Programming Assignment 0 CS671

ONLY submission due 5/26/2021 at 11:59pm, tagged 0a or 0a.#

This "dummy assignment" is designed to check that all the necessary tools are in place for the first real assignment, (Assignment 1). Students should be able to:

- Pull the starting code and sample tests from the Git repository.
- Load the code into an IDE and complete the implementation.
- Run the sample tests (from the IDE and/or the command-line).
- Commit the code to Git, tag it and push it onto the GitLab server.
- Pull the grading tests from Git after the submission date.
- Pull a grade on Git after the assignment has been graded.

Code to Be Written

The assignment task is to implement method `add` in object `Easy`. The method returns the sum of its two arguments. *Hint:* the body of method `add` can be implemented as `x + y`.

Submission

Files in `0/src/main/` should be edited (e.g., using an IDE) to implement the assignment. Then, using Git, code should be added to the `master` branch of the repository:

```
% git status
On branch master
...
% git add 0/src/main
% git commit -m 'Dummy assignment implementation.'
% git tag -a -m 'Dummy assignment submission.' 0a
% git push --tags origin master
```

The submission must be tagged with `0a`. That is a zero, not the letter O. If changes become necessary after submission (but before the deadline), new tags of the form `0a.1`, `0a.2`, . . . can be used. *Do not delete existing tags!*

Of course, multiple changes can be done to local files before an *add* and *commit*, and multiple commits can be created before a *push*. Intermediate commits don't need to be tagged; only submissions. It is recommended to commit and push often for backup purposes.

For this assignment, there is no requirement to upload a code sample on Canvas.

Notes

- You can use the GitLab web interface to check that all necessary commits and tags have been uploaded to the server. You can also use it to add tags to existing commits.
- When an assignment is tagged with a submission tag, GitLab will try to compile the sample tests. If this compilation fails, you will receive an email.
- In order to avoid conflicts and merging issues, you should refrain from modifying grading code (any code in a `grading` directory). If needed, make a copy in your own area and modify the copy.

- All the programming assignments (as well as other code) live in the same Git repository. If you use IntelliJ IDEA, however, you need to create separate IDEA projects, one for each assignment. Basically, every directory that contains an `sbt` configuration file (`build.sbt`) should be its own IDEA project (i.e., `0`, `1`, `ClassCode`, etc.). *Do not open the root of the Git repository with IntelliJ*.
- IDEA is Git-aware and can then be used to create commits and pull/push code. Make sure that the *Push Tags* checkbox is checked when pushing to the GitLab server so submission tags are included.
- Sample tests can be run directly from IntelliJ IDEA (e.g., by right-clicking the test class name). They can also be run from the `sbt` prompt:

```
> testOnly SampleTests
```

  This should produce an output of the form:

```
[info] Loading settings for project ...
[info] ...
[info] SampleTests:
[info] - sample
[info] Run completed in 1 second, 687 milliseconds.
[info] Total number of tests run: 1
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 1, failed 0, canceled 0, ignored 0,
pending 0
[info] All tests passed.
[success] Total time: 3 s, completed ...
```

  If "`testOnly SampleTests -- -oD`" is used, the duration of each test is displayed.
- To get an `sbt` prompt, you can use the `sbt shell` window of IntelliJ. You can also install the `sbt` command-line tool on your development machine (it should already be available on the department's servers):

```
% cd 0
% sbt
...
> testOnly SampleTests -- -oD
```