

Welcome to CSCI 235

Tiziana Ligorio

tligorio@hunter.cuny.edu

Today's Plan

Welcome

Logistics

What is CSCI 235?

Why Software Engineering?



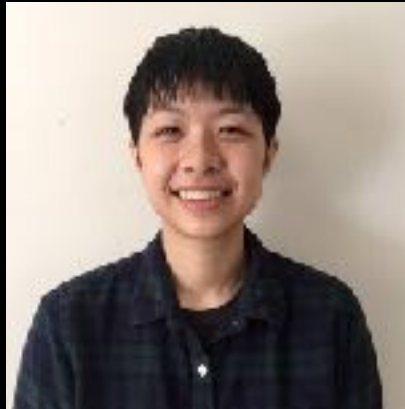
People

Instructor: (Me) Prof. Tiziana Ligorio



Undergraduate Teaching Assistants:

Carol Chau



Lily Caplan



Maria Mahin



Look for them if you need help
when you see slide titled:
In-class Task

Look for them in lab if you
need help outside of lecture
(Lab times announced next)

Acknowledgments

This course was designed with input from many great resources other than the required textbook

Many thanks for materials and inspiration to

Simon Ayzman

Susan Epstein

Keith Schwarz

Ioannis Stamos

Stewart Weiss

Logistics

Course Webpage / Syllabus

Programming Rules / Programming Projects

Linux Accounts

Communication and Help

In-class Tasks

Course Webpage

<https://tligorio.github.io/>

Visit regularly for:

Announcements

Schedule changes

Lecture Notes

Programming Projects

Programming Projects

Six programming projects (lowest dropped)

First one (review) due Tuesday 9/4

All submitted on **Gradescope**

If you haven't done so already, login to **Gradescope**
ASAP

Seek help if you have problems

READ: [Programming Rules](#) document on course
webpage

Linux Accounts

Reclaim by September 14!!!!!!

Follow instructions in:

Programming Rules document on course web page

and

http://www.geography.hunter.cuny.edu/tbw/CS.Linux.Lab.FAQ/departments_of_computer_science.faq.htm

Communication and Help

Let us hear from you!

If you find a typo or mistake let me know!!!

Blackboard forum

If you don't understand something ask!!!

Blackboard forum

Course help email : csci235.help@gmail.com

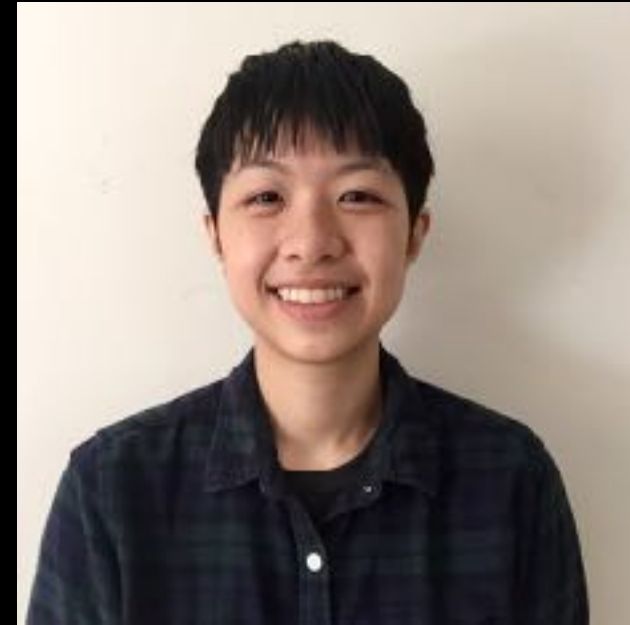
If you have concerns on something other than course content come talk to me

Office hours (subject to change, check course page for announcements) or by appointment

TAs in Lab

8/30 2 - 4pm Room: C-06

Next week **TBA**



Skirball Science Learning Center

Hunter East 7th floor



Drop-in tutoring for CSCI 235

Check the schedule on their website or pop in for a more accurate schedule for the day

<https://library.hunter.cuny.edu/skirball-science-learning-center>

Introducing In-class Tasks

5% of final grade

Bring paper and pen to lectures and **BE NEAT!!!**

I will sometimes ask you to write something down and hand it in

If today you don't have it raise your hand and a TA will give you one

Introducing In-class Tasks

On one side of a sheet of paper - (we will use other side later):

Write your name

Write if you received Gradescope email and created account

Write down if you are taking CSCI 160 right before this class

Write down if you would take advantage of tutoring on Wed between 1pm and 5:30pm at 68th street

Hold on to it for now, we will collect it soon

What is CSCI 235?

Programming => Software Analysis and Design

Think like a Computer Scientist:

Design and maintain complex programs

Software Engineering, Abstraction, OOP

Design and represent data and its management

Abstract Data Types

Implement data representation and operations

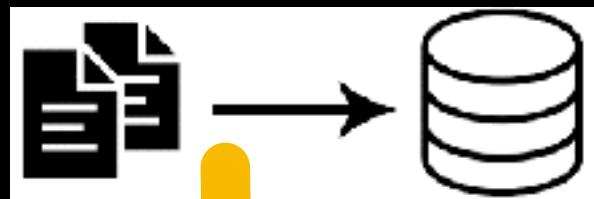
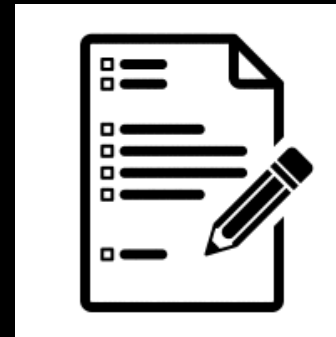
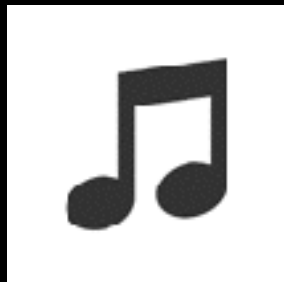
Data Structures

Algorithms

Understand Algorithm Complexity



“It’s all just bits and bytes...”



STRUCTURE

101010101010101010
010101010101010101
1001100110011001

Increasing software complexity

Society keeps digitizing more aspects of life

Software keeps getting bigger

Complexity of software systems ever increasing

Exciting!!!

Daunting for software engineers



What is software complexity?

Lines of code?

Not an exact measure but can be revealing

~10	Hello world
~100	Most STL queue implementations
~1,000	Typical Computer Science curriculum term project
~10,000	Intensive team project
~100,000	Most Linux command line utilities
~1,000,000	Linux g++ compiler
~10,000,000	Mozilla Firefox
~50,000,000	Microsoft's Windows
~2,000,000,000	Google (search, maps, docs, gmail, ...)

Illustrative example, may not be up to date

Problems of software complexity

Every bit counts!

A single incorrect bit may result in:

- negative instead of positive int
- pointer past the end of an array
- unsorted rather than sorted vector
- ...

Program performs unexpectedly

Problems of software complexity

Every bit counts!

A single incorrect bit may result in:

- negative instead of positive int
- iterator past the end of a container
- unsorted rather than sorted vector
- ...

Program performs unexpectedly



Problems of software complexity

Assume **n** lines of code

Two lines of code interact if they manipulate same data

```
int x = 5;    // if I change the x to my_var  
cout << x;    // I must change it here too
```

Any line may interact with any number of other lines

$n(n-1) = n^2 - n$ possible interactions

With **10** lines of code there may be

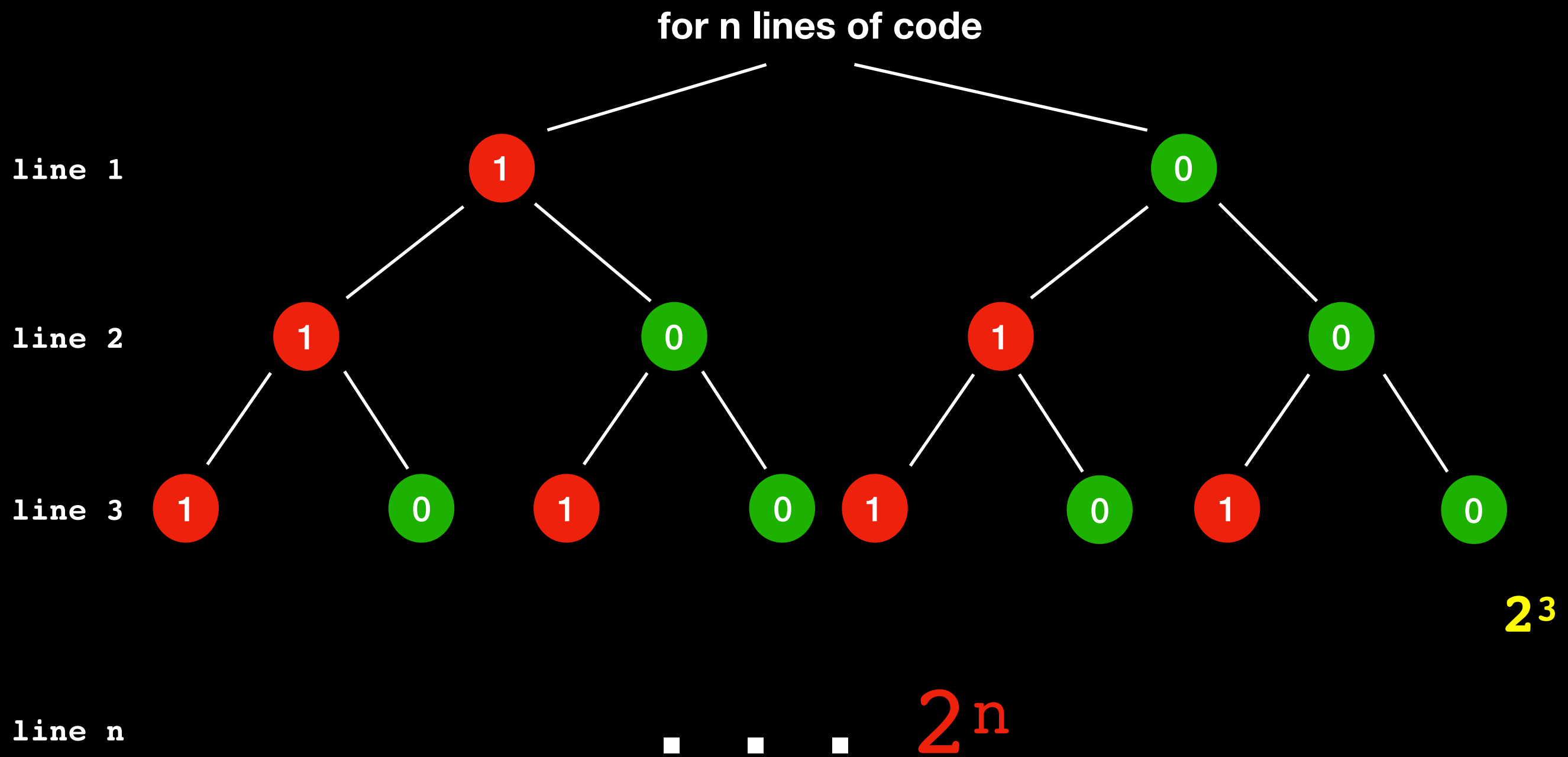
90 interactions



Problems of software complexity

Actually much worse!

More realistically interactions are not independent and may affect arbitrarily many other lines of code



Every path down the tree is an interaction among one possible subset of lines of code



0



0



0



0

1

0



0

□ □ □

 2^n

2³



Every path down the tree is an interaction among one possible subset of lines of code

In-class task

With a calculator :

Multiply $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \dots$ KEEP GOING!!!!

What is happening???

Now take a piece of paper draw a very small square on the leftmost bottom corner

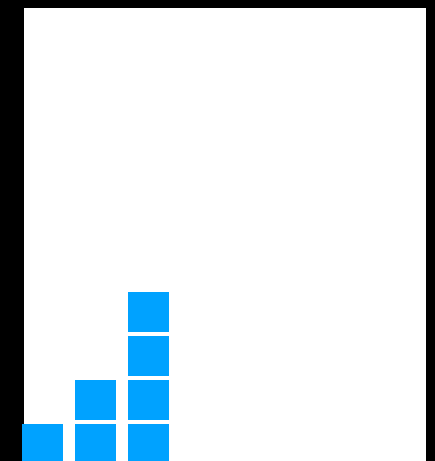
Next to it double it (2 squares one on top of other)

Next to it double it (4 squares one on top of other)

Next to it double it

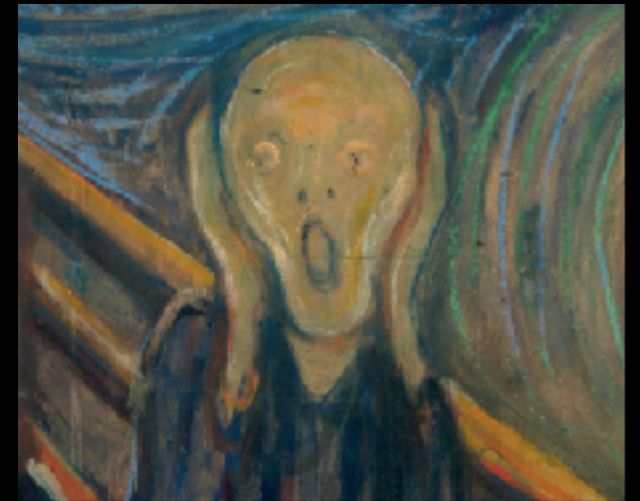
Keep going at least 7 or 8 times... What is happening?

Now we will collect it



Problems of software complexity

There are 2^n possible interactions of arbitrarily many lines of code



Get a real feeling for 2^n

AT HOME: Watch this video:

How folding paper can get you to the moon:

<https://www.youtube.com/watch?v=AmFMJJC45f1Q>

Problems of software complexity

How do you go about modifying code with many interactions?

Larger software has greater likelihood of error

More difficult to modify

Minimize complexity!!!

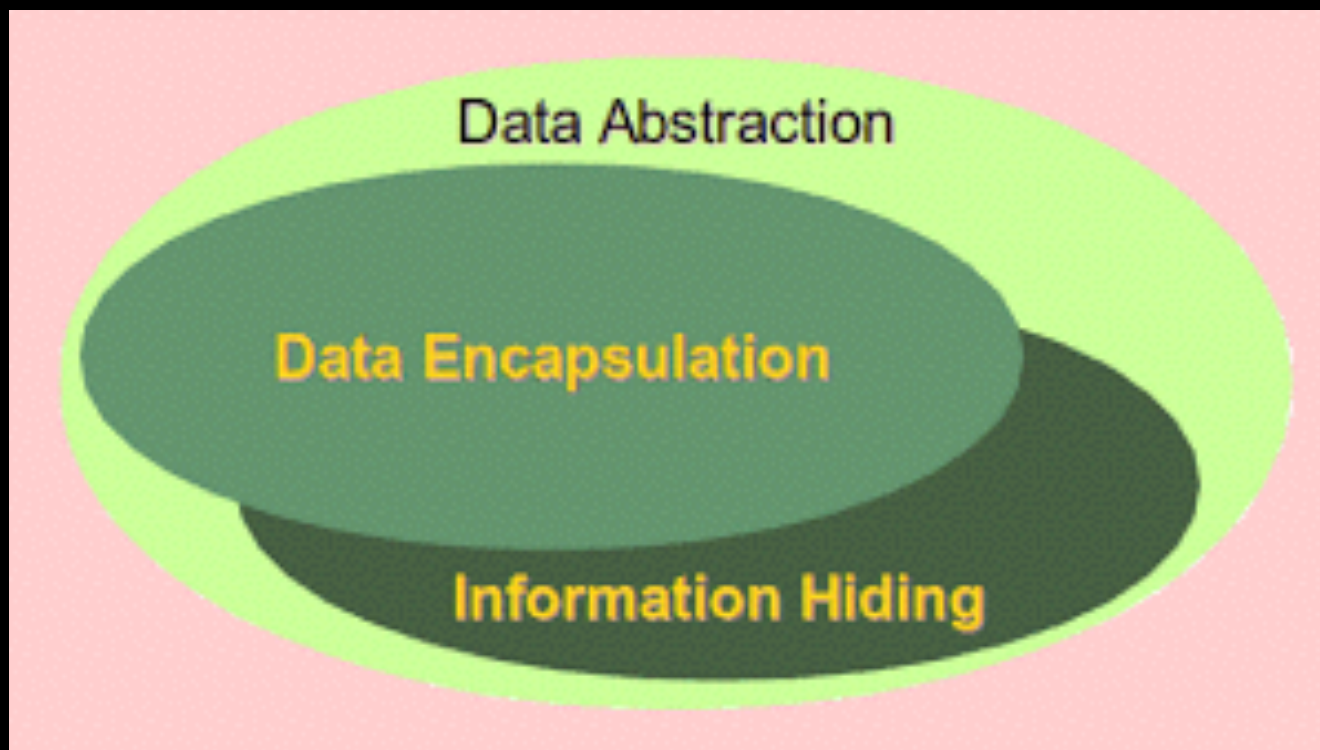
So complexity is **bad!!!!**

Write small units of code

Minimize Interactions!!!

Enforce strict rules on how code interacts

Minimize complexity!!!



O	Object
O	Oriented
P	Programming

OOP Flash Review

Classes group together variables (data) and functions (operations on the data)

Classes:

- Member variables
- Member functions (or methods)
- Constructors (special function called when instance of class declared)
- Destructors (special function called when instance of class is deleted)

Instances of a Class are called Objects

Think of Classes as types and Objects as variables

OOP Flash Review

Class members can be declared **public** or **private**

public members are the interface
for client code interaction




private members are hidden from other
code and can only be used within the class



Separate Interface From Implementation

Interface: `MyClass.h`
(or `MyClass.hpp` same thing)

Implementation: `MyClass.cpp`



We will talk
much more
about why?

Programming Project 1



1st Programming Project

Review your knowledge of Classes

Submit on Gradescope (you have email invitation to your Hunter email)

Submission will be used to confirm enrollment

Let's step back for a moment

What is Software Engineering?

"The application of a **systematic**, **disciplined**, **quantifiable** approach to the development, operation, and maintenance of software"

IEEE Standard Glossary of Software Engineering Terminology

Big Ideas of Software Engineering

Modularity

Style

Modifiability/Extensibility

Ease of Use

Fail-Safe Programming

Debugging

Testing

We will come back to these throughout the course

APPENDIX B

Control Interaction

Pass-by-value

```
bool my_method(int some_int);
```

Pass-by-reference if need to modify object

```
bool my_method(ObjectType& some_object);
```

Pass-by-constant-reference if function doesn't modify object

```
bool my_method(const ObjectType& some_object);
```

Reduce Coupling

Methods should only call other methods:

- defined within **same class**
- of **argument** objects
- of objects **created within** the method
- of objects that are **data members** of the class

Readability

Write **self-commenting** code

Important to strike balance

- don't write the obvious in comments

Bad! Don't you feel insulted?

```
int x = m * v1 / vv; //multiply m by v1 and add result to vv
```

Use descriptive names for variables and methods

```
//return: the average of values in scores
double getAverage(double* scores, int size)
{
    double total;

    for (int i = 0; i < size; ++i)
    {
        total += scores[i];
    }

    return ( total / (double)size );
}
```


Naming Conventions

<https://google.github.io/styleguide/cppguide.html>

<http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#R1-comments>

```
string my_variable;
```

or

```
string myVariable;
```

Classes ALWAYS

start with capital

```
MyClass
```

In this course I will strive for

```
class MyClass
```

```
MyClass class_instance;
```

```
string my_variable;
```

```
string my_member_variable_;
```

```
void myMethod();
```

```
int MY_CONSTANT;
```

Be consistent!!!

Modifiability

No global variables EVER!!!

Named Constants

```
const int NUMBER_OF_MAJORS = 160;  
int scores [NUMBER_OF_MAJORS];  
for(index = 0 through NUMBER_OF_MAJORS - 1)  
    Process
```

Modifiability

The typedef Statement



Renaming an
existing type

```
typedef float RealType
```

```
typedef long double RealType
```

Next Time

Abstraction and OOP