CSI/CEN 213: [H0] Hello World+

Sun, Jan 29

1 Overview

This week you'll write a program in java that prints "Hello World" to the screen. It shouldn't take any input or do anything fancy. Your programming skills are almost certainly past this. This is your opportunity to make sure you have Java running properly on your computer so you don't run into problems with more substantive assignments later in the term. You'll also make sure you have github up and running, which will make larger programming assignments easier later in the term.

2 Details

2.1 Get Java

If you aren't using one of UAlbany's information commons computers, install java if you haven't already done so. Here are some options:

- The UAlbany computers have Dr. Java on them so it's what we'll use in lecture. You can get Dr. Java for free here: `http://www.drjava.org/`
- Some people prefer Eclipse. It's a bit more complicated to set up, but it does more things for you. You can get it here: `https://www.eclipse.org/downloads/`.

2.2 Get Git

If you haven't already done so, get your Student Developer Pack from github. There are lots of components; all we care about for the purpose of this class is the unlimited private repositories. This will let you have a different private repository for each assignment. Find the Student Developer Pack here:
`https://education.github.com/pack`

Read a bit about git:
`http://www.makeuseof.com/tag/git-version-control-youre-developer/`

In this class, we won't be using the collaborative component of git. The version control aspect of git is useful on it's own. Plus learning it will mean that when you do become part of a team using git, you'll have mastered a huge component of it.

Every time you save a version of your code (which you should do relatively frequently—several times an hour or so), you'll need to write a *commit message*. Here's an article on writing good commit messages:
`http://chris.beams.io/posts/git-commit/`
The author is writing in the context of large-collaborative projects and some examples have full paragraphs of explanation. Don't feel like you have to write more than a subject line and a sentence or two of explanation in this course. Sometimes the subject line on it's own will be sufficiently explanatory.

Some people are comfortable using the command line; others will prefer to use a gui. I'll be using the SourceTree gui in class because it works on both Windows and Mac.
`https://www.sourcetreeapp.com/`

2.3 Do Stuff

Write a hello world program in Java; use git for version control.

- ☐ Create a **private** github repository for this assignment.
- ☐ Git expects projects to be contained in folders, so you'll make a different folder for each program in this class. Create a folder for this activity and give it a name formatted like this: *LastnameFirstnameHomework*. If I did this assigment, my folder would be named MagnusCristynH0.
- ☐ Start your program. While it's relatively empty, make an initial commit to your private github repository.
- ☐ Write a program in Java that prints "Hello World." to the screen. It shouldn't do anything else.
- ☐ Make another commit.
- ☐ Push your commits to github.
- ☐ Take a screenshot of your commit history and put it in a PDF file.
  Find your repository on github. You'll see a list of your file(s). Above it you'll see something that says "2 commits" (or more, if you've made more commits). If you click on that, you'll see your commit history.

  The easiest way to make the screenshot a pdf is to simply drag the image into a word processor such as Microsoft Word or Google Docs then print/export/saveAs that file to a PDF. Images not wrapped in a PDF file will not display on blackboard and will not be graded.

Note: You will be expected to use git for version control for all future activities in this class. This is the only activity where it the expectations will be described in precise detail. For larger programs you will be expected to show evidence of more than 2 commits. Think towards an initial commit then commit every time you add a new feature and test that it works.

3 Screencast

In addition to submitting your java code, you will also submit a screencast in which you show your code and demo it. This lets you show us that your code works on your system. If your code does not compile, you will receive a 0. You do not need to talk for your screencast; you are just using it to demonstrate your code.

Use jing to create your screencast. Download jing here: `http://www.techsmith.com/jing.html`

In your screencast, you should do the following:

- slowly scroll through the file (don't talk over this part; just scroll slowly enough for someone to read)
- compile your program to demonstrate that it compiles
- run your program to demonstrate that it works.
- If necessary, expand your test window if there is a lot of output. If there is more output than fits on screen, scroll slowly through this output as well.

There is a demo screencast you can use as a model here: `http://screencast.com/t/KKnT8ibrUkkZ`.

When you are done recording your screencast, choose the option to share it at `http://screencast.com`. This will put the url in your clipboard so you can paste it into your blackboard submission. Do not try to download the screencast and submit that file; just submit the url.

Submit

- ☐ In the write submission field, provide a link to your screencast at http://screencast.com.
- ☐ Attach your .java files. If there is just one, you don't need to compress it. If there are multiple files as part of the assignment, compress the project homework for the assignment using .zip. Do not use .rar or some other form of compression.
- ☐ Attach the PDF containing your screenshots.