

# Starting Out with Hello!

Due Tuesday, January 24 at 8 a.m.

Revision 0

CSE 1325 - Spring 2023 - Homework #1 - 1

## Assignment Overview

Way back in 1972, Brian Kernighan wrote the *very first program* to ever print "hello, world" to the console, as part of the Basic Combined Programming Language (BCPL) documentation. From BCPL, he derived a language called B. From B, he and Dennis Richie derived a new systems language called C, used to write a new operating system called Unix (Hi, Mac fans!). From C, James Gosling created Java, with which we'll soon be very familiar!

And the first program in *each* of those languages also printed "hello, world".

So now that you're learning Java, tradition clearly dictates that we say hello to the world! We already showed you versions of this program in Lecture 01 for several languages - C, C++, Java, and Python. **Compiling and running this program is a great way to ensure your environment is operating correctly, and to start exploring its features.** You'll find a *lot* of additional hints after the instructions on this page.

## Full Credit

Set up your Java development environment using guidance from Lecture 00. Create a GitHub account, and post its name as your solution to Assignment P01 in Canvas. Create and clone a GitHub repository **named cse1325** (LOWER CASE, no trailing punctuation - this matters!), then create a P01/full\_credit subdirectory (all lower case, with pee ZERO one not pee OH one, and with an underscore NOT a dash). In it, create and run the Java "Hello, World!" program in file **Hello.java** using System.out, **replacing "World" with your name**. Take a screenshot of your program running, and **name it Hello.png**.

Add Hello.java and Hello.png to git ( `git add Hello.java Hello.png` ), commit them with an informative commit message (e.g., `git commit -m 'P01 full credit'` ), and then push that commit to GitHub ( `git push` ).

That was easy, wasn't it?

## Bonus

Instead of hard-coding your name, *ask the user for their name*. Then print "Hello, [name]", where [name] is the name they enter. Add a bonus subdirectory to P01 ( `cse1325/P01/bonus` ) with your source file named Hello.java and one or more screenshots named Hello1.png, Hello2.png, etc. showing your program in action with *at least* 3 distinct inputs, e.g., multiple word names, Unicode characters, and a zero-length string.

Add ( `git add Hello*` ), commit ( `git commit -m 'P01 bonus'` ), and push all files to GitHub ( `git push` ).

## Extreme Bonus

Instead of asking the user for their name, **automatically identify the current user** and print "Hello, [name]!" without any input or hard-coded names in the source code, where "[name]" is something that uniquely identifies the user, such as a user name, proper name, etc. Add an "extreme\_bonus" subdirectory to P01 ( cse1325/P01/extreme\_bonus ) with your source file named Hello.java and one or more screenshots named Hello1.png, Hello2.png, etc. showing your program in action. For maximum credit, test the program *in at least two user accounts* and *on at least two operating systems*.

Add, commit, and push all files. Additional information that you may find helpful follows.

## Hint: The Required File Hierarchy

When finished, your file structure under cse1325/P01 should look like this:

Permissions	Size	User	Date	Modified	Name
drwxr-xr-x	-	ricegf	08-12	10:12	.
drwxr-xr-x	-	ricegf	08-12	10:13	bonus
.rw-rw-r--	287	ricegf	08-12	10:11	Hello.java
.rw-rw-r--	132k	ricegf	08-12	10:11	Hello1.png
.rw-rw-r--	136k	ricegf	08-12	10:11	Hello2.png
drwxr-xr-x	-	ricegf	08-12	10:12	extreme_bonus
.rw-rw-r--	174	ricegf	08-12	10:11	Hello.java
.rw-rw-r--	74k	ricegf	08-12	10:11	Hello1.png
.rw-rw-r--	115k	ricegf	08-12	10:11	Hello2.png
drwxr-xr-x	-	ricegf	08-12	10:12	full_credit
.rw-rw-r--	131	ricegf	08-12	10:11	Hello.java
.rw-rw-r--	64k	ricegf	08-12	10:11	Hello.png

## Hint: How to Take a Screenshot

I'm a Linux user, so I can't personally vouch for the Windows or Mac instructions. If these don't work, then may I recommend your favorite Internet search engine for assistance?

Area to Capture	Linux bash CLI†	Linux GUI	Windows 10	Mac OS X
Entire Screen	scrot hello.png	PrtScn	Win+PrtScn	Shift+Cmd+3
Current Window	scrot -s hello.png	Alt+PrtScn	Alt+PrtScn	Shift+Cmd+4+space
Select Region	scrot -s hello.png	Shift+PrtScn	Win+Shift+PrtScn	Shift+Cmd+4
File Location	where specified	via dialog	click preview	on the desktop

† To open a terminal, press Cntl-Alt-t or Super-"term". To install scrot, type 'sudo apt install scrot'.

You may use any screenshot utility you prefer, of course, as long as your screenshots are delivered in PNG format. **Please make your screenshots and file sizes as small as possible.**

## Hint: Working With GitHub

Professional software developers use version control, so we will, too. Don't panic – it's not as hard as you may think.

The instructions below assume you will be using the bash command line. I *strongly* recommend that you learn this first, as it will help you to understand what's really going on with git. Later you can use a slick GUI if you must.

### For the first homework only

1. As we demonstrated in the first lecture, and as discussed in the slides and in "GitHub in 5 Pages" (*did we document this enough?*), create an account at [github.com](https://github.com).
2. Using this account, create a **private** repository named `cse1325` (capitalization is important - *not* `CSE1325`, and *not* `cse_1325`!).
3. Add the professor and TAs as collaborators. See Canvas > Announcements for our GitHub usernames.
4. **Submit the URL as your solution in Canvas.** If you don't do this, we can't grade your work and you'll get a 0!
5. Clone your `cse1325` repository to your computer.

This effort should make all future assignments very simple to deliver for grading.

### Full Credit

1. In bash, change to your cloned repository. Something like `cd cse1325` may work, depending on where you cloned your repository. Or find your repository in the file manager (next to Start), right-click and select "Open in Terminal".
2. Type `mkdir -p P01/full_credit` to create and change to the directory for your full credit solution. (See Bash in 5 Pages for our custom `.bash_aliases` if you like.)
3. Create your solution, for example, `e Hello.java` to create and edit the source file. Use your own editor name if needed, for example `code Hello.java` for VS Code or `gedit Hello.java` for Gedit.
4. To compile `Hello.java`, use `javac Hello.java`. `javac` stands for "Java Compiler". If you see errors, fix them in the text editor. If not, when you type `ls` you should see a class file named `Hello.class`.
5. Run the program using `java Hello`. Note that the runtime is called `java` (logically enough) and you do NOT specify the extension. If you just type `java H` and press the Tab key, bash will correctly complete the program name without the `.class`.
6. Take the required screenshot, for example by typing `scrot -s Hello.png` and clicking the terminal window or pressing `Alt-PrtScn`.
7. Add the source file and screenshot to git using `git add Hello.java Hello.png`.
8. Commit your added files to the local repository using `git commit -m "P01 Full Credit"` (or whatever comment you like).
9. Upload your commit to GitHub using `git push`.

Your screenshot should look something like this:

```

ricegf@antares:~/dev/202108/P01/full_credit$ javac Hello.java
ricegf@antares:~/dev/202108/P01/full_credit$ java Hello
Hello, Professor Rice!
ricegf@antares:~/dev/202108/P01/full_credit$ scrot -s Hello.png

```

At this point, your full credit work is done! If you'd like to try the bonus (and you should), keep going!

### Bonus

1. Create and change to the bonus directory using `mkdir ../bonus ; cd ../bonus.`
2. Repeat steps 3-10 above to develop, add, commit, and push your bonus solution.

Your screenshot should look something like this:

```

ricegf@antares:~/dev/202108/P01/bonus$ javac Hello.java
ricegf@antares:~/dev/202108/P01/bonus$ java Hello
What is your name? George
Hello, George!
ricegf@antares:~/dev/202108/P01/bonus$ java Hello
What is your name? Professor Rice
Hello, Professor Rice!
ricegf@antares:~/dev/202108/P01/bonus$ java Hello
What is your name? 王李
Hello, 王李!
ricegf@antares:~/dev/202108/P01/bonus$ java Hello
What is your name? عَبدُ الله
Hello, عَبدُ الله!
ricegf@antares:~/dev/202108/P01/bonus$ java Hello
What is your name? əɹɪʝ ɹɒssəʝ ɔɹd
Hello, əɹɪʝ ɹɒssəʝ ɔɹd!
ricegf@antares:~/dev/202108/P01/bonus$ java Hello
What is your name? 🏠 🤖 🌾 🌑
Hello, 🏠 🤖 🌾 🌑 !
ricegf@antares:~/dev/202108/P01/bonus$ scrot -s Hello1.png

```

At this point, your bonus work is done! If you'd like to try the extreme bonus (and you should), *keep going!*

### Extreme Bonus

1. Create and change to the bonus directory using `mkdir ../extreme_bonus ; cd ../extreme_bonus.`
2. Repeat steps 3-10 above to develop, add, commit, and push your bonus solution.

Your screenshot should look something like this:

```

ricegf@antares:~/dev/202108/P01/extreme_bonus$ javac Hello.java
ricegf@antares:~/dev/202108/P01/extreme_bonus$ java Hello
Hello, ricegf!
ricegf@antares:~/dev/202108/P01/extreme_bonus$ # Or to be fancier...
ricegf@antares:~/dev/202108/P01/extreme_bonus$ javac HelloAlt.java
ricegf@antares:~/dev/202108/P01/extreme_bonus$ java HelloAlt
Hello, George F. Rice!
ricegf@antares:~/dev/202108/P01/extreme_bonus$ scrot -s Hello1.png

```

You're done!

## More on This (and Future) Homework

Don't worry, it gets MUCH more interesting (and challenging). This homework is primarily designed to help you acclimate to the new environment.

All homework must be turned in via GitHub by the deadline. **Late submissions are never graded** without an exceptionally good excuse, but the grader will grade your latest submission prior to the deadline unless you specify otherwise.

**Start and finish early.** It's a great habit to develop for life.

You may push as many times as you like, and we will only grade your latest commit on GitHub that is prior to the deadline. (We won't go snooping around your earlier commits looking for something to criticize - first, we don't have time, and second, you're *expected* to make beginner mistakes when you're a beginner!) Don't accept a 0 after completing 90% of the work! **Commit and push early and often.**

It is possible – indeed, rather easy – to earn more than a 100 on homework assignments. Going beyond the Full Credit requirements will help you to grow your skill, and also result in bonus points. If you have more than a 100 average on your homework at midterm or the end of the class, the additional credit WILL be included in calculating your final average. This is a GREAT way to recover from the occasional missed exam question.

Please do not wait until the end of the semester to seek "extra credit work" to pass the class or improve your GPA. **Extra credit work is right here! Start today and avoid the rush.**

## The Suggested Solution

Don't get *too* excited - you won't get the suggested solution until *after* the due date and time!

The professor for this class provides example code, homework resources, and (after the due date) suggested solutions via his cse1325-prof GitHub repository.

- If you haven't already, `cd` and clone the professor's cse1325-prof repository with `git clone https://github.com/prof-rice/cse1325-prof.git` to create a new directory named "cse1325-prof". This will reflect the GitHub repository contents.
- The cse1325-prof directory doesn't automatically update. To update it with the professor's latest code at any time, change to the cse1325-prof directory (`cd ~/cse1325-prof`) and pull (`git pull`). `~` (the tilde) means "my home directory" in bash.
- If the pull fails, just start over by deleting the existing copy (`cd ; rm -fr cse1325-prof`) and cloning again (`git clone https://github.com/prof-rice/cse1325-prof.git`).

If you have questions, *ask*. I'm in my office (if campus is open) during office hours, and respond quickly (I am told) to Outlook email. That's why they pay me the Big Bucks™. :-D