Chapter
Introduction to Classes

- Chapter 8, sections 1-11

Object Oriented Programming (OOP)

- Procedural programming focuses on procedures (functions) within a program
  - CS 150!
- OOP focuses on **grouping** variables and functions *that operate on those variables* together
  - What is the object *responsible* for?
- Users of the object care about the functions, not the variables

Example: String

```
#include <string>

string course = "CS150";

cout << course.at(0); // get char in position 0

cout << course.push_back('!');

cout << course.length();

cout << course;
```

Class

The ***class*** is a C++ construct used to create objects

- OOP hides the details of objects
  - Encapsulate
- Objects communicate with function calls
- OOP lends itself to more *generic* solutions because the details are hidden

Class Declaration

A class:

1. is a programmer-defined datatype
2. consists of variables and functions:

General Format

```
class ClassName {
  Declarations for member variables and
  member functions

};
```

Person Class

```
class Person {
  public:
    int getAge();
    int getBirthYear(const int CURRENT_YEAR);
    void setAge(int age);

  private:
    int mAge;                          // note prefix!
};

int main () {
  const int CURRENT_YEAR = 2019;
  Person cPerson;                      // note prefix!

  cPerson.setAge(28);
  cout << "Person is: " << cPerson.getAge() << endl;
  cout << "Person Born in: " << cPerson.getBirthYear (CURRENT_YEAR) << endl;

  return EXIT_SUCCESS;
}
```

Person Class Definitions

```
void Person::setAge (int age) {
  mAge = age;
}

int Person::getAge () {
  return mAge;
}

int Person::getBirthYear (const int CURRENT_YEAR) {
  return CURRENT_YEAR - mAge;
}
```

public: versus private:

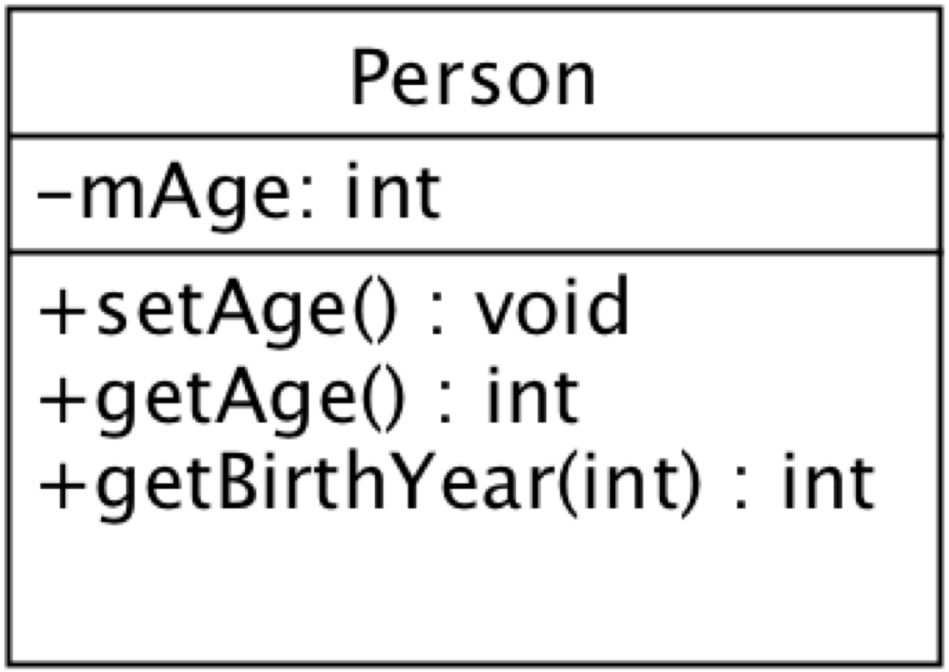
- Class data members and member functions can be either `private:` or `public:`
- `private:`
- `public:`

Annotated Person Class

```
class Person {
  public:
    void setAge(int age);   // member function prototypes
    int getAge();
    int getBirthYear(const int CURRENT_YEAR);

  private:
    int mAge;                // private member variable
};
int main () {
  Person cPerson;        // creates (instantiates) a Person o
bject

  cPerson.mAge = 28;     // error
  cPerson.setAge(28);  // correct setting of age
```

UML

- Design language that lets us describe classes



Mutator

- A mutator is any function that can change the value of a member variable

```
void Person::setAge (int age) {
  mAge = age;
}
```

Accessor

- An accessor is a function that uses a class member but cannot change the member value

```
int Person::getAge() const {
  return mAge;
}
```

Person Class

```
class Person {
  public:
    void setAge(int age);       // mutator
    int getAge()  const;        // accessor
    int getBirthYear(const int CURRENT_YEAR) const; // accessor


  private:
    int mAge;         // private member variable
};
```

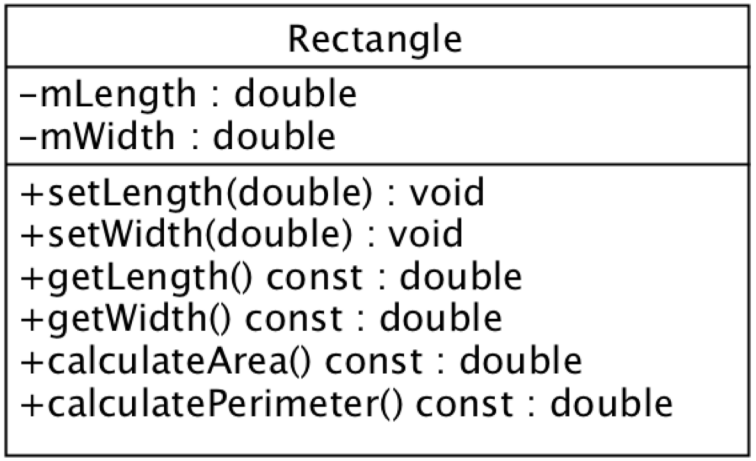
Rectangle ADT?

- What data represents a rectangle?

- *Responsibilities* the rectangle has:
  - What jobs can the rectangle perform?

OO Representation

```
class Rectangle {
  public:
    void setLength(double length);
    void setWidth(double width);

    double getLength() const;
    double getWidth() const;

    double calculateArea() const;
    double calculatePerimeter() const;

  private:
    double mLength;
    double mWidth;
};
```



class Rectangle Questions

Q1: How many data members does class Rectangle have? List them.

Q2: How many member functions does class Rectangle have? List them.

Q3: How many mutators does class Rectangle have? List them.

Q4: How many accessors does class Rectangle have? List them.

class Rectangle Questions

Q5: How do we create objects of class Rectangle?

a) A single object

b) An array of 50 objects

class Rectangle Questions

Q6: Write the C++ code that shows how to set the length and width in each of the objects in Q5 with input from the user.

Q7: Write C++ code that will print the area and perimeter of each object from Q5.

class Rectangle Questions

Q8: What other member functions might we want? What other jobs could rectangle do?

Could we get rid of the get functions?

Constructors

- Special member function to initialize data members

Constructor for Rectangle

```
Rectangle (double , double ); // prototype


Rectangle::Rectangle (double length,
                      double width)
{
  mLength = length;
  mWidth = width;
}
```

Overloaded Functions

- Multiple functions with the same name
  - Parameter lists must be different
  - Can overload constructors as well
- Choose constructor based on parameter list

Overloaded Rectangle Constructor

```
// default constructor
Rectangle::Rectangle ()
{
  mLength = mWidth = 0;
}

// overloaded constructor
Rectangle::Rectangle (double length, double width)
{
  mLength = length;
  mWidth = width;
}
```

Default Constructor

- The default constructor: no arguments

Default Arguments

```
Rectangle (double = 0.0, double = 0.0);


Rectangle::Rectangle (double length,
                      double width)
{
   mLength = length;
   mWidth = width;
}
```

Data member initialization

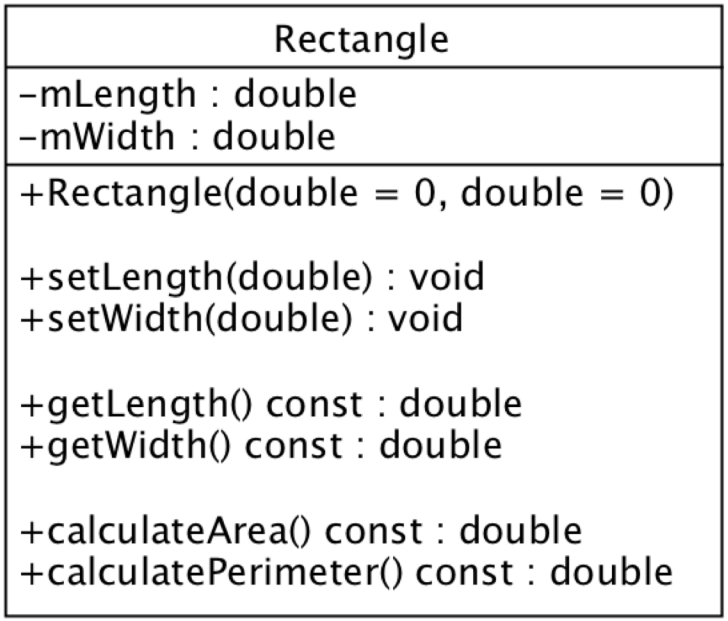
```
class Rectangle
{
  public:
    void setLength (double length);
    void setWidth (double width);

    double getLength () const;
    double getWidth () const;

    double calculateArea () const;
    double calculatePerimeter() const;

  private:
    double mLength = 0; // Initialize with constant
    double mWidth = 0;
};
```

Object Oriented Design

- Encapsulation
  - Hide the details!
  - .h file: header file with the class interface
  - .cpp file: implementation file (function bodies)
  - Users are concerned with the interface

Rectangle Interface
Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

class Rectangle
{
  public:
    Rectangle (double = 0.0, double = 0.0);

    void setLength (double);
    void setWidth (double);

    double getLength () const;
    double getWidth () const;

    double calculateArea () const;
    double calculatePerimeter() const;

  private:
    double mLength;
    double mWidth;
};

#endif
```



Rectangle Implementation
Rectangle.cpp

```
#include "Rectangle.h"

// overloaded constructor
Rectangle::Rectangle (double length, double width)
{
  mLength = length;
  mWidth = width;
}
```

Problem

- Grab the files Rectangle.h, Rectangle.cpp, and main.cpp from the folder Rectangle found in the Public directory
- Add a project Rectangle to your CS250InClass solution
- Place the three files appropriately into the project, build, and run

TODO

- Uncomment setWidth and getWidth and compile. Notice the errors and then fix the code.
- Declare an array of Rectangles of size 50.
- Create the following 50 Rectangles (LxW) 1x2, 2x3, 3x4, ...
- Print the area and perimeter for each Rectangle as follows (one per line):
  - Rectangle; Length = 1; Width = 2; Area = 2; Perimeter = 4