pinished: 3 days ago Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Duration: 16 sec Potanin authored and committed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 17 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 18 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 18 passed Image: SimpleWysern-devel Asserting false (works on Linux, so its OK). # 18 passed # 19 masserting false (works on Linux, so its OK). # 19 masserting false (works on Linux, so its OK). # 19 masserting false (works on Linux, so its OK). # 19 masserting false (works on Linux, so its OK). # 19 masserting false (works on Linux, so its OK). <t< td=""><td>Search all repositories Q</td><td>wyvernlang/wyvern 🗘 build passing</td><td></td><td></td></t<>	Search all repositories Q	wyvernlang/wyvern 🗘 build passing		
wyvernlang/wyvern # 17 SimpleWyvern-devel Asserting false (works on Linux, so its OK). # 17 passed Finished: 3 days ago SimpleWyvern-devel Asserting false (works on Linux, so its OK). # 17 passed Commit fd7be1c Compare 0e2aff.fd7b To ran for 16 sec 3 days ago This job ran on our legacy infrastructure. Please read our docs on how to upgrade Using worker: worker-linux-027f0490-1.bb.travis-ci.org:travis-linux-2 Build system information 5 git clonedepth-50branch-55mpleWyvern-devel git clonedep	Ay Repositories +	Current Branches Build History Pull Requests > Build #17	🗘 Setti	ings 🔻
potanin authored and committed 3 days ago This job ran on our legacy infrastructure. Please read our docs on how to upgrade Image: the system Image: the system <td> ✓ wyvernlang/wyvern # 17 ○ Duration: 16 sec ☐ Finished: 3 days ago </td> <td>SimpleWyvern-devel Asserting false (works on Linux, so its OK).</td> <td> # 17 passed Commit fd7be1c Compare 0e2af1ffd7b ran for 16 sec </td> <td>C</td>	 ✓ wyvernlang/wyvern # 17 ○ Duration: 16 sec ☐ Finished: 3 days ago 	SimpleWyvern-devel Asserting false (works on Linux, so its OK).	 # 17 passed Commit fd7be1c Compare 0e2af1ffd7b ran for 16 sec 	C
This job ran on our legacy infrastructure. Please read our docs on how to upgrade X* Remove Log I Download Log Using worker: worker-linux-827f9499-1.bb.travis-ci.org:travis-linux-2 I Build system information System_info Sid system information System_info System information System_info Sid system information System_info Sid system information System_info <		O potanin authored and committed	3 days ago	
<pre>K Remove Log Losing worker: worker-linux-827f64980-1.bb.travis-ci.org:travis-linux-2 Using worker: worker-linux-827f64980-1.bb.travis-ci.org:travis-linux-2 Build system information Using worker: worker-linux-827f64980 Using worker: worker-1.bb.travis-1.bb.travis-1.bb.travis-1.bt.travi</pre>		This job ran on our legacy infrastructure. Please read our docs on the second sec	now to upgrade	
1 Using worker: worker-linux-827f0490-1.bb.travis-ci.org:travis-linux-2 2 3 Build system information 6 5 5 git clonedepth-50branch-SimpleWyvern-devel git.checkout 7 5 6 5 7 5 8 5 9 Switching to Oracle JDK8 (java-8-oracle), JAVA_HOME will be set to /usr/lib/jvm/java-8-oracle 8 5 9 Sava version 10 java version 11 Java (TM) SE Runtime Environment (build 1.8.0_31-b13) 12 Java(TM) SE Runtime Environment (build 25.31-b07, mixed mode) 16 5 17 S (d tools 18 \$ cd tools* exited with 0. 19 Sava test		1	X∓ Remove Log ↓∓ Download	d Log
 Build system information System_information System_informat		1 Using worker: worker-linux-027f0490-1.bb.travis-ci.org:travis-lin	านx-2	۲
 \$ git clonedepth-50branch-SimpleMyvern-devel git.checkout 0.835 \$ jdk_switcher use oraclejdk8 \$ switching to Oracle JDKB (java-8-oracle), JAVA_HOME will be set to /usr/lib/jvm/java-8-oracle \$ java -Xmx32m -version 1 java version "1.8.0_31" 2 Java(TM) SE Runtime Environment (build 1.8.0_31-b13) 3 Java HotSpot(TM) 64-Bit Server VM (build 25.31-b67, mixed mode) \$ javac -3-Xmx32m -version \$ javac -3-Xmx32m -version \$ javac -3-Xmx32m -version \$ favac -1.8.0_31 \$ c d tools * The command "cd tools" exited with 0. * ant test 		Build system information	(system_info)	
67 88 The command "cd tools" exited with 0. 89 \$ ant test		64 \$ git clonedepth=50branch=SimpleMyvern-devel 78 \$ jdk_switcher use oraclejdk8 79 Switching to Oracle JDK8 (java-8-oracle), JAVA_HOME will be set to 80 \$ java -Xmx32m -version 81 java version "1.8.0_31" 82 Java(TM) SE Runtime Environment (build 1.8.0_31-b13) 83 Java HotSpot(TM) 64-Bit Server VM (build 25.31-b67, mixed mode) 84 \$ javac -J-Xmx32m -version 85 javac 1.8.0_31 86 \$ cd tools	git.checkout	0.815 0.015
		87 88 The command "cd tools" exited with 0. 89 \$ ant test		11.815

Continuous integration – Travis CI

You can see the results of builds over time

17-214/514

Travis Cl Blog Status	Help		Jonathan Aldrich
Search all repositories Q	wyvernlang/wyvern 🗘 🔤	passing	
My Repositories +	Current Branches Build History Pull Requests		🍄 Settings 👻
✓ wyvernlang/wyvern # 17 ⊙ Duration: 16 sec	 SimpleWyvern-devel Asserting false (works on L potanin committed 	# 17 passed fd7be1c	16 sec3 days ago
II Finisheu. 5 uays agu	 SimpleWyvern-devel Debugging mac bug. potanin committed 	# 16 passed 0e2af1f	22 sec3 days ago
	 SimpleWyvern-devel Zooming in on Mac's IRBui potanin committed 	# 14 passed 8b3606f	15 sec4 days ago
	SimpleWyvern-devel Zooming in on Mac LLVM b Dop otanin committed	# 13 passed 727fc84	16 sec4 days ago
	SimpleWyvern-devel Removed outdated tests SimpleWyvern-devel Removed outdated tests SimpleWyvern-devel Removed outdated tests	# 7 passed 4684fb5	 15 sec 11 days ago
	newlexer Merge branch 'master' of https://githu image: Source of the source o	# 6 passed # 876a074	3 14 sec11 days ago
	 master Build with JDK 8 Jonathan Aldrich committed 	# 5 passed b15273c	 13 sec 11 days ago
	master fixed Travis build script syntax error	# 4 failed	S 5 sec

Outline

- On Homework 1
- 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?



Anyone care to guess?

• Can be based on something you've seen, but think will boom





AI Powered Programming

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

🔄 GitHub Copilot	Sign up >
(Technical Preview)	
Your Al pair programme)r
With GitHub Copilot, get suggestions for whole lines or entire functions right inside your editor.	
Sign up >	
sentiment.ts 👓 write_sql.go 🔹 parse_expenses.py 🔏 addresses.rb	
1 #!/usr/bin/env ts-node	
<pre>2 import { fetch } from "fetch-h2"; 4 5 // Determine whether the sentiment of text is positive 6 // Use a web service</pre>	
7 async function isPositive(text: stri 8 9 10	



Collaborative online coding

- Think: Google Docs for code
- E.g. VS Life Share
- How will this change "commits"?





Tighter IDE-to-cloud integration

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

CLO	OUD CODE - CLOUD RUN: CLOUD RUN EXPLORER	廿	⊳	嵏	ଦ୍ଧ	U	?
:	jawindsor-test-gke						
> 🤇	hello-world-11 us-central1						
> 🤇	hello-world-2 us-central1						
-> 🤇	hello-world-3 us-central1						
~ 🤇	hello-world-34 us-central1						
	hello-world-34-00001-fif 100%						
	Properties						
	Revision Name: hello-world-34-00001-fif						
	Container Image: gcr.io/jawindsor-test-gke	/hello	-wor	ld-34			
	Traffic: 100%						
	Container Port: 8080						
	Autoscaling: Up to 1000 container instance	es					
	CPU Allocated: 1						
	Memory Allocated: 256Mi						
	Concurrency: 80						
	Request Timeout: 300						
	Non-serving revisions 0 Revisions						
	Properties						
> 🤇	hello-world-37 us-central1						
-> 🤇	hello-world-6 us-central1						
> 🤇	hello-world-7 us-central1						
> 🤇	hello-world-9 us-central1						
> 🤇	pantheon-test-1 us-central1						
> 🤇	pantheon-test-2 us-central1						



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
 - Tomorrow's recitation offers help



