CS409
Software Testing

TAN, Shin Hwei
陈馨慧

Southern University of Science and Technology

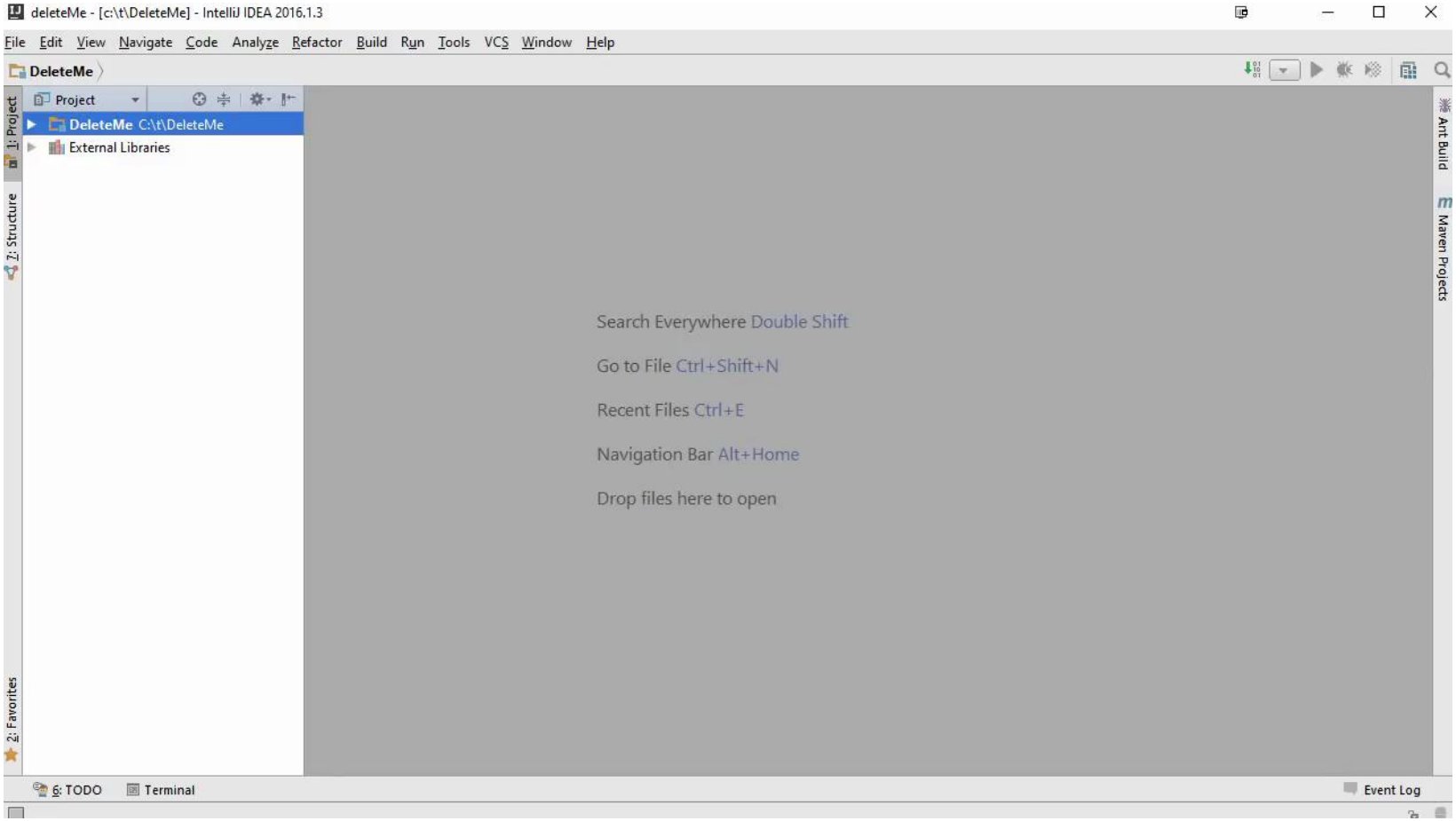
Slides adapted from cs498dm (UIUC) and cs4218 (NUS)

Sakai

- https://sakai.sustech.edu.cn/portal/site/1a46e1d5-fae1-418a-b4a2-f9b7537c7b8c
- Contains all lecture notes, lab notes and mp0 instructions
- New this semester: Integration with GitHub Classroom
  - The class roaster will be synchronize with Sakai

Effective usage of features in IDE

How to create tests in IDE fast



Shortcuts

- Hot Keys:
  - Alt-Insert: Generate code -- constructor, getter, test case
  - Alt-Enter: Quick-Fix something, such as create a test class, method, field.
  - Ctrl-Shift-F10: Set current run config.

  If done inside a test method, will run just that method. Otherwise runs whole class
  - Shift-F10: Re-run last run config.

Try it yourself

Steps:

- Create new IntelliJ Java project
- Create class under test: PetRock
- Create test folder in project, and set as 'Test' folder
  - File, Project Structure, Sources, select 'test' folder; click TESTS on top
- Create test class from existing class
  - Click class name; press Alt-Enter, Create Test
  - Configure project for JUnit 4
- Write testGetName, run tests
  - assertEquals

- Run with CTRL+Shift+F10 to set what to run
  - If in a test method, only runs that method
  - If in test class, runs full class
- Shift F10 = Re-run last "configuration"
- Ctrl + Shift + T: Toggle between test and code-under-test
- Create new test method for isHappy; run tests.
- in JUnit class, Alt+Insert, Test Method
- Alt + Enter: on non-existant methods to create.
- use assertTrue, assertFalse

How to create tests in IDE fast

Find the shortcut for your system in the links below:

- Windows:
  - https://www.jetbrains.com/help/idea/tdd-with-intellij-idea.html?keymap=primary_default_for_windows
- Mac:
  - https://www.jetbrains.com/help/idea/tdd-with-intellij-idea.html?keymap=primary_intellij_idea_classic_macos

RECAP: VERSION CONTROL WITH GIT

slides created by Ruth Anderson, images from http://git-scm.com/book/en/

http://www.cs.washington.edu/390a/

Git Resources

- At the command line: (where verb = config, add, commit, etc.)

  $ git help <verb>

  $ git <verb> --help

  $ man git-<verb>

- Free on-line book: http://git-scm.com/book
- Git tutorial: http://schacon.github.com/git/gittutorial.html
- Reference page for Git: http://gitref.org/index.html
- Git website: http://git-scm.com/

- Git for Computer Scientists (http://eagain.net/articles/git-for-computer-scientists/)

Committing files

- The first time we ask a file to be tracked, *and* **every time before we commit a file** we must add it to the staging area:

```
$ git add README.txt hello.java
```

This takes a snapshot of these files at this point in time and adds it to the staging area.

- To move staged changes into the repo we commit:

```
$ git commit –m “Fixing bug #22”
```

Note: To unstage a change on a file before you have committed it:

```
$ git reset HEAD  --  filename
```

Note: To unmodify a modified file:

```
$ git checkout  --  filename
```

Note: These commands are just acting on **<u>your local version of repo</u>**.

Status and Diff

- To view the **status** of your files in the working directory and staging area:

```
$ git status
```

or

```
$ git status -s
```

(`-s` shows a short one line version similar to svn)

- To see what is modified but unstaged:

```
$ git diff
```

- To see staged changes:

```
$ git diff --cached
```

Pulling and Pushing

Good practice:

1. **Add** and **Commit** your changes to your local repo
2. **Pull** from remote repo to get most recent changes (fix conflicts if necessary, add and commit them to your local repo)
3. **Push** your changes to the remote repo

~~To fetch the most recent updates from the remote repo into your local~~ repo, and put them into your working directory:

```
$ git pull origin master
```

To push your changes from your local repo to the remote repo:

```
$ git push origin master
```

Notes: **`origin`** = an alias for the URL you cloned from

`master` = the remote branch you are pulling from/pushing to, (the local branch you are pulling to/pushing from is your current branch)

Branching

To create a branch called experimental:

- `$ git branch experimental`

To list all branches: (* shows which one you are currently on)

- `$ git branch`

To switch to the experimental branch:

- `$ git checkout experimental`

Later on, changes between the two branches differ, to merge changes from experimental into the master:

- `$ git checkout master`
- `$ git merge experimental`

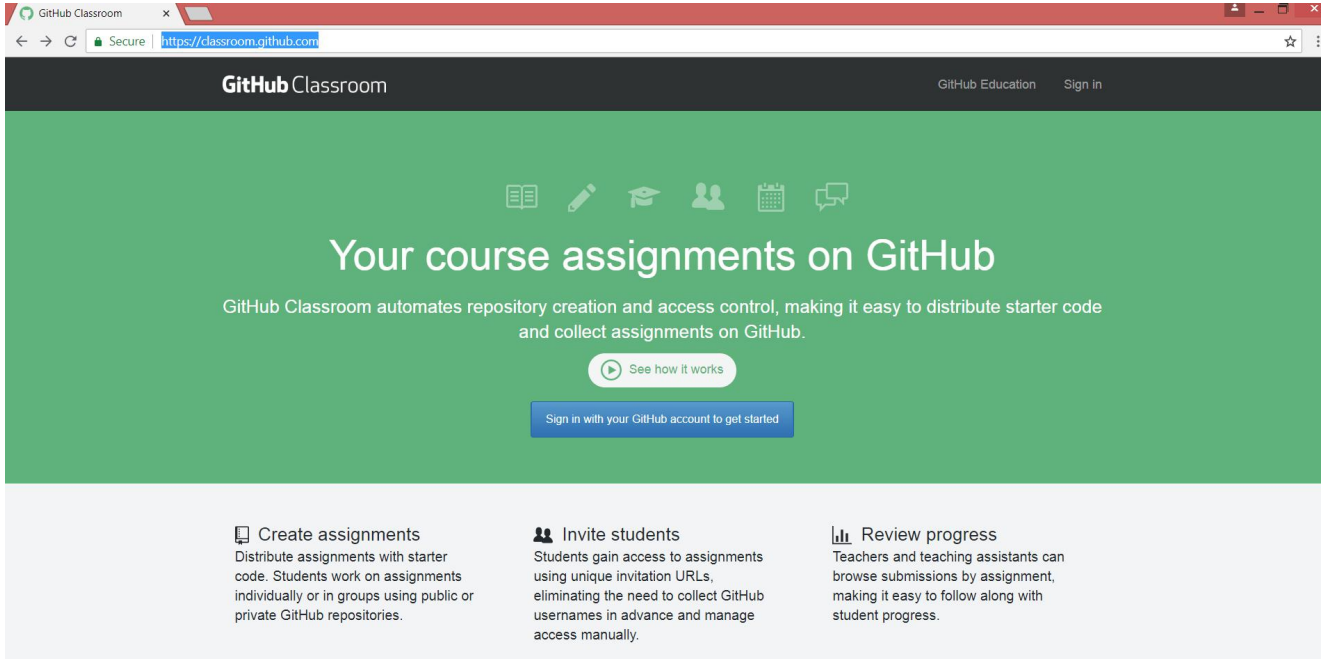
Note: **git log --graph** can be useful for showing branches.

Note: These branches are in *your local repo*!

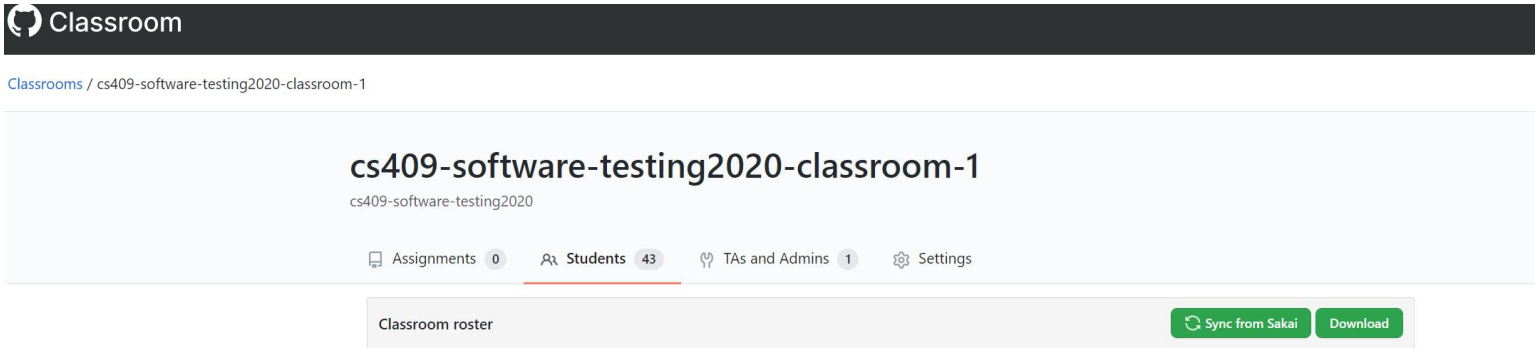
Collaboration through GitHub classroom

Main Page

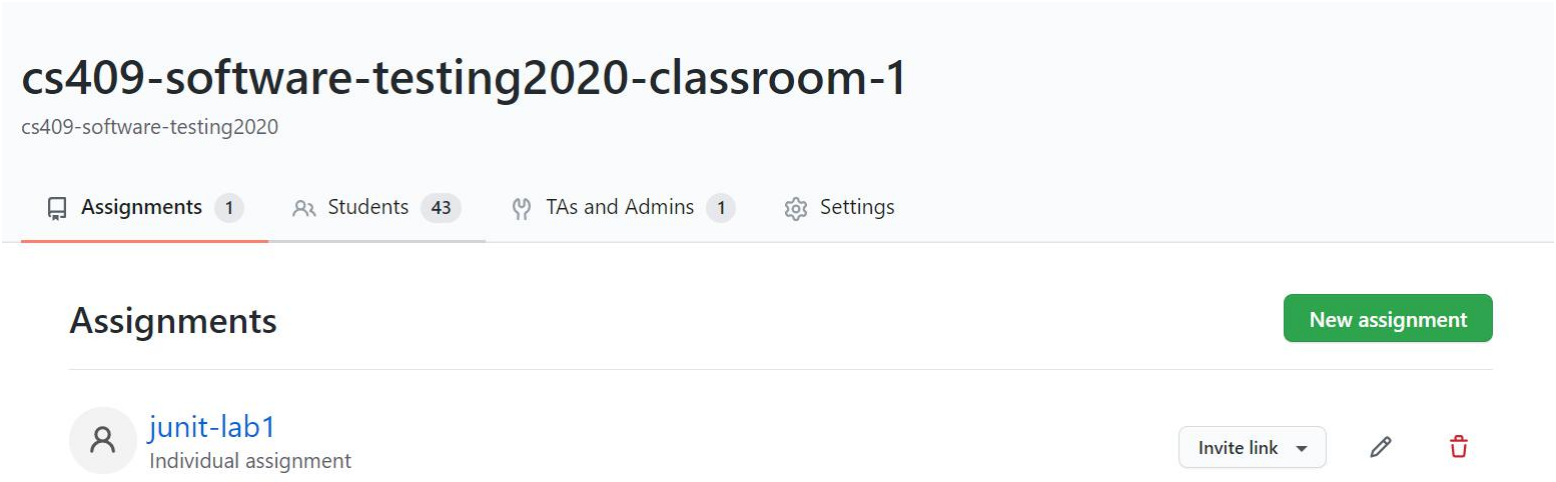
- https://classroom.github.com/



Our organization: cs409-software-testing2020-classroom-1



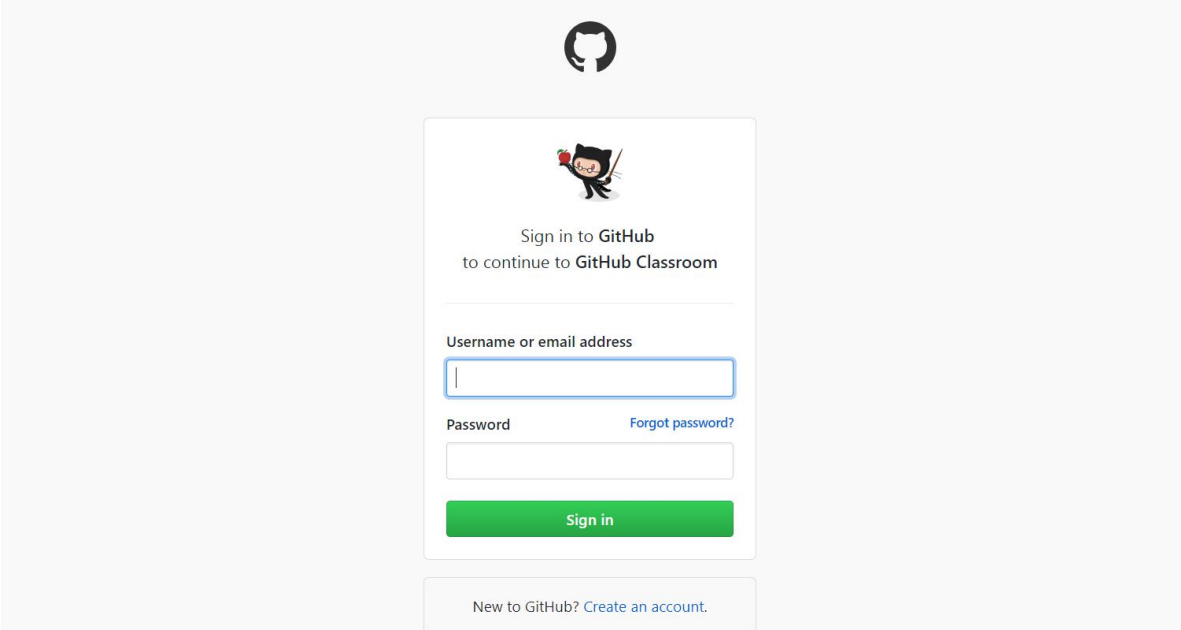
List of Assignments: More coming soon...



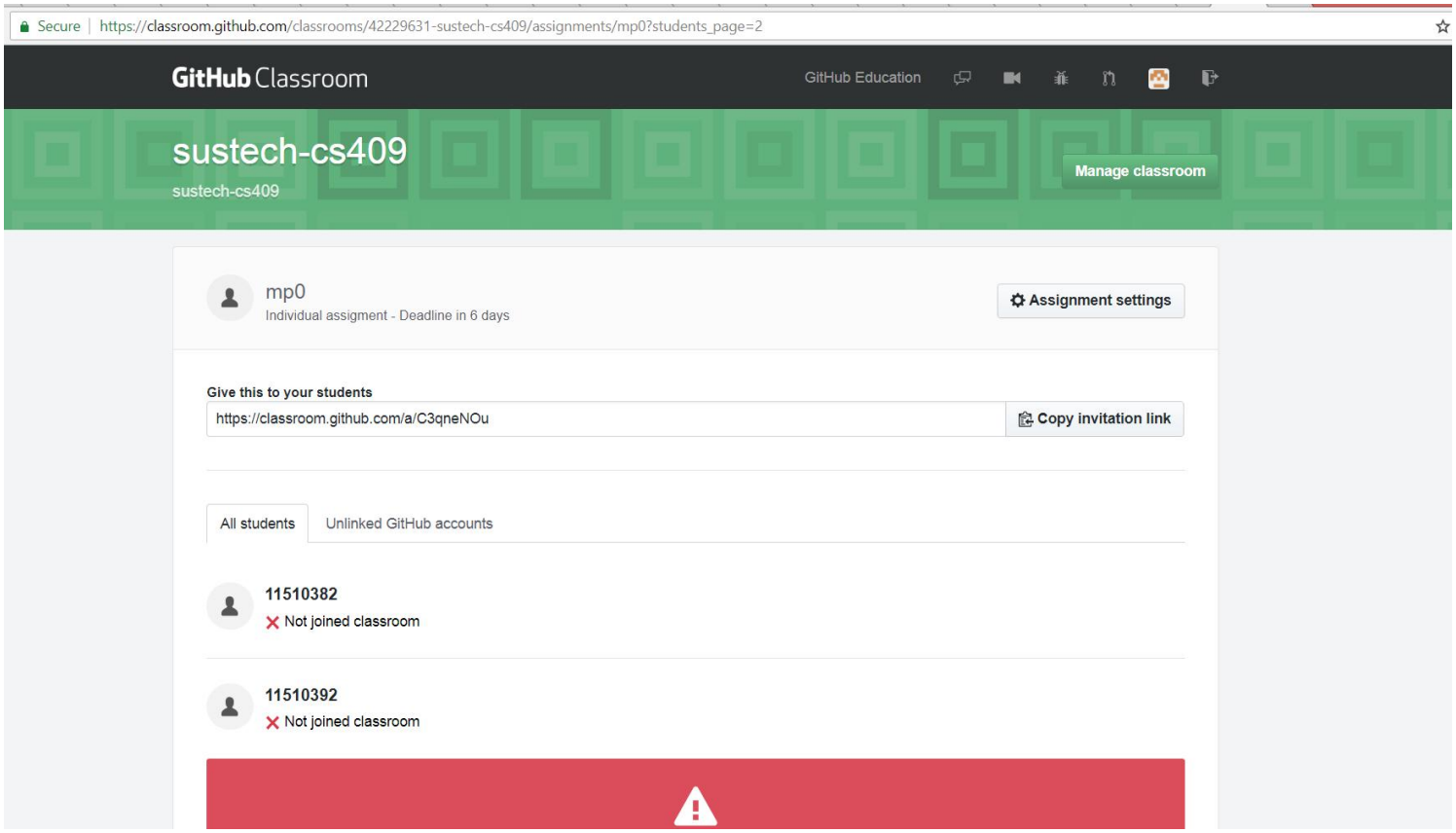
Accept Invitation for the lab1

- Go to the invitation links:
- https://classroom.github.com/a/TnI4NoVY

Step 2:
Sign in to your existing account/ Create New Account



Select Your Student ID



Please let me know if you have problem selecting your ID in this step:

- Select wrong id
- Your id doesn't exist

Write tests for the starter code

- Write at least one JUnit test for each method
- Write tests for checking:
  - All outputs (including system.out)
- Don’t forget to use the shortcut for creating tests fast in IDE
- What to submit?
  - **TestPrimitiveParameters.java file**