CS 485
Advanced Object Oriented Design

Spring 2019

Version Control

- Assignment 1 will be distributed Friday
- You MUST use version control
  - Subversion via Zeus (as in CS 300)
    - https://ankhsvn.open.collab.net/
  - Git via GitHub/GitLab
    - built into Visual Studio (instructions provided)
- You MUST email me by 5pm Thursday to let me know which system you choose
  - You will need a private Git repository!
    OR
  - You need to **`svnadmin create CS485S19`** on Zeus

Object Oriented Design

- Read Shalloway Chapter 1 by Wednesday
  - 30 pages
- Read Shalloway Chapter 2 by Friday
  - 15 pages

Syllabus

- Grade distribution
  - Outside Class Projects
  - Labs/Quizzes
  - Midterms (2)
  - Final Exam
- Grading 40/20/40 Exec/CodingStd/Design
  - Design and communication is more important than hacking together a working solution. Projects/Exams
  - Visual Studio Community Edition 2017
    - visualstudio.com - free!
    - www.umlet.com - free UML design tool
- Important Dates

Topics

Schedule

- CS 250 Review
  - Simple object hierarchy
  - virtual functions
  - pointers in C++/new/delete
- new C++ Topics
  - Copy Constructors, Operating Overloading
  - RAII
  - std::/STL/templates
  - Exceptions
  - C++11, C++14
    - move/smart pointers/rvalue reference/runtime type information/auto/lambda
- Design tools & techniques
- Design Patterns

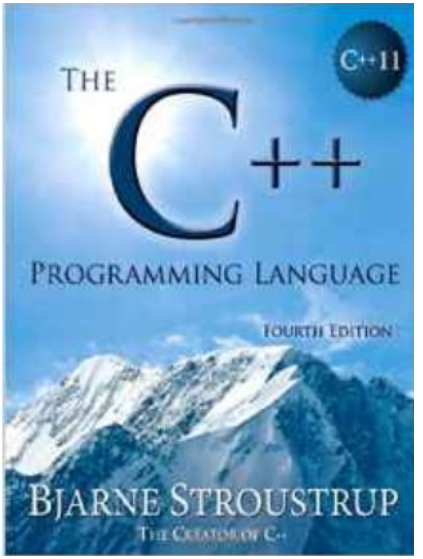
Required Text Books

- Design Patterns Explained: A New Perspective on Object Oriented Design, 2nd Edition, Alan Shalloway
  - http://www.netobjectives.com/resources/books/design-patterns-explained

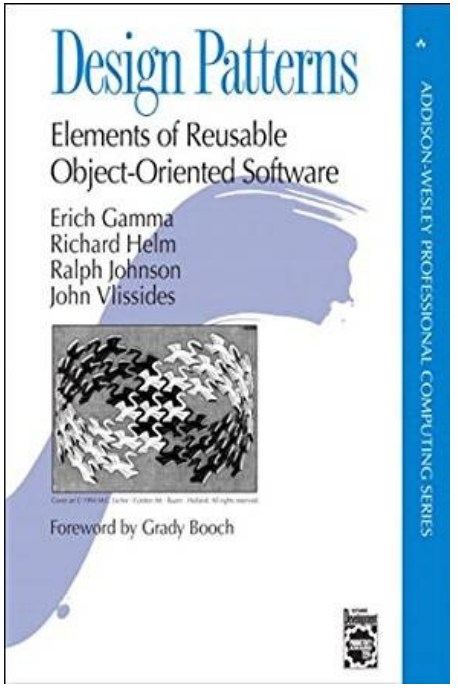
- Microsoft Developer Network:
  - https://msdn.microsoft.com/en-us/library/hh279654.aspx

Primary Sources

- The C++ Programming Language, **4th Edition**
  - Bjarne Stroustrup
  - http://www.stroustrup.com/
- Microsoft Developer Network:
  - https://msdn.microsoft.com/en-us/library/hh279654.aspx



- Gang Of Four
  - Gamma, Helm, Johnson, Vlissdes
- Christopher Alexander
  - architect (buildings, not software)
  - what are common patterns in architecture that humans enjoy



Other good books

★★★☆☆ Good technical introduction, but too preachy

- Object Oriented Software Construction
  - Bertrand Meyer - not C++, a classic text
- Effective Modern C++, Scott Meyers
  - Good notes on OO design specific to C++

- Starting Out with C++ From Control Structures through Objects, Gaddis, 8th Edition (CS150/CS250)
- UML Distilled: Martin Fowler
- Pattern Hatching: Vlissides
- Head First Design Patterns
- Head First Object Oriented Analysis & Design
- Holub on Patterns: Appendix
  - http://www.holub.com/goodies/holub_design_patterns.pdf
- Game Programming Patterns
  - http://gameprogrammingpatterns.com/

Object Oriented Design Principles

- Design Goals:

SOLID

- https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)

Cohesion/Coupling

CS 250 Review: Syntax

-
- C++ Syntax
  - public/private/protected
  - constructor/destructor
  - copy constructor
  - static members

What is an object?

- CS 250

- more theoretical CS 250

Object Lifecycle

Create/Initialize/Read

loop
    use/update

destroy

What is an object?

- CS 485

What is a class?

Class

```
class
{
  public:




  private:




}
```

Special Member Functions

```
default constructor

destructor

copy constructor


copy assignment operator
```

A small class

```
// you write
class small
{
  public:

  private:
    int *mpData=nullptr;
}
```

```
// the compiler builds
class small
{
  public:

  private:
    int *mpData=nullptr;
}
```

Using small

```
small s1;

small s2(s1);

small s3;

s3=s1;
```

- Effective C++, Meyers, Item 5

Copy Constructor and =

```
class bigger
{
public:
  bigger() { std::cout << "ctor\n"; }

  bigger(int x) : mData(x) {std::cout << "ctor(i)\n";}

  ~bigger() { std::cout << "dtor\n"; }

  bigger(const bigger&rcData)
  {
    std::cout << "cctor\n"; mData = rcData.mData;
  }

  bigger& operator=(const bigger&rcData)
  {
    std::cout << "op=\n"; mData = rcData.mData;
    return *this;
  }

private:
  int mData = 0;
};
```

Who is called?

```
void foo(bigger b4);
bigger bar(bigger b5);

  cout << "b1\n";
  bigger b1;
  cout << "end b1\n";

  cout << "b2\n";
  bigger b2(b1);
  cout << "end b2\n";

  cout << "b3\n";
  bigger b3 = b1;
  cout << "end b3\n";

  cout << "b6\n";
  bigger b6;
  cout << "end b6\n";

  cout << "foo\n";
  foo(b1);
  cout << "end foo\n";

  cout << "bar\n";
  b6 = bar(b1);
  cout << "end bar\n";
  cout << "b7\n";
  bigger b7=1;
  cout << "end b7\n";
```

https://gitlab.com/chaddcw/CS485_Student_Examples

CS 250 Review: Design

- Composition
- Aggregation
- Inheritance
- Polymorphism
  - virtual functions/pure virtual
  - abstract classes

Terms

- What is a class?
- Abstract vs Concrete
- What is an interface?
- What is a method signature?

What is inheritance?

- CS 250

- CS 485

- Base/Super class
- Derived/child class

Polymorphism

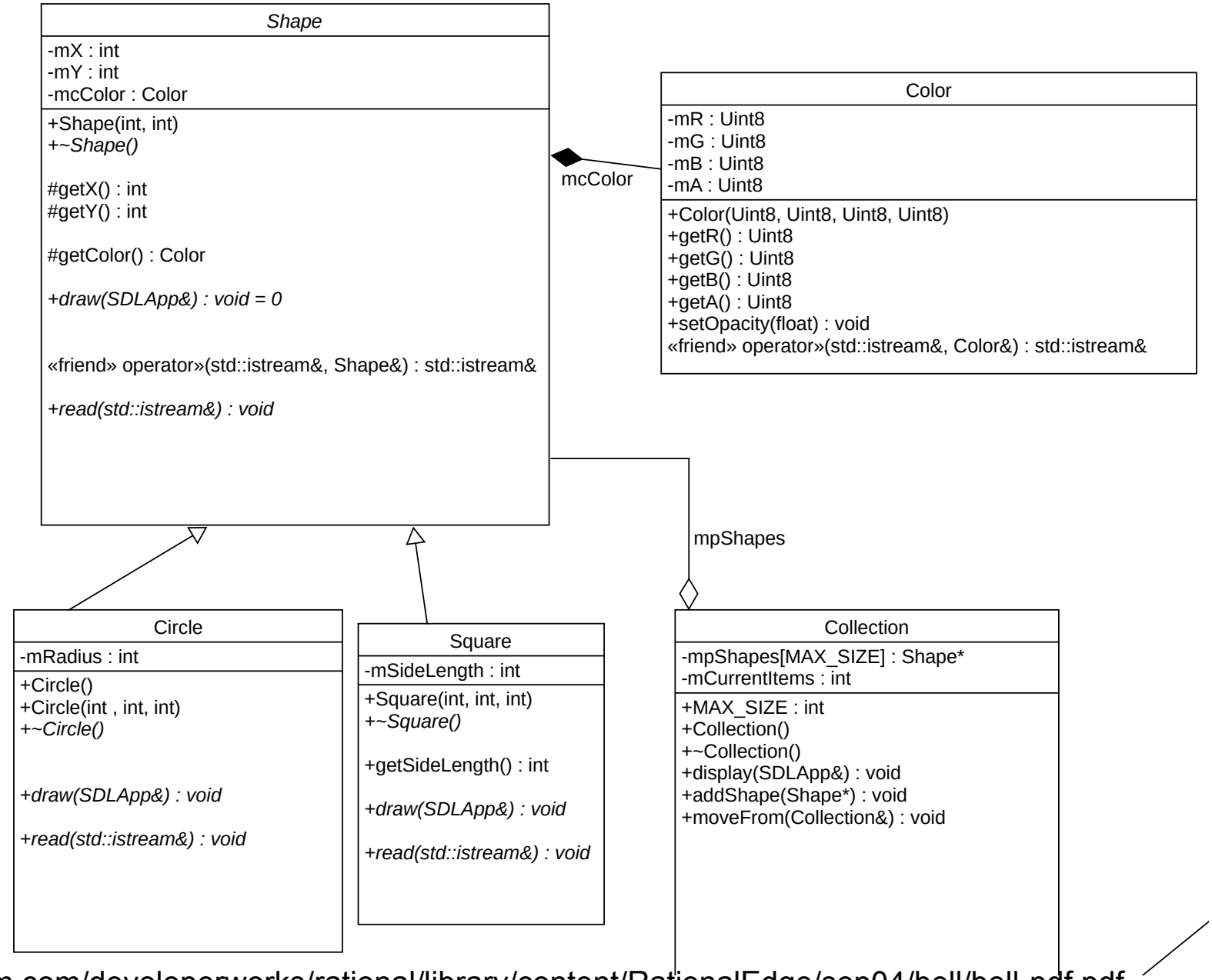
- virtual function

Example

```
class
{
  public:




  private:




}
```

```
class
{
  public:




  private:




}
```

UML



https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/bell-pdf.pdf
https://www.martinfowler.com/bliki/UmlAsSketch.html
http://umich.edu/~eecs381/handouts/UMLNotationSummary.pdf

Inhertiance vs Composition

Inheritance

- Subclasses
  - benefits
  - disadvantages

Composition

- One object contains another
  - benefits
  - disadvantages

Differences

Use Cases

- Inheritance

- Composition

setters/getters