CS 485
Advanced Object Oriented Design

Spring 2017

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 (as in CS 260/360)
    - built into Visual Studio (instructions provided)
- You MUST email me by 5pm Thursday to let me know which system you choose
  - I will create private GitHub repositories
    - email me your GitHub account!
  - You need to **`svnadmin create CS485S17`** on Zeus

Object Oriented Design

- Read Shalloway Chapter 1 by Wednesday
- Read Shalloway Chapter 2 by Friday

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 2015
    - 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

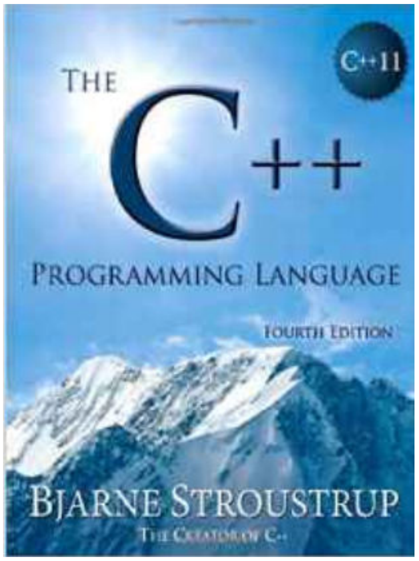
- Design Patterns Explained: A New Perspective on Object Oriented Design, 2nd Edition, Alan Shalloway
  - http://www.netobjectives.com/resources/books/design-patterns-explained
- Effective Modern C++, Scott Meyers

  http://www.aristeia.com/BookErrata/emc++-errata.html

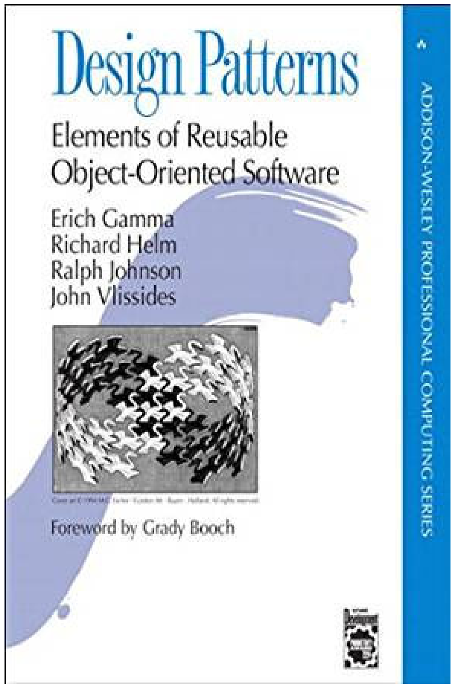
- Microsoft Developer Network:
  - https://msdn.microsoft.com/en-us/library/hh279654.aspx

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


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Other good books

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

- Object Oriented Software Construction
  - Bertrand Meyer - not C++, a classic text
- Effective C++, Scott Meyers
  - dated syntax but 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

Agile Software Development

- Value:

  Individuals and interactions over processes and tools

  Working software over comprehensive documentation

  Customer collaboration over contract negotiation

  Responding to change over following a plan

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

http://agilemanifesto.org/

http://www.hillside.net/

Object Oriented Design Principles

- Design: isolate what may change to lessen impact (data or functionality)
  - encapsulate
  - polymorphism
- Communicate your design
  - Speak the language
- Less code intensive, more theory.
- No perfect solution: trade offs
  - some terribly wrong solutions, though.

CS 250 Review

- Encapsulation
  - isolate what could change
    - data or functionality
  - Code to an interface
    - the general, not the specific
- Class
  - public/private/protected
  - constructor/destructor
  - copy constructor
  - static members

What is an object?

- CS 250
- more theoretical CS 250
- CS 485

Object Lifecycle

Create/Initialize/Read

loop
    use/update

destroy

What is a class?

Class

```
class
{
  public:









  private:






}
```

A small class

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

  private:
    int mData=0;
}
```

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









  private:
    int mData;
}
```

Using small

```
small s1;

small s2(s1);

small s3;

s3=s1;
```

- Effective C++, Meyers, Item 5

Copy Constructor and =

- Who is called?

```
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;
};
```

```
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://github.com/cs485s17/CS485_Student_Examples

CS 250 Review

- Composition
- Aggregation
- Inheritance

- Pointers
  - new/delete
  - this
  - NULL/nullptr
- Polymorphism
  - virtual functions/pure virtual
  - abstract classes

Terms

- What is a class?

- Base/Super class
- Derived/child class - what can you inherit?

- What is an abstract class? Concrete class?

- What is an interface?

- What is a method signature?

What is inheritance?

- CS 250

- CS 485

Example

```
class
{
  public:








  private:






}
```

```
class
{
  public:








  private:






}
```

Stop Lec1

pointers! new/delete

- Dynamic memory
  - where does it come from?
  - how do we get it?

Classes that contain dynamic memory

- Or any dynamic resource (file, network, ...)

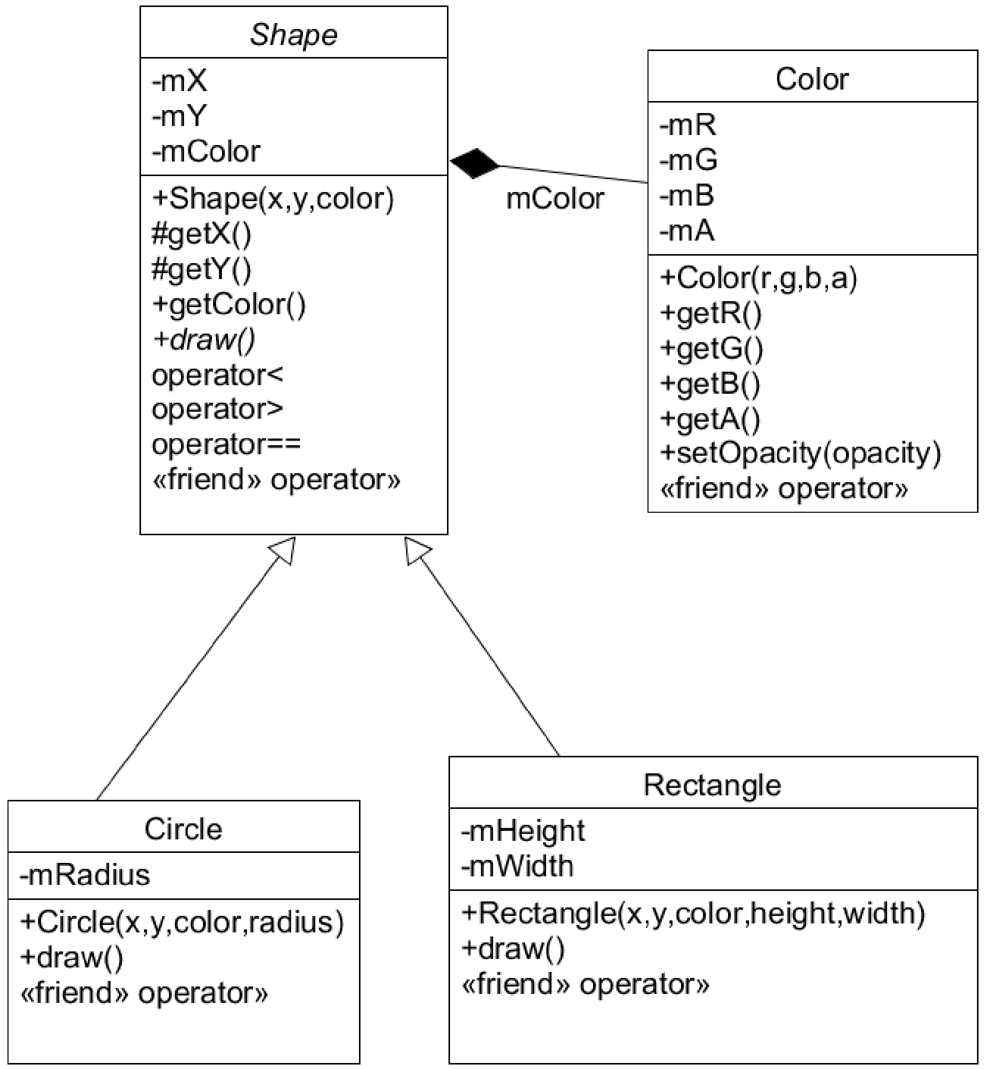
```
class bigData
{
  public:
    bigData();
    ~bigData();
    bigData(const bigData &rcData);

    bigData& operator=(const bigData &rcData);

    bigData& operator=(bigData cData); //force copy constructor to be called


  private:
    int *mpHugeData;
}
```

UML Design



http://umich.edu/~eecs381/handouts/UMLNotationSummary.pdf

https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/bell-pdf.pdf

https://www.martinfowler.com/bliki/UmlAsSketch.html

SOLID

- https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)
- http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Cohesion/Coupling

setters/getters

Rvalue refences & Move Semantics

Exceptions

- Safety
  - none
  - basic
  - strong
  - no-throw guarantee