Class Design in OO

Ridi Ferdiana | http://ugm.id/ridif
Version 3.0.0

Overview

- C# Overview
- Abstraction in Class
- Encapsulation in Class
- Inheritence in Class

C# as programming language

C# Structure

```
using System;

class Hello
{
  public static void Main()
  {
    Console.WriteLine("Hello, DTETI");
  }
}
```

The Class

- A C# application is a collection of classes, structures, and types
- Syntax

```
class name
{
        ...
}
```

- A C# application can consist of many files
- A class can be spanned multiple files (partial)

The Main Method

- When writing Main, you should:
  - Use an uppercase “M”, as in “Main”
  - Designate one **Main** as the entry point to the program
  - Declare **Main** as **public static void Main**
- Multiple classes can have a Main but is not Recomended
- When Main finishes, or returns, the application quits

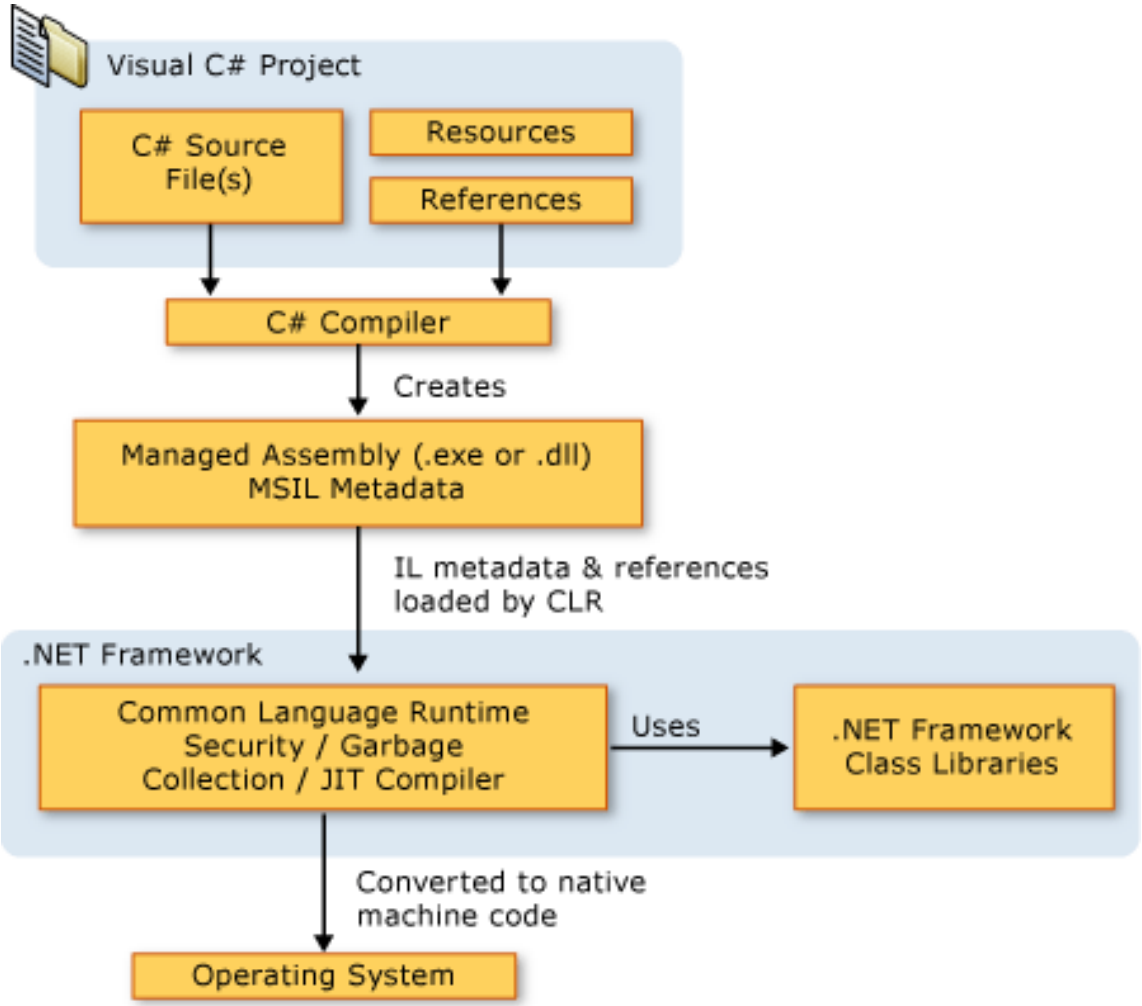
The Directive

- The .NET Framework provides many utility classes
  - Organized into namespaces
- System is the most commonly used namespace
- Refer to classes by their namespace

```
using System;
…
Console.WriteLine("Hello, World");
```

- The using directive

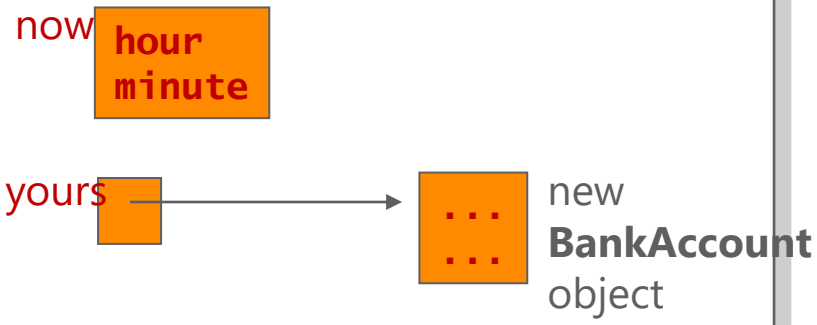
C# Build Model



Instantiating New Objects

- Declaring a class variable does not create an object
  - Use the **new** operator to create an object

```
class Program
{
    static void Main( )
    {
        Time now;
        now.hour = 11;
        BankAccount yours = new BankAccount( );
        yours.Deposit(999999M);
    }
}
```



Using the this Keyword

- The this keyword refers to the object used to call the method
  - Useful when identifiers from different scopes clash

```
class BankAccount
{
    ...
    public void SetName(string name)
    {
        this.name = name;
    }
    private string name;
}
```

If this statement were
name = name;
What would happen?

Creating Nested Classes

Classes can be nested inside other classes

```
class Program
{
    static void Main( )
    {
        Bank.Account yours = new Bank.Account( );
    }
}
class Bank
{
    ... class Account { ... }
}
```

The full name of the nested class includes the name of the outer class

Accessing Nested Classes

Nested classes can also be declared as public or private

```
class Bank
{
    public  class Account { ... }
    private class AccountNumberGenerator { ... }
}
class Program
{
    static void Main( )
    {
        Bank.Account                    accessible; ✓
        Bank.AccountNumberGenerator inaccessible; ✗
    }
}
```

C# on Windows Form

Demo

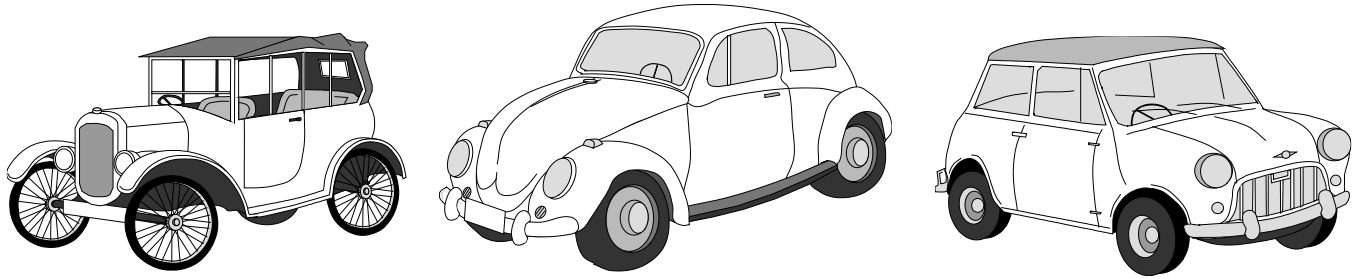
Asbtraction in Class

What Is a Class?

- For the philosopher…
  - An artifact of human *class*ification!
  - *Class*ify based on common behavior or attributes
  - Agree on descriptions and names of useful *class*es
  - Create vocabulary; we communicate; we think!
- For the object-oriented programmer…
  - A named syntactic construct that describes common behavior and attributes
  - A data structure that includes both data and functions

What Is an Object?

- An object is an instance of a class
- Objects exhibit:
  - Identity: objects uniques ID (object name)
  - Behavior: Objects can perform tasks (method)
  - State: Objects store information (attribute)



Comparing Classes to Structs

- A struct is a blueprint for a value
  - Composition of types value
  - Better use for computation purposes
- A class is a blueprint for an object
  - Composition of types value and other objects
  - Better use for complex transaction purposes

```
struct Time                         class BankAccount
{                                   {
    public int hour;                    ...
    public int minute;                  ...
}                                   }
```

Struct vs Class

Demo

Class Category

Instanced Class (Object)	• Common implementation of class
Static Class	• Class that has no state
Abstract Class	• Class that works as a template of others class
Sealed Class	• Class that cannot be inherited

Abstraction is implemented by Default in a class

Abstraction is selective ignorance

Decide what is important and what is not

Focus and depend on what is important

Ignore and do not depend on what is unimportant

Use encapsulation to enforce an abstraction

The purpose of abstraction is not to be vague,
but to create a new semantic level in which one can be absolutely precise.
Edsger Dijkstra

Abstraction in Class

Demo

Encapsulation in Class

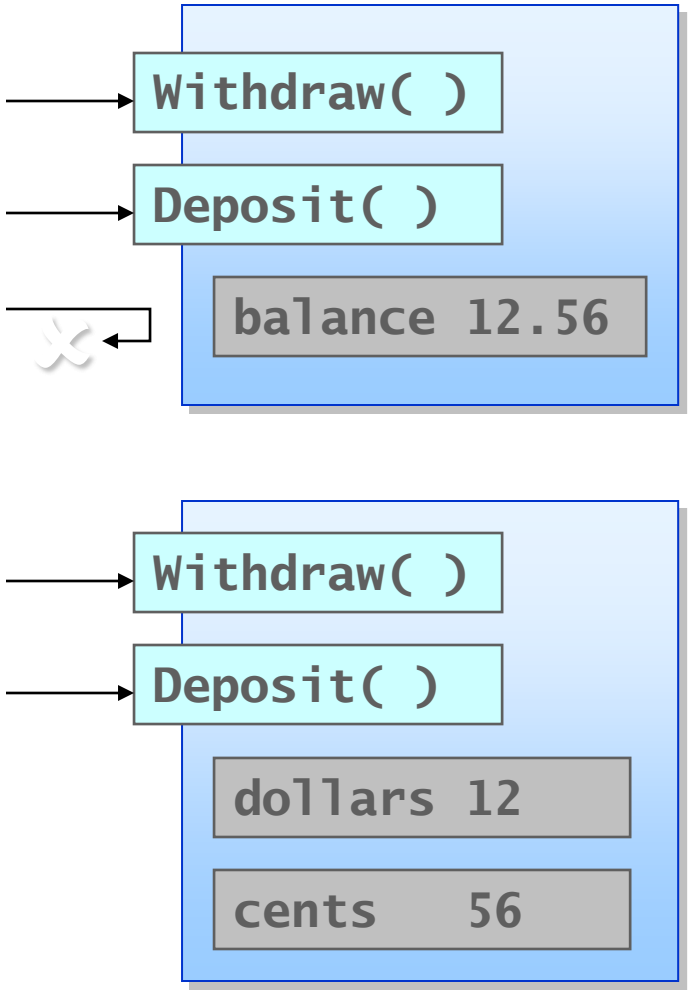
Why Encapsulate?

Allows control

Use of the object is solely through the public methods

Allows change

Use of the object is unaffected if the private data type changes

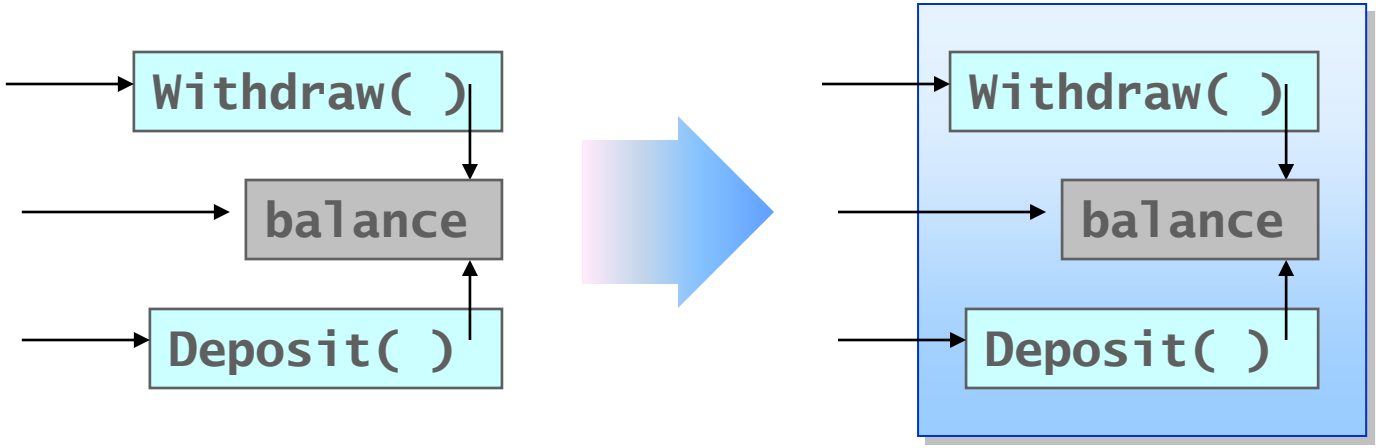


Implement Encapsulation in Class

- Combining Data and Methods
- Controlling Access Visibility
- Object Data
- Using Static Data
- Using Static Methods

Combining Data and Methods

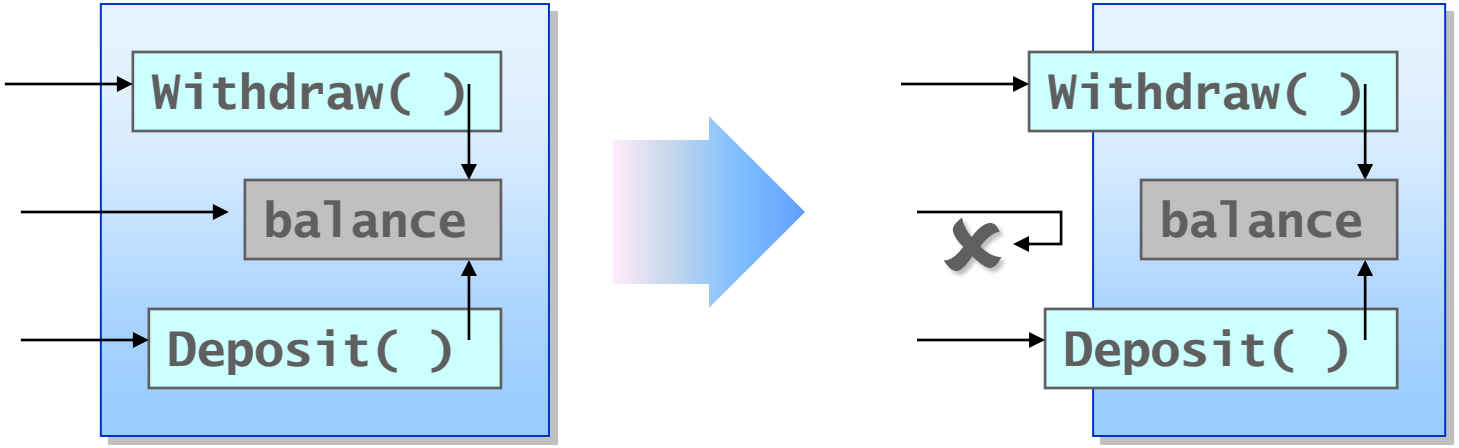
- Combine the data and methods in a single *capsule*
- The capsule boundary forms an inside and an outside



BankAccount ?

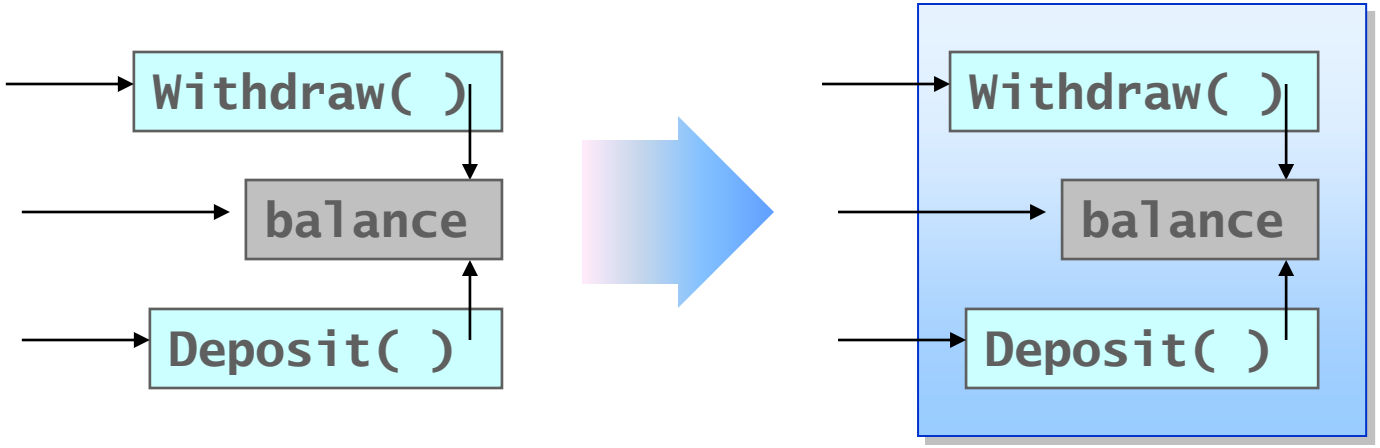
Controlling Access Visibility

- Methods are *public*, accessible from the outside
- Data is *private*, accessible only from the inside



Combining Data and Methods

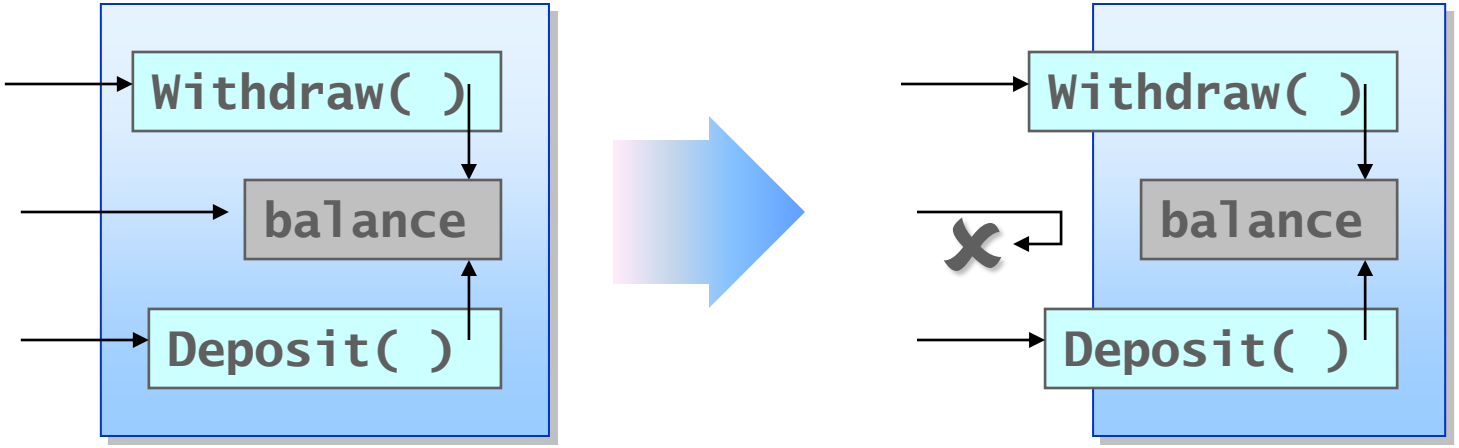
- Combine the data and methods in a single *capsule*
- The capsule boundary forms an inside and an outside



BankAccount ?

Controlling Access Visibility

- Methods are *public*, accessible from the outside
- Data is *private*, accessible only from the inside



Encapsulation in Class

Demo

Inheritence in Class

Revisiting Principles of Good Classes

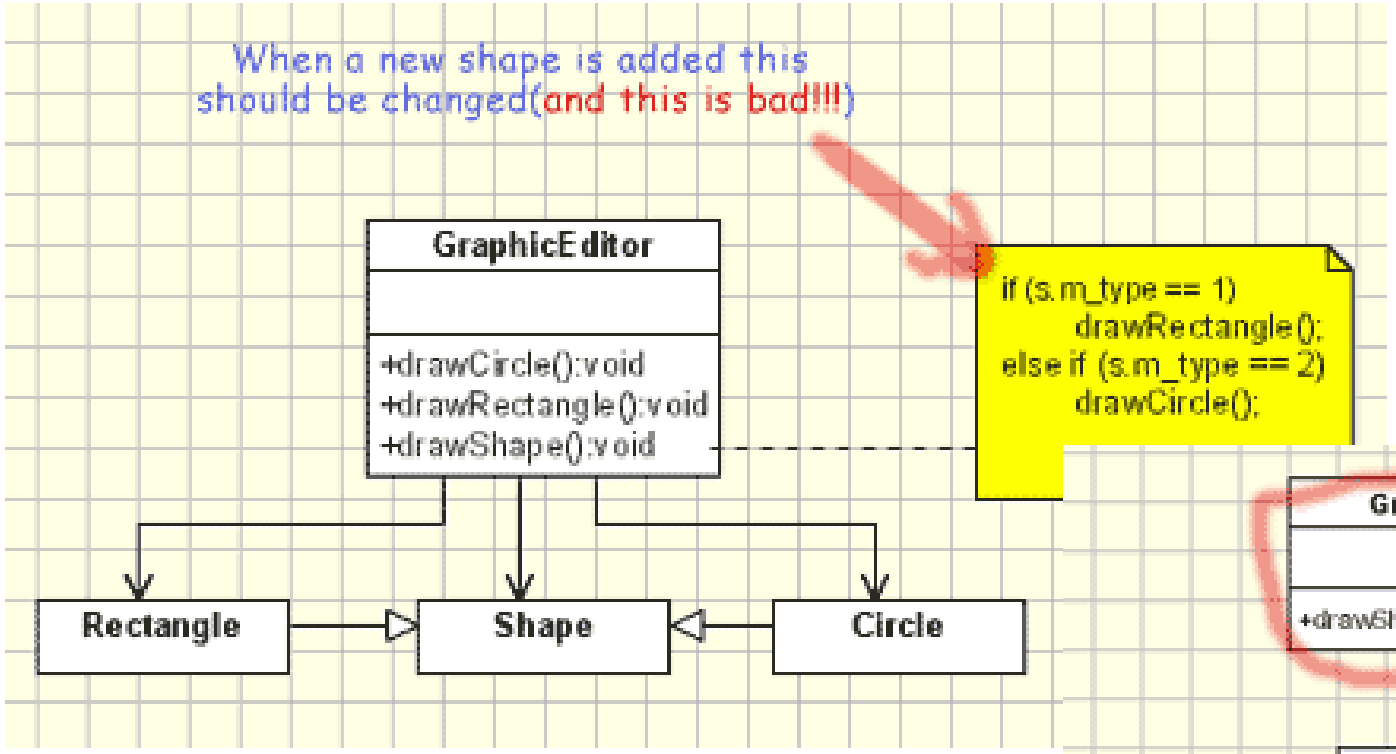
- Data and methods together inside a class
- Methods are public, data is private

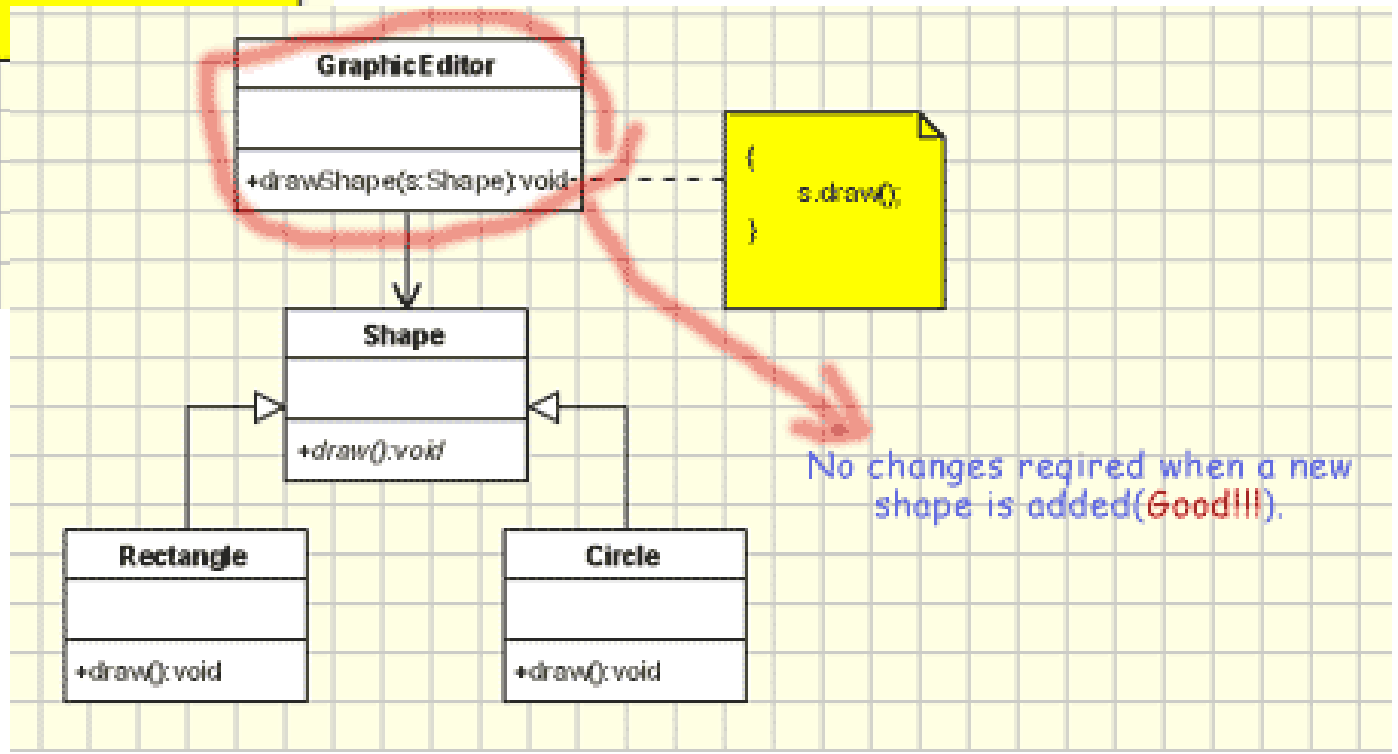
```
class BankAccount
{
    public void Withdraw(decimal amount)
    { ... }
    public void Deposit(decimal amount)
    { ... }
    private decimal balance;
    private string name;
}
```

Public methods describe accessible behaviour

Private fields describe inaccessible state

OCP (Open Close Principle)

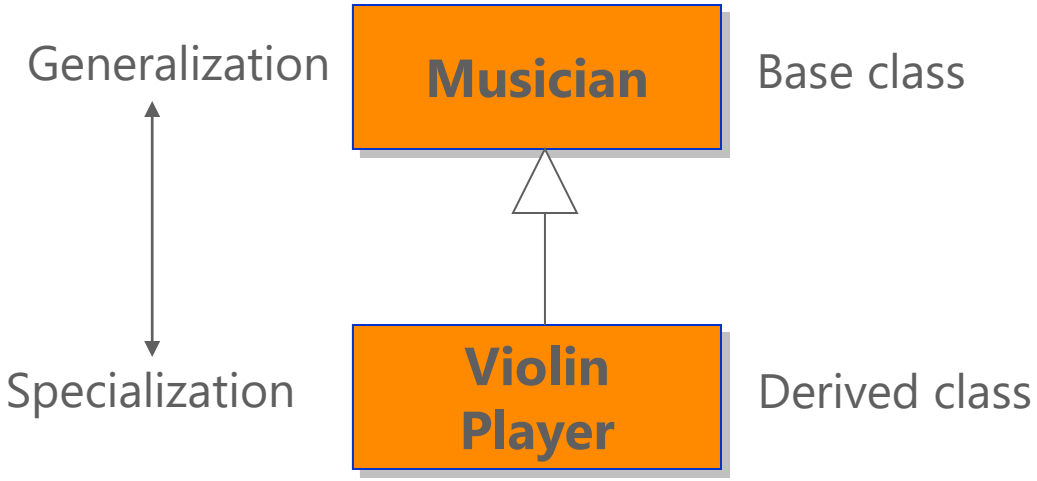




[Open Close Principle | Object Oriented Design (oodesign.com)]

Inheritance

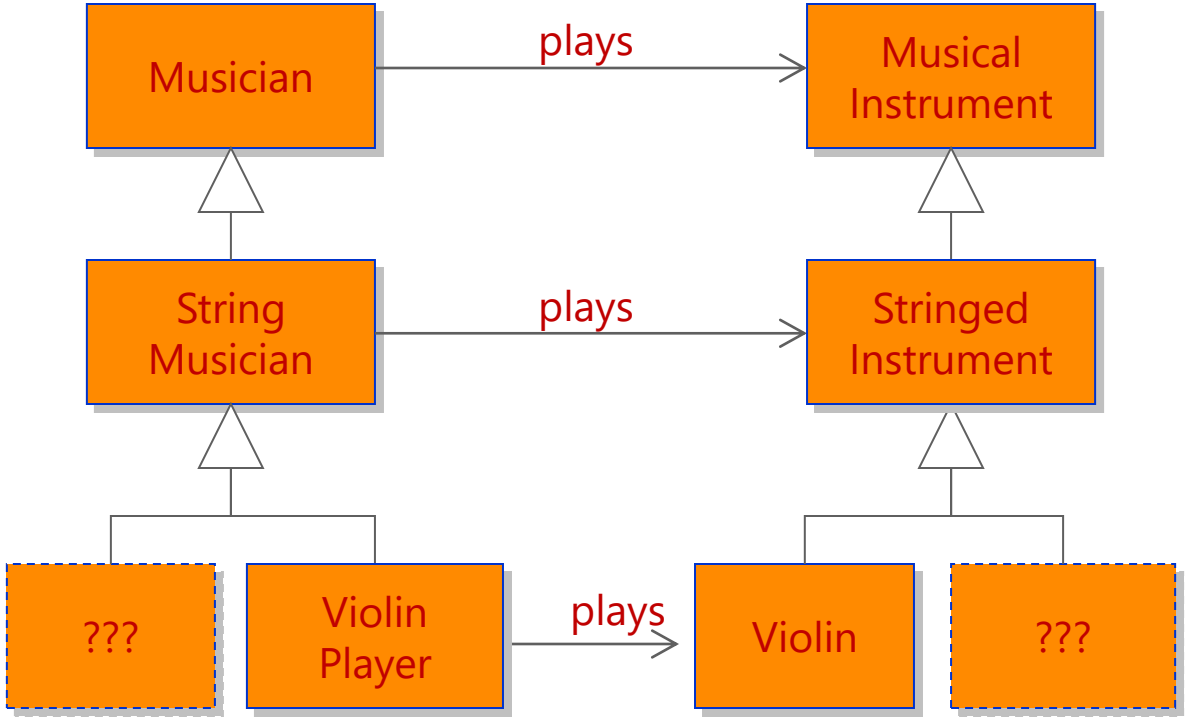
- Inheritance specifies an “is a kind of" relationship
  - Inheritance is a class relationship
  - New classes specialize existing classes



Is this a good example of inheritance ?

Class Hierarchies

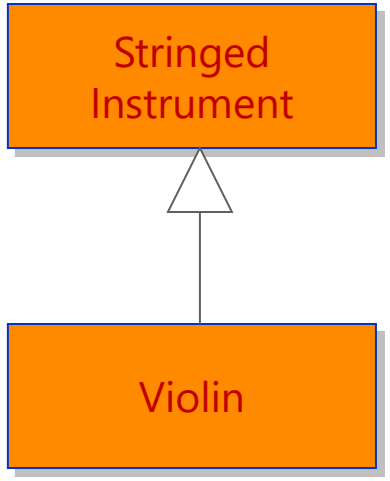
- Classes related by inheritance form class hierarchies



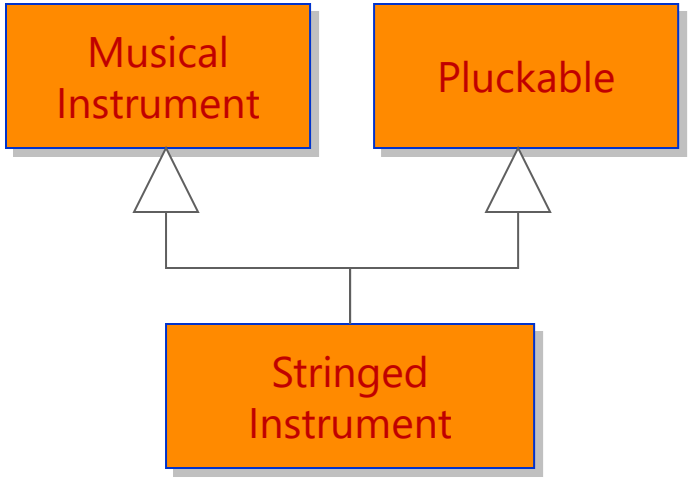
Single and Multiple Inheritance

Single inheritance: deriving from one base class

Multiple inheritance: deriving from two or more base classes



Violin has a single direct base class



Stringed Instrument has two direct base classes

Inheritance in Class

Demo

Conclusion

- C# programming language provides full support of OO concept
- Class, or Struct? Is about performance, reusable, and purpose
- Class concept fulfill 3 concepts of OO – Abstraction, Encapsulation, and Inheritance