Lecture 1

Introduction

- People
- Course Overview
- Java
- Next Steps

People

Students

All students entering the course have the following background:

- Completed CSCI 134 (in Python), or
- Passed the Computer Science A AP exam (in Java), or
- Programming Experience (i.e., internship).

Faculty

Duane Bailey

- Wrote the textbook.
- Wrote the `structures` software library.
- Teaching 2 Sections + 3 Labs.
- Look for mathematical artwork in the Schow Library.
- cs.williams.edu/~bailey/ bailey@cs.williams.edu

Duane Bailey

Aaron Williams

- First time teaching this course.
- Some students will know more about Java!
- Teaching 1 Section + 2 Labs.
- Research includes combinatorial algorithms, computational complexity, video game history.
- cs.williams.edu/~aaron aw14@williams.edu

Aaron Williams

Staff

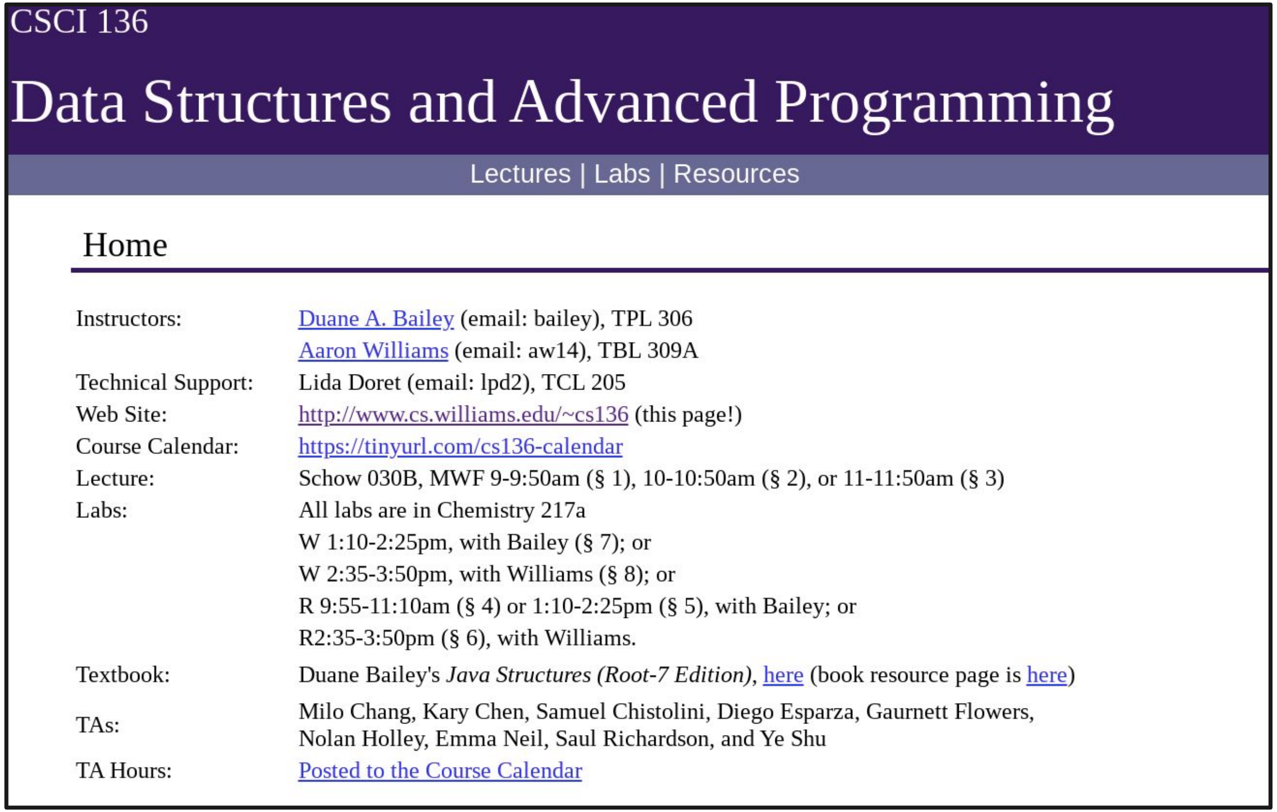
Lida Doret

- Instructional Support.
- `git` magic!

Note: Our team supports the Computing Environment (including lab computers and servers) and <u>not</u> your personal computer.

Lida Doret

Course Overview



CSCI 136

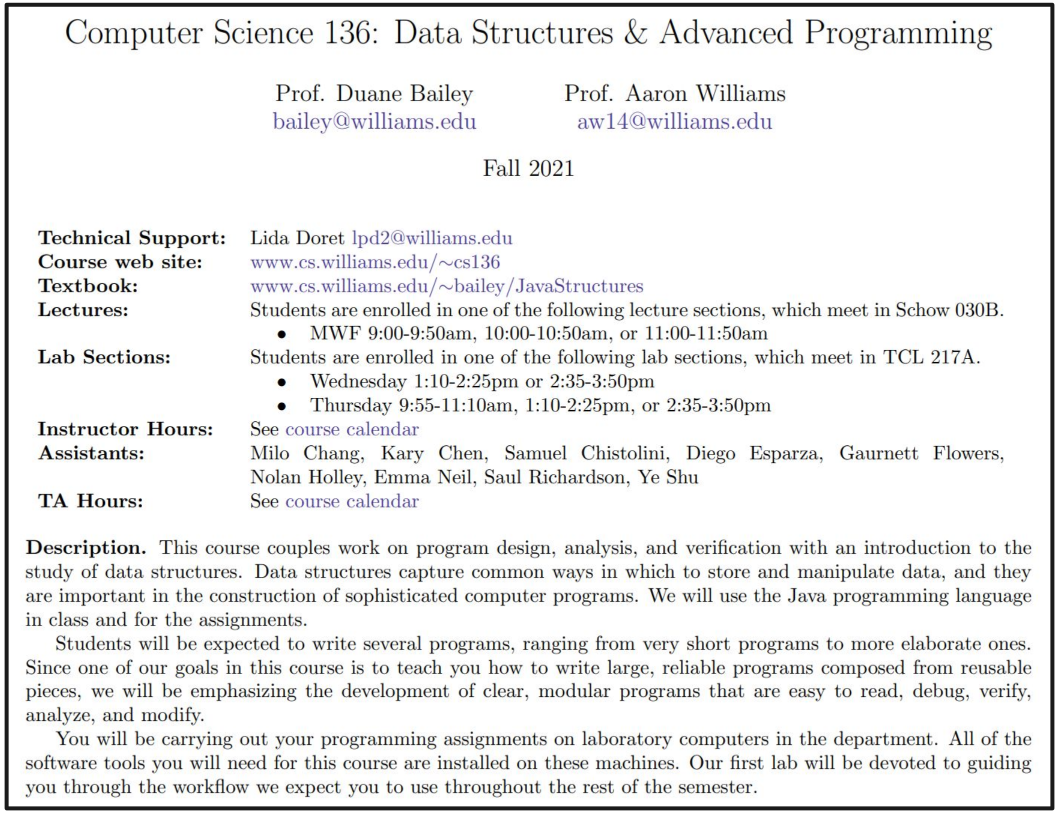
Data Structures and Advanced Programming

Lectures | Labs | Resources

Home

Instructors:	Duane A. Bailey (email: bailey), TPL 306
	Aaron Williams (email: aw14), TBL 309A
Technical Support:	Lida Doret (email: lpd2), TCL 205
Web Site:	http://www.cs.williams.edu/~cs136 (this page!)
Course Calendar:	https://tinyurl.com/cs136-calendar
Lecture:	Schow 030B, MWF 9-9:50am (§ 1), 10-10:50am (§ 2), or 11-11:50am (§ 3)
Labs:	All labs are in Chemistry 217a
	W 1:10-2:25pm, with Bailey (§ 7); or
	W 2:35-3:50pm, with Williams (§ 8); or
	R 9:55-11:10am (§ 4) or 1:10-2:25pm (§ 5), with Bailey; or
	R2:35-3:50pm (§ 6), with Williams.
Textbook:	Duane Bailey's *Java Structures (Root-7 Edition)*, here (book resource page is here)
TAs:	Milo Chang, Kary Chen, Samuel Chistolini, Diego Esparza, Gaurnett Flowers, Nolan Holley, Emma Neil, Saul Richardson, and Ye Shu
TA Hours:	Posted to the Course Calendar

Information is found on the course webpage: www.cs.williams.edu/~cs136



Computer Science 136: Data Structures & Advanced Programming

Prof. Duane Bailey bailey@williams.edu
Prof. Aaron Williams aw14@williams.edu

Fall 2021

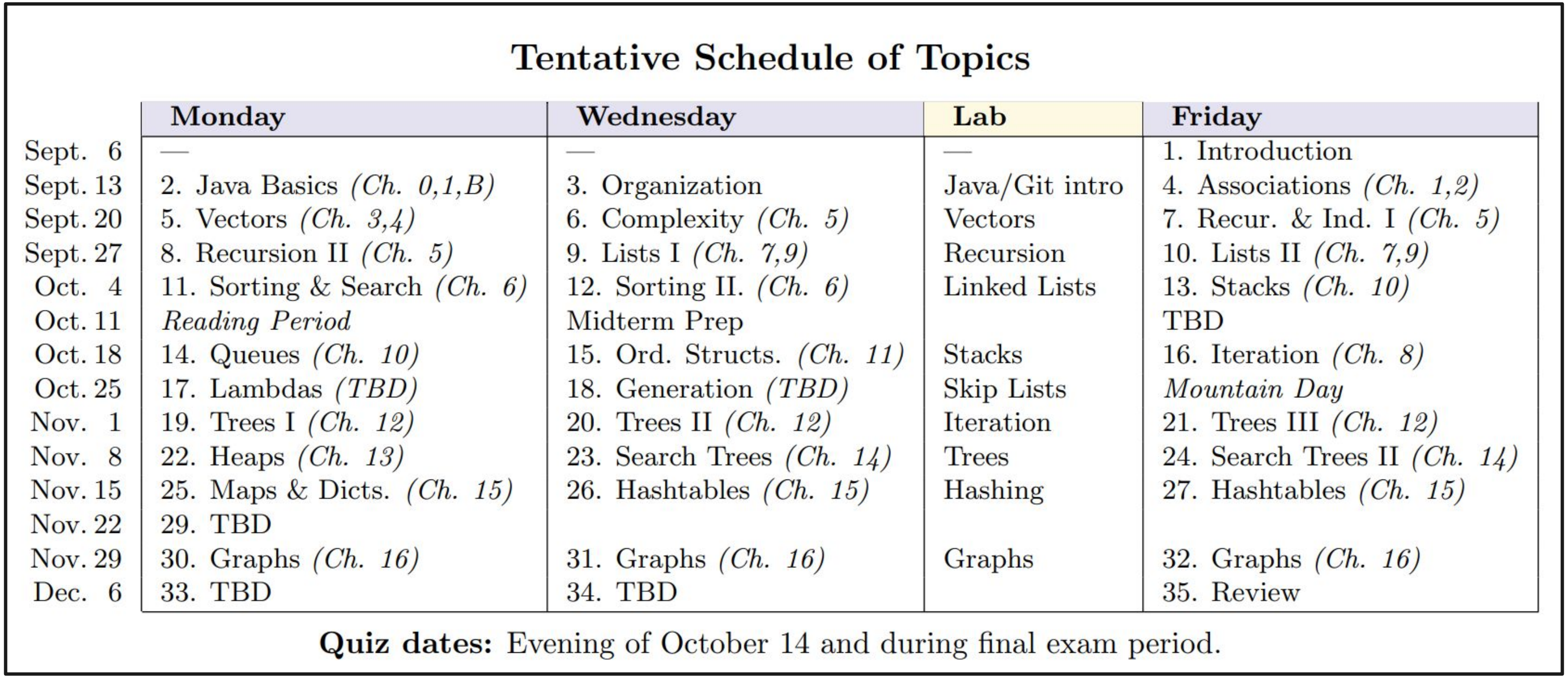
Technical Support:	Lida Doret lpd2@williams.edu
Course web site:	www.cs.williams.edu/~cs136
Textbook:	www.cs.williams.edu/~bailey/JavaStructures
Lectures:	Students are enrolled in one of the following lecture sections, which meet in Schow 030B. • MWF 9:00-9:50am, 10:00-10:50am, or 11:00-11:50am
Lab Sections:	Students are enrolled in one of the following lab sections, which meet in TCL 217A. • Wednesday 1:10-2:25pm or 2:35-3:50pm • Thursday 9:55-11:10am, 1:10-2:25pm, or 2:35-3:50pm
Instructor Hours:	See course calendar
Assistants:	Milo Chang, Kary Chen, Samuel Chistolini, Diego Esparza, Gaurnett Flowers, Nolan Holley, Emma Neil, Saul Richardson, Ye Shu
TA Hours:	See course calendar

Description. This course couples work on program design, analysis, and verification with an introduction to the study of data structures. Data structures capture common ways in which to store and manipulate data, and they are important in the construction of sophisticated computer programs. We will use the Java programming language in class and for the assignments.

Students will be expected to write several programs, ranging from very short programs to more elaborate ones. Since one of our goals in this course is to teach you how to write large, reliable programs composed from reusable pieces, we will be emphasizing the development of clear, modular programs that are easy to read, debug, verify, analyze, and modify.

You will be carrying out your programming assignments on laboratory computers in the department. All of the software tools you will need for this course are installed on these machines. Our first lab will be devoted to guiding you through the workflow we expect you to use throughout the rest of the semester.

Course syllabus: www.cs.williams.edu/~cs136/syllabus.pdf

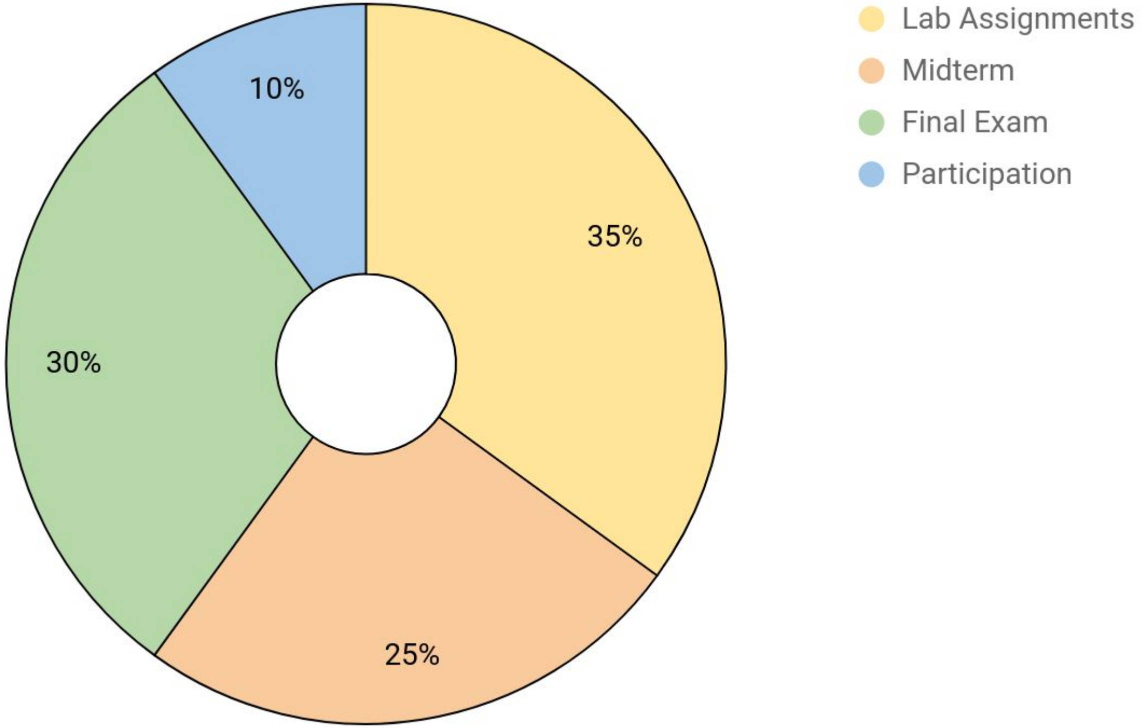


Tentative Schedule of Topics

	Monday	Wednesday	Lab	Friday
Sept. 6	—	—	—	1. Introduction
Sept. 13	2. Java Basics *(Ch. 0,1,B)*	3. Organization	Java/Git intro	4. Associations *(Ch. 1,2)*
Sept. 20	5. Vectors *(Ch. 3,4)*	6. Complexity *(Ch. 5)*	Vectors	7. Recur. & Ind. I *(Ch. 5)*
Sept. 27	8. Recursion II *(Ch. 5)*	9. Lists I *(Ch. 7,9)*	Recursion	10. Lists II *(Ch. 7,9)*
Oct. 4	11. Sorting & Search *(Ch. 6)*	12. Sorting II. *(Ch. 6)*	Linked Lists	13. Stacks *(Ch. 10)*
Oct. 11	*Reading Period*	Midterm Prep		TBD
Oct. 18	14. Queues *(Ch. 10)*	15. Ord. Structs. *(Ch. 11)*	Stacks	16. Iteration *(Ch. 8)*
Oct. 25	17. Lambdas *(TBD)*	18. Generation *(TBD)*	Skip Lists	*Mountain Day*
Nov. 1	19. Trees I *(Ch. 12)*	20. Trees II *(Ch. 12)*	Iteration	21. Trees III *(Ch. 12)*
Nov. 8	22. Heaps *(Ch. 13)*	23. Search Trees *(Ch. 14)*	Trees	24. Search Trees II *(Ch. 14)*
Nov. 15	25. Maps & Dicts. *(Ch. 15)*	26. Hashtables *(Ch. 15)*	Hashing	27. Hashtables *(Ch. 15)*
Nov. 22	29. TBD			
Nov. 29	30. Graphs *(Ch. 16)*	31. Graphs *(Ch. 16)*	Graphs	32. Graphs *(Ch. 16)*
Dec. 6	33. TBD	34. TBD		35. Review

Quiz dates: Evening of October 14 and during final exam period.

(Tentative) Course Schedule from the Syllabus.


35%
25%
30%
10%
Lab Assignments
Midterm
Final Exam
Participation

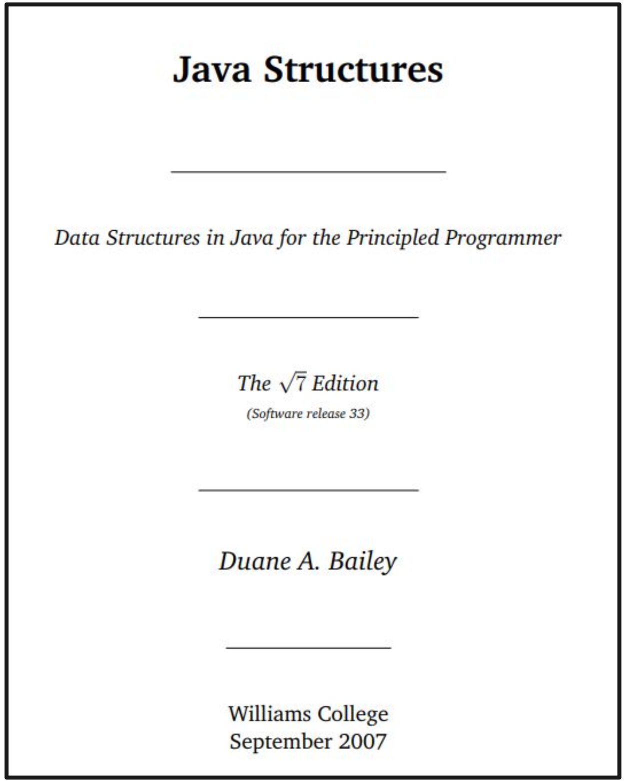
Grading.

Mac Lab
TCL 217A
392781

Unix Lab
TCL 312
#392781

Servers

Computing Environment.



Java Structures

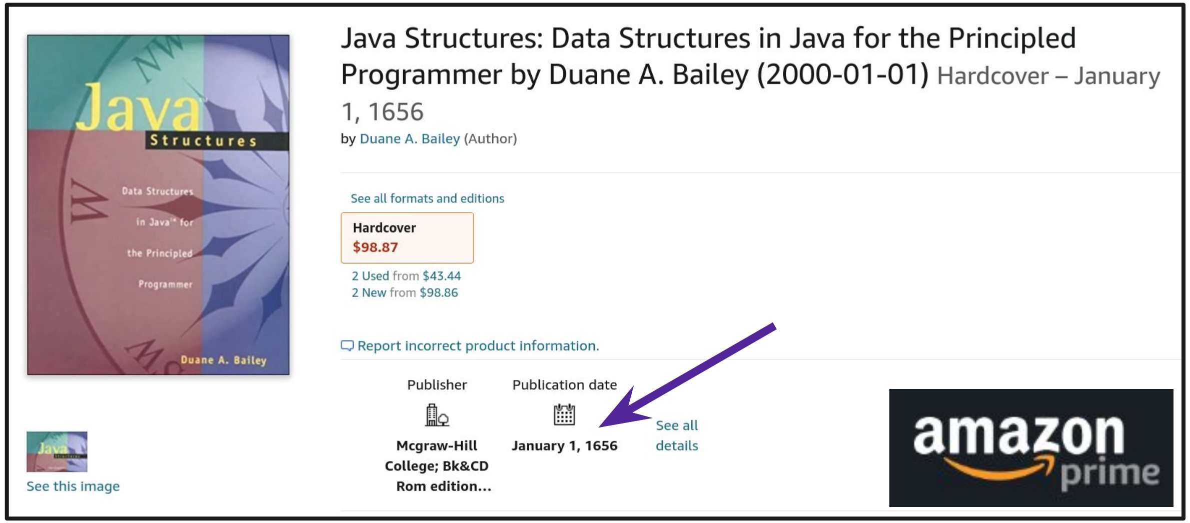
Data Structures in Java for the Principled Programmer

The $\sqrt{7}$ Edition

(Software release 33)

Duane A. Bailey

Williams College
September 2007



Java Structures: Data Structures in Java for the Principled Programmer by Duane A. Bailey (2000-01-01) Hardcover – January 1, 1656

by Duane A. Bailey (Author)

See all formats and editions

Hardcover
$98.87

2 Used from $43.44
2 New from $98.86

Report incorrect product information.

Publisher	Publication date
Mcgraw-Hill College; Bk&CD Rom edition...	January 1, 1656

See all details

Textbook.

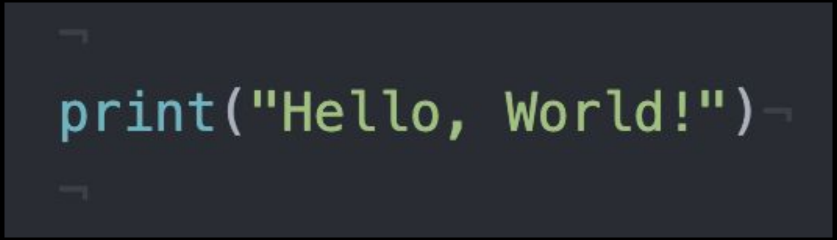
- We will use an electronic version.
- Older (but not that old!) versions have been printed in hardcover.

Succeeding in the Course

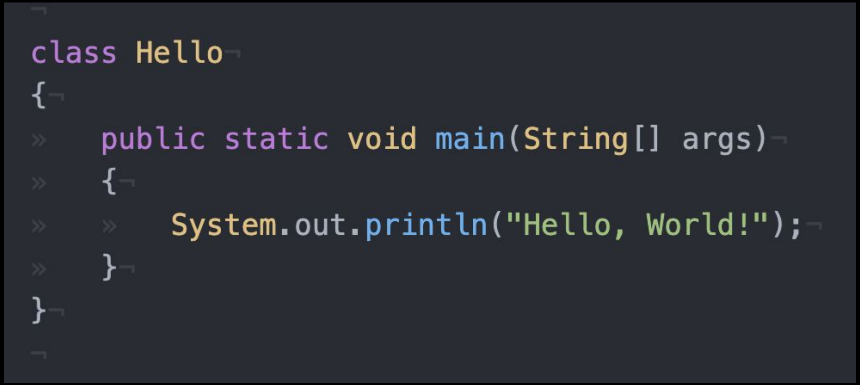
Below are some steps that will ensure that you are successful in the course.

- **Read** the textbook.
  - The readings are given in the course schedule (see previous slide).
  - Try to understanding everything – don't skim.
- Be **engaged** during lecture.
  - Refocus yourself when necessary.
  - Sit closer to the front if that helps.
- **Prepare** for the labs.
  - Start reading and working on the lab before your lab section. Handouts are available on Tuesdays.
  - Work through difficult parts of the lab during lab time.
  - Continue working on the lab when the lab time is finished.
- Use the course and college **resources**.
  - TA hours.
  - Instructor office hours.
  - Conversations with other students.
  - Check syllabus for additional support.
- Have **fun**!

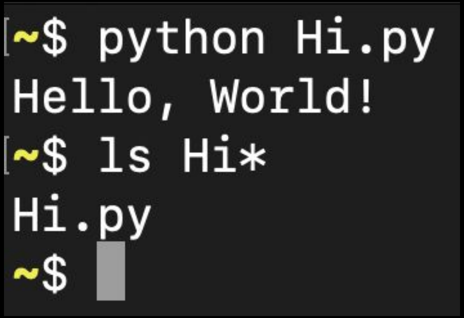
Java



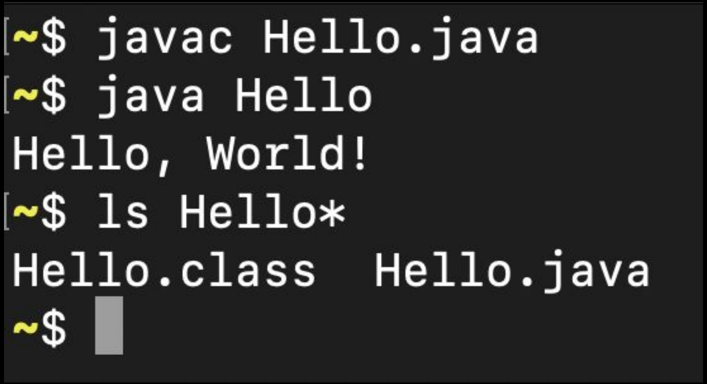
```python
print("Hello, World!")
```



```java
class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```

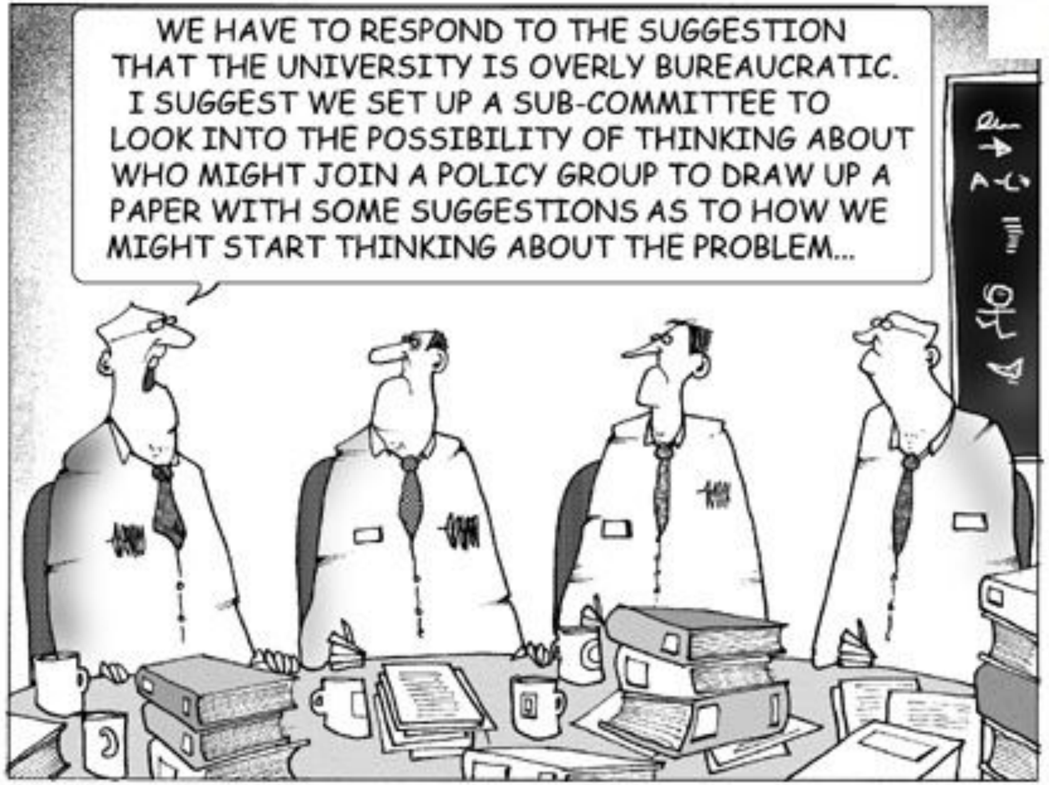


```
~$ python Hi.py
Hello, World!
~$ ls Hi*
Hi.py
~$
```



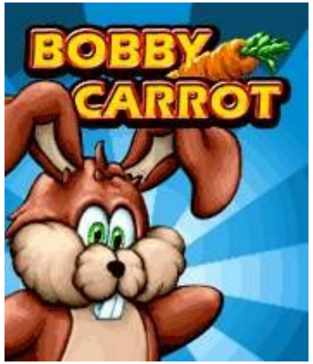
```
~$ javac Hello.java
~$ java Hello
Hello, World!
~$ ls Hello*
Hello.class  Hello.java
~$
```

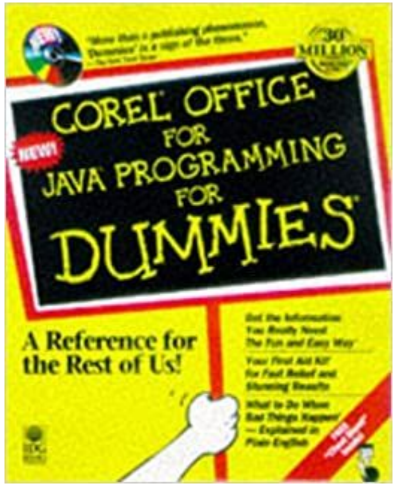
Saying hello in Python (left) and Java (right).


WE HAVE TO RESPOND TO THE SUGGESTION THAT THE UNIVERSITY IS OVERLY BUREAUCRATIC. I SUGGEST WE SET UP A SUB-COMMITTEE TO LOOK INTO THE POSSIBILITY OF THINKING ABOUT WHO MIGHT JOIN A POLICY GROUP TO DRAW UP A PAPER WITH SOME SUGGESTIONS AS TO HOW WE MIGHT START THINKING ABOUT THE PROBLEM...

Java often feels like a bureaucratic language.

- There are advantages and disadvantages to this style.




COREL OFFICE FOR JAVA PROGRAMMING FOR DUMMIES
NEW!
A Reference for the Rest of Us!

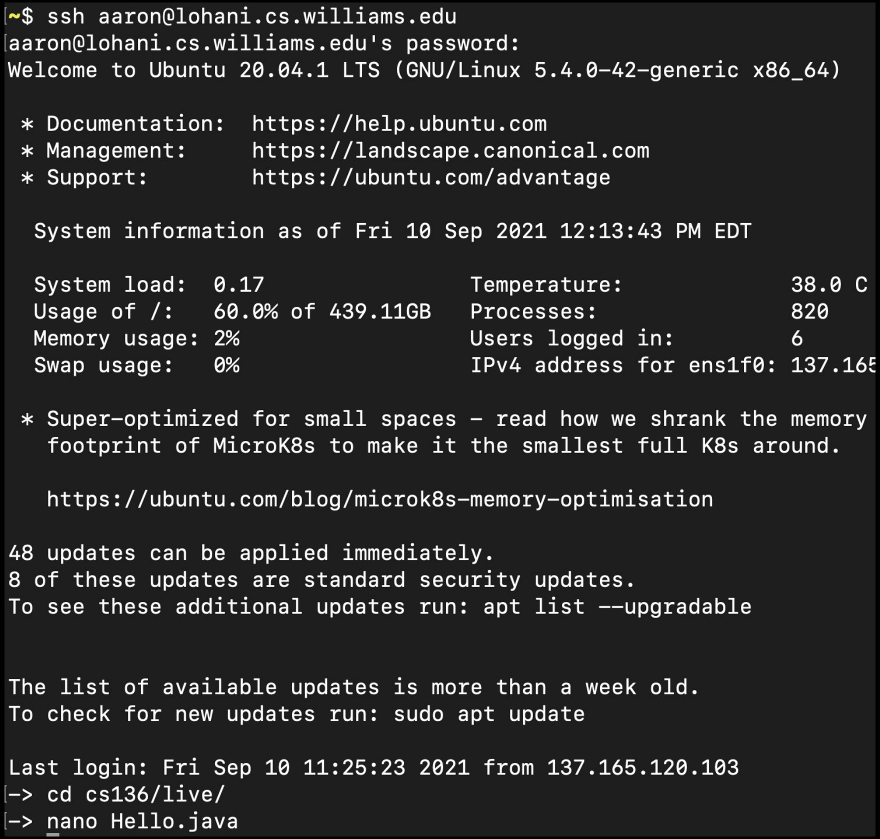
I am <u>personally</u> excited to learn more about Java for many reasons:

- *Retrogame archeology*. Java is the language used in many pre-iPhone mobile devices.
- *History*. As an undergraduate student in the late-1990s, I had three internships at Corel.

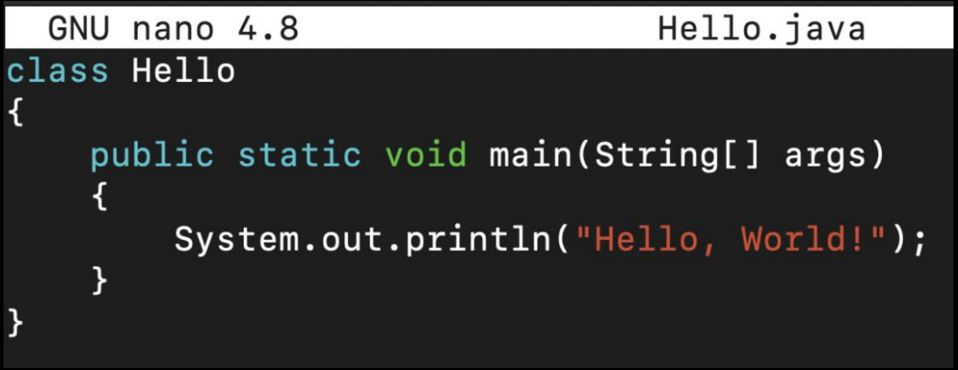
Hello, World!
Our first program

Live Coding: Hello, World! (`Hello.java`)

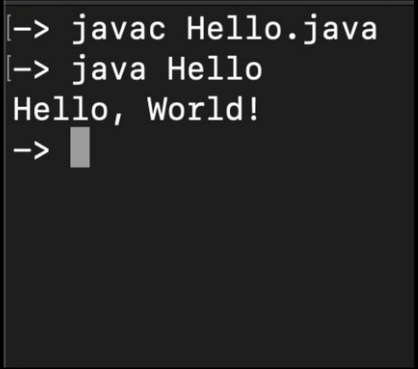
- Write a Java program that prints something to the console.
- Compile and run the program from the command-line.
- Discuss the syntax and various keywords.
  - `class`, `public`, `static`, `void`, `main`, `System.out`, `{}`, `;`



```
~$ ssh aaron@lohani.cs.williams.edu
aaron@lohani.cs.williams.edu's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 10 Sep 2021 12:13:43 PM EDT

  System load:  0.17               Temperature:            38.0 C
  Usage of /:   60.0% of 439.11GB  Processes:              820
  Memory usage: 2%                 Users logged in:        6
  Swap usage:   0%                 IPv4 address for ens1f0: 137.165

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

48 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Fri Sep 10 11:25:23 2021 from 137.165.120.103
-> cd cs136/live/
-> nano Hello.java
```



```
  GNU nano 4.8                    Hello.java
class Hello
{
    public static void main(String[] args)
    {
        System.out.println("Hello, World!");
    }
}
```



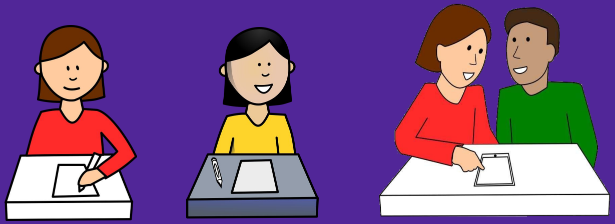
```
-> javac Hello.java
-> java Hello
Hello, World!
->
```

Live coding from class.

Challenge: Recall the Program

Try to recreate Hello, World!

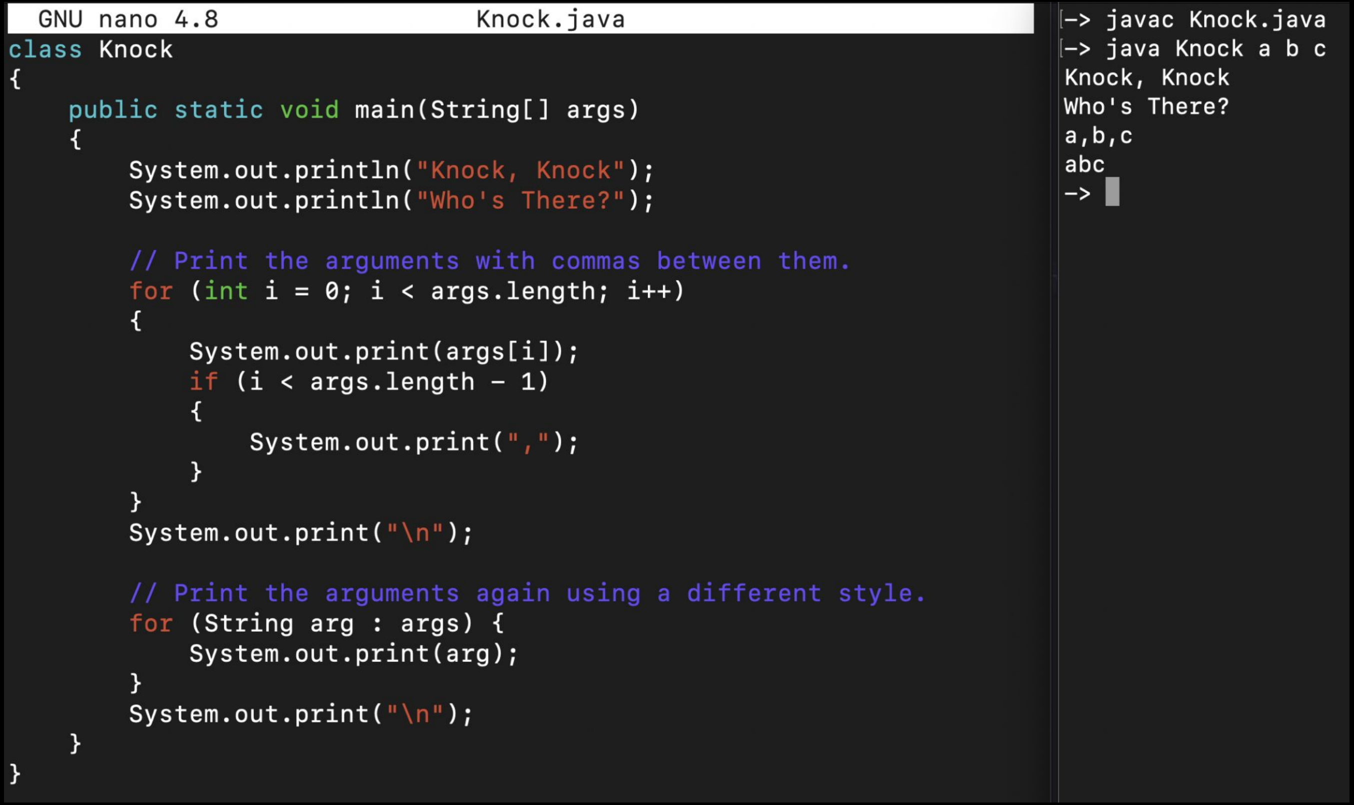
- Write the program down on paper, or on your computer.
- Do not refer to any of your notes.
- (Time Permitting) Compare your program with a neighbor.



Knock, Knock:
Command-Line Arguments

Live Coding: Knock, Knock (`Knock.java`)

- Write a Java program that reads and prints command-line arguments.
- Use an if statement to put a comma between the arguments.
- Use two different types of for loops to iterate over the arguments.
- (Time Permitting) Discuss the `nano` editor.
- Discuss additional syntax and keywords.
    - `String, int, [], .length, :`

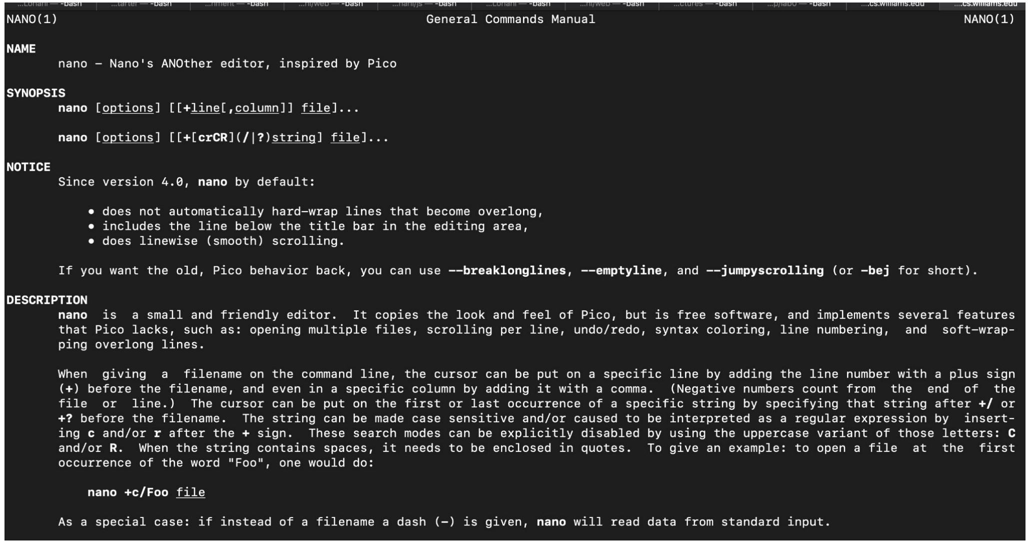


```
  GNU nano 4.8                    Knock.java
class Knock
{
    public static void main(String[] args)
    {
        System.out.println("Knock, Knock");
        System.out.println("Who's There?");

        // Print the arguments with commas between them.
        for (int i = 0; i < args.length; i++)
        {
            System.out.print(args[i]);
            if (i < args.length - 1)
            {
                System.out.print(",");
            }
        }
        System.out.print("\n");

        // Print the arguments again using a different style.
        for (String arg : args) {
            System.out.print(arg);
        }
        System.out.print("\n");
    }
}
```

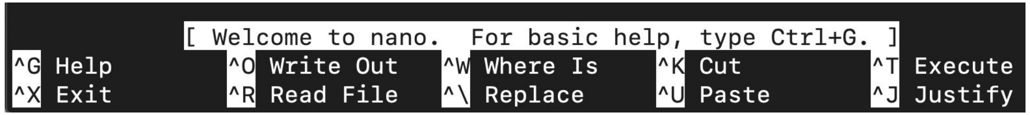
```
-> javac Knock.java
-> java Knock a b c
Knock, Knock
Who's There?
a,b,c
abc
->
```

Live coding from class (with some extra comments and printing added).

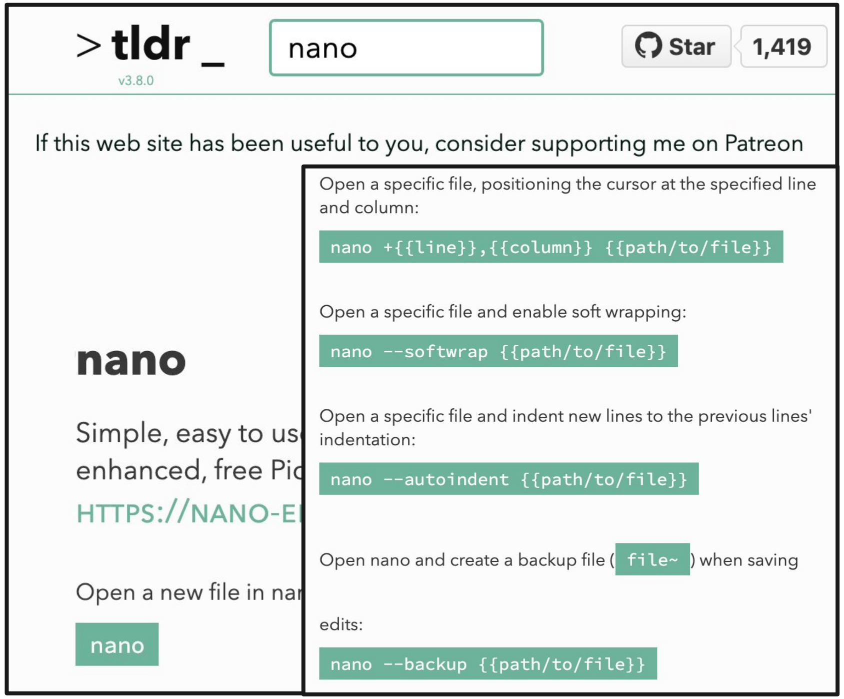
nano:
the simple terminal editor



```
NANO(1)                           General Commands Manual                           NANO(1)

NAME
       nano - Nano's ANOther editor, inspired by Pico

SYNOPSIS
       nano [options] [[+line[,column]] file]...

       nano [options] [[+[crCR](/|?)string] file]...

NOTICE
       Since version 4.0, nano by default:

           • does not automatically hard-wrap lines that become overlong,
           • includes the line below the title bar in the editing area,
           • does linewise (smooth) scrolling.

       If you want the old, Pico behavior back, you can use --breaklonglines, --emptyline, and --jumpyscrolling (or -bej for short).

DESCRIPTION
       nano  is  a small and friendly editor.  It copies the look and feel of Pico, but is free software, and implements several features
       that Pico lacks, such as: opening multiple files, scrolling per line, undo/redo, syntax coloring, line numbering,  and  soft-wrap-
       ping overlong lines.

       When  giving  a  filename on the command line, the cursor can be put on a specific line by adding the line number with a plus sign
       (+) before the filename, and even in a specific column by adding it with a comma.  (Negative numbers count from  the  end  of  the
       file  or  line.)  The cursor can be put on the first or last occurrence of a specific string by specifying that string after +/ or
       +? before the filename.  The string can be made case sensitive and/or caused to be interpreted as a regular expression by  insert-
       ing c and/or r after the + sign.  These search modes can be explicitly disabled by using the uppercase variant of those letters: C
       and/or R.  When the string contains spaces, it needs to be enclosed in quotes.  To give an example: to open a file  at  the  first
       occurrence of the word "Foo", one would do:

           nano +c/Foo file

       As a special case: if instead of a filename a dash (-) is given, nano will read data from standard input.
```



```
             [ Welcome to nano.  For basic help, type Ctrl+G. ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```



If this web site has been useful to you, consider supporting me on Patreon

nano

Simple, easy to us
enhanced, free Pic
HTTPS://NANO-E

Open a new file in nar

`nano`

Open a specific file, positioning the cursor at the specified line and column:

`nano +{{line}},{{column}} {{path/to/file}}`

Open a specific file and enable soft wrapping:

`nano --softwrap {{path/to/file}}`

Open a specific file and indent new lines to the previous lines' indentation:

`nano --autoindent {{path/to/file}}`

Open nano and create a backup file (`file~`) when saving edits:

`nano --backup {{path/to/file}}`

`nano` is one of the the simplest **terminal-based** text editors.

- Learn more: `man nano` or `tldr nano` (online version: tldr.sh) or `Ctrl+g` in the program.
- For configuration refer to the `~/.nano` folder and the `~/.nanorc` file.
- Our Unix machines have Version 4+ but the Mac machines may only have Version 2.
- Other terminal options: `emacs` or `vi(m)`. Atom is an excellent non-terminal text editor.

Next Steps

Next Steps

In this lecture we gave an overview of the course, and wrote our first Java programs.

Lecture 2: Java Basics

- Classes, objects, …

Lecture 3: Organizing Code

- Tools: git, ssh, …

Lab 0: Computing Environment

- Getting used to how our labs will work …

Lecture 4: Associations

- Our first real data structure …
- The `structure` package …

Note: Duane will be substituting for me during Lecture 2.