

# CSE 1325: Object-Oriented Programming

## Lecture 00

# Introduction

(These slides are at Canvas > Modules > Lecture 00)

**Mr. George F. Rice**

[george.rice@uta.edu](mailto:george.rice@uta.edu)

**Office Hours:**

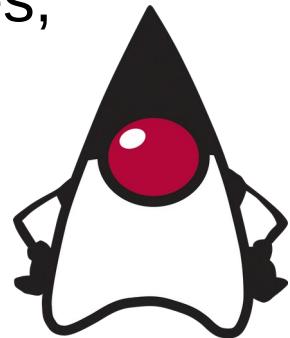
**Prof Rice 12:30 Tuesday and  
Thursday in ERB 336  
TAs [see this web page](#)**

Some thought  
required.

Shortest distance between two jokes = A straight line.

# Today's Topics

- Overview of CSE 1325
  - What you need (textbooks, tools, and lots of time)
  - How we grade (pop quizzes, post-lecture quizzes, homework, exams, and maybe benchmarks)
- Java introduction
  - Brief history and context
  - How to compile and run a Java program
- Version control
  - How not to become a case study in my future lectures
- Backups (DO them – or else!)
- Final thoughts



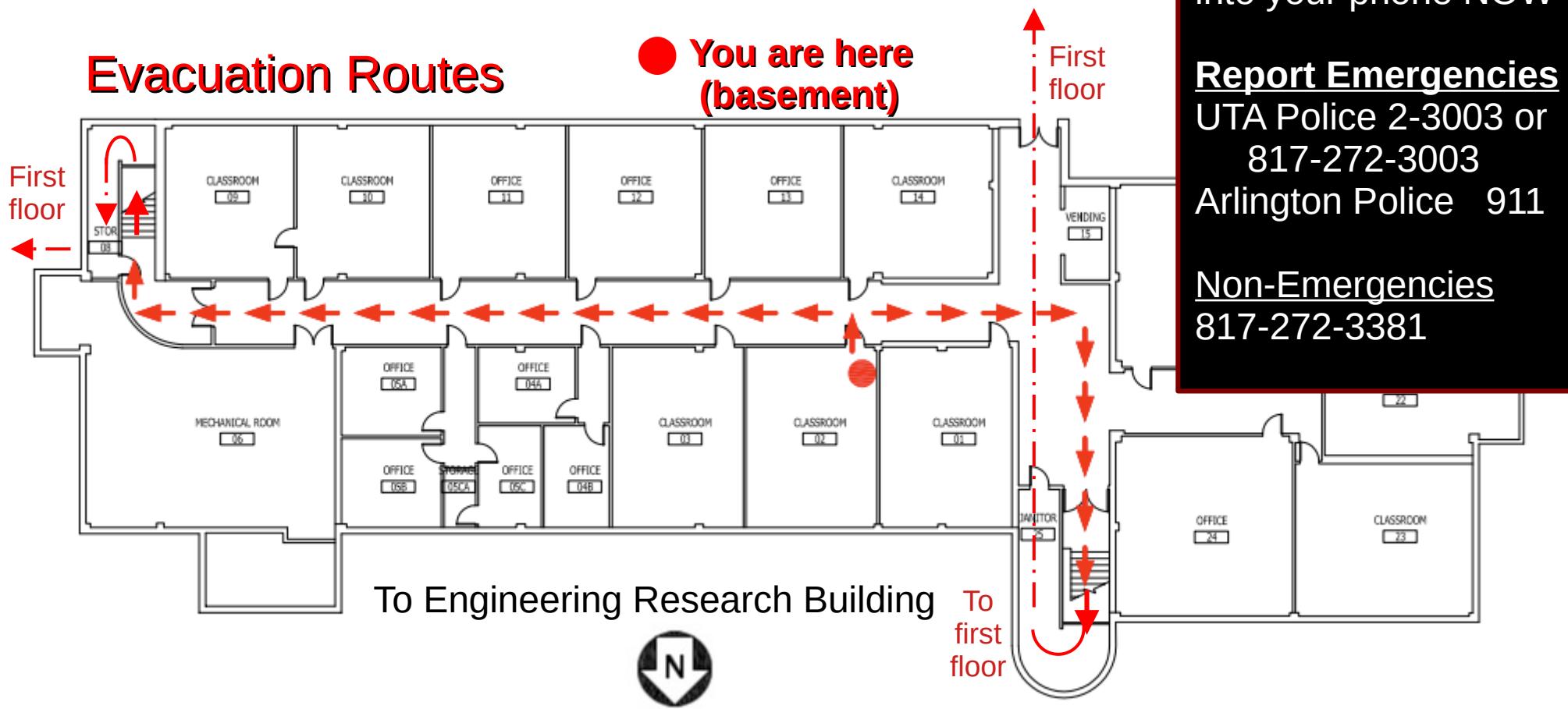
Hi! Pleased to meet you!

# Emergency Planning

## Trimble Hall (TH 02 in the basement)

Subscribe to MavAlert <https://mavalert.uta.edu/>

### Evacuation Routes





# About Me

- 45+ years of computer experience with NASA, Waterways Experiment Station, and General Dynamics / Lockheed Martin
  - Retired at the end of 2015 and began teaching part-time at UTA
- Worked with mainframes, minicomputers, and especially microcomputers
  - Designed early computer architectures, OS, and compilers
  - Programmed early CP/M, Atari, DOS 2.0+, and other 8-bit machines
  - Macintosh OS 1.0+, Windows 1.0+, OS/2 2.0+, and Unix System 7
  - Windows NT 1.0+, Linux 2.0+ including Red Hat 6.0+, Mandrake 7.0+, Ubuntu 4.10+, and SUSE Linux Enterprise Real-Time (SLERT) 10.2+
- Designed & programmed hard real-time embedded chips / motherboards
  - Aircraft test equipment and trainers
  - Data acquisition and processing equipment
  - IT architecture and applications for world's largest contract

*My real office!*



# CSE 1325: Expanding Your Programming Toolkit

- From procedural to object-oriented paradigm
  - Polymorphism + Inheritance + Encapsulation (PIE)
- Adding *two* new languages, Java and C++
  - And a *little* graphic object-oriented notation using the Unified Modeling Language (UML)
- Broadening your knowledge to areas relevant to Science, Technology, Engineering, and Math
  - Respecting “intellectual property”, command line interfaces, debuggers, sprints, version control...

# Class Schedule

Subject to Change

Class Date	Lecture	Topic
Tue, Jan 14	0	Syllabus & Intro to Java
Thu, Jan 16	1	From C to Java
	2	(Video) Java Fundamentals
Tue, Jan 21	3	<b>Encapsulation:</b> Intro to Classes
Thu, Jan 23	4	Class Members
Tue, Jan 28	5	Error Handling, Testing, and Debugging
Thu, Jan 30	6	Class Relationships
Tue, Feb 4	7	<b>Inheritance</b> & Custom Exceptions
Thu, Feb 6	8	Interfaces, Package-Private, & JavaDoc
Tue, Feb 11	9	Writing an Object-Oriented Java CLI Application
Thu, Feb 13	Exam #1	
Tue, Feb 18	10	Review Exam and User Interfaces
Thu, Feb 20	11	File I/O
Tue, Feb 25	12	<b>Polymorphism</b>
Thu, Feb 27	13	Generic Programming (Generics)
Tue, Mar 4	14	Concurrency (Threads)
Thu, Mar 6	15	Java Class Library
Tue, Mar 11		Spring Break
Thu, Mar 13		Spring Break
Tue, Mar 18	16	Exam Review
Thu, Mar 20	Exam #2	Drop date is Mar 28 by 4 pm
Tue, Mar 25	17	Review Exam and Intro to C++
Thu, Mar 27	18	From Java to C++
Tue, Apr 1	19	Object-Oriented C++
Thu, Apr 3	20	C++ Class Members
Tue, Apr 8	21	Operator Overloading
Thu, Apr 10	22	Polymorphism in C++
Tue, Apr 15	23	File and String Streams in C++
Thu, Apr 17	24	C++ Standard Template Library
Tue, Apr 22	25	Exam Review
Thu, Apr 24	Exam #3	
Tue, Apr 29	26	Review Exam and Final Thoughts



# Recommended Resources

(You are responsible for providing these!)

## Texts

- JDK 21 Documentation  
<https://docs.oracle.com/en/java/javase/21/>
  - Learning the Java Language  
<https://docs.oracle.com/javase/tutorial/java/TOC.html>
  - Java Tutorial by Jakob Jenkov  
<http://tutorials.jenkov.com/java/index.html>
- C++ Reference  
<http://cplusplus.com> and  
<https://en.cppreference.com/w/>
  - Learn C++  
<https://learncpp.com>
- Stack Overflow  
<http://stackoverflow.com/>

All resources are free!

## Software

- OpenJDK 21+<sup>1,2</sup>
- Ant version 1.10.14<sup>2</sup>
- Gnu Compiler Collection 11+<sup>3</sup>
- git version 2.45.2<sup>2</sup> with GitHub
- Canvas
- Your choice of editor  
(I'll use mostly gedit in class)
- Meld and cloc are optional

<sup>1</sup> OpenJDK (recommended) or Java SE is fine

<sup>2</sup> Any later version will *probably* work.

<sup>3</sup> Any version supporting C++ 20 (except modules) should work.

# May I Use Stack Overflow, CoPilot, and Other AI Coders?

- **Danger, Will Robinson!**

- Copying is easy, but understanding the code is REQUIRED
  - Using code “because AI wrote it!” is a Cargo Cult anti-pattern to avoid

- What could possibly go wrong?

- **You won't have Stack Overflow or AI on the exam!**
    - Many students last year averaged >100 on assignments but 50 on exams
  - Using another person, AI, or Internet sources to solve your homework assignments for you is *cheating*
    - **We always report those who violate the UTA Honor Code**
  - The code may include subtle bugs or side effects
  - The code may violate copyright or miss key requirements



# These Types of AI Questions are OK

CSE1325 Repository Overview and Resour... + ⋮ | ☰ ×

 prof-rice 1 minute ago

What does this repository do?

 Copilot 1 minute ago

3 references ▾

The repository `prof-rice/cse1325-prof` is designed to provide resources for the CSE1325 object-oriented programming classes. It includes:

- **Lecture Resources:** Code and materials associated with lectures, including example code, slides, and supplemental material.
- **Code Examples:** Directories for code shown in slides and during class, with instructions to build and run them using tools like `make` and `ant`.
- **Environment Testing:** Sample programs to verify that students can build and run Java and C++ command line applications, with detailed instructions for each test.

For more detailed information, you can refer to the [repository](#) and the specific README files within the subdirectories like [00-syllabus/README.md](#).

 prof-rice 1 minute ago

How would I output text to the terminal without a newline in Java?

 Copilot 1 minute ago

In Java, you can output text to the terminal without a newline by using `System.out.print()` instead of `System.out.println()`. Here is an example:

● Java

```
System.out.print("This will be on the same line.");
System.out.print(" So will this.");
```

This will print the text on the same line.



Ask Copilot



**But TYPE the suggested code into your editor yourself.  
Learn. Fight. WIN!**

# Options for Writing Java and C++

## Lean and mean (recommended)

- In terminal: **nano** →
- GUI: **Gedit** / **GTE** / **Notepad++** / **Sublime Text**

## More power! But beware of sharp corners

- In terminal: **vi** / **vim** (also **a game**)  
**emacs** (nice OS, needs a good editor – old joke)
- GUI: **VS Code** (or **VSCodium**) with **Extension Pack for Java**  
and **Extension Pack for C/C++** (See next slide and *VS Code in 5 Pages*)  
**zed** (successor to Atom) very fast but complex and **still in beta test**



A screenshot of the GTE (Gnome Text Editor) window. The title bar says "nano 2336 - PythonUser.java". The code in the editor is:

```
// NOTE: This class assumes that the String  
// is formatted correctly.  
public class PhoneNumber(String number){  
    boolean valid=true;  
  
    // This method checks to see if the area code is a Number  
    private void check1(){  
        int areaCode=Integer.parseInt(number.substring(0,3));  
        if ((areaCode==212) || (areaCode==217) || (areaCode==201))  
            valid=false;  
    }  
  
    // This method checks to see if the first number of the prefix  
    // is a valid number (2-9)  
    private void check2(){  
        int first=Integer.parseInt(number.substring(4,5));  
    }  
}
```

The status bar at the bottom shows various menu items: Aside, Write, Line, Fish, Page, Print, Copy, Paste, Quitter, Justifier, Dictionnaire, Page, suiv., Collier, Orthograph.

## Editors to AVOID in my opinion

- Limited / educational editors: **BlueJ**, **KWrite**
- Deprecated editors: **Atom**, **Brackets**, **Lime**
- Minimal Java support: **Code::Blocks**, **Kate**, **KDevelop**
- Complex IDEs: **Eclipse**, **NetBeans**, **IntelliJ**, **XCode**

Warning

→ Generating code  
with any IDE or AI  
is cheating ←

(GTE is Gnome Text Editor, a rewrite of gedit.  
Screenshot by xmodulo per the CC BY 2.0 license)

# What About VS Code?

- Pros
  - Very nice “Intellisense” auto-complete
  - Really *really* wants to help you
  - Default environment in [GitHub Codespaces](#), if you’d like a VM
- Cons
  - **You won’t have “Intellisense” on the exam**
  - It sometimes helps *too* much, adding invalid package statements and warning you about missing code that we just haven’t covered yet
- **I recommend a simpler editor unless you use Codespaces**
  - If you use VS Code, *be careful*
    - **Do NOT create a project** – use it as an editor only
    - **Do NOT install a git extension** – manage your git repo from bash / zsh only
    - Understand we can’t help you much with VS Code – you’re on your own

# Options for Running All Tools

As Java is highly portable and C++ reasonably so, you may

- Run on your laptop's local OS (Linux, Mac, Windows) or a local virtual machine (VirtualBox, VMware, Parallels) or GitHub Codespaces
- Run in Windows Subsystem for Linux (Ubuntu) or Chromebook Linux (Crostini)
- Run on Omega or other remote machine (but you'll have little control over tool versions!)

**YOU** are responsible for providing a suitable programming environment for this class.

>> You need it NOW! <<

To enjoy all tools pre-installed, or if you used it in CSE 1320,

- Use **CSE-VM** (but the Linux version here has become dated because VirtualBox is not compatible with M1 / M2 Macs<sup>†</sup>)

## Options to AVOID

- Run Java in Android Studio (very different)

<sup>†</sup> I'd like to make it compatible! Will you help?

# Installing Tools under Ubuntu\*

(if you installed your own VM or are running natively)

All CSE 1325 tools are available in the 24.04 repository via the Advanced Packaging Tool (apt)

- Press Ctrl-Alt-t
- Install using commands:
  - “sudo” means “do as the **super user**”, i.e., as the administrator.  
Your password<sup>†</sup> is required.
  - If you want the custom aliases, bash functions, and scripts I use in CSE-VM, download them from:  
<https://github.com/prof-rice/cse-vm>

```
$ #Do NOT type the $ - that's a prompt!
$ sudo apt update
$ sudo apt install openjdk-21-jdk
$ sudo apt install build-essential
$ sudo apt install ant
$ sudo apt install git-all
$ sudo apt install gedit
$ sudo apt install meld
$ sudo apt install cloc optional
```

Colors above group packages by command

\* And derivatives such as Xubuntu, Mint, and Pop!\_OS  
Gedit, meld, and cloc are NOT required, but you may find them useful.

† The CSE-VM default user name and password are both “student”.

# Installing Tools under Windows and Mac

- OpenJDK 21 (the Java compiler)

<https://docs.microsoft.com/en-us/java/openjdk/install>  
(For Mac, scroll down to “Install on macOS”)

IMPORTANT: I have no way  
to test Mac instructions

- Gnu Complier Collection (the C++ compiler)

<https://phoenixnap.com/kb/install-gcc-windows> <https://sourceware.org/cygwin/> ( Win)  
( Mac\* – I recommend “Way 2”)

<https://www.geeksforgeeks.org/different-ways-to-setting-up-environment-for-cpp-programming-in-mac/>

- Ant (Java build system) <https://ant.apache.org/manual/install.html>

- Git (version control system)

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

---

Gedit (optional – light code editor) <https://wiki.gnome.org/Apps/Gedit>

Meld (optional – text file comparisons) <https://meldmerge.org/>

CLOC (optional – count lines of code)

<https://github.com/AlDanial/cloc> (Win) <https://formulae.brew.sh/formula/cloc> (Mac)

# Installing Java under GitHub Codespaces

- To launch a new Codespace on the GitHub website, select Code > Codespaces > Create codespace on main
- Install the Java compiler

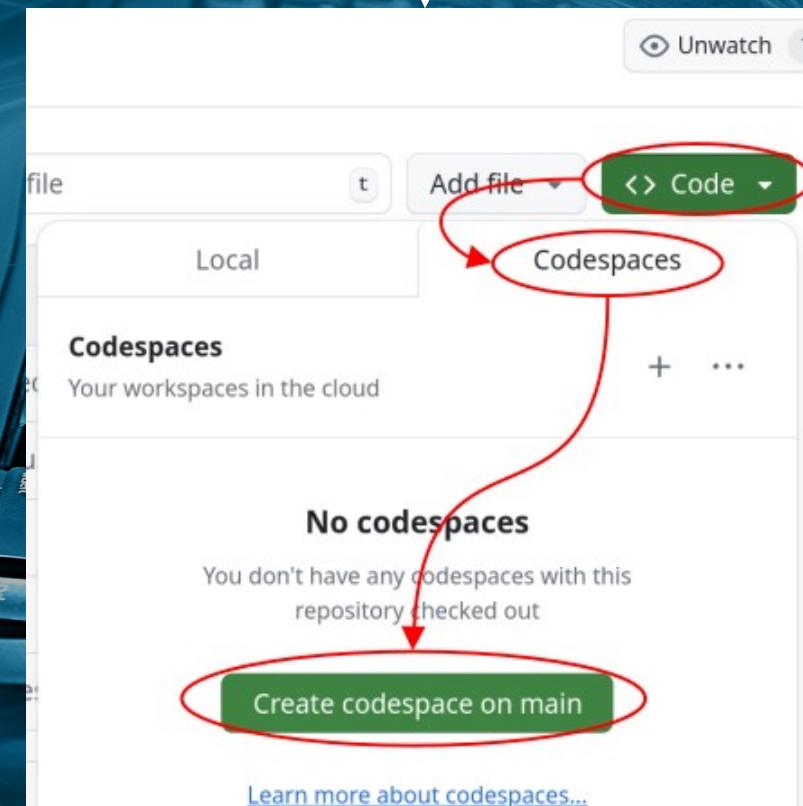
```
@prof-rice ~ /workspaces/cse1325-cpp (main) $ sdk list java
Microsoft      | >>> | 21.0.5       | ms      | installed | 21.0.5-ms
                  |        | 17.0.13     | ms      | installed | 17.0.13-ms
@prof-rice ~ /workspaces/cse1325-cpp (main) $ sdk default java 21.0.5-ms
setting java 21.0.5-ms as the default version for all shells.
@prof-rice ~ /workspaces/cse1325-cpp (main) $ javac -version
javac 21.0.5
@prof-rice ~ /workspaces/cse1325-cpp (main) $
```

## • Install Ant

```
@prof-rice ~ /workspaces/cse1325-cpp (main) $ sdk install ant 1.10.13
Downloading: ant 1.10.13
In progress...
#####
#=-=#
#####
Installing: ant 1.10.13
Done installing!

Setting ant 1.10.13 as default.
@prof-rice ~ /workspaces/cse1325-cpp (main) $
```

Git, of course,  
is pre-installed!



# Install the gcc C++ Compiler under GitHub Codespaces

- Select Ctrl-Shift-P or Cmd-Shift-P and run  
**Codespaces: Add Dev Container Configuration Files...**
  - Select the C++ container configuration (you may need to ask for all configurations first), then Ubuntu 24.04, and run
- After it finishes, edit file `.devcontainer/Dockerfile` and append:

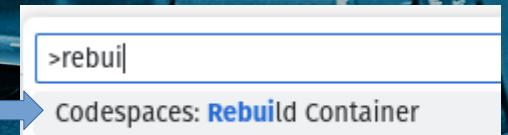
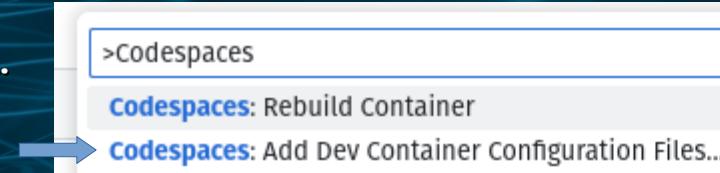
```
RUN apt-get update && export DEBIAN_FRONTEND=noninteractive \
  && apt-get -y install --no-install-recommends software-properties-common \
  && add-apt-repository ppa:ubuntu-toolchain-r/test \
  && apt-get update \
  && apt-get -y install --no-install-recommends gcc-12 g++-12 \
  && update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-12 100 \
  && update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-12 100
```

- Select Ctrl-Shift-P or Cmd-Shift-P, type "rebuild", then select  
**Codespaces: Rebuild Container**

- This will take awhile.

- Once the CodeSpace has restarted, run `gcc --version` on the command line
  - Should be version 12

- Installing the C++ Extension will get you version 9 – too old for C++ 20 . :(



```
@prof-rice ~ /workspaces/cse1325 (main) $ gcc --version
gcc (Ubuntu 12.3.0-17ubuntu1) 12.3.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

# Testing your Java Environment

- TestJavaArgs.java
  - Access program args
- TestJavaFileIO.java
  - Read text file

```
ricegf@antares:~/dev/202301/test_environment$ javac TestJavaArgs.java
ricegf@antares:~/dev/202301/test_environment$ java TestJavaArgs
Your args (command line parameters) were:
ricegf@antares:~/dev/202301/test_environment$ java TestJavaArgs hi you world
Your args (command line parameters) were:
  hi
  you
  world
ricegf@antares:~/dev/202301/test_environment$
```



```
ricegf@antares:~/dev/202301/test_environment$ javac TestJavaFileIO.java
ricegf@antares:~/dev/202301/test_environment$ java TestJavaFileIO
Here's the contents of my source file (TestJavaFileIO.java):

// These are kinda like C's #include
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
```

**It is important that you *fully* test your environment *this week!***  
Don't wait for a tight assignment deadline to discover you can't build and run your code.

# Testing your Java Environment

- TestJavaIO.java
  - Console input / output

```
ricegf@antares:~/dev/202301/test_environment$ javac TestJavaIO.java
ricegf@antares:~/dev/202301/test_environment$ java TestJavaIO
What grade would you like in CSE1325? m
Sorry, we have no M grade!
What grade would you like in CSE1325? Whatever
Sorry, we have no W grade!
What grade would you like in CSE1325? A
Here's hoping for your A!
ricegf@antares:~/dev/202301/test_environment$ █
```



Note: Source code from the lectures is always provided to you at <https://github.com/prof-rice/cse1325-prof.git>!

# Testing your C++ Environment

- test\_cpp\_io.cpp
  - Console input / output
- test\_cpp\_file\_io.cpp
  - Read text file

```
ricegf@antares:~/dev/cse1325-prof/00/test_your_environment$ g++ -std=c++17 test_cpp_io.cpp
ricegf@antares:~/dev/cse1325-prof/00/test_your_environment$ ./a.out
What grade would you like in CSE1325? m
Sorry, we have no m grade!
What grade would you like in CSE1325? Whatever
Sorry, we have no Whatever grade!
What grade would you like in CSE1325? a
Here's hoping for your a!
ricegf@antares:~/dev/cse1325-prof/00/test_your_environment$
```

**IMPORTANT:** The name of your C++ compiler and the executable file it produces may vary!

```
ricegf@antares:~/dev/cse1325-prof/00/test_your_environment$ g++ -std=c++17 test_cpp_file_io.cpp
ricegf@antares:~/dev/cse1325-prof/00/test_your_environment$ ./a.out
Here's the contents of my source file (test_cpp_file_io.cpp):

// C++ include files work much like C, except we no longer use .h for system libraries
// these are similar in function to C's stdio.h
#include <iostream>
#include <fstream>
```



Note: Source code from the lectures is always provided to you at <https://github.com/prof-rice/cse1325-prof.git>!

# Canvas

- Canvas (augmented by GitHub) is our Learning Management System

The screenshot shows the Canvas LMS homepage for course 2202-CSE-1325-002. The left sidebar contains links for Account, Dashboard, Courses, Calendar, Inbox, Commons, Studio, Help, and various administrative sections like Grades, Syllabus, Modules, Assignments, Echo360, Quizzes, People, UTA Libraries, Course Evaluations, Accessibility Report, Upload Syllabus, Collaborations, Discussions, Conferences, Files, Pages, Outcomes, and Settings. The main content area displays the course title "CSE 1325: Object-Oriented Programming in C++" and sections "002 (TR 8:00 am) and 003 (TR 9:30 am) in NH109". Below this is a banner for "The University of Texas at Arlington" with the tagline "Your University. Your Future." The bottom navigation bar includes links for Instructor, Syllabus, Modules, Announcements, Getting Started, and Q&A Discussion. A sidebar on the right shows course status (Unpublished), import options, and coming-up events.

Note: Canvas will likely report you are in section 002 regardless of your actual section.  
MyMav lists your correct section of record!

# Email

- I *strongly* prefer Outlook to Canvas or Teams
- Important: ALL actionable issues must be emailed!

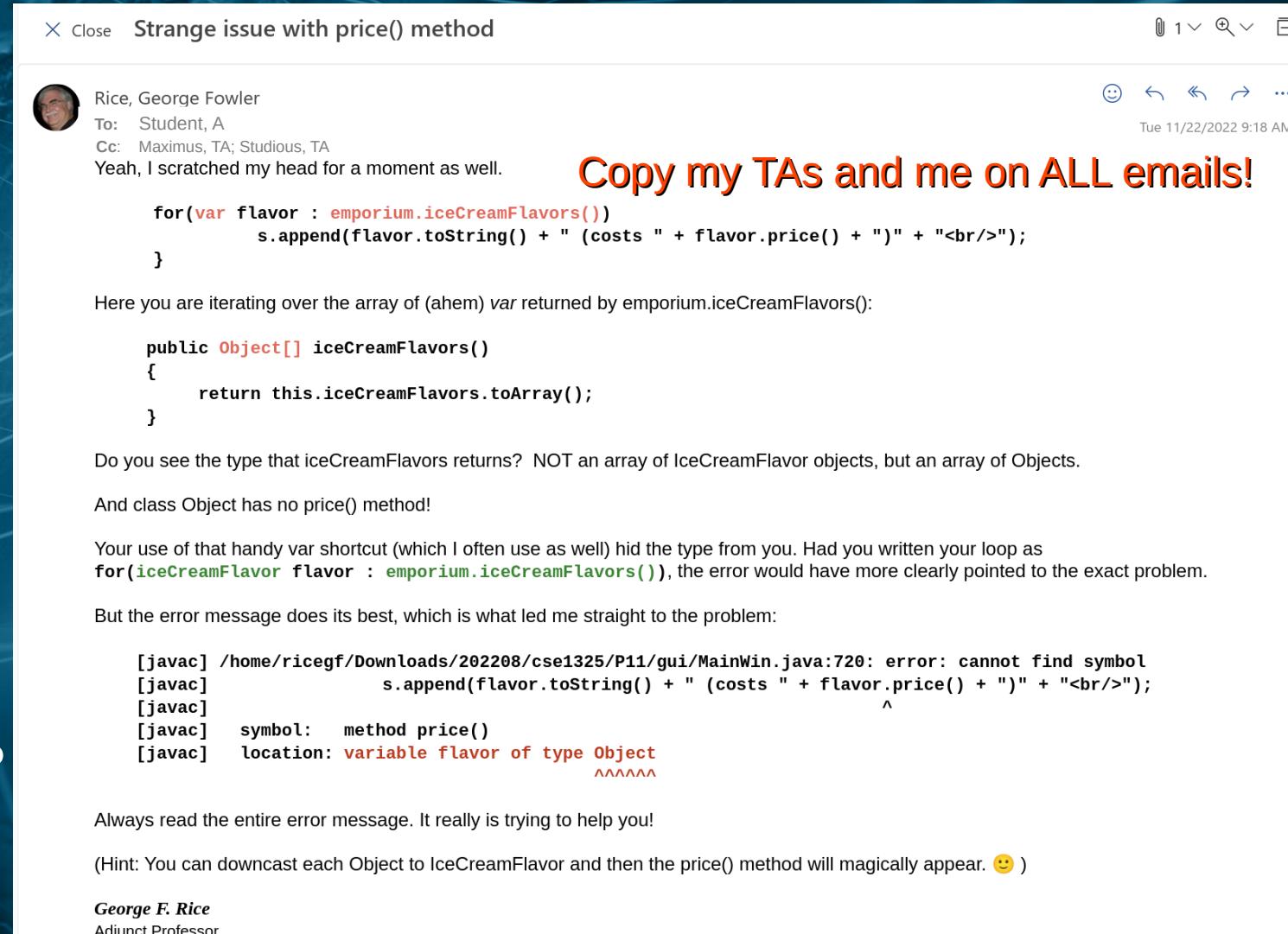
Rich text!

Central repo  
of interaction

Good search

Easy to add  
& remove TA

Did I mention  
?



**Copy my TAs and me on ALL emails!**

Rice, George Fowler

To: Student, A

Cc: Maximus, TA; Studious, TA

Yeah, I scratched my head for a moment as well.

```
for(var flavor : emporium.iceCreamFlavors())
    s.append(flavor.toString() + " (costs " + flavor.price() + ")" + "<br/>");
}
```

Here you are iterating over the array of (ahem) var returned by emporium.iceCreamFlavors():

```
public Object[] iceCreamFlavors()
{
    return this.iceCreamFlavors.toArray();
}
```

Do you see the type that iceCreamFlavors returns? NOT an array of IceCreamFlavor objects, but an array of Objects.

And class Object has no price() method!

Your use of that handy var shortcut (which I often use as well) hid the type from you. Had you written your loop as `for(iceCreamFlavor flavor : emporium.iceCreamFlavors())`, the error would have more clearly pointed to the exact problem.

But the error message does its best, which is what led me straight to the problem:

```
[javac] /home/ricegf/Downloads/202208/cse1325/P11/gui/MainWin.java:720: error: cannot find symbol
[javac]                     s.append(flavor.toString() + " (costs " + flavor.price() + ")" + "<br/>");^
[javac]   symbol:   method price()
[javac]   location: variable flavor of type Object ^^^^^^
```

Always read the entire error message. It really is trying to help you!

(Hint: You can downcast each Object to IceCreamFlavor and then the price() method will magically appear. 😊 )

# Reading Opportunities

- Reading opportunities are given in Canvas > Modules
  - The lectures will cover all of the knowledge specified in the CSE1325 Student Learning Objectives
  - The reading opportunities will give you much of the same plus some additional information
    - They offer different perspectives and additional knowledge
    - They approach the teaching differently
    - As such, you may find one or more of the texts enhances your understanding and are thus valuable for expanding your skills and “filling in the gaps”
- Reading opportunities are optional but recommended if you are having trouble keeping up with the class

5. Skim: Optional reading material<sup>1</sup> for Lecture 03

A. Java Tutorials Lesson: [What is an Object?](#), [What is a Class?](#), and [Classes and Objects](#) (excluding Nested Classes)

B. Jenkov [chapter 21 \(Classes\)](#), [chapter 22 \(Fields\)](#), [chapter 23 \(Methods\)](#), and [chapter 24 \(Constructors\)](#)

6. Contact: The professor or any TA via email or Teams with ANY issues with the above, ANY questions related to these topics, or pretty much anything about which you'd like to chat. We love questions!

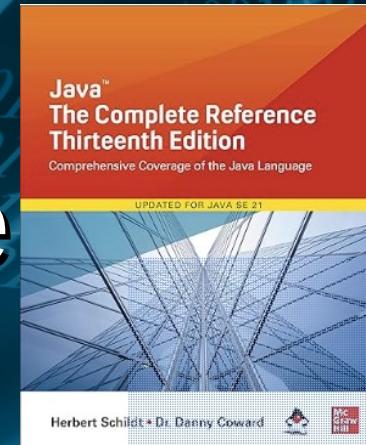
# Additional (Optional) Java Resources

- Textbooks (will be included in optional reading opportunities)
  - [Jenkov.com](#) (Java SE Tutorials only)
  - [The Java™ Tutorials](#) (omits most post-Java 8 features)
- Additional Books
  - **Java: The Complete Reference** by Herbert Schildt and Dr. Danny Coward
  - **Head First Java** by Kathy Sierra & Bert Bates
  - **Java for Dummies** by Barry A. Burd
  - [Java, Java, Java](#) by Ralph Morelli and Ralph Walde  
(omits most post-Java 8 features)
- Video Series
  - [Moshi on Java](#)

As with the reading opportunities, these resources are **OPTIONAL**.  
If they help, please let me know. A few reviews with my thoughts follow.

## Book Review (13<sup>th</sup> Edition)

# Java: The Complete Reference



Earlier versions were published by Oracle (the Java owner) as Java: A Beginners Guide, but the series is now published independent of Oracle by McGraw-Hill Publishing and renamed.

This is a book for serious students. It didn't strike me as dense at all - it's quite a bit better than the Oracle tutorials, I think - but it takes its subject seriously and covers it thoroughly.

Reviews for earlier editions are outstanding, and its longevity (13 editions?) is impressive.

And 1250 pages? This book must be listed next to "completeness" in the dictionary!

I skimmed a couple of chapters, and it meets my preference for completeness and clarity.

I'm also won over by their use of the command line (although the DOS command.exe shell - oh, well) to build and run programs early on.

**If I had a required textbook, this would be it.**

The Thirteenth Edition was published Jan 11, 2024 and covers through Java 21! This bodes well for an author who really cares and thus keeps his material up to date.

If you have a bent to more complete and thorough coverage and find light and breezy annoying, this is the choice for you.

# Book Review (3<sup>rd</sup> Edition) Head First Java

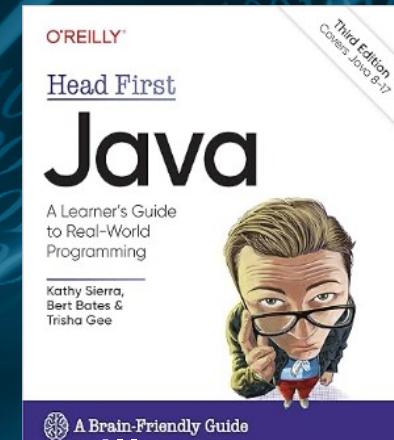
This is the complete opposite of Java: The Complete Reference – a light, breezy, artzy, cartoon-filled book that still gets to object-oriented programming (and UML - woot!) very quickly.

If you hate textbooks, but want to learn efficiently and with humor, this appears to be a good choice.

The Third Edition covers through Java 17, which is close enough to the version we are using in this class.

It has only half as many pages as Java: The Complete Reference, although it covers most of the same topics. It just covers them in a lighter fashion.

**If you want something that is not intimidating in the slightest, like an over-eager puppy, here's your choice!**



# Book Review (8<sup>th</sup> Edition) Java For Dummies

The "for Dummies" books are legendary in their approachability, and this relatively short book (just over half the length of Head First Java) doesn't disappoint in this regard.

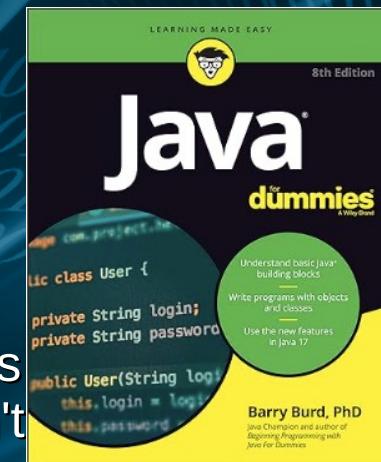
It's not as light and breezy as Head First Java nor as thorough as Java: The Complete Reference, but it's much more basic and walks more slowly through the fundamentals for those still struggling with programming in general.

**If you don't feel you had a firm foundation in C and are intimidated by Java, this book will take you back to basics and walk you in great detail through programming in general.**

If you feel you've fallen behind in CSE1325 and need to catch up quickly, this book may be very helpful. It is not as comprehensive as the first two books by any means, but it assumes much less on the part of the student and holds her hand more tightly until confidence is established.

The Eighth Edition covers through Java 17, close enough to the version we are using in this class.

If you are looking for a basic programming book that happens to focus on Java, this is the book for you.



# Video Review Moshi on Java



I looked at several YouTube series on Java, generally with disappointment. But this one did seem to complement my approach to CSE1325 rather well.

I watched several sections of the 2½ hour YouTube video and was pleased in general with his approach.

Be forewarned that he encourages the use of IntelliJ to generate code for you as you go – I again ***strongly*** recommend that you NOT do that (yet) – but otherwise his terminology and coding practices seem mostly complementary to how I teach Java.

And it's free (with ads).

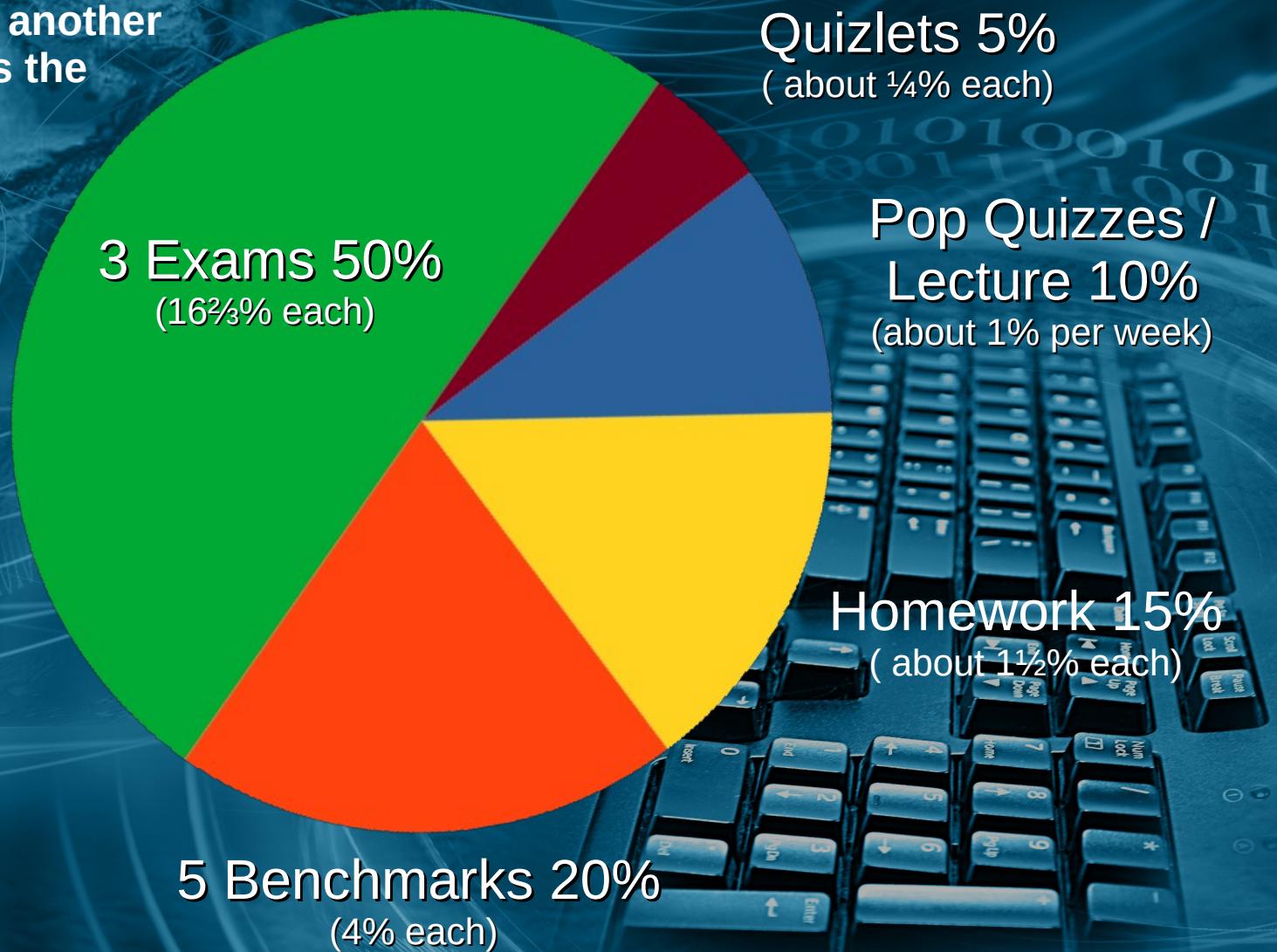
<https://www.youtube.com/watch?v=eIrMbAQSU34>

He has a \$99 video course if you find that you *really* love his teaching style. I promise I won't be offended, if you learn Java and OOP well this semester!

<https://codewithmosh.com/p/the-ultimate-java-mastery-series>

# Final Grade Composition

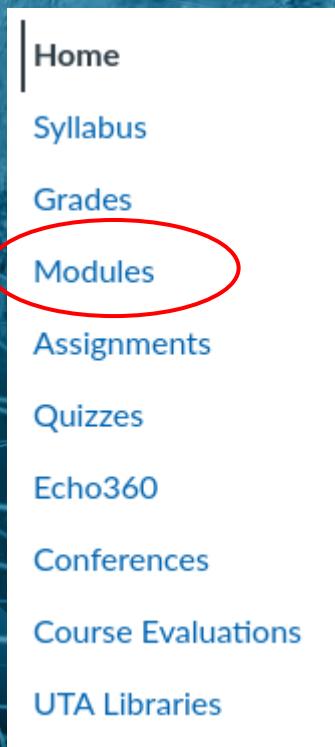
If you are ill or have another *good* reason to miss the lecture, **email me IN ADVANCE** and we will work out a way for you to avoid losing the pop quiz points.



If Benchmarks are not given for any reason, Homework will count for 20% and Three Exams 65%.

# Lectures

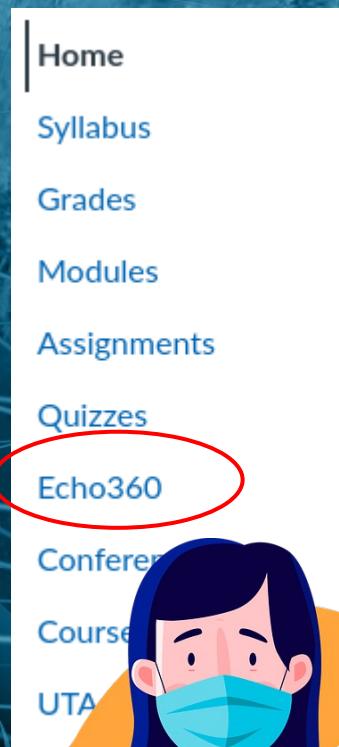
## Canvas Menu



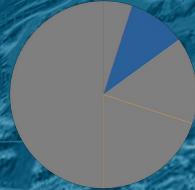
- Lecture attendance is *mandatory*
  - You learn much more by interacting in class
  - Slides, source code, goals, to-do lists, and other related material are posted in Canvas > Modules the day *before* the lecture – be prepared!
- Lectures are *mostly* face-to-face
- Since we do not have a mandatory textbook, **this is your best learning opportunity**

# Lecture Recordings

## Canvas Menu

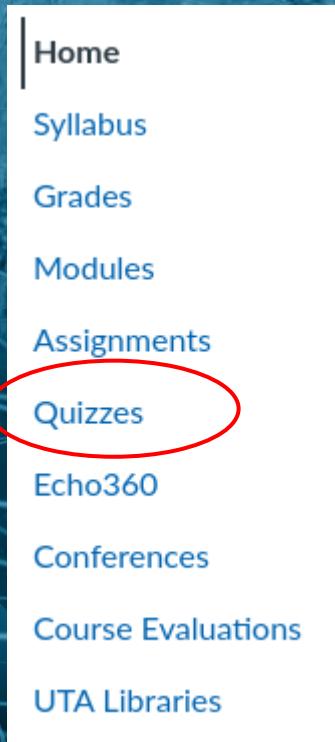


- Live lectures are both streamed & recorded
  - You will have access to *both* sections – take your pick!
  - **For occasional use only!**  
*This class is NOT for online-only students!*
  - The pop quiz passwords are *intentionally* not visible on the Echo360 recordings – pop quizzes measure your attendance, mental and physical, and cannot be made up
  - NOTE: I have no control over broadcast / recording quality
- If you are contagious, **STAY AWAY**
  - Notify me IN ADVANCE if possible or provide a doctor's note
  - Watch the lecture stream live if possible and email pop quiz answer *during* the pop quiz (NOT the end of class!) Or with a doctor's note, watch the recording and I will exempt you from that day's pop quiz



# Pop Quizzes

## Canvas Menu

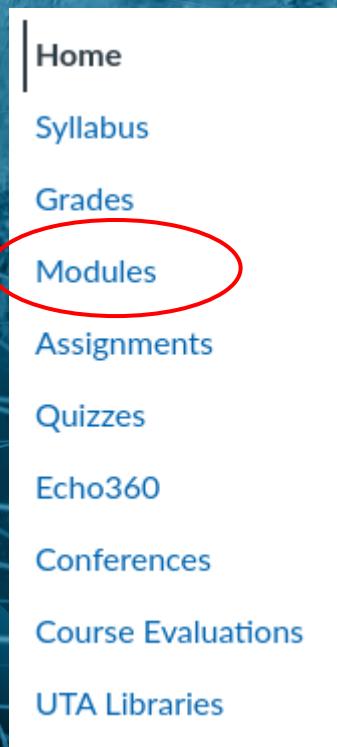


- Frequent pop quizzes measure both physical attendance and mental engagement
  - Pop quizzes are taken on Canvas – be sure you have access in class!
    - If you can't log in during class, raise your hand. Late submissions are *never* accepted!
    - If you are ill and watching video live or later at home, notify me in advance. We'll make arrangements.
  - Pop quizzes comprise 10% of your final grade

Look  
At This!

# Pre-Recorded Lecture Videos

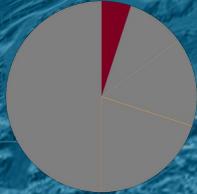
## Canvas Menu



- A few lectures are pre-recorded online via Canvas Studio and posted to Modules
  - Canvas > Modules > Lecture 02 for example
  - Online lectures are presented as several YouTube-style “chapters”, about 8 minutes each
- Pop quiz grades *may* be taken based on Canvas video access logs

The image shows a screenshot of a lecture video page. At the top, there is a blue icon of two people and the text "LECTURE VIDEOS". Below this, a message reads: "The material covered here (additional examples of the information covered in Lecture 01) is certainly on the exam and the post-lecture quizlet will certainly be graded. A pop quiz grade may be calculated based on Canvas viewing records as of 8 am on the due date." Underneath the message, the title "Chapter 01 - Multiplication Table" is displayed. To the left of the title is a small video thumbnail showing a man with glasses and a blue shirt. To the right of the title is a terminal window showing command-line output. The terminal output includes:

```
ricegf@antares:~/dev/202201/demo$ ls -l
total 0
-rw-rw-r-- 1 ricegf ricegf 0 Jan 6 15:22 MultiplicationTable.java
ricegf@antares:~/dev/202201/demo$ javac MultiplicationTable.java
ricegf@antares:~/dev/202201/demo$ java MultiplicationTable
Times 12 Table
ricegf@antares:~/dev/202201/demo$
```



# Post-Lecture Quizlets

## Canvas Menu

A screenshot of the Canvas LMS interface showing the course menu. The menu items are: Home, Syllabus, Grades, Modules, Assignments, Quizzes, Echo360, Conferences, Course Evaluations, and UTA Libraries. The 'Quizzes' link is highlighted with a red oval.

- A Canvas Quiz is required after every lecture
  - Due at start of *next* lecture
  - Helps ensure that you understand the main points
  - Typically 3-7 questions, mostly multiple choice
  - You have *unlimited* tries!
  - Only highest score counts
  - These comprise 5% of your semester grade

Look  
At This!

### Lecture 08 Quiz

Started: Aug 10 at 2:56pm

#### Quiz Instructions

Let's talk inheritance!

**Question 1** 1 pts

Inheritance is defined as

The inclusion of a header file

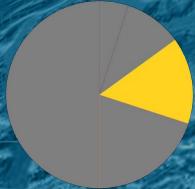
Reuse and extension of fields and method implementations from another class

Establishing the validity of a class relationship

Creating a new class from several composite classes

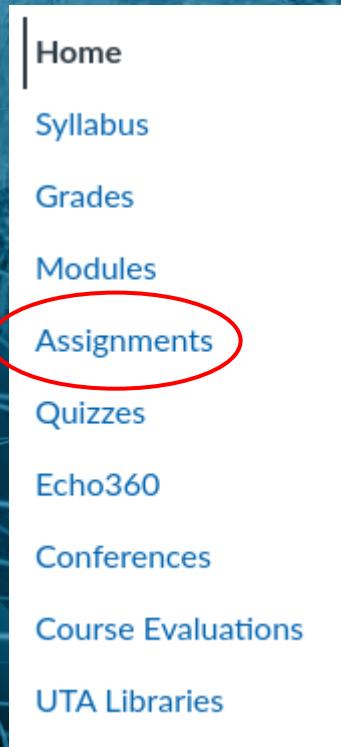
**Question 2** 1 pts

In C++, the ancestor class from which members are inherited is called the [Select] , while the inheriting class is [Select]



# Homework

## Canvas Menu



- Around 11 homework assignments are planned
- Push your solution code by 8 am each Tuesday to your private GitHub repository (details shortly)

→ If GitHub fails, *email* your solution by 8 am Tuesday

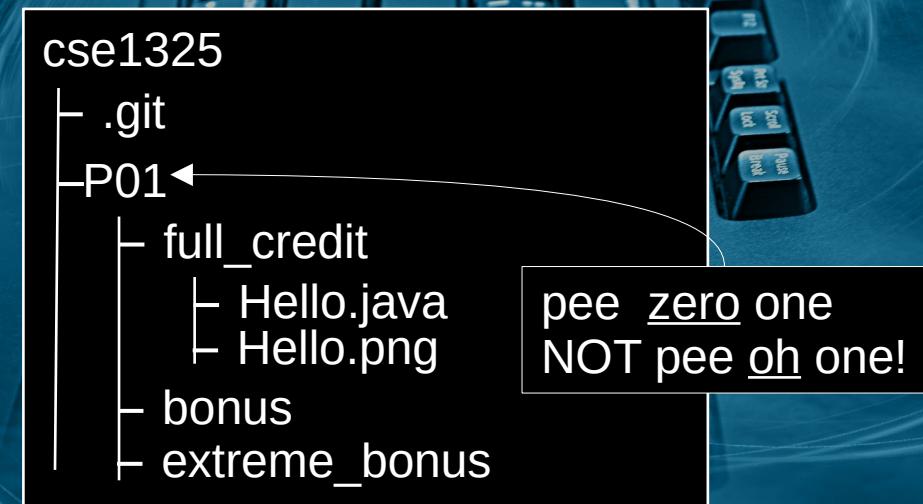
- The due date on the assignment itself is authoritative

- These comprise 15% of your semester grade

Look  
At This!

## WARNING

Submitting code that substantially duplicates another current or previous student's work will result in an F for the class, and will also be reported to the Office of Student Conduct.



This exact directory layout is required

# Extra Credit

- Homework will often offer “Bonus Levels”
  - Completing the Full Credit portion of the homework successfully earns 100%
  - Bonus levels may offer up to 10% additional credit
  - Homework averages over 100% DO count toward your final grade and MAY compensate for exam and other mistakes
- No additional extra credit work is available

Start Early and Avoid the Rush



# GitHub

(Click Me!)

Suggested solutions  
are provided via GitHub  
immediately following  
the deadline for all  
assignments AND exams.

Build, modify, explore,  
and understand them all!

<https://github.com/prof-rice/cse1325-prof.git>

*How to actually learn any new programming concept*

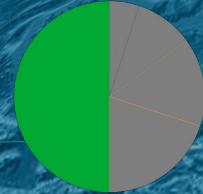


*Essential*

Changing Stuff and  
Seeing What Happens

O RLY?

@ThePracticalDev



# In-Class Exams

- Preparation resources provided\*
  - Study guide listing exam topics
  - Practice exams with keys
- Exams are always face-to-face
  - Paper and pencil only
    - Matching (vocabulary) – Talk like a professional
    - Multiple Choice (concepts) – Think like a professional
    - Free Response (coding) – Code like a professional
  - Goal is for exams to be graded by\* and reviewed next lecture
    - You will receive\* a scan of your graded exam in Canvas for keeps

These 3 exams together comprise 50% of your grade  
(16⅔% each)

\*Statement of intent, not a promise!

## Multiple Choice

Read the full question and every possible answer. Choose the one best answer for each question and write the corresponding letter in the blank next to the number. (15 at 2 points each)

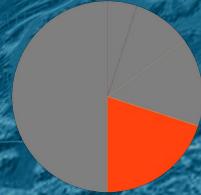
1. \_\_\_\_\_ Which is not a common debugger feature?
  - A. Git Commit
  - B. Breakpoint
  - C. View Memory
  - D. Step
2. \_\_\_\_\_ To deliver a software product update to GitHub from a cloned directory, type
  - A. git add -u ; git commit -m "Done" ; git push
  - B. git add -u ; git deliver
  - C. git add -u ; git upload
  - D. git add -u ; git checkin ; git deliver
3. \_\_\_\_\_ Which of the following is NOT a scope in C++?
  - A. local
  - B. statement
  - C. enum
  - D. global
4. \_\_\_\_\_ How are integers formatted using streams?
  - A. I/O Manipulators, e.g., std::cout << std::hex << std::setw(2) << i;
  - B. Functions, e.g., std::cout << std::operator<<(\*%2x\n", i);
  - C. Formats, e.g., std::cout << std::sprintf("%2x\n" << i;
  - D. Substrings, e.g., std::cout << substr(i, "%2x\n", 2);
5. \_\_\_\_\_ Which are TRUE about exceptions?
  - A. If an exception is not caught, it will terminate the program
  - B. To compile a program that uses exceptions, add the --except flag
  - C. To generate a runtime\_error exception, write std::runtime\_error().throw();
  - D. To catch a runtime\_error exception, write std::runtime\_error().catch();

Look  
At This!

# Make-Up Work and Appeals

- Exams, once handed to you, cannot be made up
  - A valid excuse (e.g., doctor's note) for an unexpected issue is required to avoid a zero on a missed exam
    - The make-up exam, if offered, will be different from that given to those attending on exam day
    - At my discretion, an excused missed exam grade may be replaced by the average of other exam grades
- Homework extensions are never granted
  - I provide a suggested solution instantly after the deadline, more valuable than extensions
- Grade appeals must be filed via email within 2 weeks of the day the grade was posted

Never



# Benchmarks

- These 5 *online-only* mini-exams are taken by *all* CSE1325 students
  - Each focuses on a specific OOP, Java, or C++ topic
  - Unlike in-class exams, you will NOT see your graded benchmark exam
- I personally review ALL exam questions
  - I DO know what will be on these exams (I wrote many of them!)
  - They are similar to post-lecture quizzes
  - My best effort will go to preparing you to excel at them

These 5 exams together comprise 20% of your grade  
(4% each)

Look  
At This!

## Question 4

Which lambda introducer (or capture clause) is invalid?

- [&]
- [&, &options]
- All of these are valid
- []

# Honors Contracts

- Unfortunately, CSE1325 is NOT eligible for an Honors Contract
- You can check on other courses and professors with these links

Check if your course is eligible.

Check if your instructor is eligible.

Check Honors project guidelines.



# Final Pleadings (Don't)

- Your final grade of record follows a strict 10-point scale based on your final average
  - I never give “free points” to get to next letter grade; don’t ask; and if you ask, I won’t respond
  - Any consequences you expect as a result of a poor grade never qualifies you for a higher grade.  
**If it’s that critical, earn a higher grade!**

90.00+ is A  
80.00-89.99 is B  
70.00-79.99 is C  
60.00-69.99 is D  
<59.99 is F





The UTA Logo usually indicates a change in topic

# A Brief Regression to Philosophy

## Engineers Solve Problems

“The ultimate aim of programming  
is always to produce useful systems.”

– Bjarne Stroustrup

Inventor of C++



“Talk is cheap.  
Show me the code.”

– Linus Torvolds

Inventor of Linux and git



<http://stroustrup.com/>  
By Julia Kryuchkova - Own work  
CC BY-SA 2.5

# “Most Popular” Languages

Java’s performance on these rankings continues to impress, all these years later, and as it’s shown a remarkable ability to adapt to a rapidly changing landscape it’s a language that would be difficult to bet against. – Redmonk

	TIOBE	PYPL	RedMonk	IEEE
1	Python	Python	JavaScript	Python
2	C++	Java	Python	Java
3	Java	JavaScript	Java	JavaScript
4	C	C/C++	PHP	C++
5	C#	C#	C#	TypeScript
6	JavaScript	R	TypeScript	SQL
7	Go	PHP	C++	C#
8	SQL	TypeScript	CSS	Go
9	Visual BASIC	Rust	Ruby	C
10	Fortran	Objective C	C	HTML
	December '24	August '24	June '24	Aug '24

Based on the number of skilled engineers world-wide, courses, and third party vendors

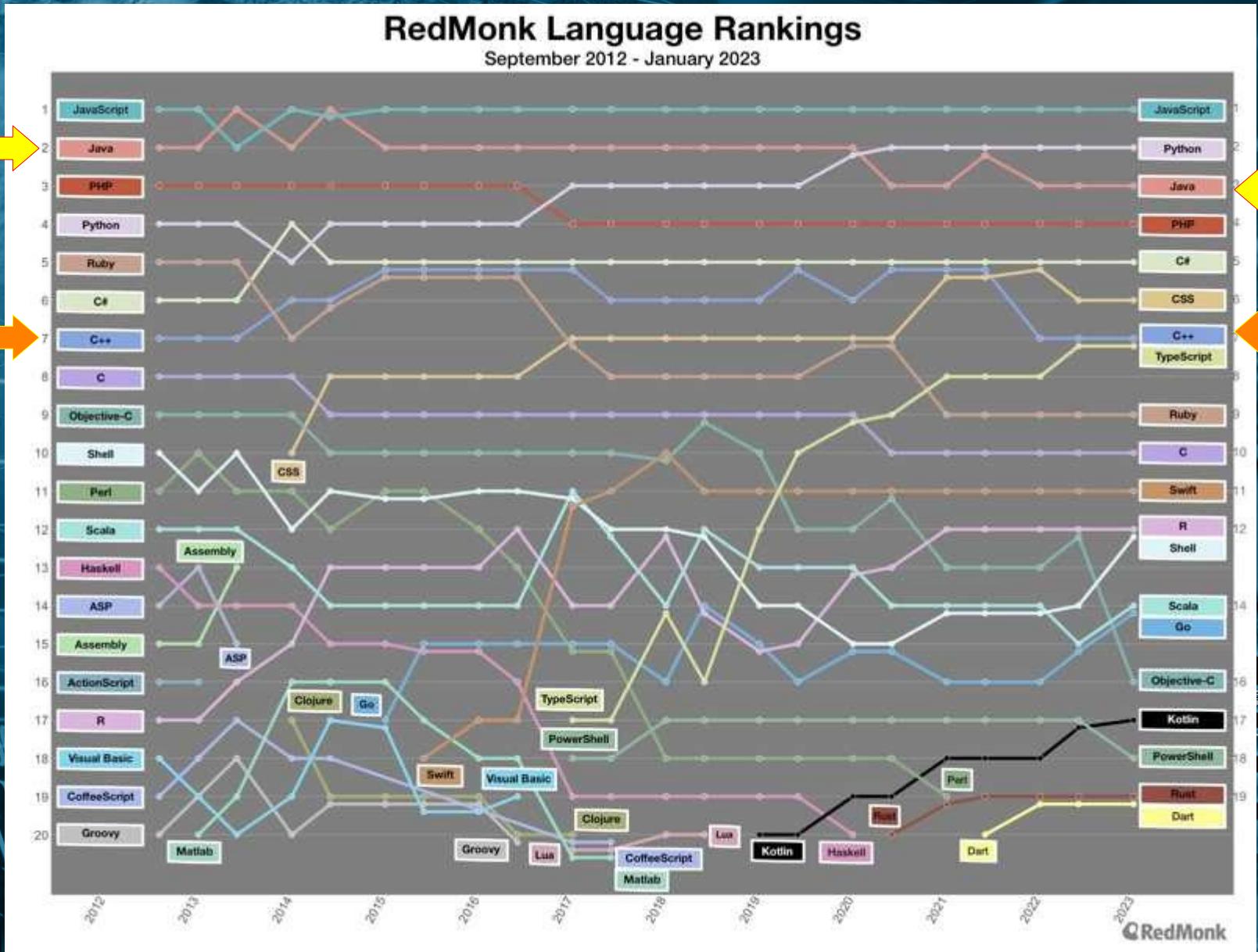
Merged language usage ([GitHub](#)) and discussion ([Stack Overflow](#)) rankings

Based on Google tutorial searches

interactive Web app interactive Web app

# Language Trends

## According to RedMonk, Winners Stay Winners





James Gosling,  
Java's inventor

Duke, Java's  
mascot

# Java: A History



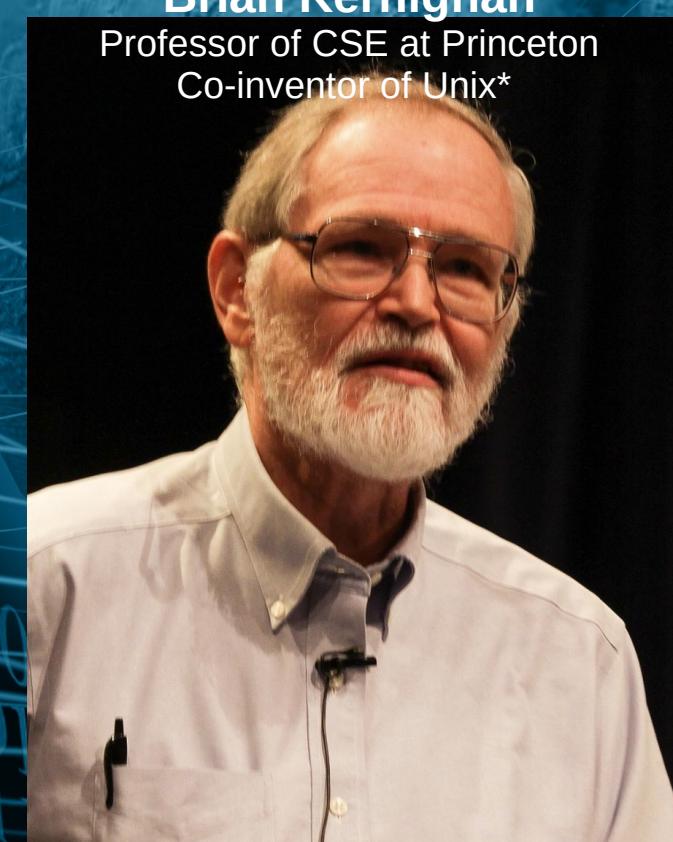
Year	Version	Summary
1996	1.0	
1997	1.1	AWT improvements, inner classes, JDBC database API, and the JavaBeans IDE
1998	1.2	Java Desktop renamed J2SE, Swing, reflection, collections, CORBA support, JIT compiler support, plug-in support
2000	1.3	Virtual Machine improvements, sound and directory APIs, debugger architecture improvements
2002	1.4	Java Community Process adopted, regular expressions, exception chaining, integrated XML
2004	5.0 (Also called 1.5)	For-each loop, generics, autoboxing, var-args
2006	6	JavaScript embedded, annotations, database and tool improvements
2011	7	String support for switch, var type inference (“auto” in C++), try with, join/fork update, new file I/O library, better support for Jython et. al. OpenJDK 7u, an open source version of Oracle JDK 7, is subsequently released.
2014	8	Lambdas, default interface methods, new date/time library, OpenJDK 8 formed the basis for Oracle JDK 8 starting with this release.
2018	11	Gnu GPL license, 2 new garbage collectors, “Flight Recorder” deep debugger
2021	17	Switch expressions (generalized ternary), text blocks, instanceof pattern matching, non-volatile buffers, packaging tool
2023	21	Record and switch patterns, String templates, virtual threads, sequenced Collections, key encapsulation

Note: Only Long-Term Support (LTS) releases are shown. Next up is Java 25, expected in September 2025.

# Taking the First Step

**Brian Kernighan**

Professor of CSE at Princeton  
Co-inventor of Unix\*

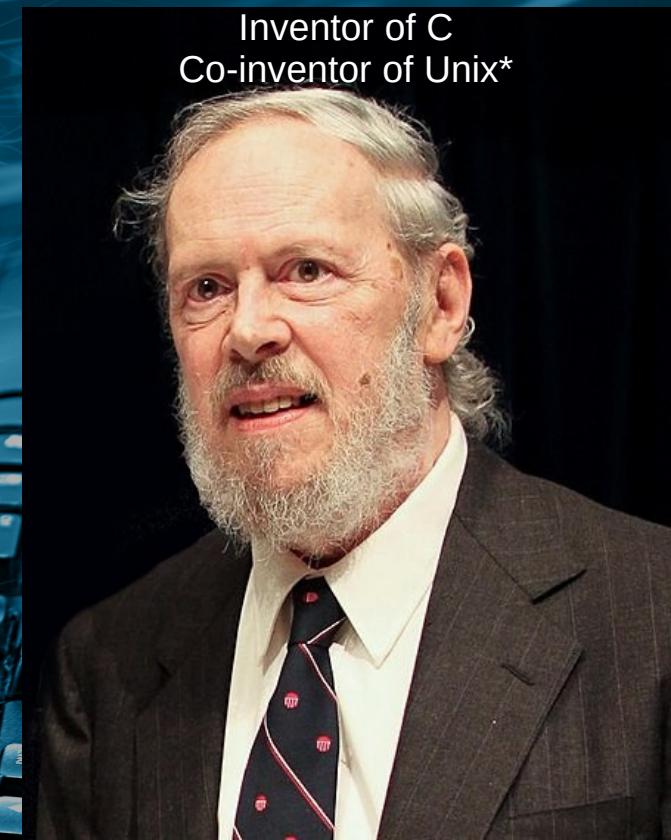


“The only way to learn a new programming language is to write programs in it. The first program to write is the same for all languages: Print the words ‘hello, world.’”

From “The C Programming Language” by K&R

**Dennis Ritchie**

Inventor of C  
Co-inventor of Unix\*



\* Like me, they were *slightly* younger then...

# hello, world

Python:

Structured  
Object-Oriented

```
print("hello, world")
```

C:

Structured

```
#include <stdio.h>
int main() {
    printf("hello, world");
}
```

C++:

Structured  
Object-Oriented

```
#include <iostream>
int main() {
    std::cout << "hello, world" << std::endl;
}
```

Java:

Object-Oriented

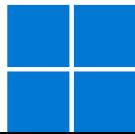
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("hello, world");
    }
}
```



Note: Source code from the lectures is always provided to you at <https://github.com/prof-rice/cse1325-prof.git>!

# Running Hello.java

Via bash



or zsh



← Click me!

```
ricegf@pluto:~/dev/202008/01/java$ ls  
Hello.java  
ricegf@pluto:~/dev/202008/01/java$ javac Hello.java  
ricegf@pluto:~/dev/202008/01/java$ ls  
Hello.class  Hello.java  
ricegf@pluto:~/dev/202008/01/java$ java Hello  
Hello, Java!  
ricegf@pluto:~/dev/202008/01/java$ █
```

Compile Hello.java into Hello.class

Run Hello.class

Note that you do NOT include the .class extension with the `java` command.  
Pressing Tab after the first few letters will correctly complete the command.

Click the logo of your operating system for additional information.

This looks suspiciously like your first homework...



# Special Topic: Version Control

- “The dog computer ate my homework!”



→ You'll see definitions marked as “Expertise” again. Soon. Likely on an exam!



**Version Control** - The task of keeping a software system consisting of many versions and configurations well organized

# A Simple Git Session...

```
student@maverick:~$ ls  
bin Desktop Documents Downloads Music Pictures Public Templates Videos  
student@maverick:~$ git clone https://github.com/prof-rice/cse1325.git  
Cloning into 'cse1325'...  
Username for 'https://github.com': prof-rice  
Password for 'https://prof-rice@github.com':  
remote: Enumerating objects: 6, done.  
remote: Counting objects: 100%, done.  
remote: Compressing objects: 100% (3/3), done.  
Unpacking objects: 100% (3/3), 838 bytes | 83.00 KiB/s, done.  
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0  
student@maverick:~$ ls  
bin cse1325 Desktop Documents Downloads Music Pictures Public Templates Videos  
student@maverick:~$ cd cse1325/  
student@maverick:~/cse1325$ ls  
README.md test.cpp  
student@maverick:~/cse1325$ e hello.cpp  
student@maverick:~/cse1325$ git add hello.cpp  
student@maverick:~/cse1325$ git status  
On branch master  
Your branch is up to date with 'origin/master'.  
  
Changes to be committed:  
(use "git restore --staged <file>..." to unstage)  
  new file:   hello.cpp  
  
Untracked files:  
(use "git add <file>..." to include in what will be committed)  
  hello.cpp  
  
student@maverick:~/cse1325$ git commit -m 'Saying hello'  
*** Please tell me who you are.  
  
Run  
  
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

Clone your GitHub repository and do all of your homework in it!

Create one or more valuable files

Add valuable files to git's list of files to be protected

They aren't protected yet, though – git just knows you care about them

Don't forget to introduce yourself to git! (You need do this only once.)

# ...with a Few Simple Commands...

```
fatal: unable to auto-detect email address (got 'student@maverick.(none)')  
student@maverick:~/cse1325$ git config --global user.email "george.rice@uta.edu"  
student@maverick:~/cse1325$ git config --global user.name "Prof Rice"  
student@maverick:~/cse1325$ git commit -m 'Saying "Hello"'
```

Tell git who you are  
Commit the file(s) to git's protection

```
[master d379fe1] Saying "Hello"  
 1 file changed, 3 insertions(+)  
 create mode 100644 hello.cpp  
student@maverick:~/cse1325$ git log
```

Ask git for a list ("log") of every commit  
you've made thus far.

The most recent commit  
(HEAD → main) is at the top.

The most recent pushed to GitHub is  
next, called origin/main. Note that  
your files are local, not yet on GitHub!  
And here's where we *created* our new  
GitHub repository! Isn't it cute?

```
Saying "Hello"  
  
commit 51b79cc0ca063b74baaealc0fac38765dcb588fe (origin/main, or... in/HEAD)  
Author: ricegf <george.rice@uta.edu>  
Date:   Wed Aug 12 19:17:28 2020 -0500
```

```
hello file  
  
commit d86a731bc122e74f97a8385d96300calec5784f3  
Author: Professor George F. Rice <george.rice@uta.edu>  
Date:   Wed Aug 12 19:15:00 2020 -0500
```

Now, let's push (upload) our latest  
commit to GitHub.

```
Initial commit  
student@maverick:~/cse1325$ git push  
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 370 bytes | 370.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://github.com/prof-rice/cse1325.git  
  51b79cc..d379fe1  master -> master  
student@maverick:~/cse1325$ git log
```

Our log now shows our "stage"  
(local repository) matches GitHub,  
meaning our files are safe!

# ...Can Save the Day...

```
student@maverick:~/cse1325$ cat hello.cpp
#include <iostream>
int main() {std::cout << "Hello\n";}
```

```
student@maverick:~/cse1325$ shred hello.cpp
student@maverick:~/cse1325$ head -1 hello.cpp
qF0T000pE8/0#a=0y000^k0040ucKtr00 &0'w/000HT0 T5y_S000cc0%_r2v00
0Vp00n000b_yvhC"C0_,C000] 0=0000*:|00]0|}{00"TU0:0000} |B02
```

```
0000>0'J0Nz0UZe00=ry7000C000tM00X00S{?200/{0>`0m000.
```

```
X~0[0N00e0?000/0003'
g00$0u0.0g0000HC0J{
0E08=010R(00#(v400Cx K:"DB0_0jX00(000L0}AJ0Z0+06e000;000N,00
v0)004r.>0
c00Ii&00
```

```
36f]`s!0G04~°002n$00,0M>0(0f0o00A10&0z#0
student@maverick:~/cse1325$ #woops
student@maverick:~/cse1325$ git checkout hello.cpp
Updated 1 path from the index
student@maverick:~/cse1325$ cat hello.cpp
#include <iostream>
int main() {std::cout << "Hello\n";}
```

```
student@maverick:~/cse1325$ g++ --std=c++17 hello.cpp
student@maverick:~/cse1325$ ./a.out
Hello
student@maverick:~/cse1325$
```

Our beloved file, blissfully unaware  
that disaster is only seconds away!

What's that compile command again?

Uh oh – something looks... different.  
When is this due again?

Every version of every file that you've  
ever “committed” to git can be  
recalled (“checked out”)  
non-destructively. If you want to  
see how a file looked 113 commits  
ago, it's less than a second away!

Saved by version control... again!!!  
But don't get cocky, kid...

# ...Or the Semester!

```
student@maverick:~/cse1325$ cd ..  
student@maverick:~$ rm -fr cse1325/  
student@maverick:~$ ls  
bin Desktop Documents Downloads Music Pictures Public Templates  
student@maverick:~$ git clone https://github.com/prof-rice/cse1325.git  
Cloning into 'cse1325'...  
remote: Enumerating objects: 9, done.  
remote: Counting objects: 100% (9/9), done.  
remote: Compressing objects: 100% (6/6), done.  
remote: Total 9 (delta 0), reused 6 (delta 0), pack-reused 0  
Unpacking objects: 100% (9/9), 1.15 KiB | 168.00 KiB/s, done.  
student@maverick:~$ ls  
bin cse1325 Desktop Documents Downloads Music Pictures Public Templates Videos  
student@maverick:~$ cd cse1325/  
student@maverick:~/cse1325$ cat hello.cpp  
#include <iostream>  
int main() {std::cout << "Hello\n";}  
  
student@maverick:~/cse1325$
```

rm -fr obliterates the local repo – pretty much “my laptop was stolen!”

My homework! It's *all gone!!!*

Oh, wait, I pushed it to the server!  
It's only a clone away.

Saved by version control... *again!!!*

If you use git consistently, your files in the virtual machine can be easily shared with your host computer. They are **safe**, and we can readily see (and grade) them. ***Even if you forget to “turn them in”!***

**So relax!**

<https://git-scm.com/documentation>



# “But My IDE Already Knows Git!”

- Your IDE will not take the exam
  - You will
- **NEVER trust magic**
  - If you can't explain what's happening at a basic level, you're not an engineer – you're a user
  - You cannot correctly design, implement, debug, or support what you do not understand



Last semester, a disappointing number of students received 100+ on assignments and 50-60 on exams. Why do you think that happened?

# GitHub Manages Git Repositories in the Cloud

- You work locally, on any machine(s) you like
- If you have questions, “push” and email me or TA\*
  - We can see your work “live”
- The TA will “clone” your solution on the due date
  - Your grade will appear on Canvas
  - The suggested solution will appear in cse1325-prof
    - `git clone https://github.com/prof-rice/cse1325-prof.git`
    - or to (more quickly) update an existing clone
    - `cd ~/cse1325-prof ; git pull`

\* TAs other than mine may also need your GitHub repository URL

# GitHub Benefits You!

- Your work is backed up – no more lost homework!
- Develop on different machines, easily merge your work
- We can see your project state when you email questions
  - no more attached code!
- You'll have a *permanent* cse1325 assignment record
  - You may set your repo public AFTER the semester ends, or copy the code to a public repo complementing your resume
- You'll practice a *professional* coding workflow
  - Very useful for interviews!

Never worry if you forgot to submit homework again!

– Your solution almost “submits itself”!

# Getting Started with GitHub

## Create a GitHub Account

- Go to <https://github.com/>
- Complete the form
  - Select any username and password
  - Use your school email address (@mavs.uta.edu)



# Getting Started with GitHub

## Create a cse1325 Repository

- Create a new repository named 'cse1325'

### Repositories

Find a repository...



Must be 'Private'!

Recommended

Any (or no) license  
is fine

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (\*).

Owner \*



prof-rice



Repository name \*

cse1325

cse1325 is available.

Must be 'cse1325'  
→ LOWER CASE ←  
→ NO punctuation ←

Great repository names are short and memorable. Need inspiration? How about [miniature-octo-journey](#) ?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: Java

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: GNU General Public License v3.0

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set main as the default branch. Change the default name in your [settings](#).

You are creating a private repository in your personal account.

Create repository

# Getting Started with GitHub

## Add Collaborators

- Add prof-rice and the TAs (in Announcements)
  - This will let us grade your assignments

The screenshot shows the GitHub Settings page for a repository. A red arrow points from the text "Get the actual names from Canvas > Announcements" to the "Collaborators" tab in the sidebar. Another red arrow points from the text "Add a collaborator to cse1325" to the search bar where "ManognaCSE1325" is typed. Red circles highlight the "Settings" tab in the top navigation bar, the "Collaborators" tab in the sidebar, the "Add people" button, and the search bar.

Manage access

You haven't invited any collaborators yet

Add people

Get the actual names from  
Canvas > Announcements

Add a collaborator to **cse1325**

ManognaCSE1325

Manogna Shashidhara  
ManognaCSE1325 • Invite collaborator

Select a collaborator above

# Submit Your GitHub URL to Canvas

- This authorizes your GitHub repository as your homework submission channel
  - “Website URL.” is for *your* repository!
- We, too, will clone it.  
What you push, we pull!

2202-CSE-1325-002 > Assignments > P01 - Starting Out With Hello! [Rev 1]

2020 Spring  
Home  
Announcements  
Syllabus  
Grades  
Modules  
**Assignments**

## P01 - Starting Out With Hello! [Rev 1]

Due Jan 28 by 8am

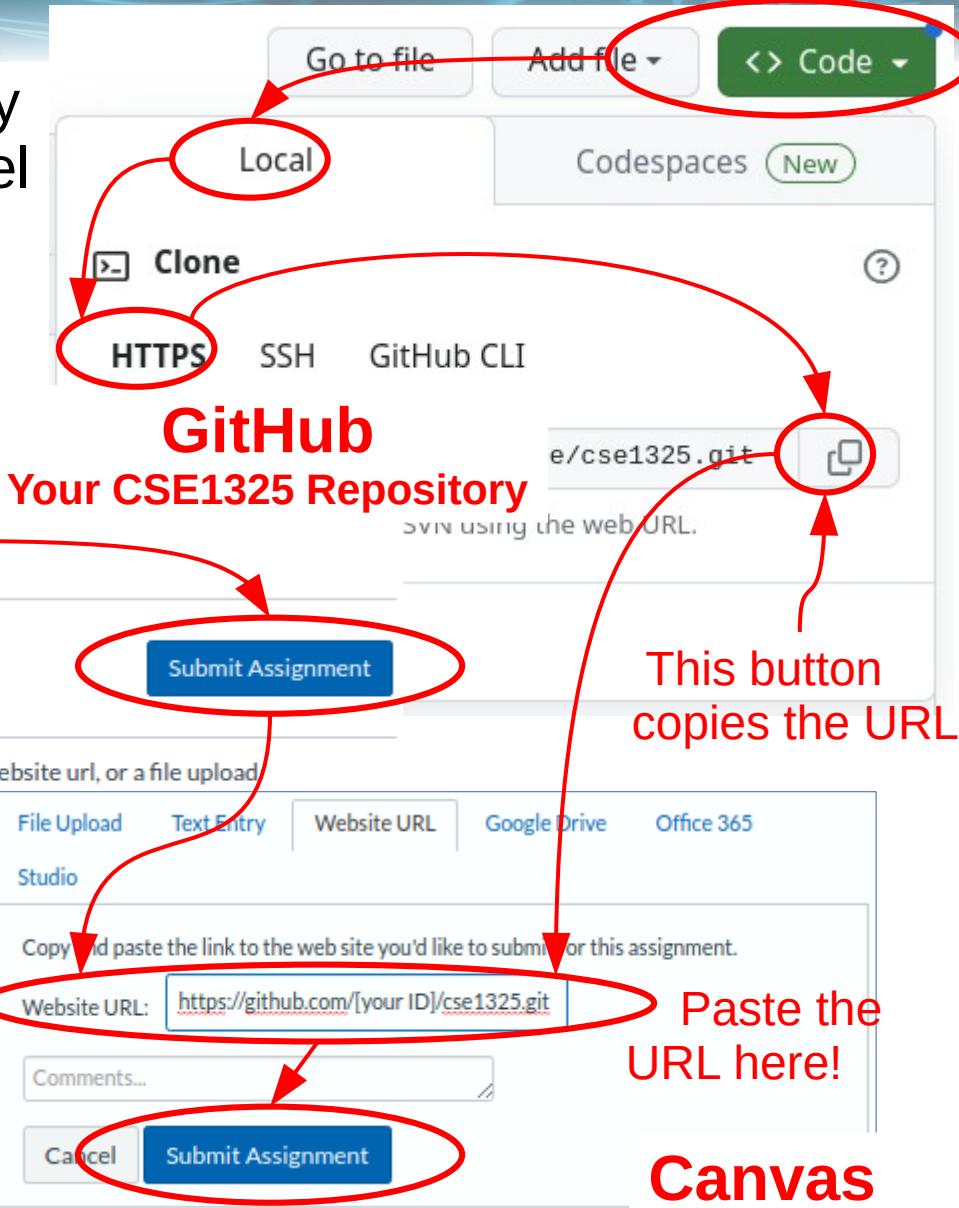
Points 100

Submitting a text entry box, a website url, or a file upload

**Canvas**

The class versions language simply prints “Hello, World!” to languages.  
Compiling and running this program, and adding and committing the files to environment is operating correctly, and exploring its features.

**NOTE: Canvas will report a ‘404 error’ because your repository is private. This is expected, and NOT a problem.**



# Personal Access Token (PAT)

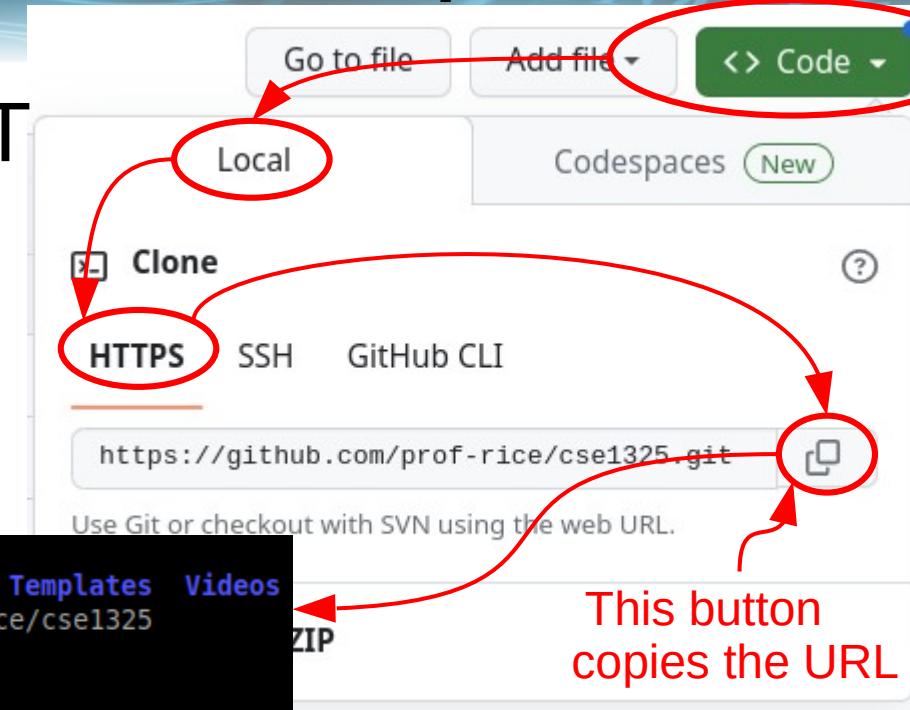
- Your GitHub password will **NOT** work on the command line!
    - You MUST use a **Personal Access Token (PAT)** instead
  - To use a PAT, follow these instructions  
<https://docs.github.com/en/github/authenticating-to-github/keeping-your-account-and-data-secure/creating-a-personal-access-token>
    - You will NOT use a “SAML SSO”, so ignore those steps
    - Once you’ve copied your PAT, **paste it to a command line so that you KNOW it’s on your clipboard!** Press Control-c instead of Enter to get back to a prompt.
  - Enter the clone command (next slide). When prompted for a password:
    - Paste your PAT with right-click > Paste. **You will see NOTHING change – but it’s there (promise!)**
    - Press ONLY Enter immediately – not a single other key – and it will work!
  - **You can cache your GitHub PAT on your private machine** using this
    - `git config --global credential.helper store`
    - If you change your mind, you can disable this using
- ```
git config --unset credential.helper
```

# Getting Started with GitHub

## “Clone” to Each VM / Computer

- Get the https URL for a PAT
- In bash, “git clone *URL*”
  - The cse1325 directory will be created

```
student@cse1325:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
student@cse1325:~$ git clone https://github.com/prof-rice/cse1325
Cloning into 'cse1325'...
Username for 'https://github.com': prof-rice
Password for 'https://prof-rice@github.com':
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 41 (delta 1), reused 38 (delta 1), pack-reused 0
Unpacking objects: 100% (41/41), done.
student@cse1325:~$ ls
cse1325 Documents Music Public Videos
Desktop Downloads Pictures Templates
student@cse1325:~$
```



Note: You can delete (rm -fr) or move (mv) the entire cse1325 directory at any time without harming GitHub or your machine.

# Git Basics

## Introduce Yourself to Git

- The first time you use git, give it your name and email:

```
git config --global user.name "Prof Rice"
```

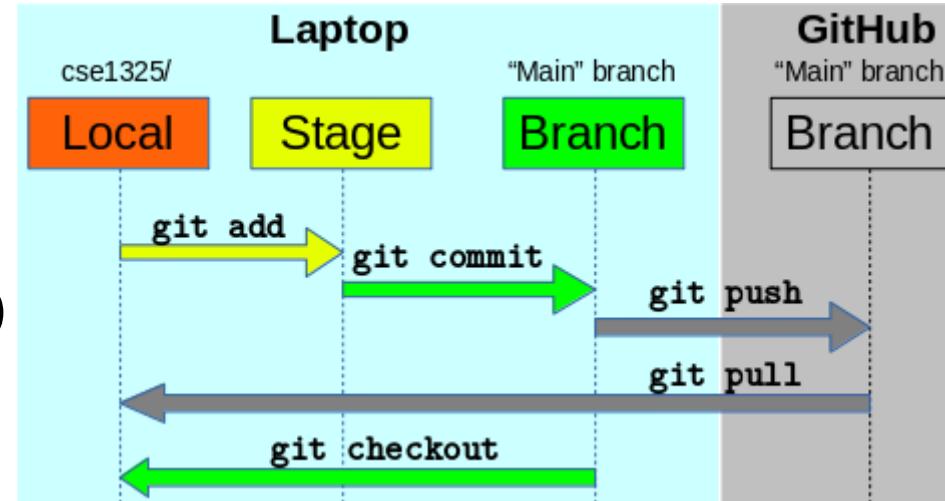
```
git config --global user.email george.rice@uta.edu
```

- In case it's not obvious, use your name and email!
- You need do this only once per account per machine

```
student@cse1325:~$ git config --global user.name "Prof Rice"
student@cse1325:~$ git config --global user.email george.rice@uta.edu
student@cse1325:~$ git config --get user.name
Prof Rice
student@cse1325:~$ git config --get user.email
george.rice@uta.edu
student@cse1325:~$ █
```

# git Basics

- Use “git clone” once to copy your remote repository on GitHub to your laptop
  - Your local repository remembers its “origin” (where from GitHub you cloned)
- When you make changes, use “git add” to copy new and changed files to the *stage* (a local copy)
- Once you’ve edited and added that batch of changes and new files, use “get commit” to copy the staged copies to your local *branch*
- After a commit, use “git push” to copy the updated *branch* to GitHub
  - If you pushed changes from another local repository, use “git pull” to update
  - To replace local changes with the last committed file version, “git checkout”



# Git Basics

## “git status”

- “git status” tells you about files in the current directory
  - “fatal: Not a git repository” means you must “git clone” or “cd ~/cse1325”
  - “nothing to commit” means the repository is up to date – good work!
  - “untracked files” lists files that git is ignoring – add using “git add <file>”
  - “changes not staged for commit” lists files git is watching, but WILL NOT add to the repository next commit – to add them all, use “get add -u” (-u stands for “update”)
  - “changes staged for commit” are the files that git WILL add to the repository next commit because you typed “get add” earlier – to remove a file from the list, use “git reset <file>” (next slide)

```
student@cse1325:~/test$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   hello.cpp
    deleted:   hello1.png

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hello.png

Untracked files:
  (use "git add <file>..." to include in what will be committed)

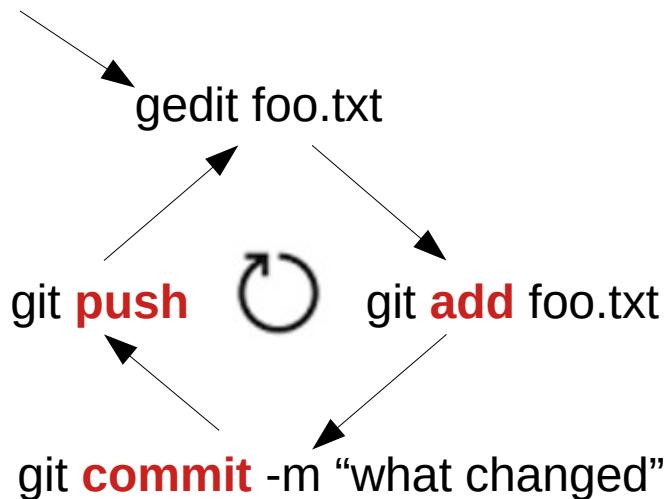
    a.out
    hello2.png

student@cse1325:~/test$
```

# Git Basics

## “git add”

- “git add <file>” tells git to add <file> to the *stage*, and thus to the local *branch* next commit (covered on the next slide)
  - If git wasn’t watching them before, it is now
  - Regardless, next commit the current file’s contents will update git’s local branch of your repository
  - You can “undo” an add with “git reset <file>”



```
ricegf@pluto:~/dev/cpp/201701/P6/fc$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Makefile

no changes added to commit (use "git add" and/or "git commit -a")
ricegf@pluto:~/dev/cpp/201701/P6/fc$ git add Makefile
ricegf@pluto:~/dev/cpp/201701/P6/fc$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   Makefile
```

## Git Basics

# “git commit -m”

- “git commit -m <message>” puts added files into git
  - Add a message in single quotes after -m
    - Keep it short – this will help you find changes later
    - For homework P01 full credit, I would write '**P01 full credit first draft**'

# Git Basics

## “git log”

- “git log” lists all commits

```
ricegf@antares:~/dev/cse1325-prof$ git log
commit 6f518d6a983ea9b7d63c7cd8c63ba7efe22e5e4a (HEAD -> main, origin/main, origin/HEAD)
Merge: 6cd0cac d643257
Author: Professor Rice <george.rice@uta.edu>
Date:   Wed Oct 30 23:08:35 2024 -0500

    Merge branch 'main' of https://github.com/prof-rice/cse1325-prof

commit 6cd0cac3b646bb6b8580ec67db6438cf3380c3d8
Author: Professor Rice <george.rice@uta.edu>
```

- This bash command permanently defines “git lg” command

```
git config --global alias.lg "log --color --graph --date=format:'%Y-%m-%d %H:%M:%S' --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset
%s %Cgreen(%cd) %C(bold blue)<%an>%Creset' --abbrev-commit"
```

```
ricegf@antares:~/dev/cse1325-prof$ git config --global alias.lg "log --color --graph --date=format:'%Y-%m-%d %H:%M:%S'
ricegf@antares:~/dev/cse1325-prof$ git lg
* 6f518d6 - (HEAD -> main, origin/main, origin/HEAD) Merge branch 'main' of https://github.com/prof-rice/cse1325-pr
| \
| * d643257 - P08 suggested solution (2024-10-29 08:01:21) <Professor Rice>
| * d2ed54f - P08 suggested solution (2024-10-29 07:57:33) <Professor Rice>
* | 6cd0cac - Lecture 20 code (2024-10-30 23:08:28) <Professor Rice>
| /
* 42d4c62 - Lecture 19 code (2024-10-28 10:00:06) <Professor Rice>
* f5db177 - Exam #2 suggested solution (2024-10-23 17:05:24) <Professor Rice>
* b2b9994 - Lecture 18 code (2024-10-23 17:03:03) <Professor Rice>
* 3ecee17 - Merge branch 'main' of https://github.com/prof-rice/cse1325-prof (2024-10-21 16:28:03) <Professor Rice>
| \
| * b557435 - P07 suggested solution (2024-10-15 07:54:34) <Professor Rice>
* | c8762f1 - Lecture 17 code (2024-10-21 16:27:59) <Professor Rice>
```

# Git Basics

## “git push”

- “git push” updates GitHub with all of your local repository commits since the last push
  - You may be asked for your GitHub login / PAT
  - It’s a good idea to push after *every* commit

```
student@cse1325:~/cse1325/01$ mkcd extreme bonus
student@cse1325:~/cse1325/01/extreme_bonus$ e hello.cpp
student@cse1325:~/cse1325/01/extreme_bonus$ g++ --std=c++17 hello.cpp
student@cse1325:~/cse1325/01/extreme_bonus$ ./a.out
Hello, student!
student@cse1325:~/cse1325/01/extreme_bonus$ git add hello.cpp
student@cse1325:~/cse1325/01/extreme_bonus$ git commit -m "Added extreme bonus"
[master 394fd97] Added extreme bonus
 1 file changed, 4 insertions(+)
  create mode 100644 01/extreme_bonus/hello.cpp
student@cse1325:~/cse1325/01/extreme_bonus$ git push
Counting objects: 5, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 521 bytes | 521.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/prof-rice/cse1325.git
  8d345da..394fd97  master -> master
student@cse1325:~/cse1325/01/extreme_bonus$
```



# Git Basics

## “git pull”

- “git pull” downloads all commits from GitHub since your last pull
  - Changes you pushed from another machine
  - Requirements and suggested solutions from the prof or TAs
- If git reports conflicts, resolve them
  - Your prof or TA can help

```
student@cse1325:~/cse1325$ ls
01 README.md
student@cse1325:~/cse1325$ git pull
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 22 (delta 3), reused 22 (delta 3), pack-reused 0
Unpacking objects: 100% (22/22), done.
From https://github.com/prof-rice/cse1325
  394fd97..90d08c3 master      -> origin/master
Updating 394fd97..90d08c3
Fast-forward
 01-ss/P1_Requirements.pdf      | Bin 0 -> 351025 bytes
 01-ss/P1_Suggested_Solution.pdf | Bin 0 -> 2024306 bytes
 01-ss/bonus/Makefile           | 8 ++++++++
 01-ss/bonus/hello.cpp          | 8 ++++++++
 01-ss/bonus/hello.png          | Bin 0 -> 124462 bytes
 01-ss/bonus/hello_getline.cpp   | 9 ++++++++
 01-ss/extreme_bonus/Makefile    | 8 ++++++++
 01-ss/extreme_bonus/hello.cpp   | 5 +++++
 01-ss/extreme_bonus/hello1.png  | Bin 0 -> 76339 bytes
 01-ss/extreme_bonus/hello2.cpp  | 11 ++++++++
 01-ss/extreme_bonus/hello2.png  | Bin 0 -> 173244 bytes
 01-ss/extreme_bonus/hello3.png  | Bin 0 -> 31796 bytes
 01-ss/extreme_bonus/hello4.png  | Bin 0 -> 31281 bytes
 01-ss/full_credit/Makefile      | 7 ++++++
 01-ss/full_credit/hello.cpp     | 5 +++++
 01-ss/full_credit/hello.png     | Bin 0 -> 54453 bytes
16 files changed, 61 insertions(+)
create mode 100644 01-ss/P1_Requirements.pdf
create mode 100644 01-ss/P1_Suggested_Solution.pdf
create mode 100644 01-ss/bonus/Makefile
create mode 100644 01-ss/bonus/hello.cpp
create mode 100644 01-ss/bonus/hello.png
create mode 100644 01-ss/bonus/hello_getline.cpp
create mode 100644 01-ss/extreme_bonus/Makefile
create mode 100644 01-ss/extreme_bonus/hello.cpp
create mode 100644 01-ss/extreme_bonus/hello1.png
create mode 100644 01-ss/extreme_bonus/hello2.cpp
create mode 100644 01-ss/extreme_bonus/hello2.png
create mode 100644 01-ss/extreme_bonus/hello3.png
create mode 100644 01-ss/extreme_bonus/hello4.png
create mode 100644 01-ss/full_credit/Makefile
create mode 100644 01-ss/full_credit/hello.cpp
create mode 100644 01-ss/full_credit/hello.png
student@cse1325:~/cse1325$ ls
01 01-ss README.md
student@cse1325:~/cse1325$
```

# [rejected] !!! Pull then Push

- If a push is rejected
  - Pull first
  - Push again!
- If that fails, clone again!  
(see next slide)

```
ricegf@antares:~/dev/202108/QB/study_sheet$ git add -u
ricegf@antares:~/dev/202108/QB/study_sheet$ git commit -m 'E2 study sheet update'
[main 6523776] E2 study sheet update
 3 files changed, 78 insertions(+), 52 deletions(-)
  rewrite QB/study_sheet/widgets.png (98%)
ricegf@antares:~/dev/202108/QB/study_sheet$ git push ↙
To https://github.com/prof-rice/202108.git
  ! [rejected]          main -> main (fetch first)
error: failed to push some refs to 'https://github.com/prof-rice/202108.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
ricegf@antares:~/dev/202108/QB/study_sheet$ git pull ↙
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 9 (delta 6), reused 9 (delta 6), pack-reused 0
Unpacking objects: 100% (9/9), 2.25 KiB | 1.13 MiB/s, done.
From https://github.com/prof-rice/202108
      6fa3989..fbcd158  main      -> origin/main
Merge made by the 'recursive' strategy.
 P07-JADE/tags/Makefile           |  2 +-
 P07-JADE/tags/sprint2.tags       | 81 ++++++-----+
 P08-JADE/full-credit/Untitled.jade | 22 ++++++-----
 3 files changed, 67 insertions(+), 38 deletions(-)
ricegf@antares:~/dev/202108/QB/study_sheet$ git push ↙
Enumerating objects: 21, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 114.86 KiB | 2.17 MiB/s, done.
Total 9 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/prof-rice/202108.git
  fbc158..996fc66  main -> main
ricegf@antares:~/dev/202108/QB/study_sheet$ █
```



Bookmark This Slide!

# [rejected] !!! – Clone Again

- **Rename** your cse1325 directory to cse1325-bad.
- **Clone** your repository again, which creates a new working cse1325 directory.
- **Copy** the files you want to push from cse1325-bad into cse1325.
- In cse1325, **add, commit, & push** the files. Life is good. Back to the homework!
- When you're sure you have all the files you need from cse1325-bad, **delete** it like a bad dream.

```
ricegf@antares:~$ lt cse1325/
Permissions Size User Date Modified Name
drwxrwxr-x   - ricegf 07-20 17:21 cse1325
drwxr-xr-x   - ricegf 07-20 17:21 P01
drwxr-xr-x   - ricegf 07-20 17:21
.rw-rw-r--  287 ricegf 07-20 17:21
.rw-rw-r-- 132k ricegf 07-20 17:21
.rw-rw-r-- 136k ricegf 07-20 17:21
drwxr-xr-x   - ricegf 07-20 17:21
.rw-rw-r-- 174 ricegf 07-20 17:21
.rw-rw-r-- 74k ricegf 07-20 17:21
.rw-rw-r-- 115k ricegf 07-20 17:21
.rw-rw-r-- 73k ricegf 07-20 17:21
.rw-rw-r-- 60k ricegf 07-20 17:21
drwxr-xr-x   - ricegf 07-20 17:21
.rw-rw-r-- 131 ricegf 07-20 17:21
.rw-rw-r-- 64k ricegf 07-20 17:21
ricegf@antares:~$ mv cse1325/ cse1325-bad/
ricegf@antares:~$ git clone https://github.com/prof-rice/cse1325.git
Cloning into 'cse1325'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 10 (delta 1), reused 5 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), 13.02 KiB | 459.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
ricegf@antares:~$ lt cse1325
Permissions Size User Date Modified Name
drwxrwxr-x   - ricegf 07-20 17:22 cse1325
.rw-rw-r-- 35k ricegf 07-20 17:22 LICENSE
drwxrwxr-x   - ricegf 07-20 17:22 P01
drwxrwxr-x   - ricegf 07-20 17:22
.rw-rw-r-- 13 ricegf 07-20 17:22 full_credit
.rw-rw-r-- 9 ricegf 07-20 17:22 Hello.java
ricegf@antares:~$ # copy important files from cse1325-bad to cse1325
```

Your ID here!

# The GitHub Web Option

- If your bash ‘git push’ command fails, upload files using the web interface
  - Select Add file > Create new file to create a new folder / file (such as P03/full\_credit)
  - Change to new folder and select Upload files

## NOTE

GitHub combines 2 or more otherwise empty folders into a single line on web!

The screenshot shows a GitHub repository page for user 'cse1325'. The repository name is 'P08/full\_credit'. The main content area displays several files: 'Makefile.mak', 'hello\_all.cpp', 'hello\_all.png', and 'hello\_me.cpp'. In the top right, there's a red arrow pointing to the 'Upload files' button in a dropdown menu. Another red arrow points from the text 'Second, change to new folder and upload your files' to the same 'Upload files' button. A large red circle highlights the 'Upload files' button. At the bottom of the page, a red box contains the text: 'For emergency use ONLY! Learn to use bash git – it's MUCH faster and may be on the exam!'.

For emergency use ONLY!  
Learn to use bash git –  
it's MUCH faster  
and may be on the exam!

Second, change to new folder  
and upload your files

First create a file in new folder

# Where's the Standard Java Docs?

- Java has a formal specification

- <https://docs.oracle.com/javase/specs/>

- Java has **comprehensive documentation**



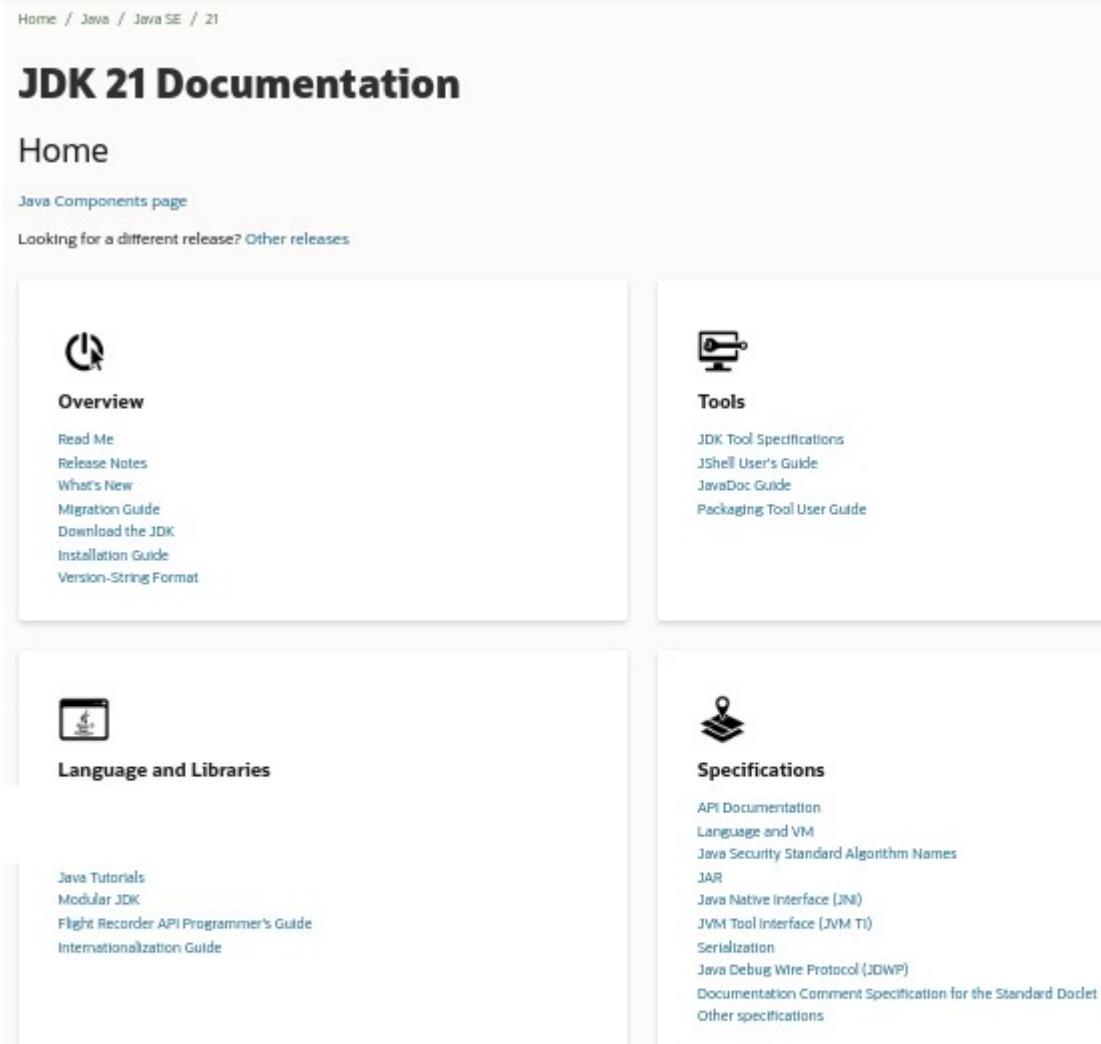
**Bookmark**

 me!

- <https://docs.oracle.com/en/java/javase/21/>
  - The Language Reference and Learning Paths (tutorials) are of primary interest to us

- Paper books are **everywhere**

- <https://codegym.cc/groups/posts/best-java-books-for-beginners>



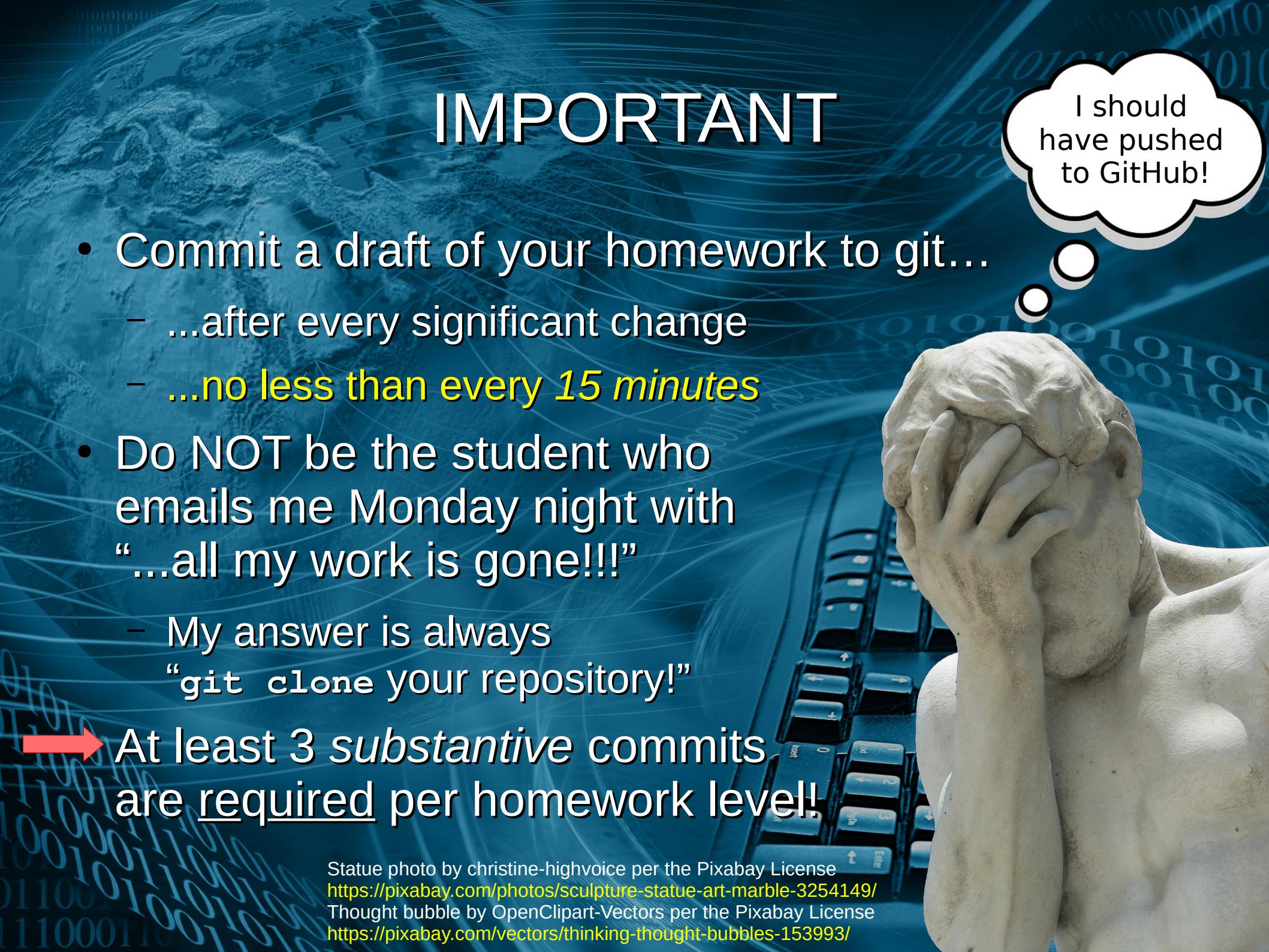
The screenshot shows the **JDK 21 Documentation** homepage. The top navigation bar includes links for Home, Java, Java SE, and the current release (21). Below the title, there are links for the Java Components page and options to look for other releases. The main content area is divided into several sections:

- Overview**: Includes links to Read Me, Release Notes, What's New, Migration Guide, Download the JDK, Installation Guide, and Version-String Format.
- Tools**: Includes links to JDK Tool Specifications, JShell User's Guide, JavaDoc Guide, and Packaging Tool User Guide.
- Language and Libraries**: Includes links to Java Tutorials, Modular JDK, Flight Recorder API Programmer's Guide, and Internationalization Guide.
- Specifications**: Includes links to API Documentation, Language and VM, Java Security Standard Algorithm Names, JAR, Java Native Interface (JNI), JVM Tool Interface (JVM TI), Serialization, Java Debug Wire Protocol (JDWP), Documentation Comment Specification for the Standard Doclet, and Other specifications.

# IMPORTANT

- Commit a draft of your homework to git...
  - ...after every significant change
  - ...no less than every *15 minutes*
- Do NOT be the student who emails me Monday night with “...all my work is gone!!!”
  - My answer is always “git clone your repository!”

➡ At least 3 *substantive* commits are required per homework level!



I should have pushed to GitHub!

# CRITICAL



FREE!

- Clone prof-rice/cse1325-prof NOW
  - ALL of the code discussed during lectures
  - Baselines for occasional upcoming assignments
  - Suggested solutions for ALL assignments
  - Suggested solutions for EVERY exam
  - Examine, modify, and *learn* it!
- From bash, use these 3 commands
  - `cd`
  - `rm -fr cse1325-prof`
  - `git clone https://github.com/prof-rice/cse1325-prof.git`

You can also access it by clicking the yellow link

# Emergency Planning

## Nedderman Hall (NH) 229

### Evacuation Routes

## In Case of Fire

Enter the  
into your

#### Report E

UTA Police  
817-21

Arlington

Non-Emer  
817-272-3

- 1) git commit
- 2) git push
- 3) exit the building



Subscribe to MavAlert  
<https://mavalert.uta.edu/>



# Get git. Learn git.



# git

--distributed-even-if-your-workflow-isnt



Search entire site...

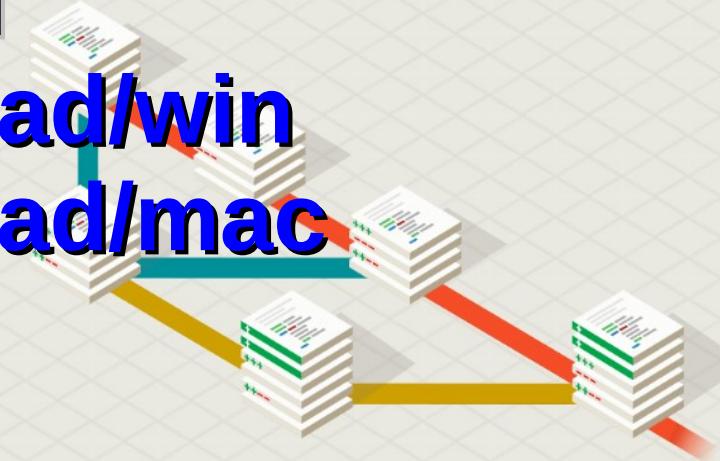
## Pre-installed in the CSE-VM

<https://git-scm.com/download/win>

<https://git-scm.com/download/mac>

`sudo apt install git-all`

`sudo yum install git-all`



## “GitHub in 5 Pages” from Canvas

<https://git-scm.com/book>

Latest source Release  
**2.22.1**  
Release Notes (2019-08-11)

Downloads for Linux

Downloads

Community

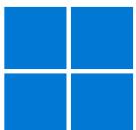
See Modules > Getting Started > GitHub in 5 Pages

# Git is Half of the Answer – The Other Half is Make Backups!



- All Technology Eventually Fails™
- Backup options (just do them all!)
  - **Duplicate** all of your valuable files periodically
    - In CSE-VM, just type “**backup**” to duplicate the current directory
    - For other bash environments, email me for how to set this up
  - Copy your Documents folder to a **flash or portable drive**
    - These usually mount in the file manager automatically
    - Keep them unmounted and off when not backing up or restoring!
  - Copy your Documents folder to the **cloud**
    - **Autosync** options like Dropbox, Google Cloud, SpiderOak, ...
    - **Manually sync** files to e.g., ownCloud, sftp, ...

More backup options for your OS!



← Click me!

Photograph of the Montparnasse derailment in Paris, France on October 22, 1895 is no longer subject to US Copyright and thus in the Public Domain



# Homework #1

- **Build and Run “hello, world” *using your own name***
  - **Bonus:** Ask the user for a name and use that
  - **Extreme Bonus:** Figure out the user’s name *without asking* and welcome them
- Complete your code on GitHub
  - Provide your URL by **Monday, August 26 at 8 am**
  - Provide your code by **Tuesday, August 27 at 8 am**
- **Details are provided on Canvas**

(HINT: These are the easiest points you'll earn all semester. Don't blow it!)

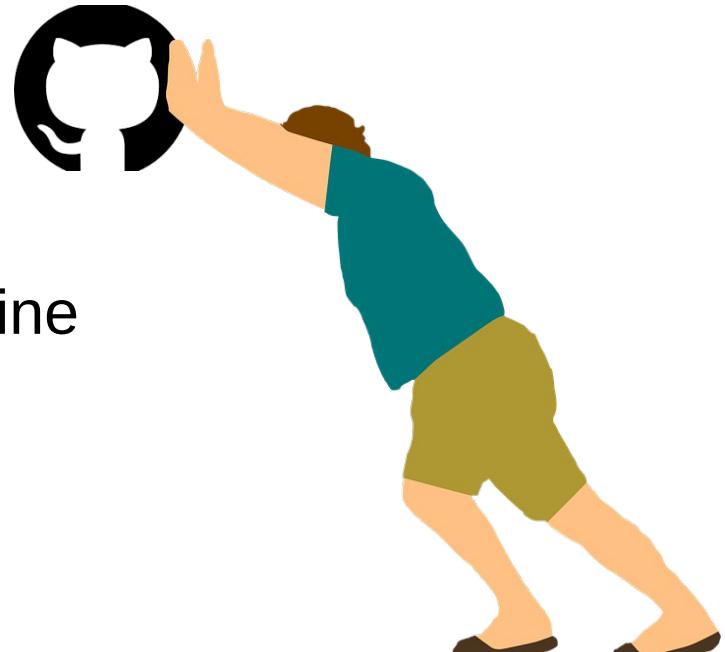
**You learn new tech by using it!**

**NOW** is the time to start – when facing a tough Java issue will be too late!



# Turn In Something

- **PUSH** your code to GitHub early and often
  - **VERIFY** your first push at <https://github.com/<your-id>/cse1325>
- **PUSH** something! We aggressively give partial credit
  - We are trying to measure knowledge gained and mastery accomplished, and those are NOT binary!
  - No homework tells us nothing, so that's what you get if you submit nothing
- **PUSH** frequently! Commit and push as often as you like
  - We grade the last push prior to the deadline
  - “Push early, push often”
  - Partial credit is better than no credit



# Why 8 a.m. on the Due Date?

- To avoid this
- Late night is traditionally “coding time”
- But most important,



we can review the Suggested Solution within minutes on the the same day you submit yours

- Faster feedback accelerates learning
- We review “Live & In Person” so you can ask questions
  - So **ASK QUESTIONS!!!**

# Asking Good Questions

- I respond to email thoroughly and almost instantly (I am told)
  - But I can only answer what I can understand

| Bad Question                                | Better Question                                                                                                                                                   |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GitHub doesn't work.                        | When I type "git push", I receive the error "Permission denied (publickey)". Screenshot attached.                                                                 |
| My code doesn't compile. Why not?           | When I type "javac hello.java", I get "error: file not found". I have pushed my code to GitHub.                                                                   |
| I don't understand the homework assignment. | You asked us to show "something that uniquely identifies the user" in the extreme bonus. Does a video of the user logging in, hacked from their video cam, count? |
| The grade the TA gave me isn't right.       | The TA wrote "missing README.md -5" in the homework feedback, but the file was pushed to GitHub before the deadline.                                              |

**If you resolve your question before I answer, tell me.**

If you waste my time today, my future answers will be shorter and less helpful.

# For Next Class

- Take the “Lecture 00 Quiz” on Canvas (5 minutes, tops!)
  - Install and configure software
    - Install VirtualBox on your laptop and load CSE-VM
    - Set up a Codespace  
See [VS Code in 5 Pages](#)
  - Test your environment! Code is provided on GitHub.
  - Read and practice [GitHub in 5 Pages](#) and [Bash in 5 Pages](#)
  - Skim the assigned reading material
    - You should already know much (but not all) of this material from CSE 1310 and 1320
- OR**
- Set up your own environment as you like and install (at least) openJDK, an editor, and git

# What We Learned Today

- Class plan, resources, and policies
  - And how to set up the required development environment
- Some (really!) basic Java
  - History and perspective with other languages
  - Online and book resources
  - How to compile and run a simple Java program
- An introduction to Version Control
  - Why you need it (yes, **you NEED git!**)
  - How to use (really!) basic git and where to learn more
  - How to backup your work (**IN ADDITION to using git**)



# BACKUP

## Early and Often!

Don't  
be me!

git commit ; git push  
THEN sleep!

