SWE 265P
Reverse Engineering and Modeling

Lecture 1: Introduction

Introductions

- Name, preferred pronouns, country/city of origin
- Prior work experience
- Future plans
- Favorite hobby
- Share something that nobody else in this room knows about you
- Who is next?

Before we start

- Please fill out the following survey (right now):
  - https://forms.gle/4f4eXcoqLkZUgQ5y7

(link is available in Canvas)

Reality

“One thing that was counterintuitive is that in my first ‘real’ software engineering jobs, I spent 80-90% of my coding time reading and 10% writing.” – Eric Dashofy [General Manager & Deputy CIO, The Aerospace Corporation]

Today

- Logistics
- Nature of the course
- Why reverse engineering
- Building large software
- First homework

Logistics

- Professor: André van der Hoek (andre@uci.edu)

- TA: Yirui He (yiruih@uci.edu)
- TA: Yu Lu (ylu31@uci.edu)

- https://canvas.eee.uci.edu/courses/54889

- Slack: SWE265p-sp2024

Logistics

- Office hours
  - Andre: Monday 10:00-11:00, in person, ISEB 2426
  - Yirui: Wednesday, 15:00-16:00, in person, ISEB 2402
  - Yu: Friday, 15:00-16:00, in person, ISEB 2402
  - by appointment

- Open door policy
  - stop by the office
  - ping us on Slack

Specific goals

- For you to be able to effectively use a set of strategies and tools for understanding, working with, and changing large software
- For you to be able to effectively model different aspects of large software
- For you to initiate or expand your professional portfolio by making contributions to a large open source software system

Overall goal

- For you to be able to intelligently talk about and use the materials of this course and smartly draw upon its content when you interview

A note on the course title

- SWE 265P – Reading code
- SWE 265P – Software understanding
- SWE 265P – Program comprehension
- SWE 265P – Making sense of source code (at scale)
- SWE 265P – Software models, visualization, and change
- SWE 265P – …

Place in the curriculum

- SWE 240P–250P
  - hands-on software development, focused topics
  - small programs
- SWE 261P
  - theory and practice of software quality
  - large-scale software
- SWE 264P
  - theory and practice of software architecture
  - large-scale software
- SWE 265P
  - theory and practice of reading code
  - large-scale software

Structure of the course

- Lecture
- Practice in class
- Learn from professionals
- Course project
  - eight parts
- Diary

Grading

- Course project (90%)
  - up to two parts of the course project can be resubmitted to improve your grade, each within five days of receiving the grade
- Diary (10%)
- Peer evaluations of teamwork
  - your contributions to the team homework
  - your enabling others to make contributions
  - *may significantly impact your letter grade*
- Attendance is mandatory
  - may influence your grade

Basic tenor of the course

- Scale
- Trial and error
- Surprises
- Weeds

Basic tenor of the course

- Hands-on
  - bring laptop to class every lecture
- Conversation
- Tenacity
- Work

Working together, working alone

- All work on your project shall be performed with your team only
- Your diary shall be updated by you alone
- Failure to do so will be considered a violation of academic integrity
  - aisc.uci.edu

Questions?

My question #1

- Why do we need to read and comprehend source code?
  - https://jamboard.google.com/d/1H4S4LBhzmSsEXGuA0734FIRP5cs3zDqFxgB5-5G5Cn4/edit?usp=sharing

(link is available in Canvas)

Answers #1

- Learning to read before learning to write
- Understanding the functionality that the software has
  - what it already does
  - what it does not do
- Understanding how the code realizes its functionality before changing it
- Compensating for documentation that might not be accurate/up-to-date
- Reviewing code (pull requests) as part of organizational processes
- Programming/debugging with a partner

Answer #1 (continued)

- Finding help on StackOverflow (or other fora) for a programming issue being faced
- Looking at how another software system might have solved a given programming problem
  - same programming language
  - other programming language
- Expanding repertoire of design/coding solutions for personal growth
- Assessing components, libraries, and services before adopting them (or not)
- Forgetting the exact details of and/or rationale for an old piece of code

Answer #1 (continued)

- Code generated by Copilot
- Code generated by ChatGPT

My question #2

- What is the largest piece of software you have worked on?
  - what did it do?
  - what specific task or tasks did you work on?

  - https://jamboard.google.com/d/1HvUVJSyBOwYjWUCdOygiMPVngieTbviDjDNqextShsk/edit?usp=sharing

(link is available in Canvas)

My question #3

- Give examples of when it was easy to read, understand, and perhaps modify a piece of code
  - why was it easy?
  - https://jamboard.google.com/d/1AcgCvMqeQZwRwNxW1q8Jk0w8NZ3jsjyNIGAs1FKebE8/edit?usp=sharing

(link is available in Canvas)

My question #4

- Give examples of when it was difficult to read, understand, and perhaps modify a piece of code
  - why was it difficult?
  - https://jamboard.google.com/d/1hFdR3OcFWDvN20a-D1bcRW475ecDd-e2sGiuK-AYIXg/edit?usp=sharing

(link is available in Canvas)

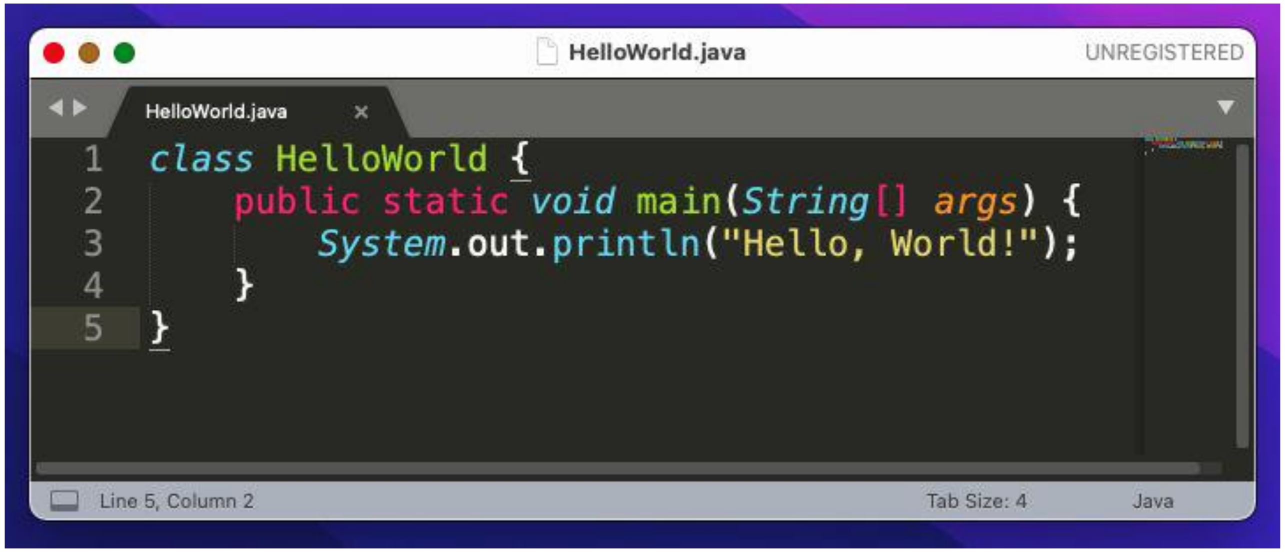
Answer #3/#4: factors affecting program readability

- Reader characteristics
- Intrinsic factors
- Representational factors
- Typographic factors
- Environmental factors

Break

Compiling source files

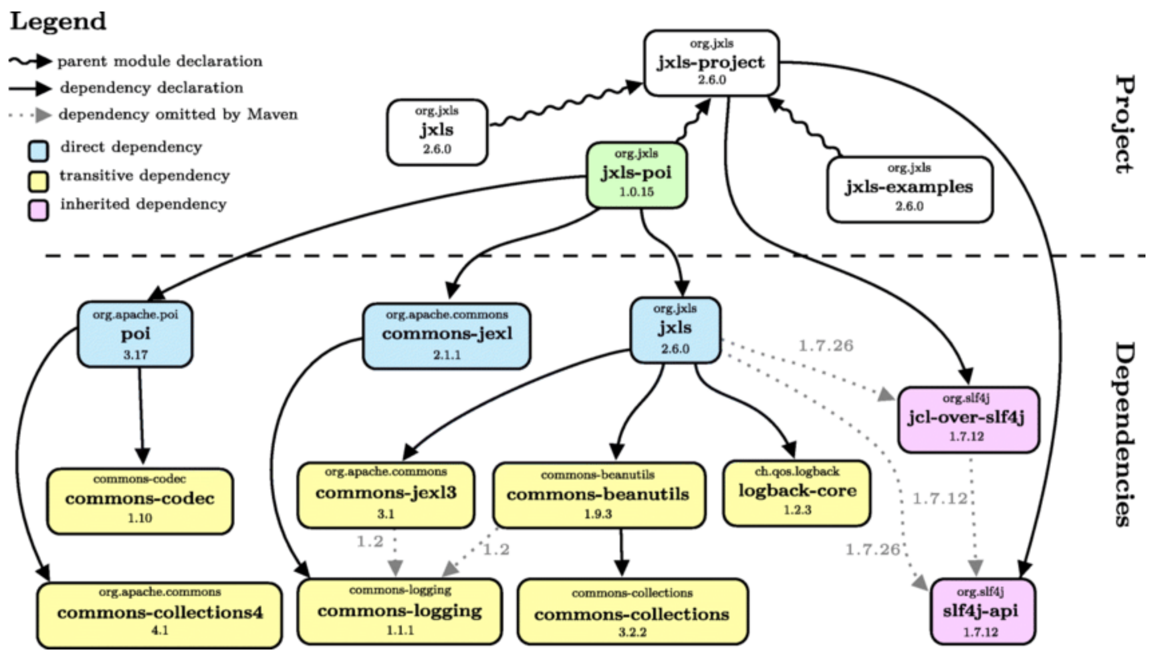
- $ javac HelloWorld.java produces HelloWorld.class
- $ java HelloWorld.class



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

Compiling software

- $ javac <<what???>>


Legend
parent module declaration
dependency declaration
dependency omitted by Maven
🟦 direct dependency
🟨 transitive dependency
🟪 inherited dependency
Project
Dependencies
org.jxls jxls-project 2.6.0
org.jxls jxls 2.6.0
org.jxls jxls-poi 1.0.15
org.jxls jxls-examples 2.6.0
org.apache.poi poi 3.17
org.apache.commons commons-jexl 2.1.1
org.jxls jxls 2.6.0
1.7.26
org.slf4j jcl-over-slf4j 1.7.12
commons-codec commons-codec 1.10
org.apache.commons commons-jexl3 3.1
commons-beanutils commons-beanutils 1.9.3
ch.qos.logback logback-core 1.2.3
1.7.12
1.2
1.2
1.7.26
org.apache.commons commons-collections4 4.1
commons-logging commons-logging 1.1.1
commons-collections commons-collections 3.2.2
org.slf4j slf4j-api 1.7.12

Building software

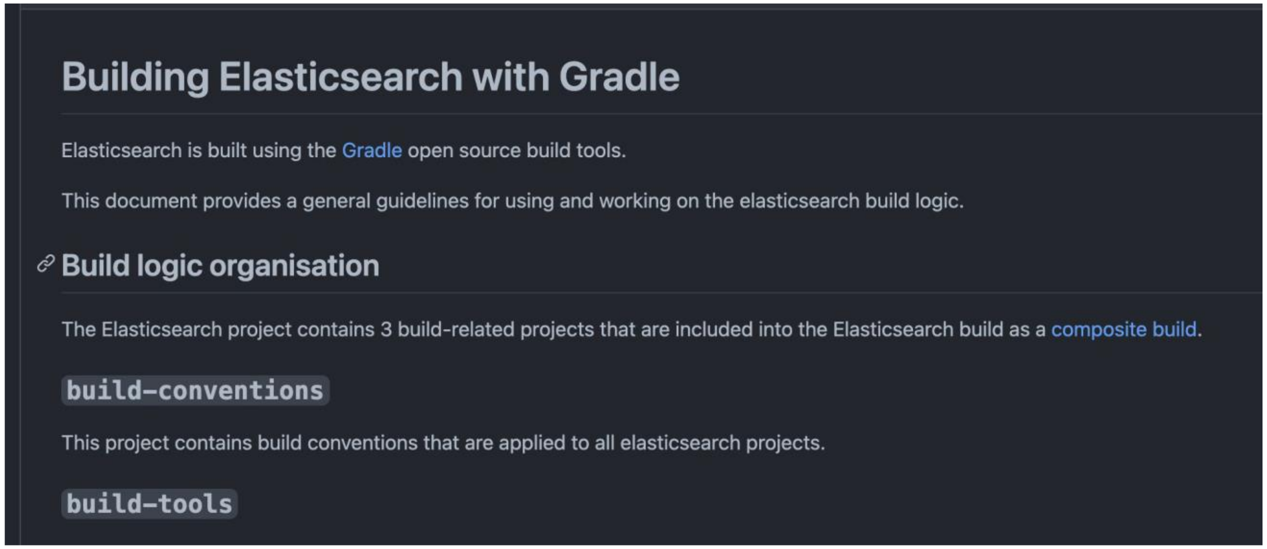
- Build automation tools use a “bill of materials”
  - pom.xml
  - build.gradle
  - ivy.xml
    build.xml

  to build an executable, and in the process automate a variety of tasks, including downloading, installing, and building output project dependencies.

Maven™ Gradle ivy

Building software

- A well-documented project will have up-to-date build instructions in a README (or some other file, usually with a .md extension)



Building Elasticsearch with Gradle

Elasticsearch is built using the Gradle open source build tools.

This document provides a general guidelines for using and working on the elasticsearch build logic.

Build logic organisation

The Elasticsearch project contains 3 build-related projects that are included into the Elasticsearch build as a composite build.

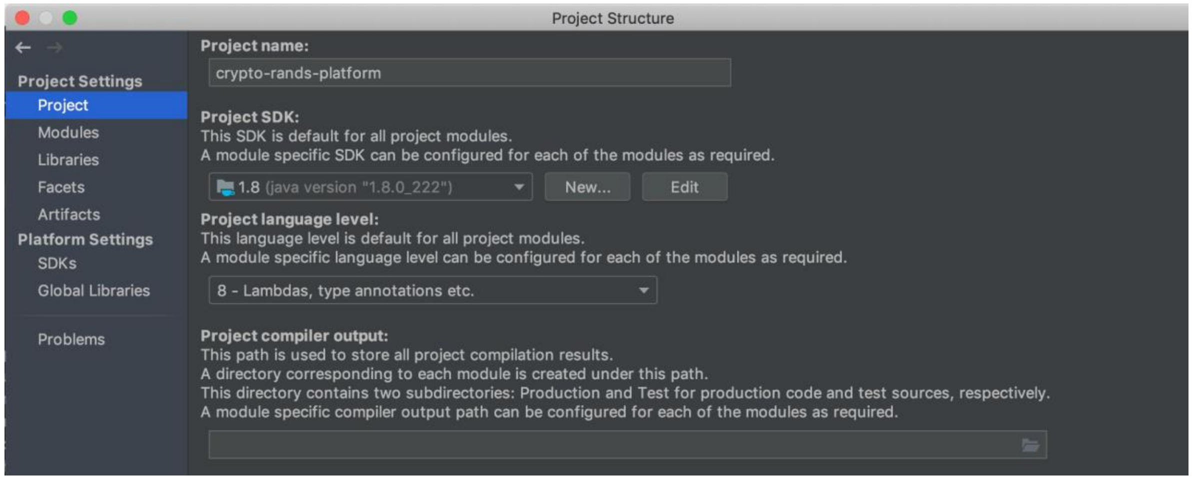
`build-conventions`

This project contains build conventions that are applied to all elasticsearch projects.

`build-tools`

Troubleshooting

- Have the right version of Java
- Make sure your Java version is up-to-date in IntelliJ
  - or make sure it is the Java version that the software requires
- Read the README instructions on GitHub again
- Set your path variable
- …



Project name:
crypto-rands-platform

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.
1.8 (java version "1.8.0_222")

Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.
8 - Lambdas, type annotations etc.

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.

Troubleshooting

- Copy and paste the exact error into Google or Stack Overflow
- Ask your team members
- Ask the TAs
- Read the community’s forum
- Ask the community (but be mindful)
- …

Diary

- Maintain an ongoing diary of class activities
  - date
  - time
  - activity
  - participants (as applicable)
  - any insights gained
    - not about the software you are studying
    - but about the nature of reverse engineering and modeling
  - mood

- Submit weekly through Canvas

Common problems with diaries

- Not doing some of the assigned activities
- Not accounting for all activities
  - self
  - with others
- Entries too short or too long
- Not reflecting on insights that you gained
- Writing what you think the instructor wants to hear
  - mood

Example insights

- I learned that the majority of open source projects are libraries. Even so, it was not too difficult to find a project.
- Gain an understanding of how to read code.
- It is an interesting project. I was always wondering how to use ssh to connect AWS EC2 instance via a mobile device.
- Everyone found some decent issues that I had not considered, based on what they were tagged as.

Example diary entries

- Code review is not just a tool to help the person getting their code reviewed, but reviewing others' code can help bring new insights to the observer as well.
- Understanding of different design pattern in OOP.
- Became more familiar with the project.
- Hard to find good issue to work on in a very active repo, especially worse if a lot of current contributors.

Homework

- Make sure you are familiar with the basics of IntelliJ IDEA
- Download a few Java open source applications, of varying sizes, build them, and run them (minimum three)
- Before 4pm Tuesday, April 9, submit your first week diary in Canvas

Homework (continued)

- Form a team of three, which will be your team for the duration of the quarter
- Be strategic
  - your team will study a single system for the entire quarter
  - what kind of system do you want to learn about to position yourself for future interviews
  - find team members with a similar interests – Slack & survey
  - find team members who want a similar challenge – Slack & survey
- Register your team in Canvas

A final word

- SWE 265P is a unique course
- SWE 265P is experimental in nature
- SWE 265P will be what you make of it
  - in engaging with the materials
  - in providing feedback on course direction and content