CS108 HW Logistics

Thanks to Nick Parlante for much of this handout

Homeworks

- The homeworks look like systems -- made of many parts
- Do not start them the night before
- We grade the homeworks mostly by their computing correct output, and we look at your source code / OOP style a little. You should aim to write reasonably clean looking code. If code has excessively bad style, we will take points off. More importantly, you should write clean code just because it's the best way to author something that is satisfying and works correctly.
- **You should not change public interfaces** -- often we have some grading code that goes through the public interfaces, methods, etc., and that will break if you start changing methods around. You are free to add methods etc. that do not break the public interfaces. Very often my solution has the main method and then a decomposed out private helper method to tame the complexity.
- The starter code will often include the prototypes for methods you need to write and sometimes other boilerplate code. A starter method may include a throwaway line like `return 0;` just so it compiles when you first load the project.
- For testing, your code should return the correct output when fed valid inputs and called in correct ways. We will not worry about what happens when your code is fed invalid data unless the assignment specifically says you must handle bad input.
- You code can assume that all inputs are formatted and structured correctly. Your solution should work correctly when presented with valid inputs, and we will not worry about invalid inputs (yet). If your code takes in a parameter such as a `String` or a `List`, you may assume that the `String` or `List` passed in is not `null`, unless `null` is specifically mentioned in the spec. In other words, `null` is not a valid `String` or `List`. The empty string and empty list are valid of course, but in Java those are different from `null`.
- Your code should never change the public interfaces given for homework problems. Very often, we have tests that call your code in various ways, and obviously that only works if your code keeps the exact interface as given in the starter code. For the same reason, you should leave your classes in the default package, so our testing code can find it. You are free to add additional or helper methods -- adding extra methods will not confuse our testing code.

Office Hours

- Office hours are listed on the course page -- both the regular hours and the special hours added for each assignment. Office hours are rather empty early on when each assignment is handed in, so early is a good time to get individual coaching.
- Send questions to cs108@cs -- medium sized questions work fine over email. If it will require stepping through code to explain, it will probably work better in-person.
- Take a look at the FAQs on the course page for each hw -- we update them with common issues.

Submit

- Comment out your little `println` lines and other extraneous output before turning anything in.
- Your homework project is in the form of an Eclipse project directory containing your .java files, .class files, etc.. To submit, copy your project directory for that homework, containing all your files (and its .dotfiles) to a leland machine, such as cardinal.stanford.edu, with a secure file copy protocol, such as scp. If you don't have a program for transferring files using scp we recommend securefx. You can use

java from the command line to compile/run your project there if you like, although in general, testing your code on your own machine is sufficient. But just know we'll be grading on the Leland machines, so it's expected of you that it works on there.

- To submit, create a `README` file as described below and log in to a leland machine, `cd` into your project directory, and run the script `/usr/class/cs108/bin/submit` script to copy your project up into the grading space. You should run `cs108/bin/cleanup` first to delete the `.class` and other files the submit script will not accept. The submit script copies your whole homework directory, including sub-directories.
- Here are the instructions for running submit (which are also in the file `108/bin/README-submit`)

```
To submit a directory of your files, first create a README file in the
directory (the file name should be exactly "README").
Near the top of that file put a line like the following that
gives your username and real name, with the name in (last, first) form...

user: jsmith05  (Smith, Jane)

Ideally your name will match up with the way the registrar has it when
we file grades at the end of the quarter. We only need your name for
the first assignment. For later assignments, we only need the
username.

For a team project, just include a line like the above for each person
on the team (the real name may be omitted), like this..
user: jsmith
user: kjones
user: chrisyay

On the later lines you can put any notes for the grader.

user: jsmith80
I honestly can't tell you how much I enjoyed this assignment.
And it made me miss the season finale of Friends.
And I didn't do part b, so sue me.

I you get the "submit success" message at the end, it worked,
and otherwise it did not.

Before submitting, remove the .class, ~file, and other leftover files
from your directory. The submit script will complain about them if
they are there, and it also does not like files larger than 500k
(if needed, you can go ahead and submit with those files anyway.)
Just run the script 108/bin/cleanup from your project directory -- it
calls the rmclass and rmtilde scripts that delete .class and ~ files
from the directory (you can, of course, use rm manually, but the
scripts have the advantage that they will not accidentally delete your
sources).

Aside: how to clean up in Eclipse: in the project menu, temporarily
turn off the Build Automatically option. Select, the Clean command,
with the Start A Build Immediately option off. This will
remove all the .class files from your project directory.

To submit, just cd to your homework directory and
run the script  /usr/class/cs108/bin/submit

The script will verify that your name is in the README, check the files,
and then copy all the files and nested directories up into submit space.
You may submit any number of times -- we grade the last one submitted and
ignore the others.


For any problems or questions with the submit script,
please email the CS108 staff.
```