

# Object Oriented Programming

C++ Lecture 5

Nick Matthews

Adam Kohl

# Motivation

- We want to group data and functions together
- We want to easily share and reuse specific data and functionality throughout our program
- We want to govern access on certain parts of our program to make it error prone.



# Lecture Goals



- Classes
- Inheritance
- Multi-file programs

# What is a Class?

- A class is a definition for a **custom data type**
- The definition is comprised of variables and functions, known as members.

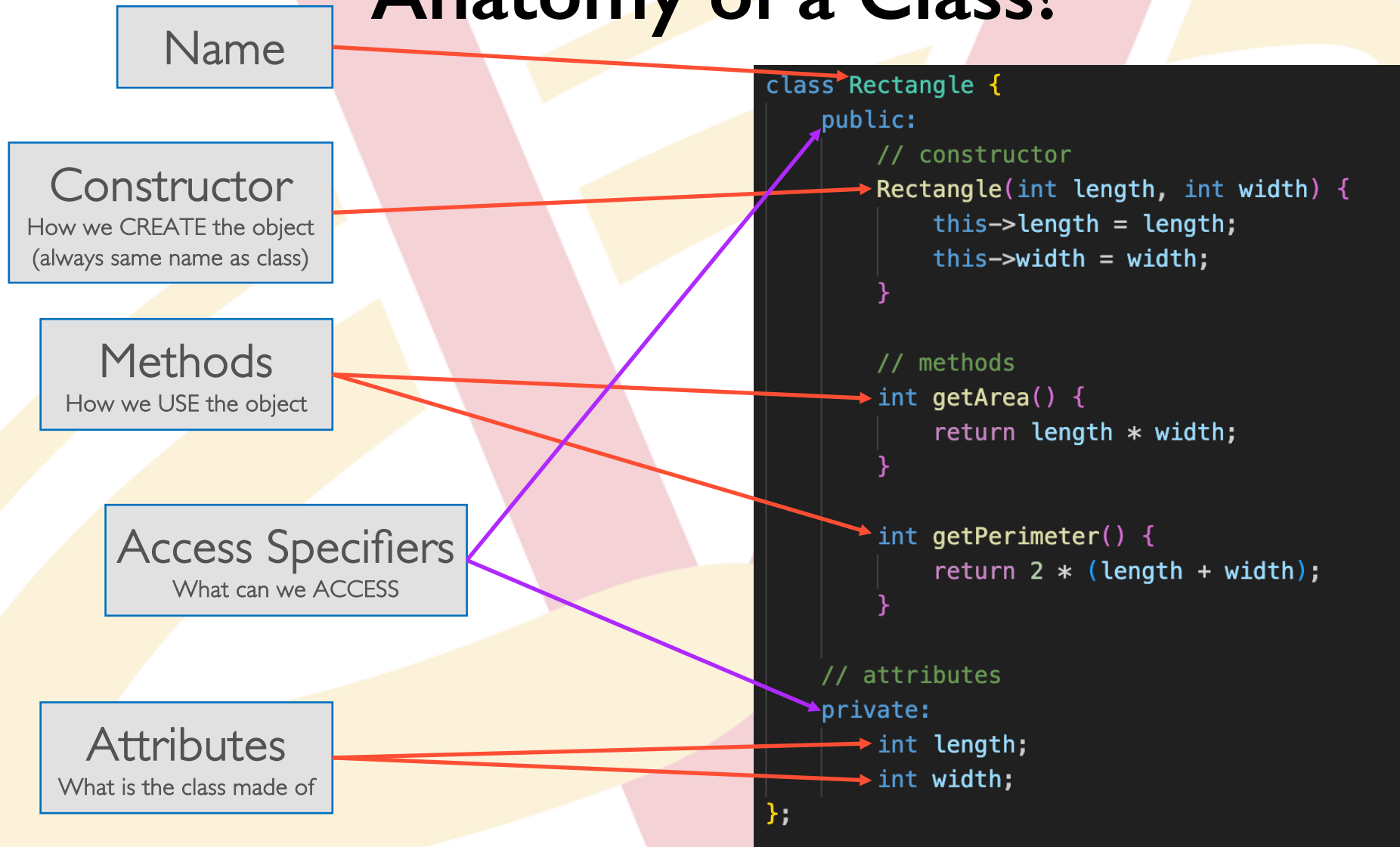
```
class Rectangle {
public:
    // constructor
    Rectangle(int length, int width) {
        this->length = length;
        this->width = width;
    }

    // methods
    int getArea() {
        return length * width;
    }

    int getPerimeter() {
        return 2 * (length + width);
    }

    // attributes
private:
    int length;
    int width;
};
```

# Anatomy of a Class?



# What is an Object?

- An object is an instance of our class

```
int main() {  
    // Create a rectangle object  
    Rectangle nicksRectangle = Rectangle(10, 5);  
  
    // Call the getArea and getPerimeter methods  
    int area = nicksRectangle.getArea();  
    int perimeter = nicksRectangle.getPerimeter();  
  
    nicksRectangle.length = 20;  
}
```

Can't access the length member directly  
because it is private

```
class Rectangle {  
public:  
    // constructor  
    Rectangle(int length, int width) {  
        this->length = length;  
        this->width = width;  
    }  
  
    // methods  
    int getArea() {  
        return length * width;  
    }  
  
    int getPerimeter() {  
        return 2 * (length + width);  
    }  
  
    // attributes  
private:  
    int length;  
    int width;  
};
```

# Instantiation

- We instantiate a class like a normal variable, except this time we use the class constructor
- Upon object creation, the constructor is called, and class variables are set.

Instantiate an integer (int type)

```
int number = 12;
```

Instantiate a class (Rectangle type)

```
Rectangle nicksRectangle = Rectangle(10, 5);
```

# Using the Object

- Once the object is made, you can use its public members and utilize the class functionality
- Access the public members using the “.” operation

```
int main() {  
    // Create a rectangle object  
    Rectangle nicksRectangle = Rectangle(10, 5);  
  
    // Call the getArea and getPerimeter methods  
    int area = nicksRectangle.getArea();  
    int perimeter = nicksRectangle.getPerimeter();  
}
```

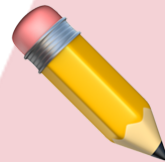




# Class Example

1. Go to [https://github.com/iastate/VRAC\\_REU\\_Programming](https://github.com/iastate/VRAC_REU_Programming)
2. Under **challenges/** read modelBankAccount.md 
3. Do modelBankAccount.cpp **TOGETHER**



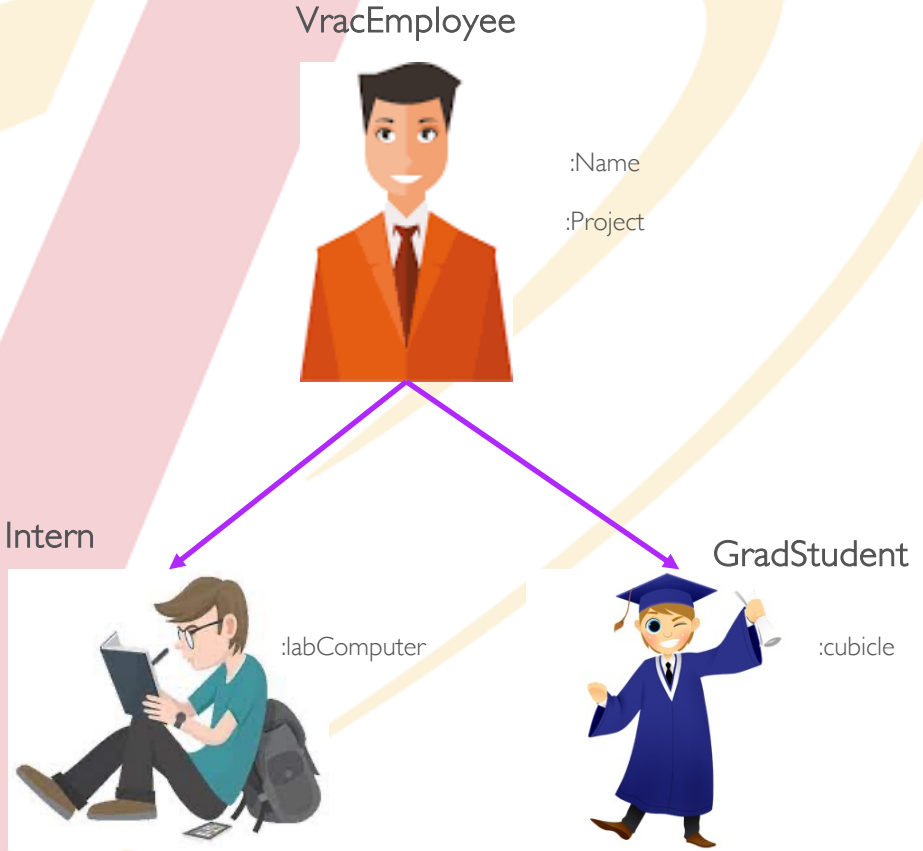
# Mini Task



1. Go to [https://github.com/iastate/VRAC\\_REU\\_Programming](https://github.com/iastate/VRAC_REU_Programming)
2. Under **challenges/** read modelDice.md 
3. Make a new project and code 

# Class Inheritance

- o Inheritance allows us to pass down attributes and methods from one class to another
- o It can further organize your system of classes and reduce redundancy of functionality



# Class Example

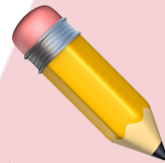
1. Model the Vrac by creating 3 classes (VracEmployee, Intern, and GradStudent)
2. Use inheritance to pass down functionality from the base class to the child classes.

# Separate Files

- As we create more variables, functions, and classes, our file begins to get cluttered and unorganized
- We can separate our code into multiple files to avoid this
- In C++, these separate files have the .h or .hpp file extension



# Mini Task



1. Open your project with the Dice class you created
2. Make a new file called dice.h
3. Move your dice class code here
4. Use `#include "dice.h"` in your file with `main()`

**Questions?**