CS 106 INTRODUCTION TO DATA STRUCTURES

SPRING 2020

PROF. SARA MATHIESON

HAVERFORD COLLEGE

ADMIN

EVERYONE

* sign in (separate sheets for registered and waitlist)

* pick up 2 handouts

* seats with computers are for registered students

JAN 21 OUTLINE

- Course overview and goals
- Introduction to Java
- Syllabus
- Logging into lab machines and first programming exercise

JAN 21 OUTLINE

- **Course overview and goals**
- Introduction to Java
- Syllabus
- Logging into lab machines and first programming exercise

COURSE STAFF

Instructor: Sara Mathieson

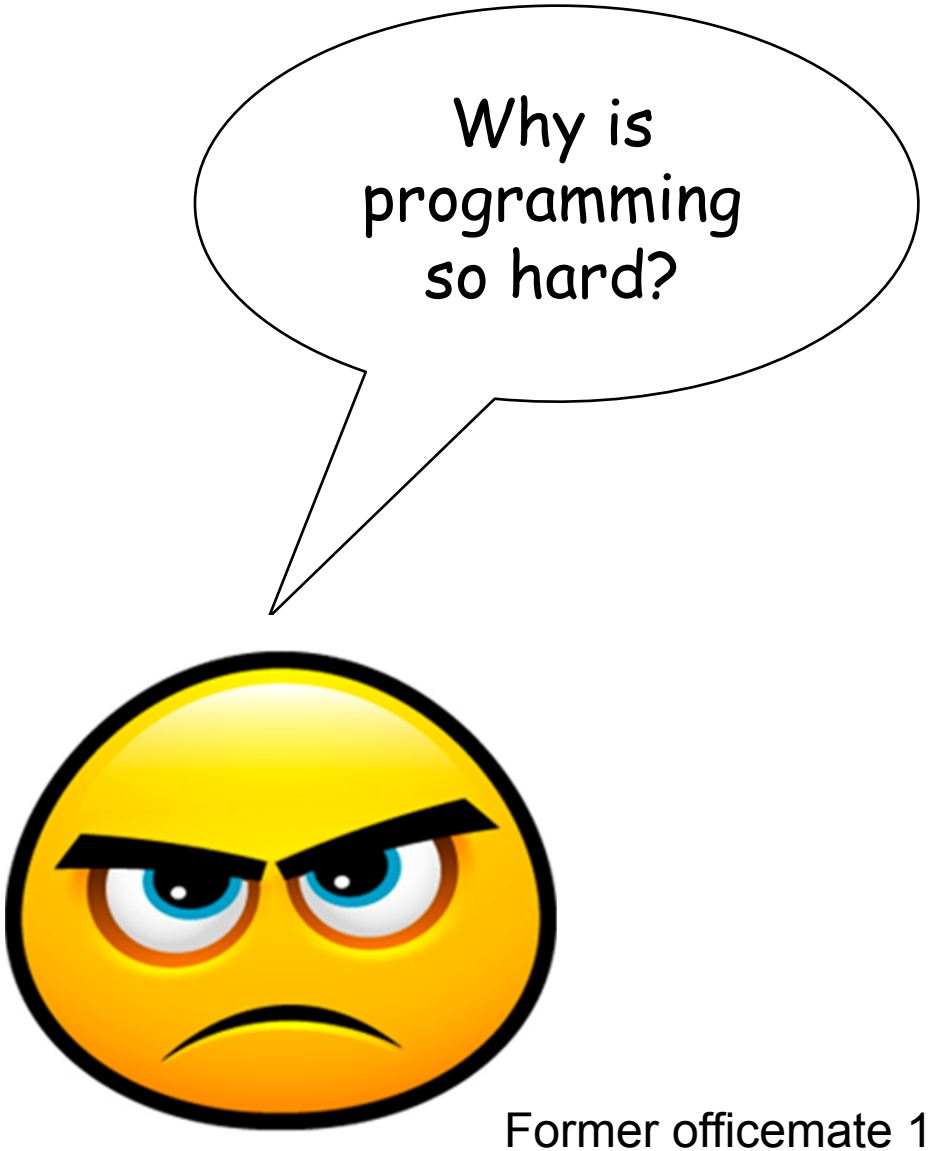
(can call me Sara or Professor Mathieson)

Lab Instructor: Suzanne Lindell

TAs: Emile Givental, Juvia Han, William Lawrence, Steve Lee, Gareth Nicholas, Lizzie Spano

Lab Monitors: evenings Sun-Thurs, 7-11pm

FOOD FOR THOUGHT



Former officemate 1

FOOD FOR THOUGHT

Former officemate 2

Why is programming so hard?

Anything that is easy on a computer… is easy because a software developer worked really hard.

Former officemate 1

COURSE GOALS

- **Essential Data Structures and Algorithms**

- **Java Programming Language**

COURSE GOALS

- **Essential Data Structures and Algorithms**
- **Java Programming Language**

- **Ability to choose an effective data structure for a new task**

COURSE GOALS

- Essential Data Structures and Algorithms
- Java Programming Language

- Ability to choose an effective data structure for a new task
- Relating theory and programming details

COURSE GOALS

- Essential Data Structures and Algorithms
- Java Programming Language

- Ability to choose an effective data structure for a new task
- Relating theory and programming details
- Confidence to learn a new language in the future

COURSE GOALS

- Essential Data Structures and Algorithms
- Java Programming Language

- Ability to choose an effective data structure for a new task
- Relating theory and programming details
- Confidence to learn a new language in the future
- Coding style and workflow practices

COURSE GOALS

Understanding your responsibilities as a computer scientist

- Documentation and reproducibility
- Efficiency and sustainability
- Careful, ethical use of data

THE ART OF PROGRAMMING

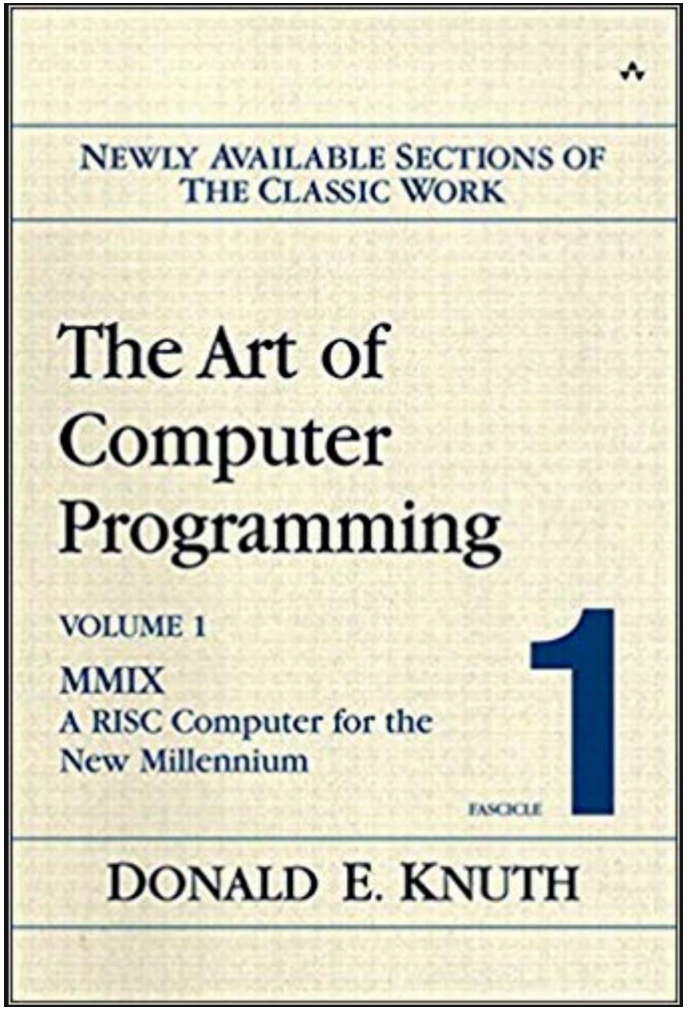
With a partner:

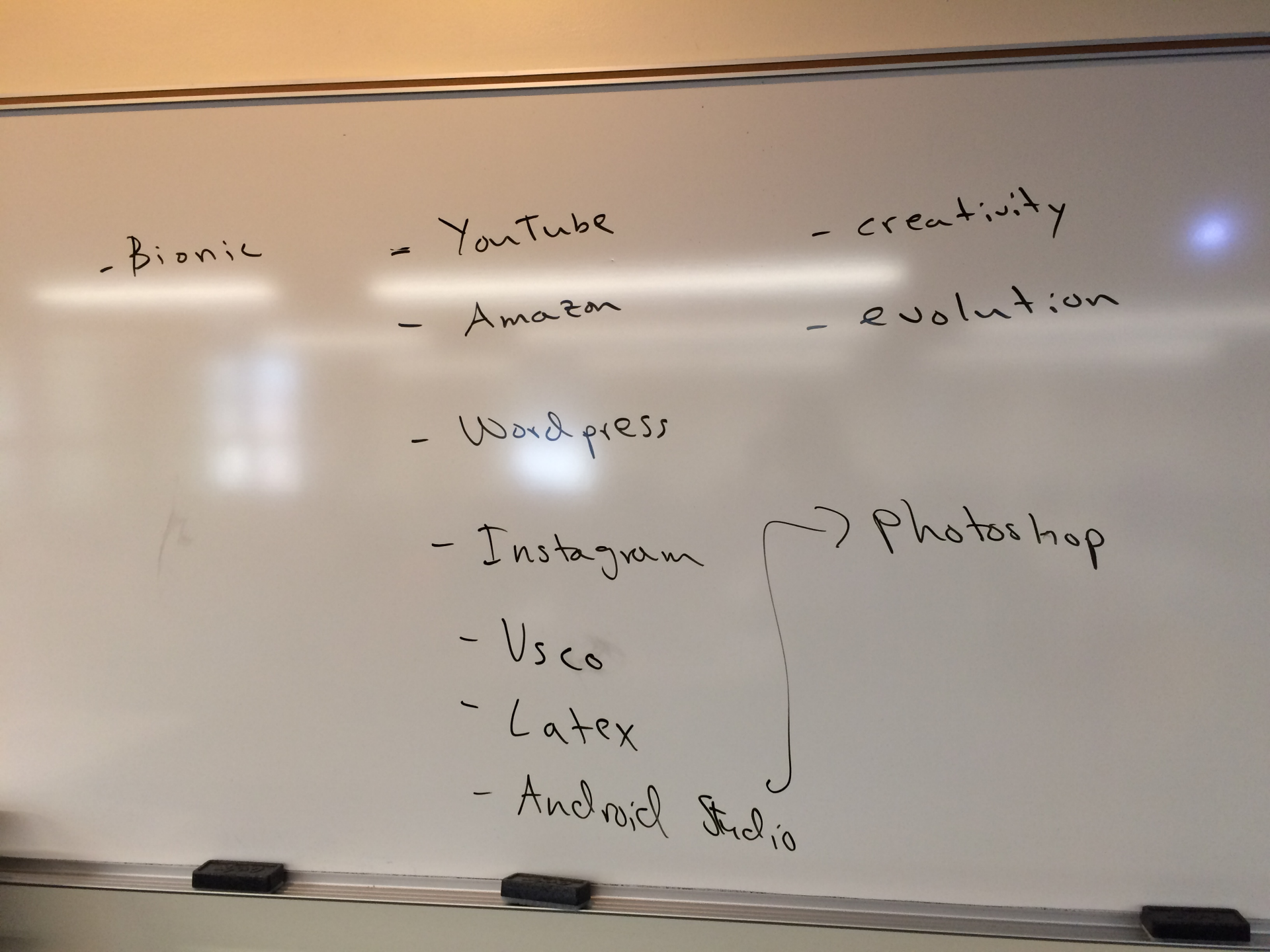
1) **Names (be ready to introduce your partner to the class!)**

2) **What you did for winter break**

3) **Brainstorm how or why programming might be considered an art**

4) **Example of software you have used that you would consider elegant or artistic**




- Bionic
= YouTube
- Amazon
- creativity
- evolution
- Wordpress
- Instagram
- Vsco
- Latex
- Android Studio
→ Photoshop

JAN 21 OUTLINE

- Course overview and goals
- **Introduction to Java**
- Syllabus
- Logging into lab machines and first programming exercise

JAVA BASICS

PYTHON

```python
y = int(input("Enter graduation year: "))

if y <= 2019:
    print("You have graduated.")
elif y <= 2023:
    print("You are in college.")
else:
    print("You are not in college.")
```

JAVA BASICS

PYTHON

```python
y = int(input("Enter graduation year: "))

if y <= 2019:
    print("You have graduated.")
elif y <= 2023:
    print("You are in college.")
else:
    print("You are not in college.")
```

JAVA

```java
System.out.print("Enter graduation year: ");
String line = stdin.readLine();
int y = Integer.parseInt(line);

if (y <= 2019) {
    System.out.println("You have graduated.");
} else if (y <= 2023) {
    System.out.println("You are in college.");
} else {
    System.out.println("You are not in college.");
}
```

JAVA BASICS

PYTHON

```python
for i in range(10):
    print(i)

x = 4;
while x < 10:
    print(x)
    x += 1;
```

JAVA BASICS

PYTHON

```python
for i in range(10):
    print(i)

x = 4;
while x < 10:
    print(x)
    x += 1;
```

JAVA

```java
for (int i=0; i < 10; i++) {
    System.out.println(i);
}

int x = 4;
while (x < 10) {
    System.out.println(x);
    x += 1;
}
```

JAVA BASICS

PYTHON

and, or, not

```
def func():
    print(“Hello, World!”)
```

JAVA

&&, ||, !

```
public static void func() {
    System.out.println("Hello, World!");
}
```

JAVADOC COMMENTS

```
/**
 * This is a block comment that describes
 * what a function does.
 * @param  param1 the first parameter described
 * @param  param2 the second parameter
 * @return the return value described
 */
 public static int func(int param1, int param2) {

    // short single line comments
    return 0;
 }
```

THE MAIN METHOD

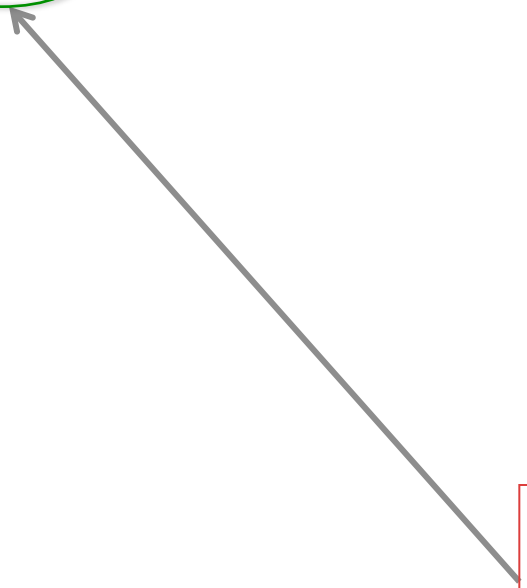
```
public class Main {

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

    public static int add(int x, int y) {
        return x + y;
    }
}
```

BUILDING A JAVA PROGRAM

```
public class AddTax {
```



public: visible to the world
private: visible only within the class
(usually classes are public)

```
}
```

BUILDING A JAVA PROGRAM

```
public class AddTax {

        public static final double TAXRATE = 0.06;
```

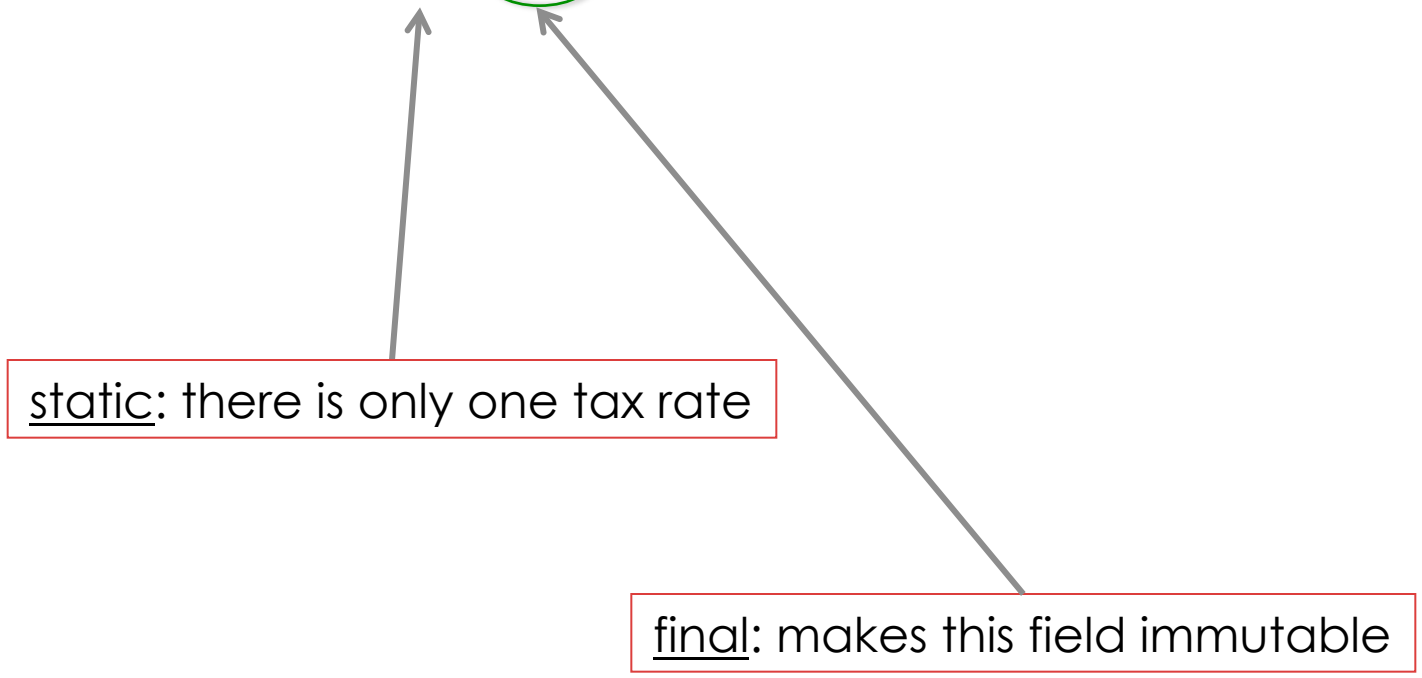
static: there is only one tax rate

```
}
```

BUILDING A JAVA PROGRAM

```
public class AddTax {

        public static final double TAXRATE = 0.06;
```



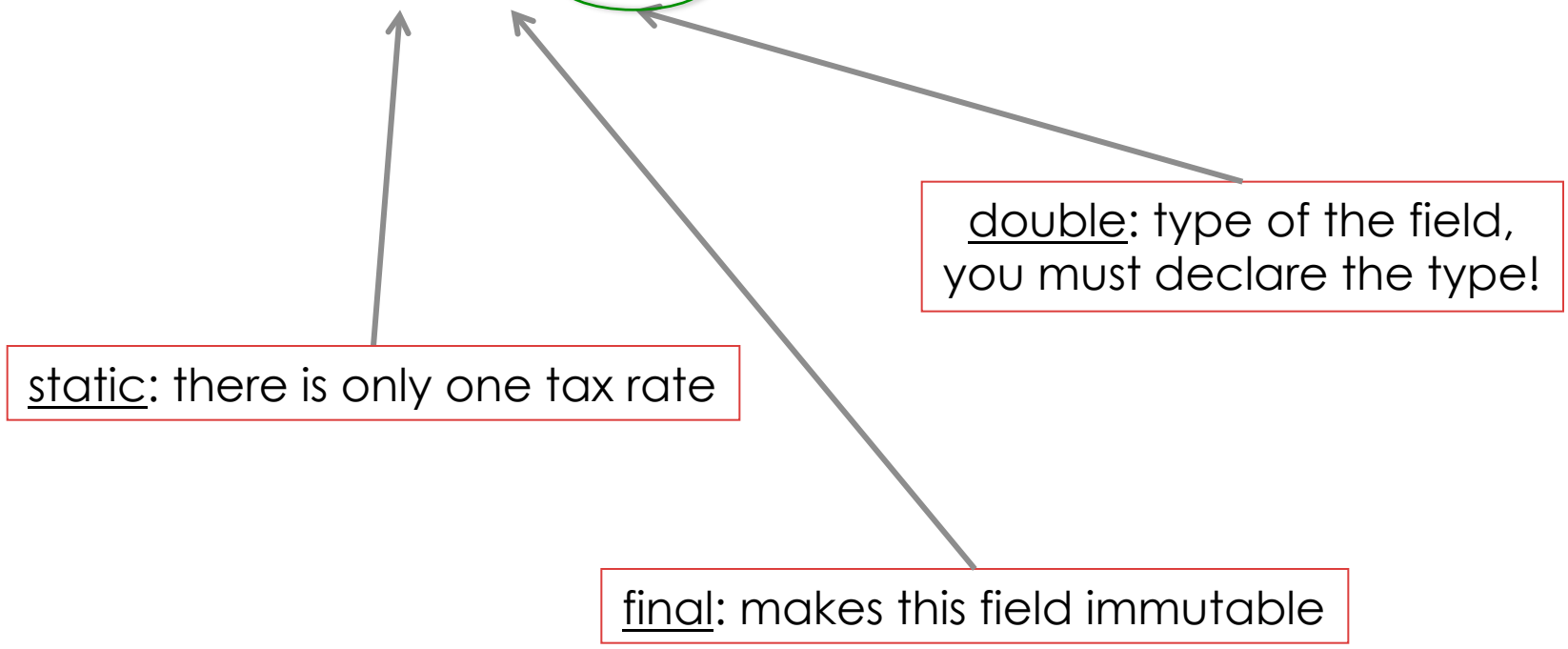

```
}
```

BUILDING A JAVA PROGRAM

```
public class AddTax {

        public static final double TAXRATE = 0.06;

}
```



BUILDING A JAVA PROGRAM

```
public class AddTax {

        public static final double TAXRATE = 0.06;



        private static double addTax(double stickerPrice) {
                double finalCost = (1 + TAXRATE) * stickerPrice;
                return finalCost;
        }

}
```

argument(s): like python, but you must declare their type

BUILDING A JAVA PROGRAM

```
public class AddTax {

        public static final double TAXRATE = 0.06;



        private static double addTax(double stickerPrice) {
                double finalCost = (1 + TAXRATE) * stickerPrice;
                return finalCost;
        }

}
```

argument(s): like python, but you must declare their type

return type: methods can either return a type or return "void"

BUILDING A JAVA PROGRAM

```
public class AddTax {
        public static final double TAXRATE = 0.06;
        private static BufferedReader stdin =
                new BufferedReader(new InputStreamReader(System.in));
        private static double addTax(double stickerPrice) {
                double finalCost = (1 + TAXRATE) * stickerPrice;
                return finalCost;
        }
        public static void main(String[] args) throws IOException {
        }
}
```

return type for main is void

BUILDING A JAVA PROGRAM

```
public class AddTax {
        public static final double TAXRATE = 0.06;
        private static BufferedReader stdin =
                new BufferedReader(new InputStreamReader(System.in));
        private static double addTax(double stickerPrice) {
                double finalCost = (1 + TAXRATE) * stickerPrice;
                return finalCost;
        }
        public static void main(String[] args) throws IOException {
                System.out.print("Please enter the price:  ");
                String line = stdin.readLine();
                double stickerPrice = Double.parseDouble(line);
        }
}
```

return type for main is void

convert a string to a number

BUILDING A JAVA PROGRAM

```
public class AddTax {
        public static final double TAXRATE = 0.06;
        private static BufferedReader stdin =
                new BufferedReader(new InputStreamReader(System.in));
        private static double addTax(double stickerPrice) {
                double finalCost = (1 + TAXRATE) * stickerPrice;
                return finalCost;
        }
        public static void main(String[] args) throws IOException {
                System.out.print("Please enter the price:  ");
                String line = stdin.readLine();
                double stickerPrice = Double.parseDouble(line);
                double finalPrice = addTax(stickerPrice);
                System.out.println("With tax, that comes to $" + finalPrice + ".");
        }
}
```

return type for main is void

convert a string to a number

printing

BUILDING A JAVA PROGRAM

```
public class AddTax {
        public static final double TAXRATE = 0.06;
        private static BufferedReader stdin =
                new BufferedReader(new InputStreamReader(System.in));
        private static double addTax(double stickerPrice) {
                double finalCost = (1 + TAXRATE) * stickerPrice;
                return finalCost;
        }
        public static void main(String[] args) throws IOException {
                System.out.print("Please enter the price:  ");
                String line = stdin.readLine();
                double stickerPrice = Double.parseDouble(line);
                double finalPrice = addTax(stickerPrice);
                System.out.println("With tax, that comes to $" + finalPrice + ".");
        }
}
```

COMMON JAVA ERRORS

- **Missing bracket**
- **No type specifier**
- **Omit static on a field or main**
- **Argument/parameter type mismatch**
- **Field/method out of place**
- **No class**

Demo: AddTax in Java (Eclipse)

GROUP ACTIVITY: REVERSE ENGINEERING

In groups of 2-3, write down what this same program would look like in Python.

GROUP ACTIVITY: REVERSE ENGINEERING

```
TAXRATE = 0.06

def add_tax(sticker_price):
    final_price = (1+TAXRATE)*sticker_price
    return final_price

def main():
    sticker_price = float(input("Please enter the price: "))
    final_price = add_tax(sticker_price)
    print("With tax, that comes to $" + str(final_price))

main()
```

JAN 21 OUTLINE

- Course overview and goals
- Introduction to Java
- **Syllabus**
- Logging into lab machines and first programming exercise

COURSE COMPONENTS

Labs (8 total, every 1-2 weeks, may include intro coding exercises): 45%

Midterms (2 in class, week 7 and week 14): 40% (20% each)

Final project: 10%

Participation (includes attendance, participating in class, and Piazza): 5%

MY EXPECTATIONS

Come to class (Tu/Th) and lab (F), ON TIME, and actively participate

- Email me if you will be absent from class, email Suzanne if you will be absent from lab
- If you are sick, do not come to class/lab!

Complete the weekly reading

Come to office hours (this week: Thurs: 1-2pm, by appointment)

- **KINSC L302**

Post questions on Piazza

WEEK	DAY	ANNOUNCEMENTS	TOPIC & READING	LABS
1	Jan 21		**Introduction** • Java basics and syntax • Documentation, Javadoc comments • Types and type conversion • Variable scope • Basics of classes • Programming style Reading: • Chapter 1	**Lab 0**: **Java intro** **Eclipse/GIT movie**
	Jan 23			

SYLLABUS NOTES

(Note: you are responsible for reading the entire syllabus on the course webpage)

http://cs.haverford.edu/faculty/smathieson/teaching/s20/

1. **Notes and slides will be posted *after* class on the course webpage**
2. **Lab is mandatory (attendance will be taken)**
3. **Attendance will be taken during class and counts toward participation**
4. **You will get 2 late days during the semester**
5. **Extensions beyond these two days must be arranged with your class dean**
6. **Email: allow 24 hours for a response (more during weekends)**
7. **Piazza: should be used for all content/logistics questions**

PARTICIPATION

What counts as participation?

Asking and answering questions in class (very important!)

- Raise your hand (because some people are more/less comfortable shouting out answers)
- Will call on groups, but only after giving you a few minutes to think/discuss

Actively participating in in-class activities (group work, handouts, etc)

Working thoughtfully during lab

- Thinking about concepts instead of just trying to get to the end of the lab

Asking and answering questions on Piazza

- Avoid long blocks of code and giving away answers
- Only non-anonymous posts count toward participation grade

Attending office hours

Sometimes participation goes too far

- Try to avoid dominating class discussion, office hours, Piazza, etc

ELECTRONIC DEVICES

In class and lab we will be using the lab computers to practice

If you want to use your own computer, that's fine, but it must be directed toward class material (exercise, taking notes)

Use of phones and or laptops for other sites is distracting, not only to you, but your neighbors as well

ACADEMIC INTEGRITY

Faculty statement on academic integrity

In a community that thrives on relationships between students and faculty that are based on trust and respect, it is crucial that students understand a professor's expectations and what it means to do academic work with integrity. Plagiarism and cheating, even if unintentional, undermine the values of the **Honor Code** and the ability of all students to benefit from the academic freedom and relationships of trust the Code facilitates. Plagiarism is using someone else's work or ideas and presenting them as your own without attribution. Plagiarism can also occur in more subtle forms, such as inadequate paraphrasing, failure to cite another person's idea even if not directly quoted, failure to attribute the synthesis of various sources in a review article to that author, or accidental incorporation of another's words into your own paper as a result of careless note-taking. Cheating is another form of academic dishonesty, and it includes not only copying, but also inappropriate collaboration, exceeding the time allowed, and discussion of the form, content, or degree of difficulty of an exam. Please be conscientious about your work, and check with me if anything is unclear.

Note for this course

Discussing ideas and approaches to problems with others on a general level is fine (in fact, we encourage you to discuss general strategies with each other), but you should never read anyone else's code or let anyone else read your code.

- No code from online
- No code from students who took this course previously

COLLABORATION POLICY

Allowed sources:

- your classmates in this class – work together, erase your work, then write it up separately
- the slides and your notes
- me, Suzanne, the TAs, and lab monitors
- the internet, though it *must be cited and only used for reference*

Rule of thumb: you may not copy code

this includes copy-paste or manual copying

ACADEMIC ACCOMMODATIONS

Faculty statement on accommodations

Haverford College is committed to providing equal access to students with a disability. If you have (or think you have) a learning difference or disability – including mental health, medical, or physical impairment - please contact the Office of Access and Disability Services (ADS) at **hc-ads@haverford.edu**. The Coordinator will confidentially discuss the process to establish reasonable accommodations.

Students who have already been approved to receive academic accommodations and want to use their accommodations in this course should share their verification letter with me and also make arrangements to meet with me as soon as possible to discuss their specific accommodations. Please note that accommodations are **not retroactive** and require advance notice to implement.

It is a state law in Pennsylvania that individuals must be given advance notice if they are to be recorded. Therefore, any student who has a disability-related need to audio record this class must first be approved for this accommodation from the Coordinator of Access and Disability Services and then must speak with me. Other class members will need to be aware that this class may be recorded.

https://www.haverford.edu/access-and-disability-services/accommodations/receiving-accommodations

CLASS DEANS

Raquel Esteves-Joyce
Assistant Dean of First Generation/Low Income (FGLI) Student Support & Programming

Stokes 118F
(610) 795-6121
rjoyce@haverford.edu

Make an Appointment
Profile

Amy Feifer
Dean of Career and Professional Advising & Associate Dean of the College

Stokes 300H
(610) 896-1181
afeifer@haverford.edu

Profile

Katrina Glanzer '02
Dean of First Year & Pre-Major Advising, Associate Dean of the College

Stokes 118C
(610) 896-1227
Assigned Students:
Class of 2023 (A-Z)

kglanzer@haverford.edu
Profile

Martha Denney
Dean of the College

Interim Title IX Coordinator

Stokes 103
(610) 896-1232
Assigned Students:
C-F ('20, '21, '22)

mdenney@haverford.edu
Profile
Assistant:
Susan Lewis

Nathan Diehl
Dean of Residential and Community Life & Associate Dean of the College

Stokes 022
(610) 896-1003
Assigned Students:
S ('20, '21, '22)

ndiehl@haverford.edu
Profile

Michael Elias
Dean of Student Engagement & Leadership and Divisional Initiatives, Associate Dean of the College

Stokes 022C
(610) 896-1228
Assigned Students:
T-Z ('20, '21, '22)

melias@haverford.edu
Profile

James Keane
Registrar, Director of Academic Assessment and Operations, Associate Dean of the College

Stokes 108
(610) 896-1023
Assigned Students:
M, Transfer Students ('20, '21, '22)

jkeane@haverford.edu
Profile

Michelle Leao
Director of Student Engagement and Leadership, and Assistant Dean of the College

Stokes 022
(610) 896-1298
Assigned Students:
N-R ('20, '21, '22)

mleao@haverford.edu
Profile

Theresa Tensuan
Assoc. Dean of the College; Dean for Diversity, Access and Community Engagement; Director of the Office of Multicultural Affairs

Stokes 118 A-1
(610) 896-1420
Assigned Students:
A-B ('20, '21, '22)

ttensuan@haverford.edu
Profile
Assistant:
Roxanne Madden

Brian Cuzzolina
Director of the OAR, Associate Dean for Student Academic Success and Persistence

Stokes 118G
(610) 795-6139
Assigned Students:
G-J ('20, '21, '22)

bcuzzolina@haverford.edu

Kelly Wilcox
Dean for Student Health & Learning Resources

Stokes 203D
(610) 795-6140
Assigned Students:
K-L ('20, '21, '22)

kwilcox@haverford.edu

Make an Appointment
Profile

JAN 21 OUTLINE

- Course overview and goals
- Introduction to Java
- Syllabus
- **Logging into lab machines and first programming exercise**

Lab 0 – Due Next Wednesday

Goal: know the basics of how to write python-like code in Java

See the website for the full lab description

Go through the video tutorial for help with github classroom!

Start before lab on Friday

CODING EXERCISE

Note: work with a partner if you are having trouble logging in!

1. **Log on to the lab computers after you get your password**
2. **Go to the Terminal and type "kpasswd" to change your password (you won't see your password being typed)**
3. **Open Eclipse and go to the Workbench**
4. **Create a new Java project called "Sum"**
5. **In "src", create a new class called "Sum"**
6. **Make a function `sum` that takes an integer n and returns the sum of the numbers from 1 to n. Call the function from the main function in order to test it.**

Example: if n=4, should return:

1+2+3+4 = 10

Credit: Ryan North (2008)