Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM. T 2:00 – 3:00 PM

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email: See syllabus

Supplemental Instruction: TBA

INTRODUCTION TO CS1- WEEK 1

COMPUTER SCIENCE I

OBJECTIVES

- Syllabus and Course Goals
- How to Study for This Class (And CS in general!)
- Types of Programming Languages
- An Overview of Java

HI, I'M KAITLIN! NICE TO MEET YOU

- Undergrad in Nutrition and Dietetics
- Worked as a cook then a dietetic technician
- M.S. in Computer Science here at SUNY New Paltz
- Love all things web development!
- Hobbies include making jewelry, cross stitch, baking, and annoying my 4 cats (Harry, Jerry, Earl and Gigi)

STRUCTURE OF LECTURE

- Beginning of class will start with you trying a quick review question (not handing in, but please try!!)
- Lecture
- Examples/demo going through each new concept
- Problems for you to try on your own that I will go over after a few minutes

STRUCTURE OF LAB

- Quiz that will be given at the beginning of lab. You will have 30 minutes to complete it. Lowest quiz grade will be dropped at end of semester.
- Work on your lab which will be based on the previous lecture's content. Ask the TA any questions you may have. Don't be afraid to ask for help! That is why they are there.
- Towards the end of lab, your TA will look over your work. This will be graded out of **2 points** based on participation. Meaning, if it looks like you made an effort and tried to solve the problems, you will receive the full 2 points. However, if it's obvious you did not try, you will only be given 1 point. If you don't attend lab without a valid excuse, you will receive 0 points.
- Solutions for the labs will be released on Fridays. Be sure to look over them!!

EXAMS

- There are a total of **4** exams which includes the final. Exam dates are on the syllabus. It is your duty to work your schedule around your tests.
- If you have an accommodation for extended time, you MUST schedule your exams with the DRC.
- Exams will be similar in structure to the labs. If you do well on the labs, you should do well on the exams.

TLDR:

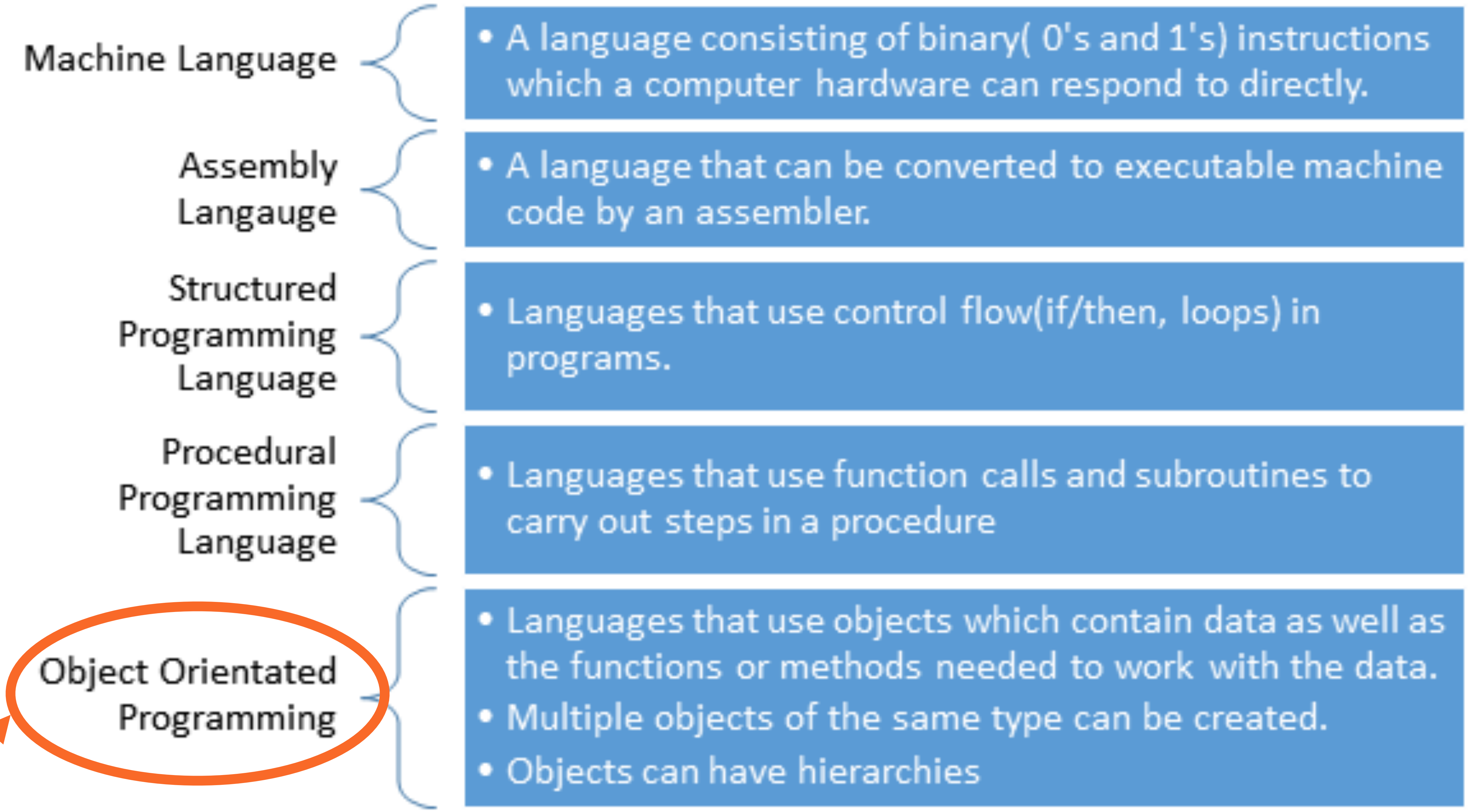
- Textbook is not mandatory, but if you want to use one, I’m referencing: ***Introduction to JAVA Programming, Comprehensive Version, 10th edition by Y. Daniel Liang***, ISBN: 978-0-13-376131-3
- Class is a mix of exams, weekly quizzes, and labs (exams are given during lecture, quizzes and labs are during lab)
- **4 Exams** - 3 Exams during the semester and a final.
- **Weekly Quizzes** - Given during the beginning of Lab. The lowest quiz grade is dropped at the end of semester. No quizzes given during exam weeks.
- **Weekly Labs** - It is expected that you complete the ENTIRE lab. It is crucial to doing well in this course. Solutions will be released at the end of the week.

HOW TO STUDY FOR COMPUTER SCIENCE

- Computer Science can be super confusing at first when learning. It is a different way of thinking that many of you may not be used to (which is okay!). **Best way to study for this class is to**:
  - Come to every lecture and ask questions whenever you are confused. PLEASE try the problems in lecture to get practice and used to writing code. Paper and pen is fine!
  - **Actually complete the labs!!** With computer science, the only way to learn is to practice, practice, practice! You will not pass this class by simply going to lectures and looking at the solutions. You actually have to try them out for your own. Studying for computer science is a lot like studying for math – by doing a lot of problems!
  - If you get stuck on a lab problem, mark it down somewhere, look over my solution and/or get an explanation from me or your TA, and leave it alone for a day or two. Revisit the question and try it again on your own.
  - Come to office hours if you need clarification on anything.
  - To study for an exam, do the lab problems again, especially any that confused you. This will force you to practice and reenforce CS concepts. You can't study by just looking at the solutions – you actually have to do it yourself.

ANY QUESTIONS?

TYPES OF PROGRAMMING LANGUAGES



Machine Language	• A language consisting of binary(0's and 1's) instructions which a computer hardware can respond to directly.
Assembly Langauge	• A language that can be converted to executable machine code by an assembler.
Structured Programming Language	• Languages that use control flow(if/then, loops) in programs.
Procedural Programming Language	• Languages that use function calls and subroutines to carry out steps in a procedure
Object Orientated Programming	• Languages that use objects which contain data as well as the functions or methods needed to work with the data. • Multiple objects of the same type can be created. • Objects can have hierarchies

What Java is and what we will be learning!

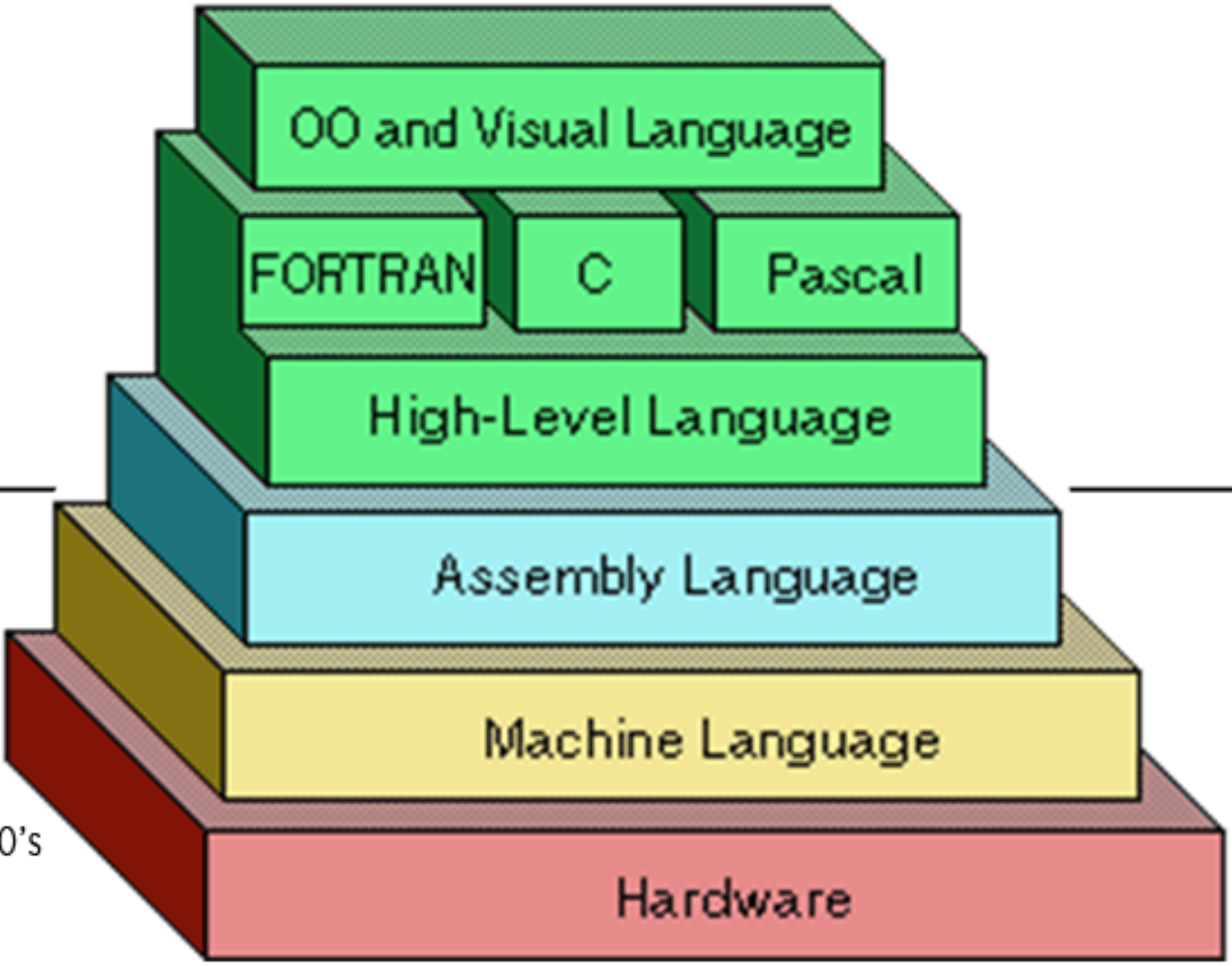
TYPES OF PROGRAMMING LANGUAGES

High Level Language

- Easy for Programmers to understand
- Contains Engilish Words

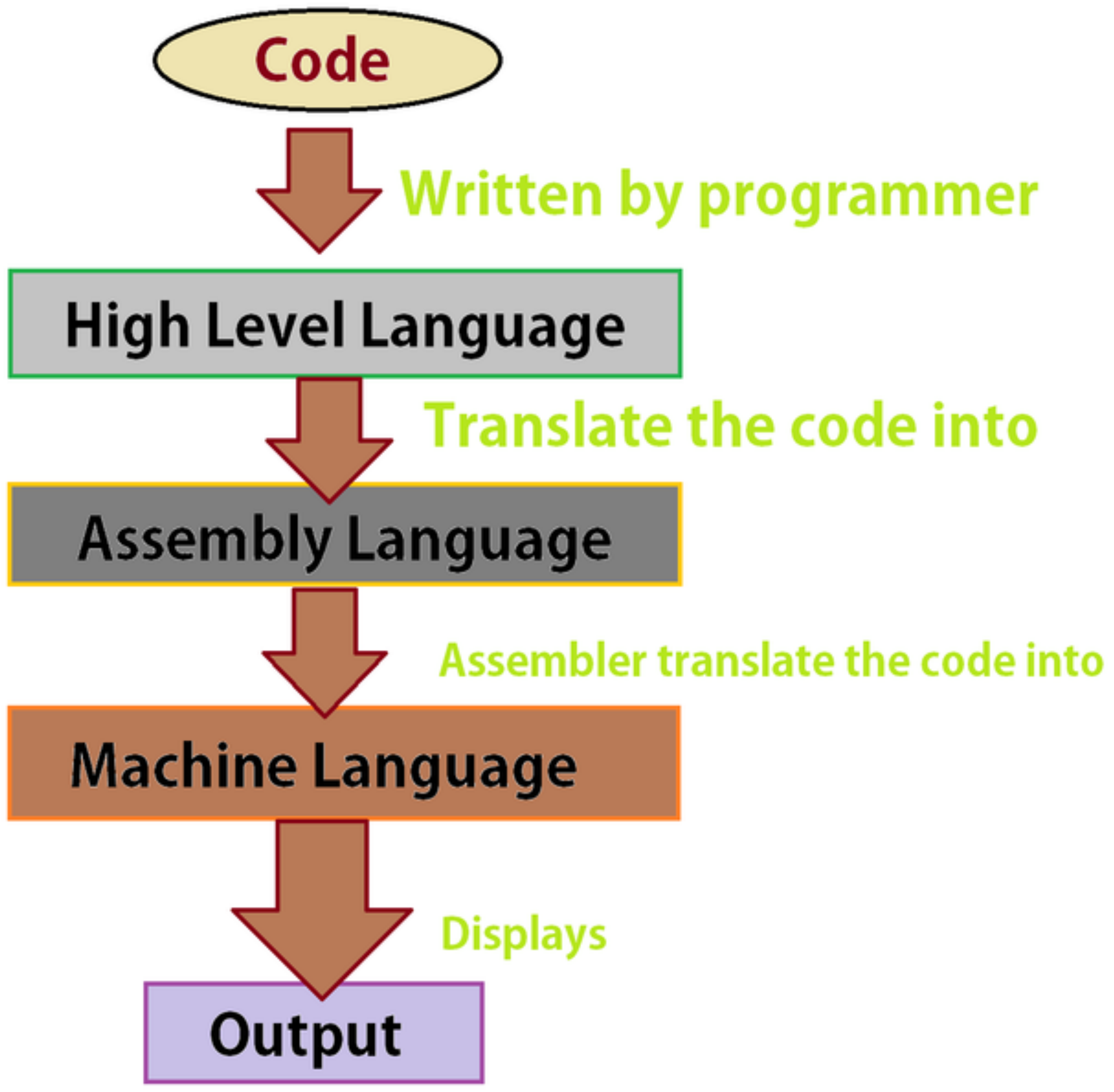
Low Level Langugae

- The computer's own Language
- Binary numbers, in 1's and 0's



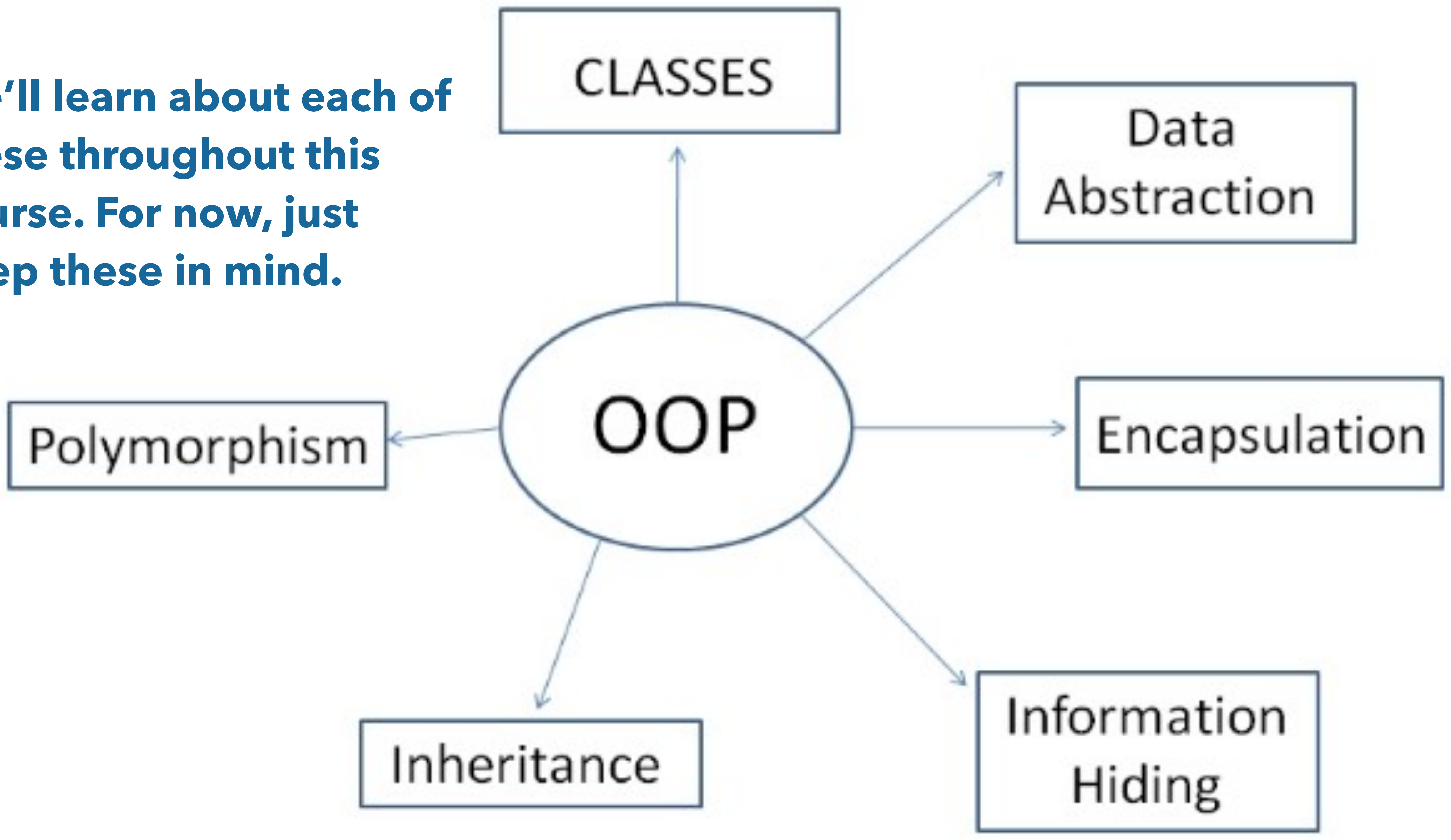
justcode.me

HOW A PROGRAM IS EXECUTED



WHAT MAKES UP AN OBJECT ORIENTED PROGRAMMING LANGUAGE (OOP)

We'll learn about each of these throughout this course. For now, just keep these in mind.



JAVA

There are many different programming languages are out there, such as Python, C++, JavaScript and Java. You'll be learning the Object Oriented Programming Language, Java!

Where is Java used?

Primarily on the Server-Side (Back End) of applications and websites:

- Minecraft
- Google.com
- Youtube.com
- Amazon.com
- LinkedIn.com

BACK END VS FRONT END: FRONT END

When you use applications, such as for example, a website like Facebook.com, you as a user sees text, pictures, and pretty colors and a clean format.

- That's the Client-Side (or **Front End**). It describes any parts that the **average user will be able to access**. This includes menus, buttons, forms, sliders, and even the fonts of the words you see on the screen.
- This is written usually with **HTML**, **CSS**, and **JavaScript**.
- **Example**: A button that says Submit that a user can click
- You will **not be learning** about HTML, CSS or JavaScript in this course.

BACK END VS FRONT END: BACK END

The **Server-Side (or Back End)** describes all of the elements that **users can't access**, at least not directly. This includes servers, databases, operating systems, and APIs, all with the purpose of supporting front-end systems.

- The front end and back end **work together** to make a complete application, such as a website, an app, or game. Java is primarily used in the back end!
- **Example**:
    1. You enter your login info on a website then **click** the submit button.
    2. The **client-side** sends your info to the **server-side** where Java may be used to access a database to see if your login credentials are correct.
    3. If correct, the back-end sends your profile page back to the client-side. If not, the back-end returns an error which is sent to the client-side which in turn is seen by you, the user.

OKAY, SO HOW DO WE USE JAVA?

- Java is **portable** meaning it is runnable/executable on all operating systems.
- All you need is to have Java installed on your computer and a text editor of your choice!
- Code is the Java program that you write. It must be saved with the file extension **.java**
    - For example, the extension of a pdf is **.pdf**, the extension of a word document is **.doc** – Java always uses **.java**

BASIC JAVA FORMAT – OUR FIRST PROGRAM

All Java programs will start with a **Class Header** that you name (in this case, we'll name this **HelloWorld**) and a **Main Method**:

```
public class HelloWorld {

    public static void main(String[ ] args) {

        System.out.println(“Hello World!”);

    }

}
```

THE CLASS HEADER

Remember how slide 14 said OOP Languages are made up of classes? Well this explains why the first line contains **class — this line is stating that this Java program is a class called HelloWorld.** HelloWorld is the **class name** that you as the programmer makes up. A class contains all of our code for this program.

```
public class HelloWorld {



}
```

public is not mandatory (we'll learn more about *public*, *private*, etc. later on)

THE CLASS HEADER

The first line should be the word **class** followed by the **class name** (HelloWorld in this example). The class name is something we choose.

Some guidelines for naming the class:

- the name should be **descriptive.** I will suggest to name them something like Q1 or Lab1Q1 etc.
- class names should start with a **capital letter**
- there should be **no** spaces or periods (.)
- The name of your Java file **MUST** be the class name! **Example**:
  - The class name is **HelloWorld** so the file should be saved as **HelloWorld.java**

THE MAIN METHOD

This will always look the same. This is **inside** the class header.

The **class** and **main method** will be the start of all of our programs for now. We will learn what each part/word means as we go through the semester.

For now, all the code we want to write will go inside the main method. The main method is where **program execution** starts from in Java. Java then follows **line by line**.

```
public static void main(String[ ] args) {



}
```

OUR CODE!

There are so many things you can write in Java, but this line of code **prints out** Hello World! — **Notice the semi-colon ;** Don't forget this! Most Java statements end with a ;

The line System.out.println(); can be used to **print/display** things to the screen.

NOTE: the s in System is capital and it is a lowercase L after print. **JAVA IS CASE SENSITIVE!!!**

```
System.out.println("Hello World!");
```

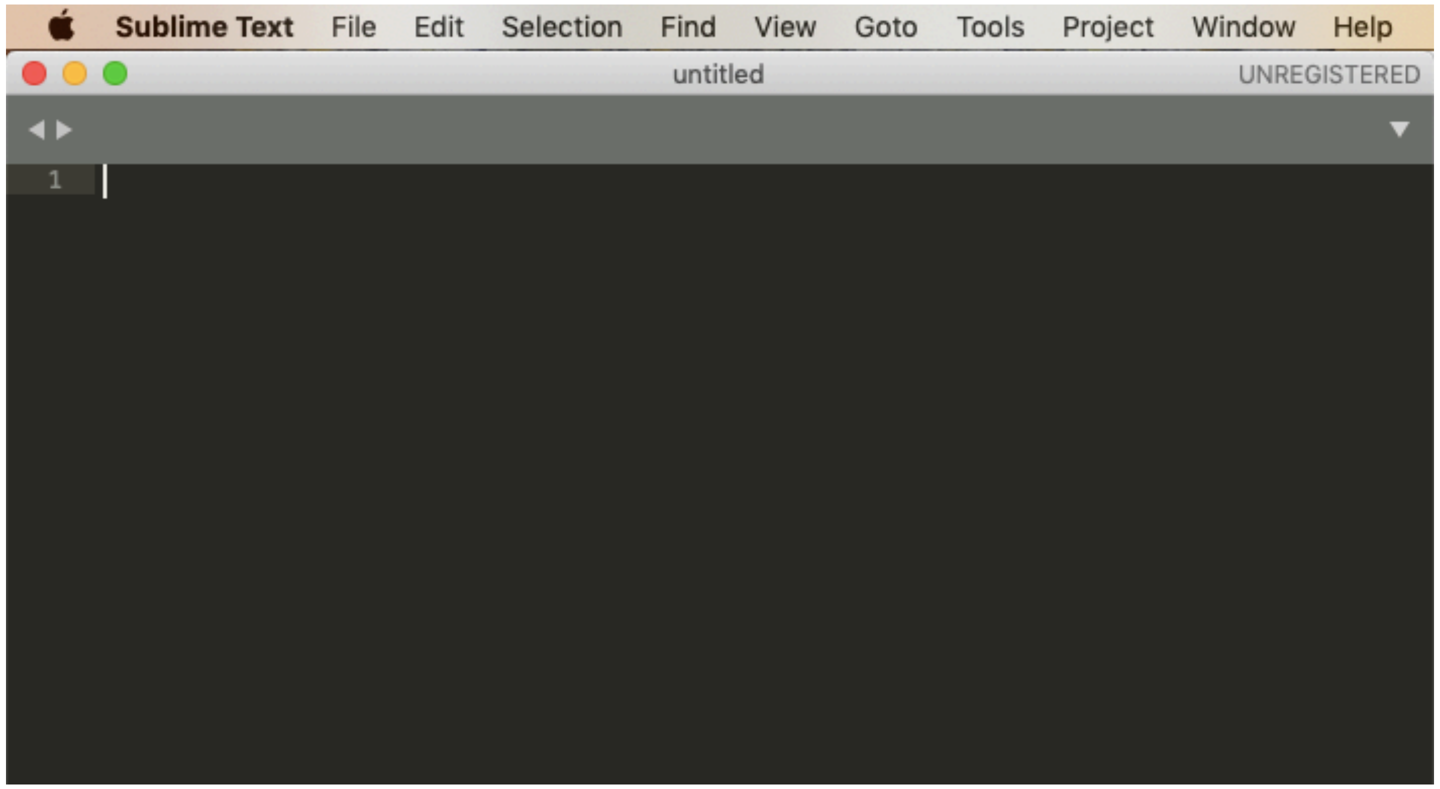
OUR FIRST PROGRAM!

- I will do a demo with you on a Windows Computer. The following slides are on a Mac. However, both are very very similar, so if you get stuck, use these as a reference or ask for help.
- Pay attention to how I do this. You will be doing this in lab!

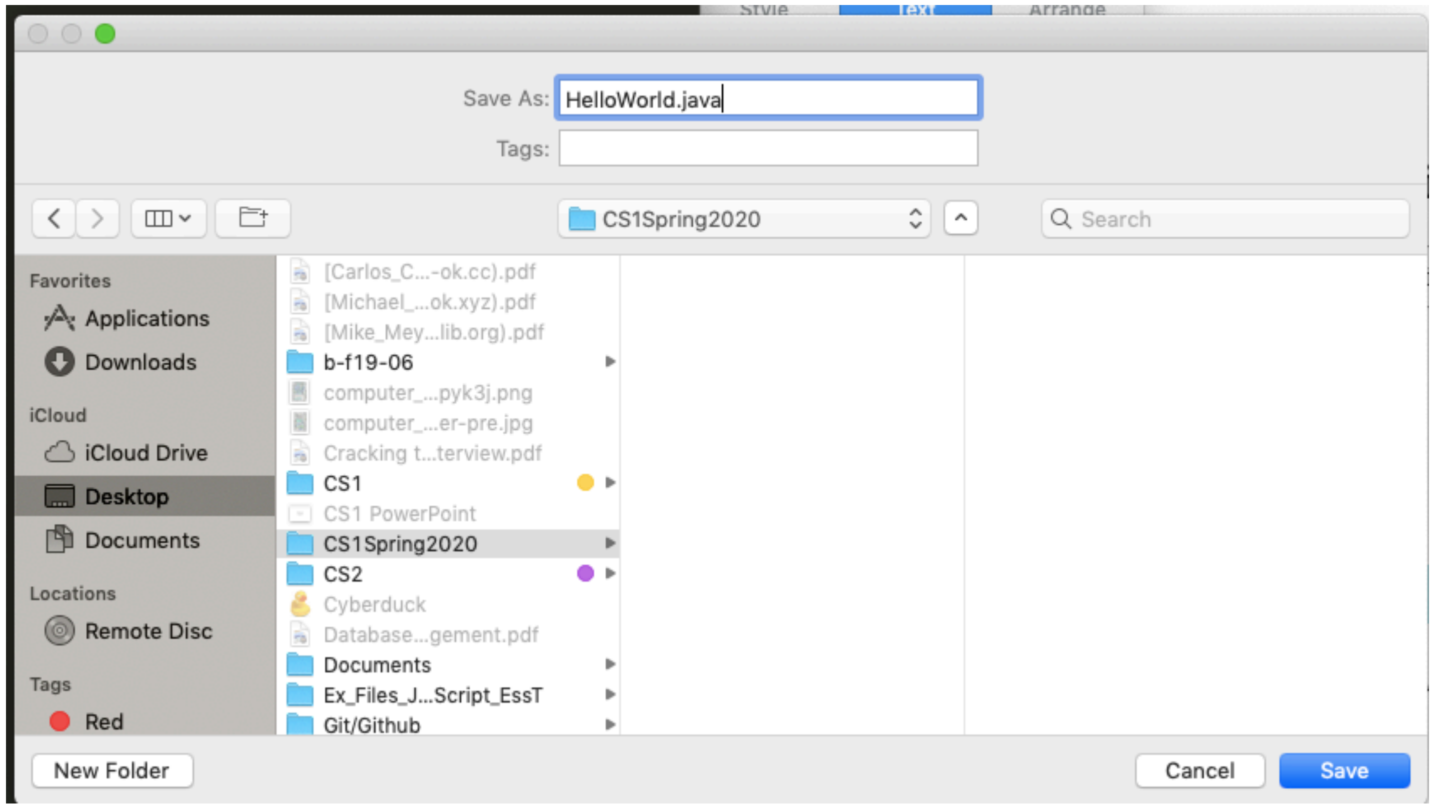
MAC USERS

How to create a simple Java program with Sublime and terminal/command line:

1. Open Sublime:

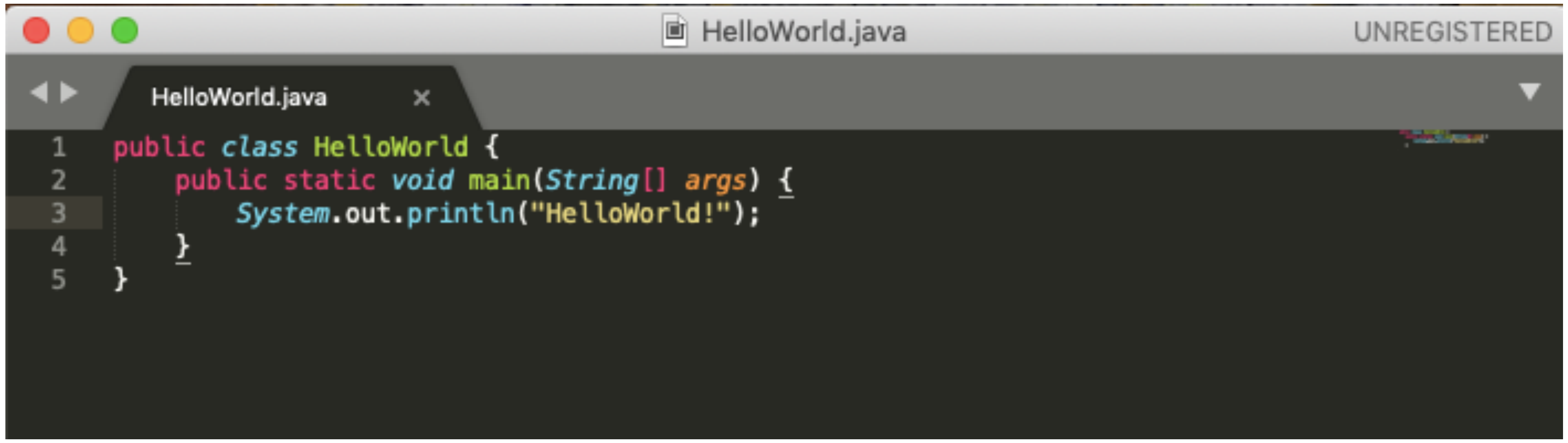

Sublime Text File Edit Selection Find View Goto Tools Project Window Help
untitled
UNREGISTERED

2. Save a file as the name of your class name (in this case, HelloWorld) and .java:


Save As: HelloWorld.java

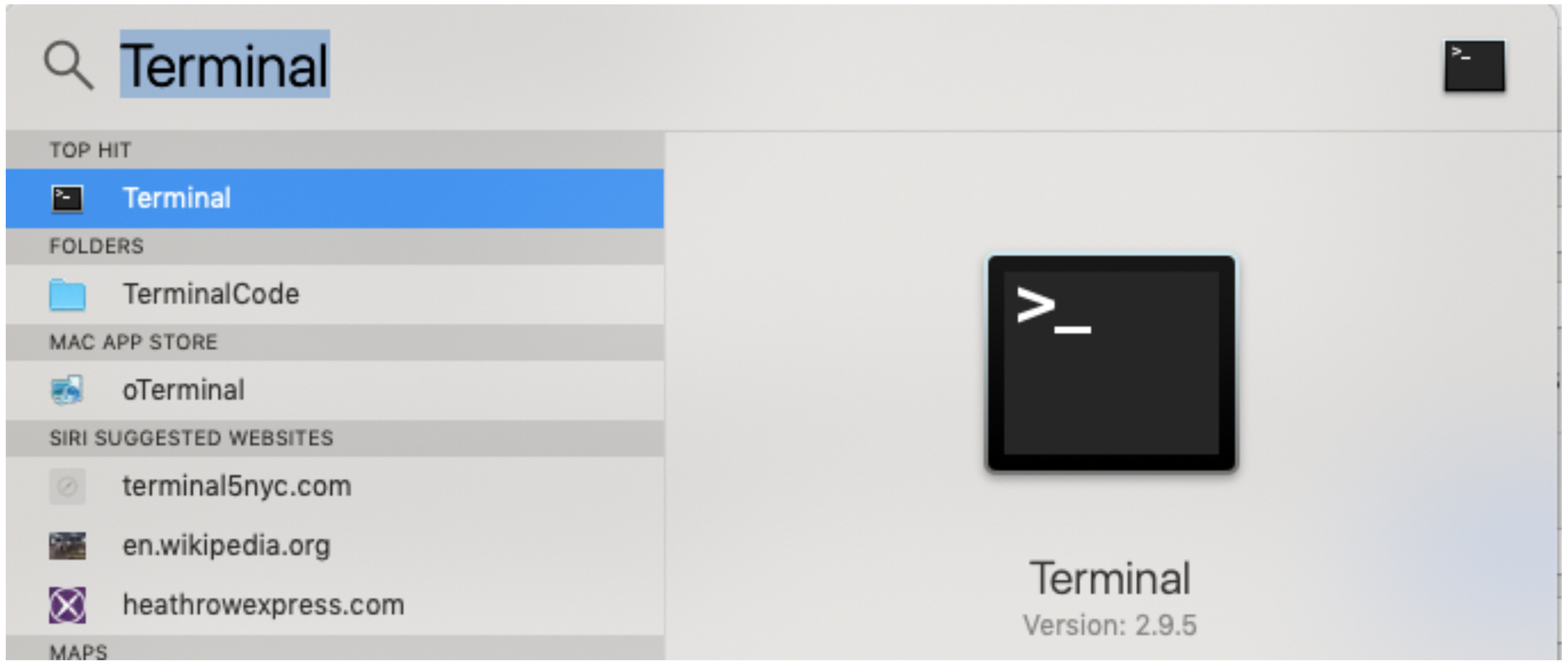
Tags:

3. Write your program (you need 3 things: your class header, your main header, and your code):

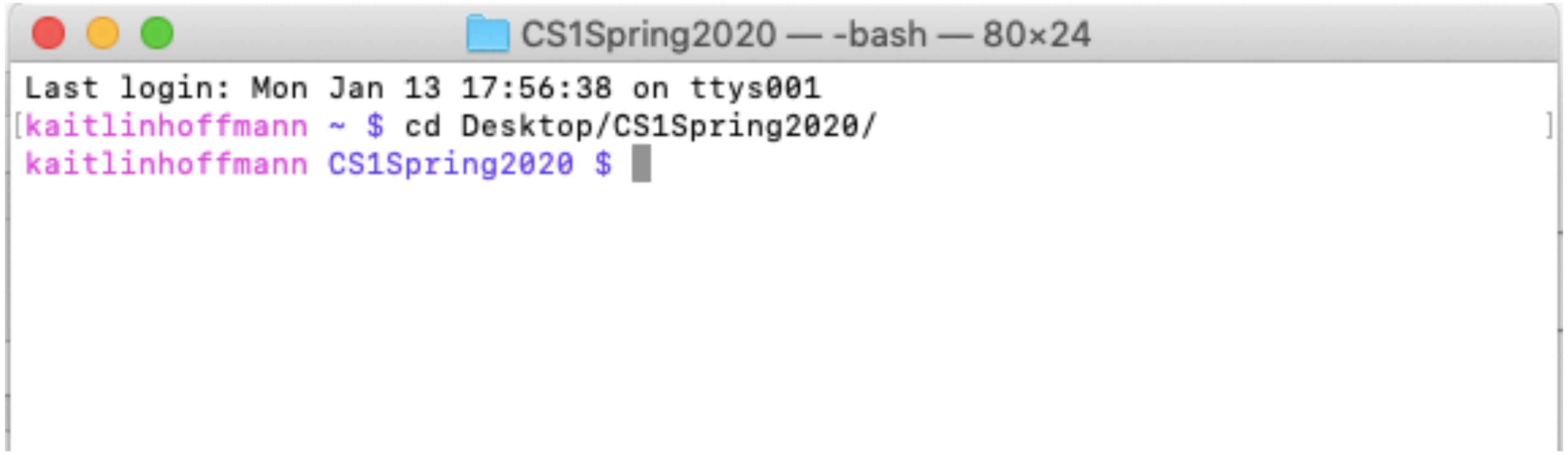


```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("HelloWorld!");
    }
}
```

4. Save it. Open your terminal/command Line. To find it, go to your search bar on your computer and type in **terminal** (**command** for Windows). Click it open:

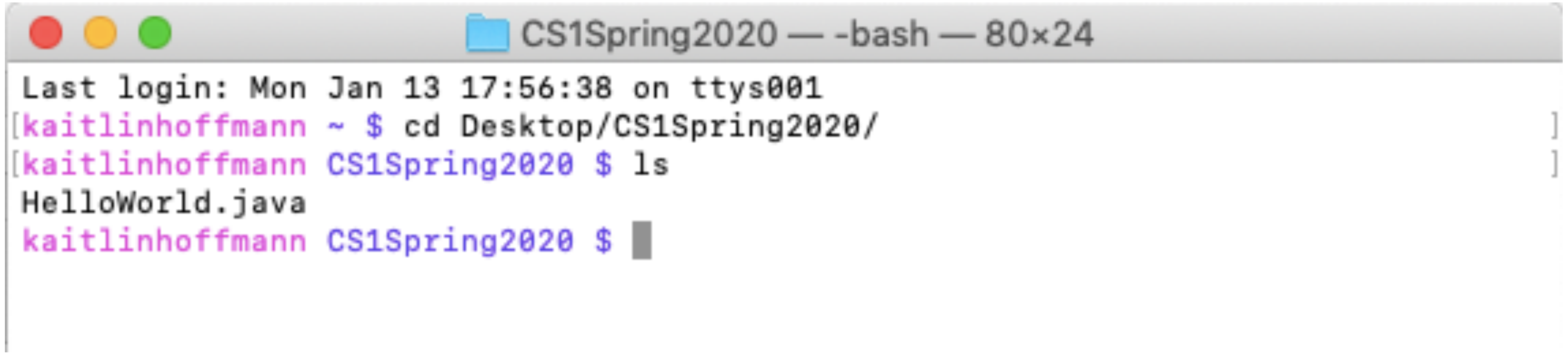


5. Change your directory to where your **HelloWorld.java** file is kept. If you need help with this in lab, let your TA know. For example, my program is on my Desktop in a folder called CS1Spring2020. So to change directories from my home directory, I write in: **cd Desktop/CS1Spring2020/**



```
Last login: Mon Jan 13 17:56:38 on ttys001
kaitlinhoffmann ~ $ cd Desktop/CS1Spring2020/
kaitlinhoffmann CS1Spring2020 $ 
```

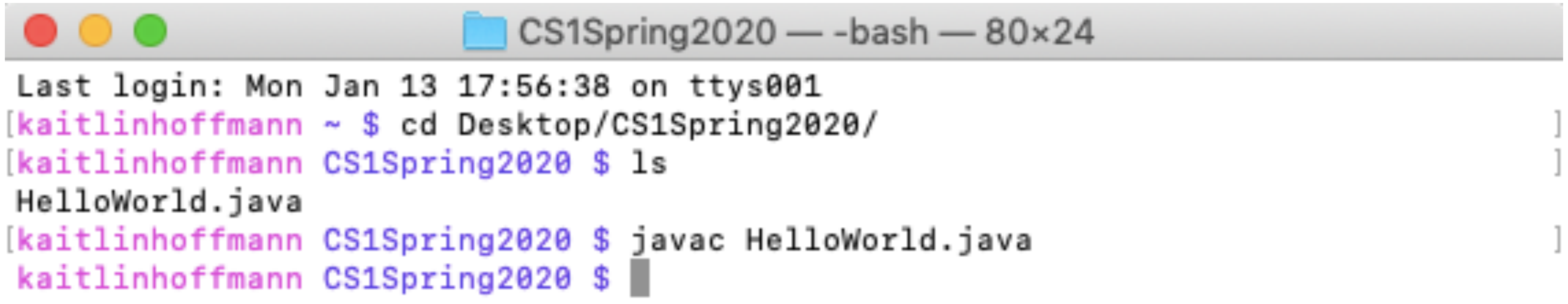
-To see the different files in this folder, enter **ls**:



```
Last login: Mon Jan 13 17:56:38 on ttys001
kaitlinhoffmann ~ $ cd Desktop/CS1Spring2020/
kaitlinhoffmann CS1Spring2020 $ ls
HelloWorld.java
kaitlinhoffmann CS1Spring2020 $ 
```

-I only have my HelloWorld program in there.

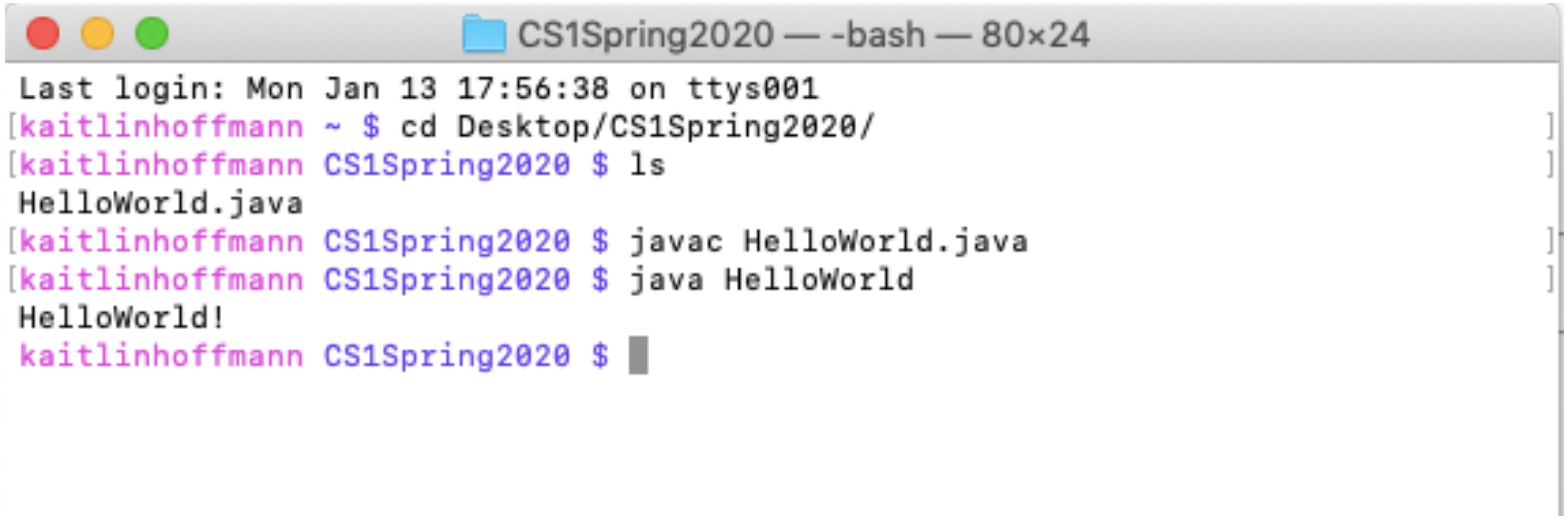
6. To run a java program, you need to first **compile** it. Compiling transforms a high level language (in this case Java) from source code to object code. (**Remember slide 13??**) To compile, enter: **javac HelloWorld.java**



```
Last login: Mon Jan 13 17:56:38 on ttys001
[kaitlinhoffmann ~ $ cd Desktop/CS1Spring2020/                                 ]
[kaitlinhoffmann CS1Spring2020 $ ls                                            ]
HelloWorld.java
[kaitlinhoffmann CS1Spring2020 $ javac HelloWorld.java                         ]
kaitlinhoffmann CS1Spring2020 $ 
```

-Nothing came up so that means no bugs or errors! If the program has errors or bugs, it will throw an error. If this happens, see if you can figure it out. Ask for help if needed!

7. Now let's run it! To run your compiled Java program, enter:
java HelloWorld



```
Last login: Mon Jan 13 17:56:38 on ttys001
[kaitlinhoffmann ~ $ cd Desktop/CS1Spring2020/                              ]
[kaitlinhoffmann CS1Spring2020 $ ls                                         ]
HelloWorld.java
[kaitlinhoffmann CS1Spring2020 $ javac HelloWorld.java                      ]
[kaitlinhoffmann CS1Spring2020 $ java HelloWorld                            ]
HelloWorld!
kaitlinhoffmann CS1Spring2020 $ 
```

SUCCESS!

LAB 1

- **No quiz** since this is the first lab
- First lab will have you download Java and a text editor of your choice. You will code a "Hello World" program. **Directions** can be found under the **Lab** tab on Blackboard.