Individual Assignment & Automation Tools

Individual Assignment

- A chance to experience aspects of test driven development
  - Reading specifications in the form of unit tests
  - Writing code against clearly defined specifications
  - Different than real TDD - You get all of the tests in advance
- Get used to using GitHub (pull-requests, issues, etc.)

Individual Assignment

- Each student will
  - Get a read-only, private repo
    (We call this repo the *handout repo*).
  - Fork the handout
  - Clone the fork locally, and work on the assignment
    - Write Java code to pass JUnit tests
    - Push commits and, optionally, open issues
  - Submit their solution as a PR.
- After the deadline, we will merge your PR.

Individual Assignment

- Start early
- Make sure to check Travis CI results.
- Try to give us an insight into your work:
  - Use issues, if needed
  - Make small commits, with concise and meaningful messages
- And ... Try to have fun

Individual Assignment

- Learn how to use your IDE (e.g. Eclipse) efficiently
  - Use autocomplete, *Ctrl + space*
  - Use auto-correct, *Ctrl + 1* (*Cmd + 1* ,in OSX)
  - Use refactor tools (right click ➜ refactor)

    Ex: Rename a variable/method using *Alt+Shift+R* (*Cmd+Option+R* in OSX)
  - Pay attention to compile warnings
  - Learn to use the debugger
- General tip: Try to pick up a new convenient shortcut every week.

Individual Assignment

One more (minor) goal for this assignment - Introduce a couple of useful automation tools ...

Automating Tests

- In Eclipse, you run unit tests by clicking through some menus, but what if we need to automate?
    - Run unit tests every time someone submits (or updates) a pull-request.. aka Continuous Integration
    - Run long-running and/or resource-intensive tests during off hours
    - etc.

Automating Tests

- Can use Maven to automate JUnit test runs
- We follow some conventions
    - A specific directory structure
    - A configuration file, called pom.xml, that provides Maven with the information it needs
- Maven provides us with easy automation
    - `mvn test`
    - If we can run it in the shell (i.e. terminal) we can script (i.e. automate) it.

Automating Tests & Travis CI

In this assignment, whenever you submit (or update) a pull-request against the handout repo

1. Travis CI downloads the code from GitHub
2. Uses Maven to compile the code and run the tests
3. Reports the results back to GitHub
   (you will see them with the pull-request)

Note: It usually takes a few minutes (sometimes a bit longer) until you can see the test results.

Automating Tests & CI

- Continuous Integration is a useful tool.
    - Helps us avoid merging broken code into our repo
    - Extremely useful in open-source, where contributors may not trust each other’s code
    - Super convenient when test run on diverse OS, CPUs
    - Allows us to confidently merge code into production
- In the past, companies invested millions in server farms for CI.
    - Adobe Flash, Intel Android CI ran on 100’s of CPUs
    - now you can have it too.

Automating Tests & CI

- CI is not truly needed for your assignment , but we still wanted you to see it, because
  - It can still catch a few naive mistakes that can prevent your code from compiling.
    Ex: Forgot to add one of the files, before committing.
  - It can reveal other build-related bugs
  - You will most likely run into it at your first job
  - We think it’s cool, and the people at Travis-CI were generous to let us use their pro version for free.

More On Maven

- Maven can automate many tasks, not just running JUnit tests.
    - Your pom describes what your code depends on
        - Compile the code
        - Generate Java Docs
        - Download dependencies from the Internet
        - And many more (Maven is extensible via plugins)

How is that relevant to CSC301?

Maven in CSC301

- Our Java utility library
    - Code shared between assignments
    - Uses Maven to
        - Compile the code (into a Jar file)
        - Generate Jar files with Java Docs and source
    - Maven tasks run on JitPack's servers.
        - JitPack downloads the source from GitHub
        - Runs Maven tasks, based on pom.xml
        - Makes Jar files available on the Internet

Maven in CSC301

- Assignment code uses Maven to download the utility library from JitPack's servers
    - `pom.xml` specifies JitPack's servers
    - `pom.xml` specifies the name (and version) of our utility library
    - Maven takes care of the rest

Automation Tools

- Why are we telling you about these automation tools?
    - Chance to “show off” open-source software
        - Heads up for PEY, internships, summer jobs
    - Motivate teamwork:
        - There are always problems you haven’t faced.
        - There always technologies you don’t know about
        - Larger team = larger knowledge base