Welcome!

- Welcome to ECE 496.03:
  - Engineering Software for Maintainability
- Quick introductions:
  - Please feel free to just call me Drew
  - ...and I’d like you all to introduce yourselves

What this class is about

- Real software has a long lifespan
  - In industry, you might work the same code base for years or decades
- Contrast with code you write in school:
  - Turn it in, forget about it.
- Real world software’s requirements evolve
  - New features
  - Changing requirements
  - ...
- How do we design software to ease later changes?
  - Goal of this class: learn this by doing and reflection

What this class is not

- This class is not about learning to program
  - I assume you already are a competent programmer
  - Must have completed at least ECE551 or CS/ECE250
    - In the later case, CS 308 is recommended
- This is not a lecture class
  - Today is the only time I will have prepared slides
  - I could talk about software design all day, but...
  - You will really only learn by doing
  - “I’m not the sage on the stage, I’m the guide on the side”
    - So many profs say that, but in this case, I mean it.

What are we doing?

- One semester long project:
  - Really Interesting Strategic Conquest
  - Requirements staged into 4 evolutions.
    - Add or change requirements from prior evolutions
  - With each evolutions, submit a writeup. Two major parts:
    - Forward looking (analysis of current design):
      - What are its key features?
      - Why did you design it this way?
      - What do you see as its strengths?
      - How about its weaknesses?
    - Retrospective (analysis of past design choices):
      - How did your past designs set you up to win or struggle?
      - How did these outcomes align with your prior analyses?

Project Groups

- You will do your project in groups of 2 or 3
  - Pick carefully: fixed for the semester
  - Recommended 3 per group
    - Groups of 2 have slightly reduced requirements
- Considerations:
  - Language choices
  - Note: subject of next Monday's discussion
  - Other tool choices
  - Revision control, ...
  - Skills and expertise
    - Ideal: strong skills, complimentary expertise
- End of class: find groups, start planning ev1
Writeups

- No specific page limit/requirement
  
  - Say what you need to say. Don't say more, don't say less.

- Highly recommend LaTeX + version control (svn or git)
  
  - You are all scientists and/or engineers: make good use of your tools
  
  - Submit pdfs only (no Word docs)

- Expect document to be
  
  - Well-written:
    
    - Organized, clear, precise.
    
    - Include figures if they help
  
  - Analytical:
    
    - Delve into why your design is/was good/bad
    
    - Tell me what was bad, and how it could have been better
    
    - Hindsight is 20/20

Class Time: Two ways

- Class discussions:
  
  - Topics posted on class webpage (all posted now)
    
    - Feel free to prepare for other topics than those listed
  
  - Prepare ~1--2 pages of outline/notes on discussion
    
    - Should take ~30 minutes per discussion
    
    - Typed up, will hand in during class
  
  - Some discussions: general topics (good design, documentation...)
  
  - Others: specific to project (after each evolution)
    
    - Compare designs: discuss your strengths + weaknesses
    
    - Contemplate upcoming evolution's requirements
    
    - No page of notes needed for these

- Workdays
  
  - Work with your group on your project
  
  - I'll circulate around, answer questions, offer advice, etc.

Grading

- 50% software deliverables:
  
  - How well did your code work?
  
  - How well was it designed?
    
    - Note: poor design will be mitigated by recognizing and analyzing it in your writeup

- 30% written deliverables:
  
  - Technical/analytical content: how well did you describe/analyze?
  
  - Writing: how well written are your documents?
    
    - More strict as semester progresses

- 20% class attendance/participation:
  
  - Come to class regularly (2 free absences).
  
  - Have your discussion notes prepared (grading: 0, 70, or 100)
  
  - Actively participate in the discussions

- No exams

Specifics of the Project

- Project: Really Interesting Strategic Conquest (RISC)
  
  - Pun on RISC (the ISA category) and RISK (the board game!)

- Turn based strategy game
  
  - Where all turns happen simultaneously.

- Networked client/server setup
  
  - Players enter their orders in their clients
  
  - "Commit" their orders for a turn
  
  - Server takes all players orders and processes them
  
  - Server sends outcomes to clients
  
  - Next turn starts

Academic Integrity

- Expect academic integrity from all of you
  
  - Duke community standard
    
    - I will not lie, cheat, or steal in my academic endeavors, nor will I accept the actions of those who do
    
    - I will conduct myself responsibly and honorably in all my activities as a Duke student.

- Concrete rules:
  
  - Discuss anything you want
    
    - Give credit where its due if you use other groups’ ideas
  
  - All code should be produced within your group
  
  - Don't share code outside your group
  
  - Can use libraries for graphics, sound, etc (e.g., SDL) as needed

- Not sure? **ASK**
Requirements (Continued)

- Requirements will be distributed as pdfs
  - New requirements in blue
  - Changed requirements have old requirements in grey
    - (replacements in blue)
  - P.S. one LaTeX source, macros to control evolutions

- Unclear on requirements? Ask
  - Happy to clarify anything

- Unspecified requirements/behavior?
  - Do anything reasonable

- Don’t need to be artistic
  - Though feel free to make it look nice if you want

Questions?

- Any questions?
  - Before I turn you loose on evolution 1...

Evolution 1: Go!

- You all have a copy of the Evolution 1 requirements
  - Time to get started!
- Find your groups
- Start talking about your design
  - Sketch out some UML?
  - Decide how to split up the work?
  - What do you think the main challenges will be?
  - How should you design to accommodate whatever changes I throw at you?
  - What programming language do you want to use?
    - Detailed discussion on Monday.
- Also, a good chance to ask for clarifications on the reqs