

CS 635 Advanced Object-Oriented Design & Programming  
Spring Semester, 2011  
Doc 1 Introduction  
Jan 20, 2011

Copyright ©, All rights reserved. 2011 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (<http://www.opencontent.org/opl.shtml>) license defines the copyright on this document.

# References

Object-Oriented Design Heuristics, Riel,  
Addison Wesley, 1996

# Reading

Jan 25 - Big Ball of Mud, <http://www.laputan.org/mud/mud.html>

Jan 27 - Refactoring, Chapters 1 & 2

Feb 1 - Refactoring, Chapters 3 & 4

# Crashing

Last Day to Drop

Feb 1

Last Day to Add

Feb 3

# Course Web Site

<http://www.eli.sdsu.edu/index.html>

CS 635 Spring 11

Lecture Notes

Assignments

Wiki

~~Mailing List~~

Course Portal

Syllabus

Reading Assignments

# Languages

Java, C++, C#, Ruby, Objective C or Smalltalk

# Preferred Languages

Java  
Smalltalk

Objective C

Ruby

# C#

Programs have to run in Mono

It is your responsibility to insure this

No support



# C++ is **STRONGLY** Discouraged

I have not used C++ in over 10 years

I don't like the language

It is very difficult to grade

Each additional language make grading harder

It is extremely hard to deal with GUI assignments in C++

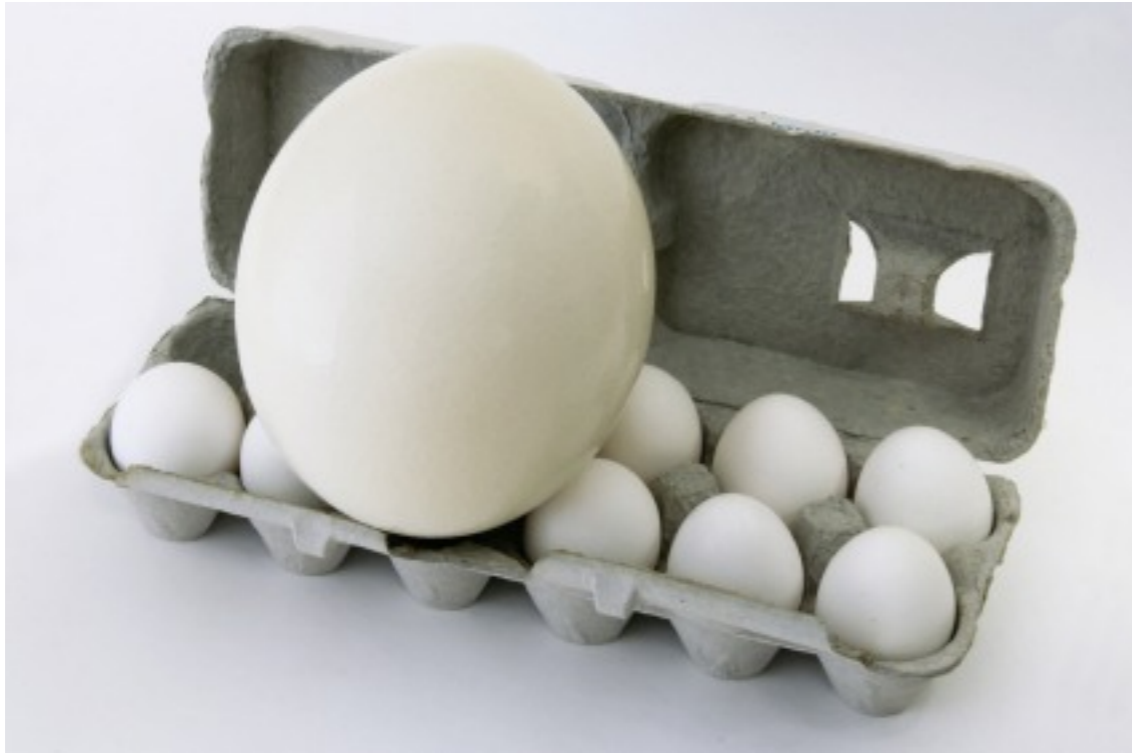
Assignments are often harder in C++

# What this course is about

Writing quality OO code  
Design Patterns  
Coupling & Cohesion

Unit Testing  
Refactoring

# Scale Changes Everything



# Review

# Define

Object  
Class

# What are the Benefits of OO

# Issues?

```
public class A {  
    public int x;  
    public int y;  
    public int z;  
}
```

# Issues?

```
class Stack
  def initialize
    @elements = Array.new
  end

  def empty?
    return @elements.empty?
  end

  def push(element)
    @elements.push(element)
  end

  def pop
    @elements.pop
    return elements
  end
end
```



# A verses B

```
public class A {  
    public int x;  
    public int y;  
    public int z;  
}
```

```
public class B {  
    private int x;  
    private int y;  
    private int z;  
  
    public int getX() { return x;}  
    public int getY() { return y;}  
    public int getZ() { return z;}  
    public void setX(int value) {x = value;}  
    public void setY(int value) {y = value;}  
    public void setZ(int value) {z = value;}  
}
```

# Heuristics

Keep related data and behavior in one place

A class should capture one and only one key abstraction

# Heuristics

Beware of classes that have many accessor methods defined in their public interface

Do not create god classes/objects in your system

Beware of classes that have too much noncommunicating behavior