Guide to Slides

- Take notes of:
  - Sweep-in Text: Blanked out text.
  - Extra content not on slides:
    - Notes on board, in-class examples
    - Most notes will be on board.
- Joke:
  - If you put a million monkeys at a million keyboards, one of them will eventually
  - Texting
  - Talking (vs participating!)
  - Coming late
- If sending me an email:
  - Give me a little context (class, your name, topic, ...)
  - You should write the msg so I won't reed it.
- If sick, please email vs coming to office hours.
  (baby at home; don't want to get sick!)
Discussion

In groups of 3 to 4 people:
- Exchange email address;
- Answer the following:

1. Should you usually spend more time making your code more efficient or more readable?

2. What are some ways to make your code more maintainable?

You already known:

- Java, or C++ (and are eager to learn Java).
- 2nd Year Software Development
  - Data structures
  - Some team work
  - Some OOD
- You don’t have to be a coding guru. *(but try to become one this semester!)*
- If you don’t, please come talk to me!

Course Objective

- Goals
  - ..
  - Scrum process
  - Tools like Git, IntelliJ, etc.
  - ..

- OOD design
- design patterns
Course Components

- **Lecture**
  - Cover topics for applied software development.
  - Relevant to project work, and your career.

- **Project**
  - I make 8-person groups; tutorial attendance mandatory.
  - Individual grades for contributions to project.
  - I hope to release project code under BSD (form).
  - Code may be discussed ‘anonymously’ in class.
  - I make projects happen, not know all answers.

Readings

- Every 2 weeks; hand in written “responses”
- See web for marking/chapters. 1st due next week!
- In-class discussion of thoughts.

Exercises

- Activities throughout semester to learn tools
- Git exercise due next week!

In Class

- Quizzes (announced), discussion participation
- No exams
  - Course graded on a curve

Project Overview

- Each team given a project topic
  - Real external customer, or me as customer.
  - First tutorial is with customer to gather project requirements

- 3 iterations using Scrum (Agile)
  - in tutorial demos & customer feedback
  - in lecture retrospectives

- Each student responsible for have something “reasonable” to work on.

Peer Evaluations

- Done at end of each iteration
- Formative pseudo-anon feedback
- Some affect on grade

Tutorials

- weekly checkup on how things are going
- ask customer questions
- get new feature requests / changes
- have 25 min with customer (or me)
  - have 25 min with TA
Project Technical Points

- Code in Java with IntelliJ
  - (unless whole team and I agree otherwise)

- Must Use Git
  - Exercise 1 teaches Git use
  - Tell Git your email & name to earn marks
    $ git config --global user.name "John Doe"
    $ git config --global user.email johndoe@sfu.ca

Advice from Previous Students

- Be Assertive
  - speak-up about your ideas
  - pickup more tasks

- Be proactive
  - look for more tasks to do
  - don't leave work to end of iteration

- Learn from teammates; support teammates
  - ask for help faster;
    if stuck for 4 hours, ask your team.
  - Course is a lot of work!

Project Logistics

- Monday & Friday: Lecture
- Wednesday has 3 tutorial times: (and no lectures)
  - 1:30-2:20
  - 2:30-3:20
  - 3:30-4:20
  - Complete online survey with your availability;
    I will assign students to times and groups.

- Discuss permission for customer to use software & complete feedback form.

Keys to Success

- It's Project base! You'll get out of the course what you put into it.
- No final; but in-class quizzes and project performance will reward those who learn lecture content
- Honest effort on readings to internalize ideas