CSE 331
Software Design & Implementation

Kevin Zatloukal

Spring 2021

Lecture 1 – Administrivia

(Based on slides by Mike Ernst, Dan Grossman, and many others)

Motivation

How do we ensure correctness?

Best practice: use three techniques (we'll study each)

1. **Tools**
   – Type checkers, test runners, libraries, etc.
2. **Inspection**
   – Think through your code carefully
   – Have another person review your code (code review)
3. **Testing**
   – Usually >50% of the work in building software

Each removes ~2/3 of bugs. Together >97%

How do we cope with complexity?

We tackle complexity with **modularity**

- Split code into pieces that can be built independently
- Each must be documented so others can use it
- Also helps understandability and changeability

Administrivia

Who: Course staff

- **Instructor**: Kevin Zatloukal (kevinz at cs)
  - 15 years in industry, ~5th year teaching

- 17 great **TA**s
  - mix of new and veteran

- Office hours posted soon
  - (starting later this week)

Get to know us!

- We're here to help you succeed

Who: Students

- Assuming you have mastered CSE142 and CSE143
- Hoping (but not assuming) have you taken 311
  - will connect to 311 material where it arises
- Assuming you are in your second year of CS courses
  - seniors may be bored

Prerequisites

- Knowing Java is a prerequisite

Examples:

- Difference between **`int`** and **`Integer`**
- Distinction between **`x == y`** and **`x.equals(y)`**
- Aliasing: multiple references to the same object, what does assignment (**`x=y;`**) *really* mean?
- Subtyping via **`extends`** (classes) and **`implements`** (interfaces)
- Method calls: inheritance and overriding; dynamic dispatch
- Difference between compile-time and run-time type

Staying in touch

- Ed message board (link on course web page)
  - should have access already
  - best place to ask questions
- Course staff: **`cse331-staff@cs.washington.edu`**
  - For things that don’t make sense to post on message board
- Course email list: **`cse331{a,b}_sp21@u.washington.edu`**
  - students already subscribed (your UW email address)
  - Section A: infrequent, but important emails
  - Section B: frequent emails from me (one for each lecture)

Lectures

- Providing both synchronous and asynchronous versions
  - **Section A**: synchronous (live) lectures
  - **Section B**: asynchronous (recorded) lectures + live Q&A
- Okay for any of you to attend any of live lecture or Q&A
- Register for the section with the lecture type you will normally use
- If you are in the wrong section, email ugrad-advisor@cs to change
  - those using asynchronous lectures will want to be in Section B

Lectures: Section A (10:30)

Format: Live lectures via Zoom

- Will also be recorded in case you miss one
  - see the Zoom tab in Canvas

- Ask questions at any time via the chat window

- May occasionally ask you to watch part of a recorded video
  - backup plan since I often struggle to lecture in <50 minutes
  - may also do this if I want to spend lecture time demoing etc.

Lectures: Section B (2:30)

Format: pre-recorded videos + live Q&A

- Videos recorded during fall & last spring

- Total lecture time will average **more than 50 minutes**
  - required reading was reduced to compensate
    - feel free to watch at 1.25x speed
  - my bias is toward more teaching & learning, not less

Lectures: Section B (2:30)

Format: pre-recorded videos + live Q&A

- Regular lecture time used for live Q&A session
  - these will also be recorded
- Will email links *at least 24 hours before* the Q&A session
  - only sent to Section B students
- Fine to ask questions about earlier lectures
  - (e.g., if you fall behind by a lecture)

Section

- Will be focused on **helping with homework**
  - typically fall on day before a new HW is released
  - get you get you started with the work to be done
  - they should be very useful

- Live via Zoom video
  - links in Zoom app in Canvas

Homework Assignments

- Roughly 1 assignment per week
  - exception: week 3 has two assignments but one is short
- First 3 are paper assignments
  - submit these in Gradescope
  - should get an invite email today
    - let me know if you don't by tomorrow
- Remaining 8 are coding assignments
  - generally due on Thursday by 11pm
  - submit and tag your code in Gitlab
    - TAs will grade and get feedback to you

Homework Assignments

- Biggest misconception (?) about CSE 331

  "Homework was programming projects that seemed disconnected from lecture"

- If you think so, you are making them harder!
  - approaching them as CSE143 homework won't work well
  - each HW designed to teach topics from prior lectures
  - seek out the connections by before typing

- (Tip: this is also true of quizzes & exams)

Late Policy: Written Assignments

- Allowed only in special situations
  - let us know at least 30 hours beforehand
    - do not start the night before
  - will make exceptions for emergencies

Late Policy: Coding Assignments

- Same special situations as written assignments
- And also:
  - Up to **4** times this quarter you can turn in a homework assignment **one** day late
  - Not accepted for credit after that.
  - Late days are 24-hour chunks
- Why?
  - keep you on schedule (real world has deadlines)
  - get feedback to you before next deadline

Resubmission: Coding Assignments

- We will allow re-submission of coding assignments
  - first five coding assignments (HW4–HW7) only
  - allowed for 1 week after these grades are first published

- Aim of the policy is to limit the deductions for minor mistakes that end up causing a disproportionate number of test failures

- We will re-calculate the correctness score up to a maximum score of 80%
  - other scores (design, style, etc.) are not changed

Academic Integrity

- "The code you submit must be your own"
  - no copying from other students, web pages, etc.
- Read the full course policy carefully
  - ask questions if you are unsure
- Always explain in your HW any unconventional action
  - worst result then is some points lost
  - worst result otherwise is expulsion
- Violations are unfair to other students and yourself

Tests

- Will have 6 **quizzes** during the quarter *in lieu of a midterm*
  - ~30 minutes each
  - will have 24-hour period in which to take it
  - *mostly* multiple-choice questions (will mix in other types)
  - each test can be taken twice, with higher score used
  - (taken during weeks 4–9… none weeks 1–3 or week 10)

- Take-home **exam** during finals week
  - ~90 minutes
  - will have 36-hour period in which to take it
  - will aim for this to be straightforward
  - final chance to practice working on paper again

Grading

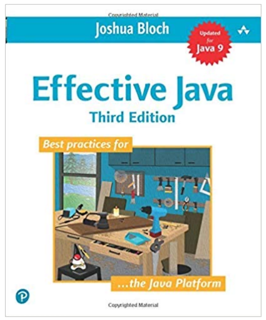
Approximate weighting (subject to change):

70%	Homework
20%	Quizzes
10%	Final Exam

Books

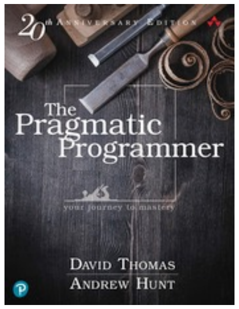
Required book

- *Effective Java* 3rd ed, Bloch (EJ)



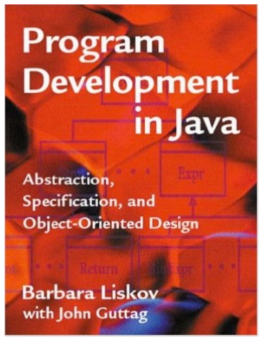
Optional book

- *Pragmatic Programmer*, new 20th anniversary (2nd) edition, Hunt & Thomas (PP)



Other books

- *Program Development in Java*, Liskov & Guttag
  - would be the textbook if not from 2001
- *Core Java* Vol I, Horstmann
  - good reference on language & libraries


Program Development in Java
Abstraction, Specification, and Object-Oriented Design
Barbara Liskov with John Guttag

Books? In the 21st century?

- Why not just use Google, Stack Overflow, Reddit, Quora, …?
- Web-search good for
  - Finding the parameters of a Java API function
- (can be) Bad for
  - Why does it work this way?
  - What is the intended use?
  - How does my issue fit into the bigger picture?

- Beware:
  - Answers on the web are often **quickly** out of date
    - aim is to answer the question at the time when asked
  - “This incantation solved my problem”
    - give that to users without knowing how it works?

Readings

- Calendar will include book sections for you to read
  – EJ = required, PP = optional

- These are “real” books about software, approachable in 331
  – occasionally slight reach: accept the challenge

- Overlap only partially with lectures
  – books include lots of other useful information

- Readings are fair game for quizzes & final
  – want to make sure you do it

CSE 331 can be challenging

- Past experience tells us CSE 331 is **hard**
  - not my intention to make it difficult!

- Big change to move
  - **from** programming by trial & error
    - technique that does not work for building large scale software
  - **to** programming by careful design, reasoning, and testing

- Programming itself can be hard
  - surprisingly difficult to specify, design, implement, test, debug, and maintain even a simple program

CSE 331 can be challenging

- We strive to create assignments that are reasonable if you apply the techniques taught in class…

  … but likely hard to do in a trial & error manner

  … and almost certainly impossible to finish if you put them off until a few days before they're due

- Assignments will take more time than you think (**start early**)
  - even professionals *routinely* underestimate by 3x
  - these assignments will be a step up in difficulty
  - aim to finish by Wednesday, not Thursday

- If you are having trouble, *think* before you act
  - then, look for help

Other Advice

- Don’t be afraid to make mistakes
  - accepting that you will make mistakes is perhaps the <u>most important</u> lesson of this course
  - we often learn best from our mistakes
  - if you’re not making mistakes, you’re not challenging yourself
    - “Never promote someone who hasn’t made some bad mistakes because, if you do, you are promoting someone who has never done anything”
      — Dr. Herbert Dow (founder of the Dow Chemical Company)

- Don’t expect everything to be spelled out for you
  - real-world problems don’t come that way
    - if there are detailed instructions for solving a problem, then there should already be a program that does it
  - world needs you for your intuition, creativity, & intelligence

HW0

An exercise before next class

- Do HW0 (90 minutes max) before lecture on Wednesday
  - practice interview question
  - **write** an algorithm to rearrange array elements as described
  - should run in O(n) time
    - (optional challenge: can you do it in a single pass?)
  - **argue** in concise, convincing English that it is correct
    - don't just explain *what the code does!*
  - do not actually run your code! (pretend it's on a whiteboard)
    - know that is correct *without* running it (a necessary skill)

- Start trying to **reason** about the code you write
  - this may be difficult... if so, remember that!
  - next, we will learn to use a set of tools that will make this easy

Before next class...

1. Familiarize yourself with website:

   http://courses.cs.washington.edu/courses/cse331/21sp/

   – skim the syllabus
   – read the academic integrity policy
   – find the homework list
   – find the link to Canvas

2. Do HW0 before lecture on Wednesday!
   – limit this to 90 minutes
   – submit a PDF on Gradescope (invite coming today)
   – graded for effort