What is the goal of CSE 331?

In short: to help you become better programmers

Specifically, to teach you how to write code of

• higher quality
• increased complexity

We will discuss tools and techniques to help with these
What is high quality?

Code is high quality when it is

1. **Correct**
   - everything else is of secondary importance
2. Easy to **change**
   - most work is making changes to existing systems
3. Easy to **understand**
   - needed for 1 & 2 above
How do we ensure correctness?

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

1. **Tools**
   - e.g., type checking compiler

2. **Inspection**
   - think through your code carefully
   - have another person review your code

3. **Testing**
   - usually >50% of the work in building software

Each removes ~2/3 of bugs. Together >97%
What is increased complexity?

Analogy to building physical objects:

- 100 well-tested LOC = a nice cabinet
- 2,500 LOC = a room with furniture
- 2,500,000 LOC = 1000 rooms ≈

North Carolina class WW2 battleship
≈

the entire British Naval fleet in WW2
Actually, software is more complex…

- Every bit of code is unique, individually designed
  - US built 10 identical Essex carriers

  - Software equivalent would be one carrier 10 times as large:

- Defects can be even more destructive
  - a defect in one room can sink the ship
  - but a defect OS could sink the whole fleet

- And more reasons we will see shortly…
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

In summary, we want our code to be:
1. correct
2. easy to change
3. easy to understand
4. easy to scale (modular)
Scale makes everything harder

Modularity makes scale **possible** but it’s still **hard**…

- Time to write N-line program grows faster than linear
  - good estimate is $O(N^{1.05})$ [Boehm, ‘81]
- Bugs grow like $\Theta(N \log N)$ [Jones, ‘12’]
  - 10% are errors are btw modules [Seaman, ‘08]
  - corner cases are more important with more users
- Comm. costs dominate schedules [Brooks, ‘75]

**Corollary**: quality must be even higher, per line, in order to achieve overall quality in a *large* program
What we will cover in CSE 331

- Everything we cover relates to the 4 goals
- We’ll use Java but the principles apply in any setting

**Correctness**
1. Tools
   - Git, Eclipse, JUnit, Javadoc, …
   - Java libraries: equality & hashing
   - Adv. Java: generics, assertions, …
   - debugging
2. Inspection
   - reasoning about code
   - specifications
3. Testing
   - test design
   - coverage

**Changeability**
- specifications, ADTs
- listeners & callbacks

**Understandability**
- specifications, ADTs
- Adv. Java: exceptions
- subtypes

**Modularity**
- module design & design patterns
- event-driven programming, MVC, GUIs
Administrivia
Course staff

- **Lecturer**
  - Kevin Zatloukal (kevinz@cs, zat@uw)

- **TAs**

  - **AA**
    - Bryan Van Draanen (bryanvd@cs) teaching
    - Waylon Huang (waylonh@cs) grading
  
  - **AB**
    - Josh Katz (katzjm@cs) teaching
    - Su Ye (yes23@cs) grading
  
  - **AC**
    - Belinda Li (lib49@cs) teaching
    - Yiyang Xu (xu517@cs) grading
  
  - Ruby Li (liz67@cs)
Office Hours

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Location</th>
<th>With</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>1:30 – 2:30pm</td>
<td>CSE 006</td>
<td>Bryan</td>
</tr>
<tr>
<td>Tuesday</td>
<td>10:30 – 11:30am</td>
<td>CSE 220</td>
<td>Josh</td>
</tr>
<tr>
<td></td>
<td>3:30 – 4:20pm</td>
<td>CSE 006</td>
<td>Belinda</td>
</tr>
<tr>
<td>Wednesday</td>
<td>2:30 – 3:30pm</td>
<td>CSE 006</td>
<td>Yiyang</td>
</tr>
<tr>
<td></td>
<td>3:30 – 4:20pm</td>
<td>CSE 007</td>
<td>Ruby</td>
</tr>
<tr>
<td>Thursday</td>
<td>2:30 – 3:30pm</td>
<td>CSE 021</td>
<td>Waylon</td>
</tr>
<tr>
<td>Friday</td>
<td>2:30 – 3:20pm</td>
<td>CSE 006</td>
<td>Su</td>
</tr>
</tbody>
</table>

- My OOs: by appointment (send me email)
  - usually available Mon & Fri
Staying in touch

- Course email list: cse331a_au17@u.washington.edu
  - for class announcements
  - students and staff already subscribed
  - fairly low traffic

- Message Board
  - for class discussion (staff will monitor and participate)
  - help each other out and stay in touch outside of class

- Course staff: cse331-staff@cs.washington.edu
  - for things that don’t make sense to post on message board
  - can also email your section TAs (earlier slide)
Prerequisites

Only prerequisite is Java knowledge
  – we assume you have mastered CSE142 and CSE143

Examples

• Difference between int and Integer
• Basic java classes like Object, String, Number, Integer, Double
• Subtyping via extends (classes) and implements (interfaces)
• Difference between compile-time and run-time type
• Object-oriented dispatch with inheritance and overriding
Lecture and section

• Both are required

• All materials posted, but they are visual aids
  – arrive punctually and pay attention
  – if doing so doesn’t save you time, one of us is messing up (!)

• Section will often be more tools- and homework-focused

• Will post other handouts related to class material on web site
  http://courses.cs.washington.edu/courses/cse331/
Homework

• Homework assignments will
  1. give you more practice
  2. require you to apply the techniques learned in class
     • Pro Tip: think about which techniques are intended

• We will have 10 homework assignments
  – first 3 are on paper, then all coding
Late Policy

• Late work will be penalized:
  – 10% for 1 day (<= 24 hours)
  – 20% for 2+ days (> 24 hours)

• Notify grader or cse331-staff if you need to use 2
  – we will normally start grading after 24 hours

• Three (3) free late days
  – for emergencies (life happens, we know that)

• Re-submission allowed for coding assignments, but...
  – only for correctness points (not style, design, etc.)
  – maximum score is 80% on correctness (since 2+ days late)
  – intended for fixing minor mistakes that saw many lost points
Academic Integrity

"The code you write must be your own."

- Read the course policy **carefully**
  - collaboration is encouraged, but…
  - do not share your HW code with others
- When in doubt, document your collaboration in your HW
  - at worst, you will lose a few points
- Cheating disrespects your colleagues and yourself
Books

**Required** textbooks

- *Effective Java* 2nd ed, Bloch (EJ)
- *Pragmatic Programmer*, Hunt & Thomas (PP)

Other useful 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
Books? In the 21st century?

• Why not just use Google, Stack Overflow, Reddit, Quora, …?

• Web articles can
  – be out of date (without any indication this is so)
    • even 2014 is like 1960 in Internet years
  – rely on context that is not apparent on that page

• Books usually give better presentation of high level ideas
  – the purpose of a language feature or library
  – key reasons for its design

• Do use the Java 8 APIs (link on web site)
Readings & Quizzes

• We will have readings from the required textbooks
  – these books are also on reserve at the library

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

• Quizzes to make sure you don’t skip the readings
  – short: 2-6 questions, usually multiple choice
  – take as many times as you want
Exams

• Midterm in class on Friday, November 3rd
  – main focus on reasoning, specifications, ADTs, & testing
  – these are the most important topics in the class

• Final in class on Friday, August 18th
  – comprehensive but first half most important
Grading

Approximate weighting (subject to change):

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>55%</td>
<td>Homework</td>
</tr>
<tr>
<td>5%</td>
<td>Reading quizzes</td>
</tr>
<tr>
<td>15%</td>
<td>Midterm exam</td>
</tr>
<tr>
<td>25%</td>
<td>Final exam</td>
</tr>
</tbody>
</table>
Acknowledgments

• Course designed/created/evolved/edited by others
  – Michael D. Ernst
  – Dan Grossman
  – David Notkin
  – Hal Perkins
  – Zach Tatlock (newcomer last quarter)
  – A couple dozen amazing TAs

• Hoping my own perspective offers benefits

• [Because you are unlikely to care, I won’t carefully attribute authorship of course materials]
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 brute-force, trial & error
  – to programming by careful design, reasoning, and testing

• 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

• Learning to program well is worth the effort
  – create solely with the power of your imagination
  – create software that positively affects the lives of many people
Questions?
You have homework!

• HW0, due in dropbox by 10:30am Friday
  – write an algorithm to rearrange array elements as described
  – argue in concise, convincing English that it is correct!
  – should run in $O(n)$ time
    • (optional challenge: can you do it in a single pass?)
  – do not actually run your code!

• Start learning to reason about the code you write
  – this is the one homework that is intentionally difficult
  – spend 2 hours max (if stuck after 90m, write up what you tried)
    • this HW grade is for participation not results
  – this will be easy in a week or so
To-Do List

Before the next class…

1. Familiarize yourself with website:
   
   http://courses.cs.washington.edu/courses/cse331/
   
   – read the syllabus (esp. the advice section)
   – read the academic integrity policy
   – find the homework list

2. Do HW0 by 10:30 am on Friday!
   – limit this to 2 hours
   – submit a PDF into the dropbox