Chapter 8 – Object-Based Programming

8.3 Class Scope

8.4 Controlling Access to Members

8.5 Initializing Class Objects: Constructors

8.6 Using Overloaded Constructors

8.7 Properties

8.8 Composition: Objects References as Instance Variables of Other Classes

8.9 Using the `this` Reference

8.10 Garbage Collection

8.11 `static` Class Members

OOP

- Object Oriented Programming (OOP)
  - *Encapsulates* data (attributes) and methods (behaviors)
- Program use objects to simulate the real entity of the world
  - Attributes simulate the entity's properties,
    - like CustomerNumber, name, address of a customer's entity
  - Methods simulate the entity's actions,
    - like order, changeOrder

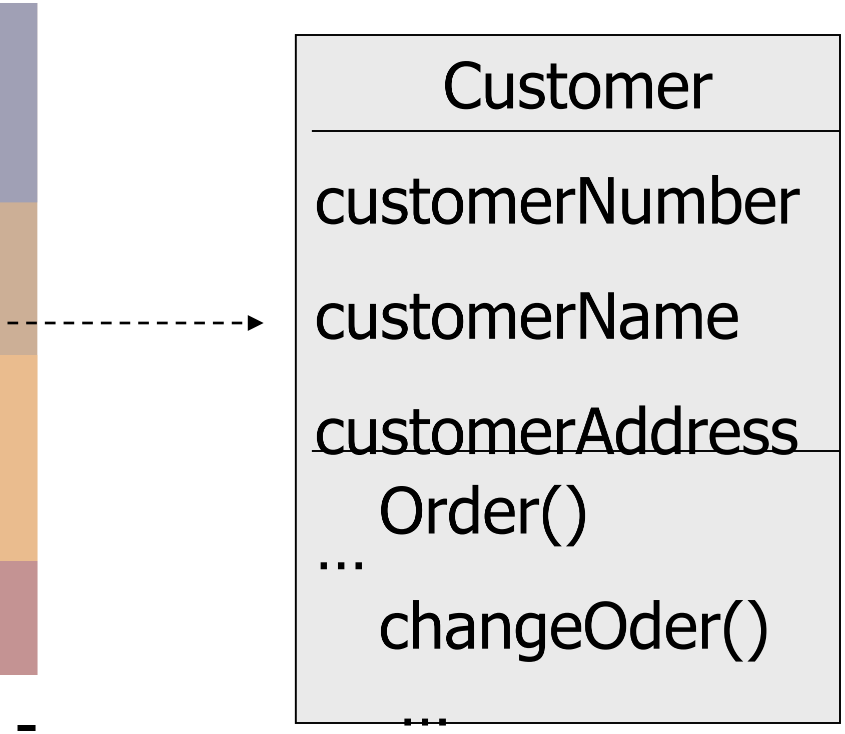
◀ ▶

Modeling related things with a class

Physical reality -

Person who is a

customer



Customer

customerNumber

customerName

customerAddress

…

Order()

changeOder()

…

– Class name

– Class attributes

– Class behaviors (methods, processes/functions working with data)

a class Customer

(DB term "Entity")

Legend of Class

- Suppose you need to compute the average score of a class (算班平均).

In C (loose), you may write

```
….
string  name[60];
int eng[60], math[60];
float  sumEng, sumMath;
…..
for  (i=1; i<60; i++){
    sumEng+= eng[i];
    sumMath+=math[i];
}
…..
```

in pascal (not solid enough), you may write

```
….
type score = record
        name : string;
        eng, math: integer;
end;
….
sumEng, sumMath: float;
exam[60]:  score;
…..
for  (i=1; i<60; i++){
     sumEng+= exam[i].eng;
     sumMath+=exam[i].math;
}
```

in C#, Java (solid), you may write….

```
Class Score {
      public  string  name;
      private  int  eng, math;
      public void calAvg(score s[]) {
        float sumEng, sumMath;
         ……
         for  (i=1; i<s.length; i++){
            sumEng+= s[i].eng;
            sumMath+=s[i].math;
           …..   }      }
};
Score  exam[60];
….
calAvg(exam);
….
```

Legend of Class

- In traditional C,
  - You need to be aware of the grouping of data scattered in different sets of variables
    - Difficult in maintaining if abundant data
- In Pascal, …
  - You use "record" to group a related data as a unit for easily maintaining
  - But you still separate the methods for the data away from the data unit
- In C#,
  - You use "class" to group data and its related methods as a unit
    - Capture the abstract attributes and behaviors of a real entity as a class at first
    - Some data can be sealed from outside to avoid inappropriate use

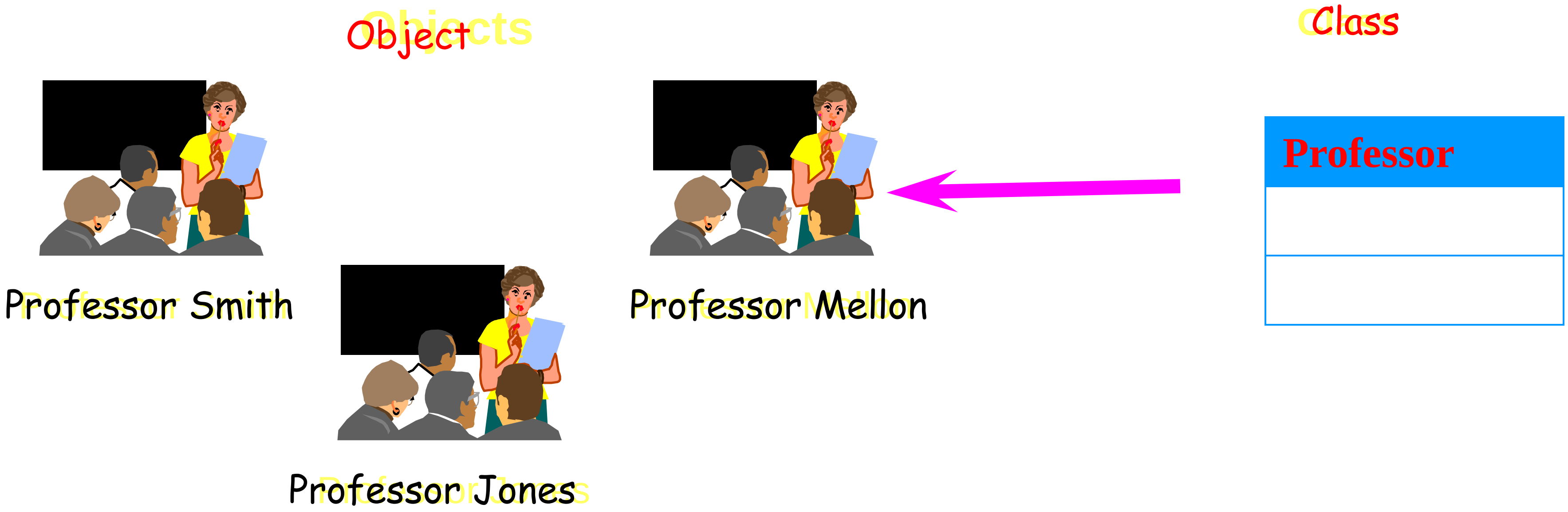
Object vs. class

- An Object is an instance of class
  - something that is perceived as an entity and referred to by name, like John, Marry
- A class is a template of objects
  - It contains a set of objects that have the same traits (attributes and operations)
- class is like a blueprint
  - reusable
  - A class can produce many its duplication
- Objects are instantiated (created) from the class
  - For example, a house is an instance of a "blueprint"

◀ ▶

The Relationship Between Classes and Objects

- A class is an abstract definition of an object
  - It defines the structure and behavior of each object in the class
  - It serves as a template for creating objects
  - For example, a specific professor is an instance of a “professor” class



Object

Objects

Class

Professor

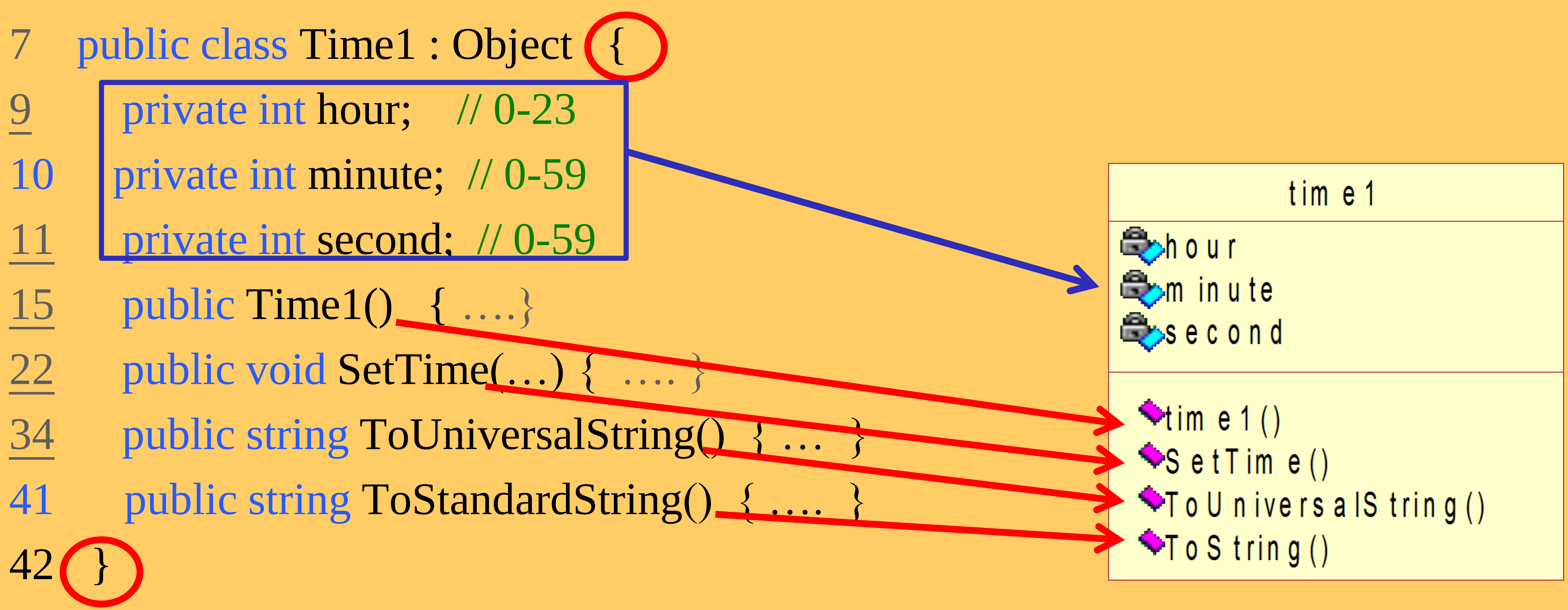
Professor Smith

Professor Mellon

Professor Jones

Time Abstract Data Type with a Class

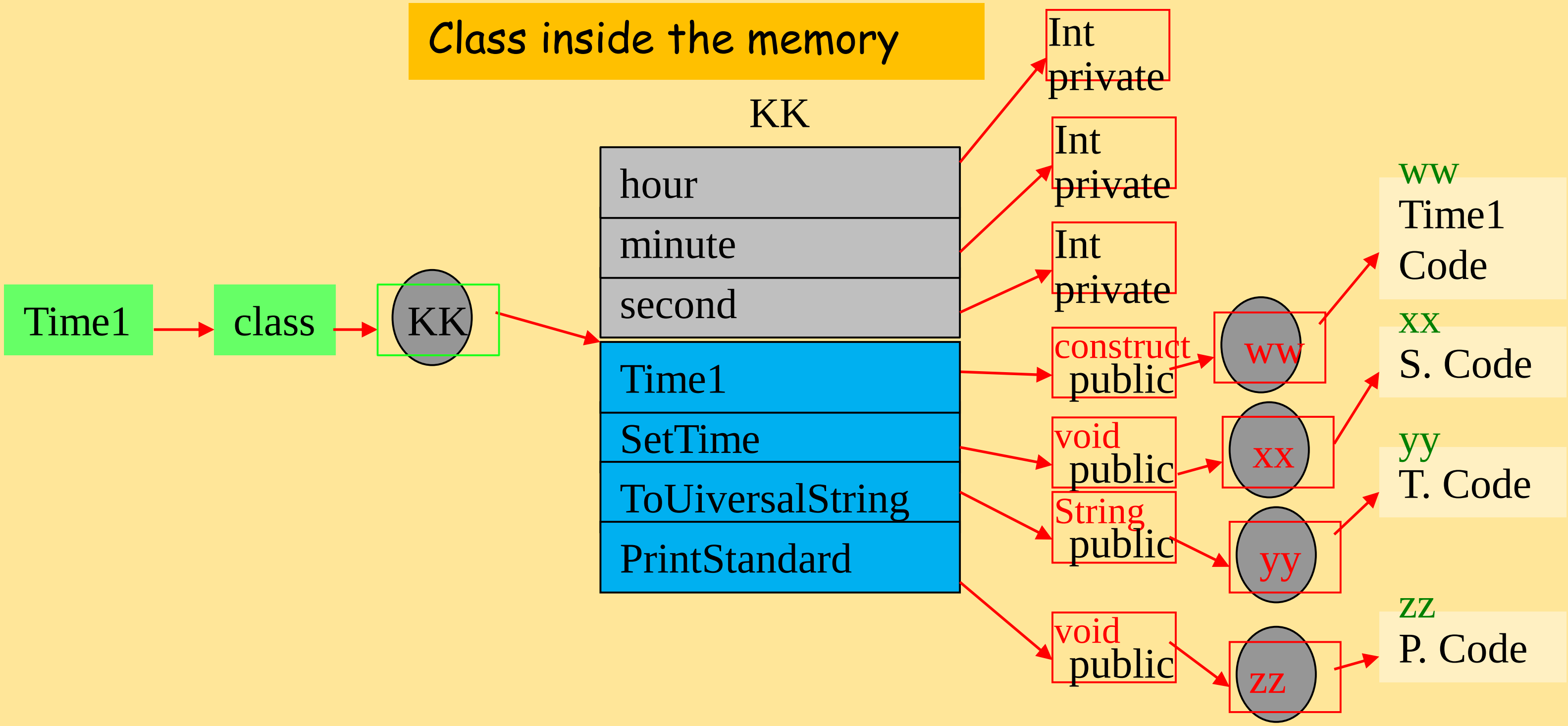
- Classes
  - Model objects that have attributes (data members) and behaviors (member functions)
  - Defined using keyword **class**
  - Have a body delineated with braces ({ and })



```
public class Time1 : Object {
    private int hour;     // 0-23
   private int minute;  // 0-59
    private int second;  // 0-59
    public Time1()   { ….}
    public void SetTime(…) {  …. }
    public string ToUniversalString()  { …  }
    public string ToStandardString()  { ….  }
}
```

```
public class Time1 : Object    {
    private int hour;     // 0-23
    private int minute;   // 0-59
    private int second;   // 0-59
    public Time1()    { ….}
    public void SetTime(…)  {   ….  }
    public string ToUniversalString()   { …   }
    public string ToStandardString()   { ….   }
}
```

Class inside the memory



Implementing a Time Data Type with a Class

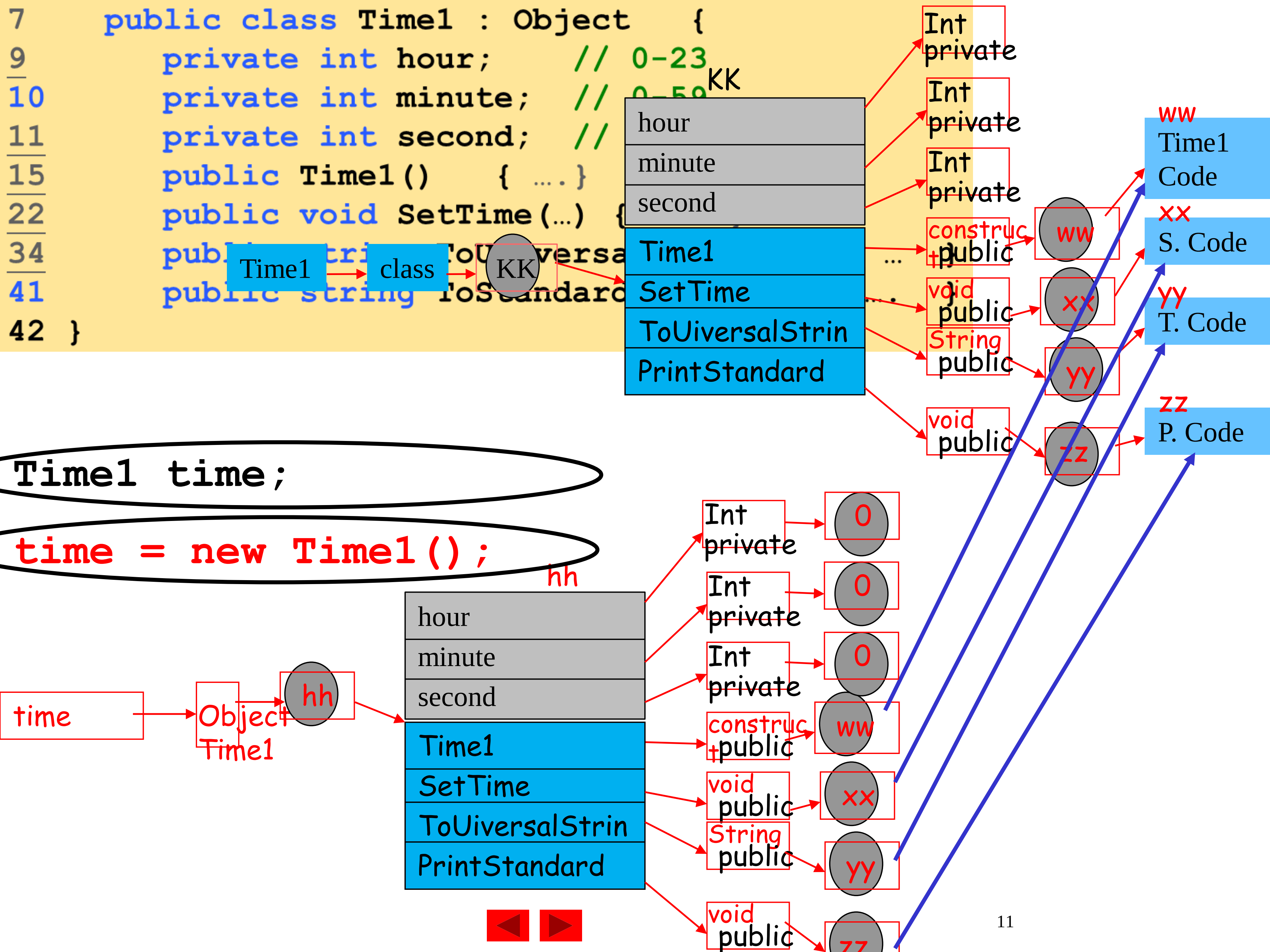
- Class definition and declaration
  - Once a class has been defined, it can be used as a data type in object, array
  - Example:

```
Time1 sunset = new Time1(); //object of classTime

Time1 classtime[];
```

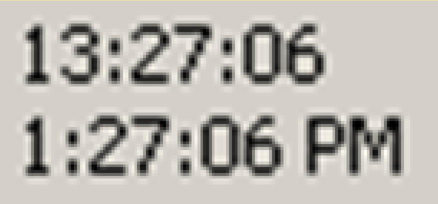
```
public class Time1 : Object    {
    private int hour;      // 0-23
    private int minute;    // 0-59
    private int second;    //
    public Time1()    { ....}
    public void SetTime(...)  {
    public string ToUniversa
    public string ToStandard
}
```

Time1 time;

time = new Time1();

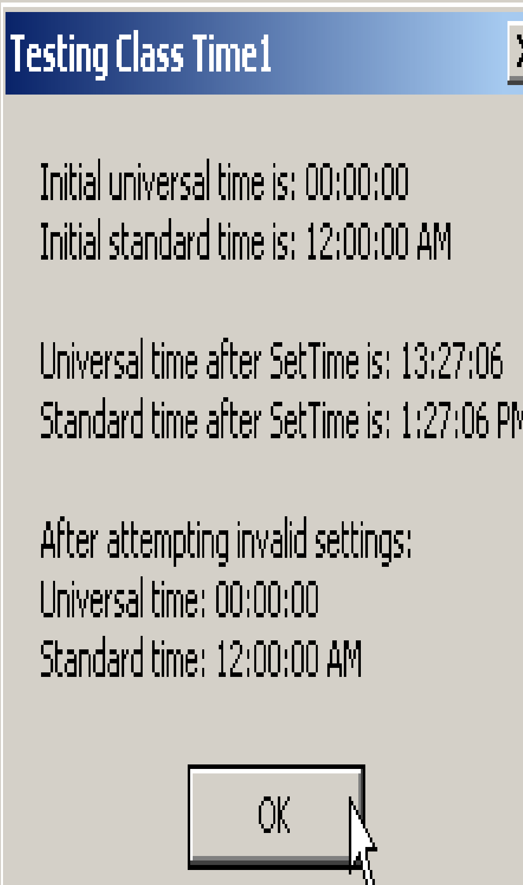


```
public class Time1 : Object   {
   private int hour;     // 0-23
    private int minute;  // 0-59
    private int second;  // 0-59
    public Time1() {
       SetTime( 0, 0, 0 );
    }
    public void SetTime(int hourValue, int minuteValue, int secondValue ) {
       hour=(hourValue>=0 && hourValue<24)? hourValue:0;
       minute=(minuteValue >= 0 && minuteValue < 60 ) ? minuteValue : 0;
       second =(secondValue >= 0 && secondValue < 60 ) ? secondValue : 0;
       }
    public string ToUniversalString() {
      return String.Format("{0:D2}:{1:D2}:{2:D2}", hour, minute, second );
     }
    public string ToStandardString() {
      return String.Format("{0}:{1:D2}:{2:D2} {3}",
         ((hour == 12||hour == 0 ) ? 12 : hour % 12 ),
         minute, second, (hour < 12 ? "AM" : "PM" ) ); }
     }
 }
```



```
13:27:06
1:27:06 PM
```

```
using System;
class TimeTest1  {
   static void Main( string[] args )    {
      Time1 time = new Time1();
      string output;
      output= “Initial universal time is: ” + time.ToUniversalString()
          + "\nInitial standard time is:" +time.ToStandardString();
      time.SetTime( 13, 27, 6 );
      output += "\n\nUniversal time after SetTime is: " +
        time.ToUniversalString() + "\nStandard time after
        SetTime is: " + time.ToStandardString();
      time.SetTime( 99, 99, 99 );
      output += "\n\nAfter attempting invalid settings: " +
        "\nUniversal time: " + time.ToUniversalString() +
        "\nStandard time: " + time.ToStandardString();
      MessageBox.Show( output, "Testing Class Time1" );
   }
 }
```



Testing Class Time1

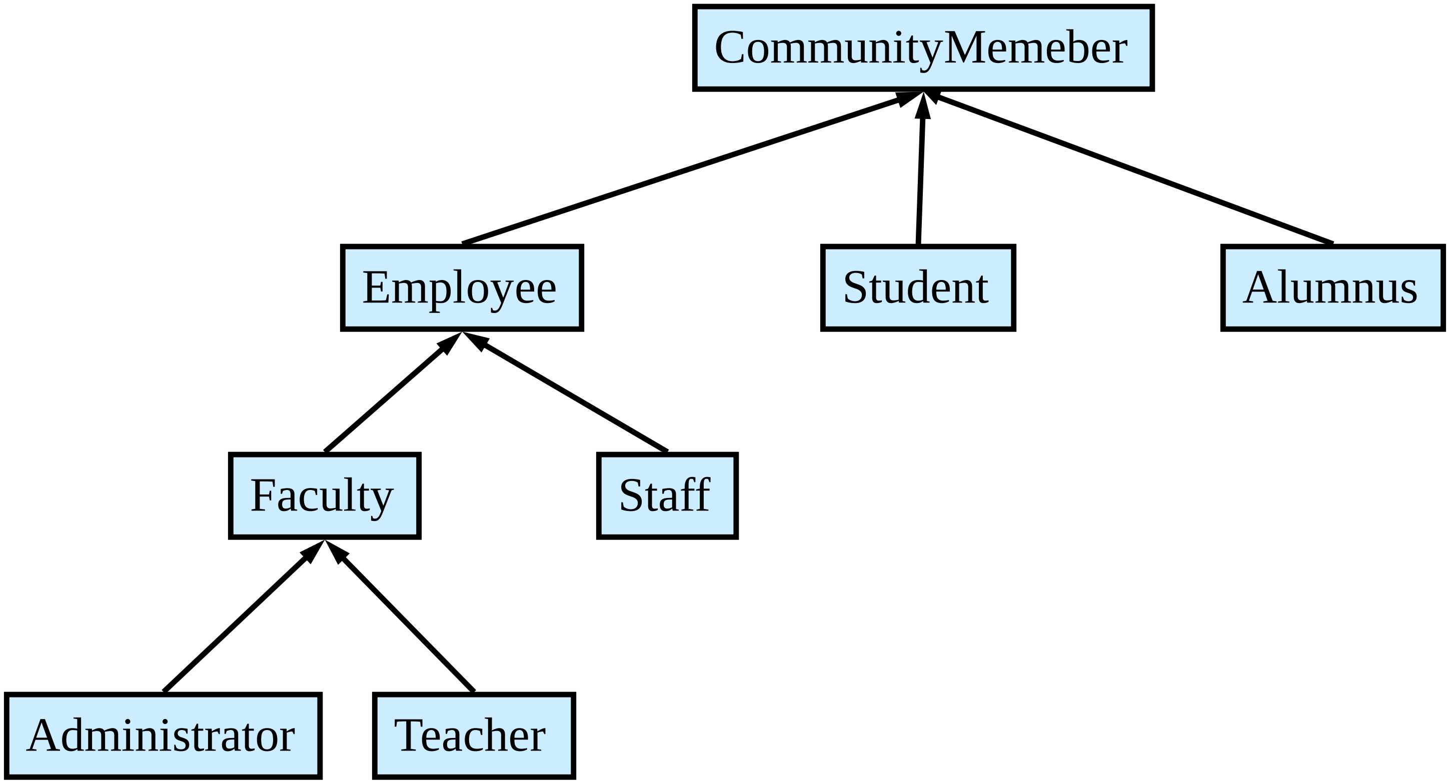
```
Initial universal time is: 00:00:00
Initial standard time is: 12:00:00 AM

Universal time after SetTime is: 13:27:06
Standard time after SetTime is: 1:27:06 PM

After attempting invalid settings:
Universal time: 00:00:00
Standard time: 12:00:00 AM
```

“object” Class

```
public class Time1: Object {
```

- Every C# class must extend from another class
  - If class does not explicitly **extend** another class
    - class implicitly extends **Object**
    - **Object is the superclass of all classes**

Supper and sub Classes



Constructor declaration

```
    public Time1() {
```

- Constructor
  - A special method with the name as its class's name
    - Cannot return values, but can take arguments
    - Void specifier is of no need
  - it is called implicitly (through "new")
    - Called when program instantiates an object of that class
  - may be more than one constructor per class (through *overloading*)

Initializing Class Objects: Constructors

- If a class has no constructor, a default constructor is provided
  - no code for the constructor (and takes no parameters)
- if the constructor have constructor definition but no statement in the constructor,
  - all data members are initialized
    - Primitive numeric types are set to 0
    - Boolean types are set to false
    - Reference types are set to null

Calling Constructor: by“new”

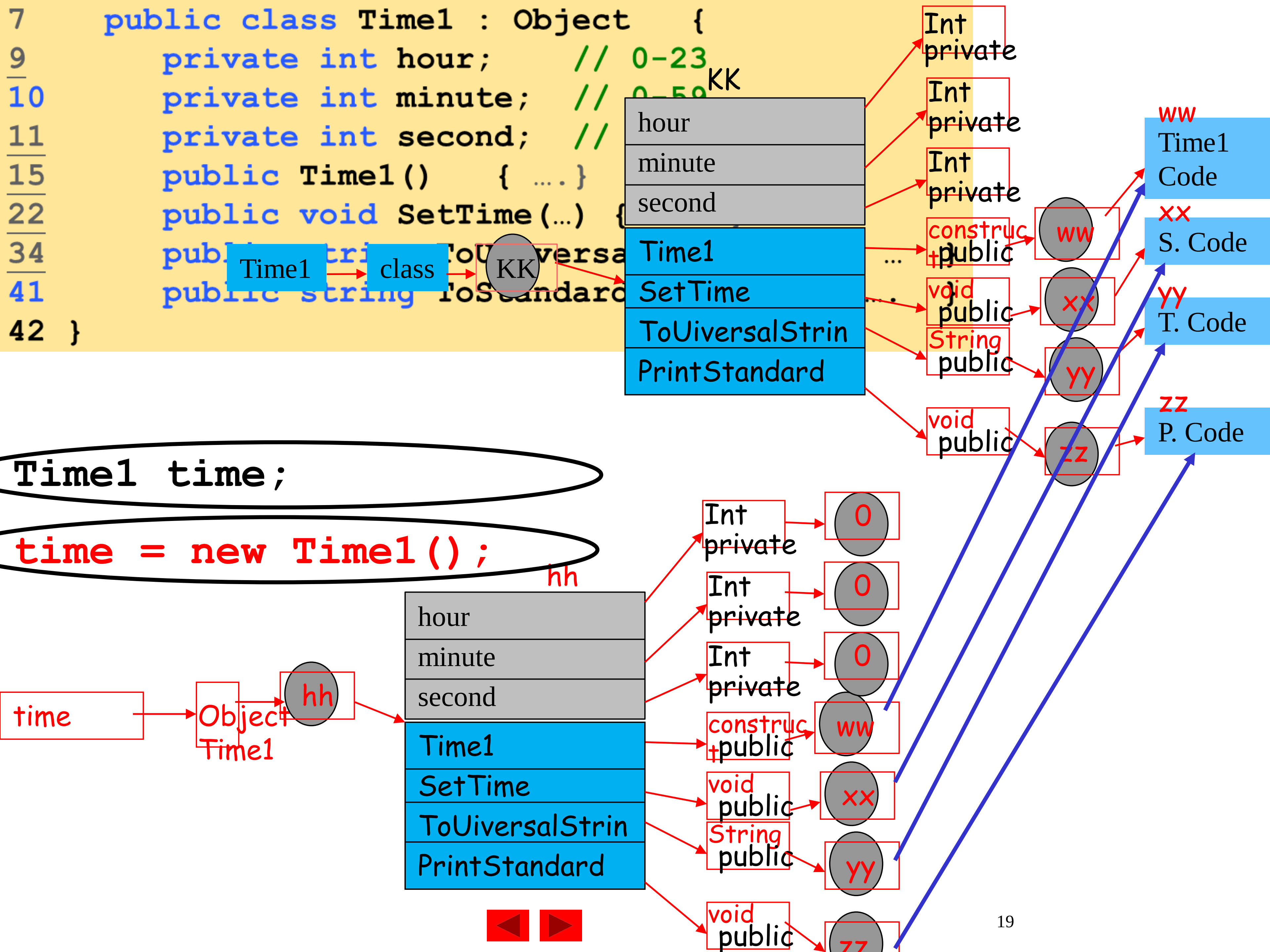
- Class definition and declaration
  - Once a class has been defined, it can be used as a data type
  - Example:

```
Time1 time = new Time1(); //time is an object of
                          //    type Time
```

  - Object is a reference variable (not primitive variable)
  - You need to use “new” command to ask a memory to store the contents of the variable
- Member access operators:
  - Dot operator (.) to get the object’s method and variable
  - Eq: **time.hour;**

```
time.SetTime(13, 27, 6);
```

```
public class Time1 : Object    {
    private int hour;      // 0-23
    private int minute;    // 0-59
    private int second;    //
    public Time1()    { ....}
    public void SetTime(...)  {
    public string ToUniversa
    public string ToStandard
}
```

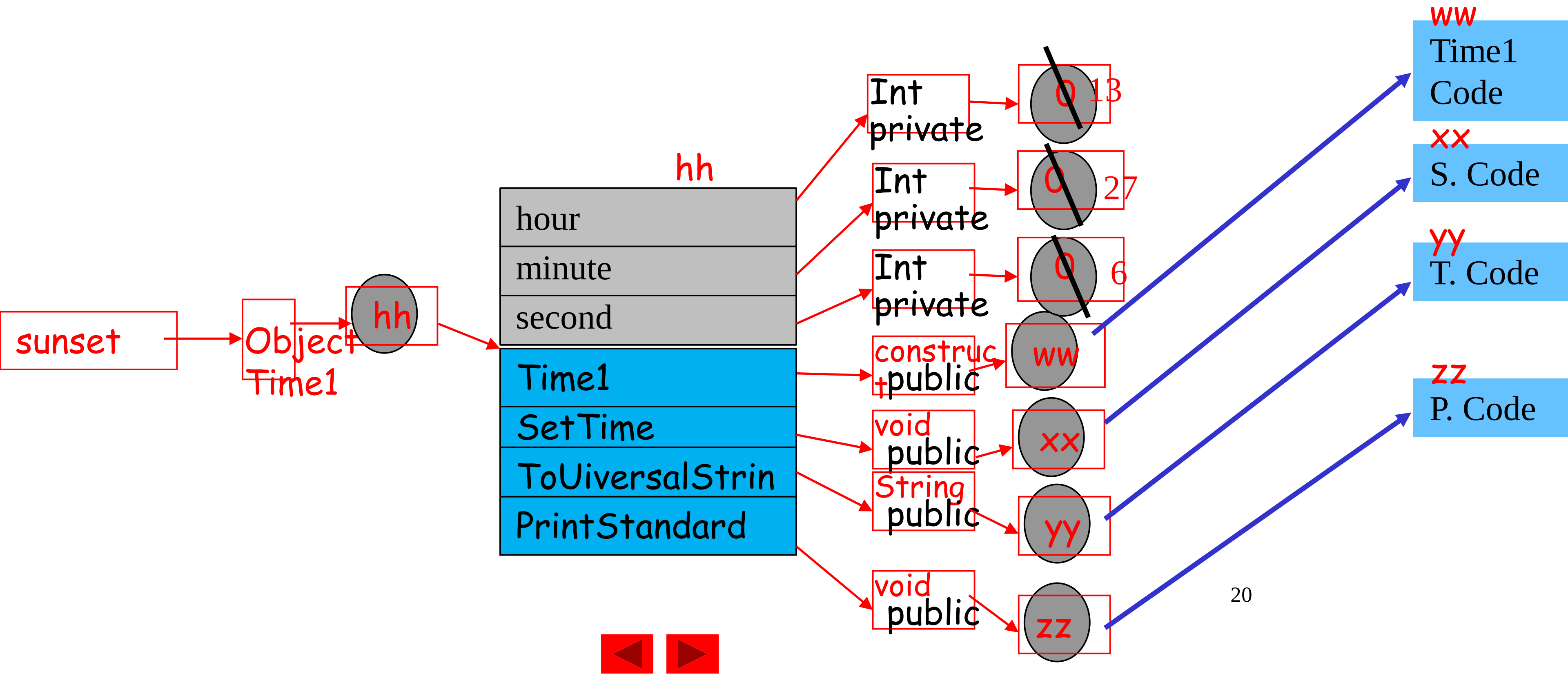


Time1 time;

time = new Time1();

```
using System;
class TimeTest1  {
   static void Main( string[] args )
      Time1 time = new Time1();
      string output;
        ....
      time.SetTime( 13, 27, 6 );
```

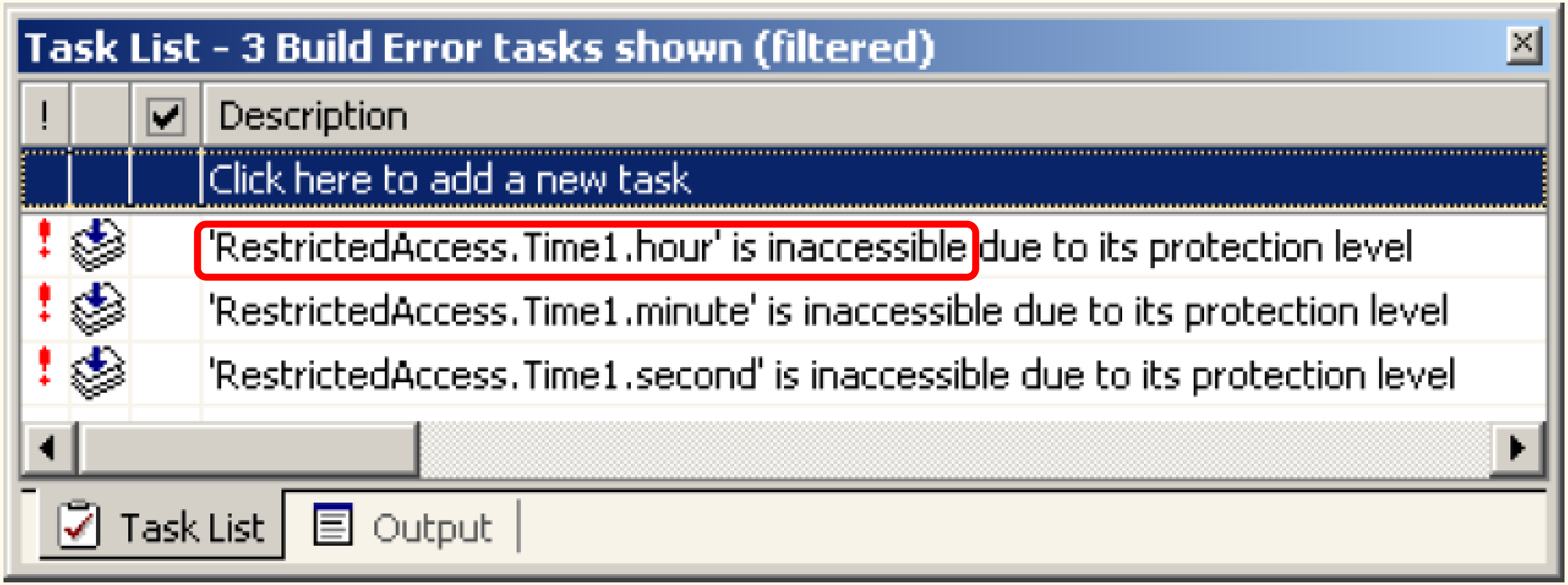
```
public void setTime( int h, int m, int s ) {
 hour = ( ( h >= 0 && h < 24 ) ? h : 0 );
 minute = ( ( m >= 0 && m < 60 ) ? m : 0 );
 second = ( ( s >= 0 && s < 60 ) ? s : 0 );
}
```



Member access specifiers

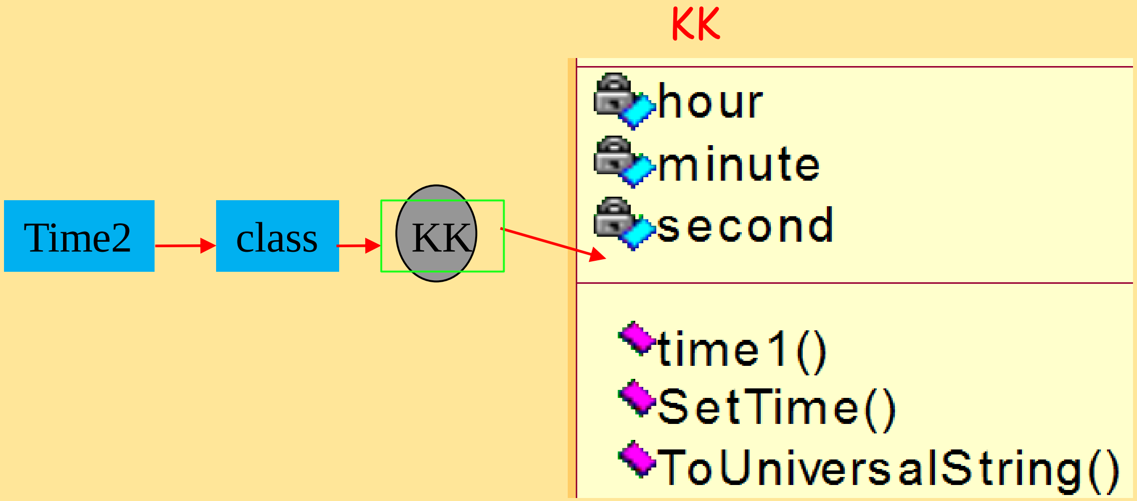
- Member access specifiers
  - Classes can limit the access to their member functions and data
  - The three types of access a class can grant are:
    - **Public**
      - 普遍級 *Accessible wherever the program has access to an object of the class*
    - **private**
      - 限制級 *Accessible only to member functions of the class*
    - **Protected**
      - 保護級 *Accessed only by subclass methods*

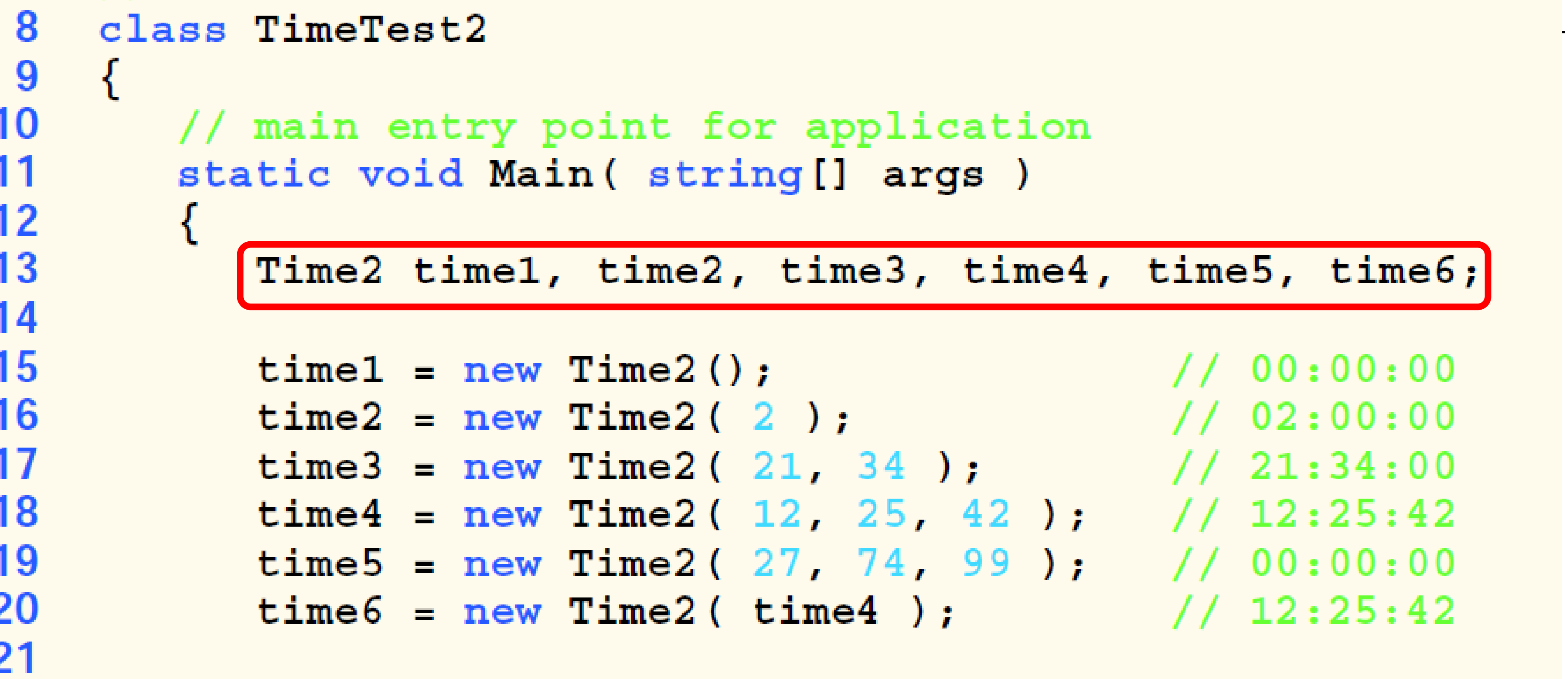
```
class RestrictedAccess
{
   // main entry point for application
   static void Main( string[] args )
   {
      Time1 time = new Time1();

      time.hour = 7;
      time.minute = 15;
      time.second = 30;
   }

} // end class RestrictedAccess
```


Task List - 3 Build Error tasks shown (filtered)
!
✓
Description
Click here to add a new task
Task List
Output

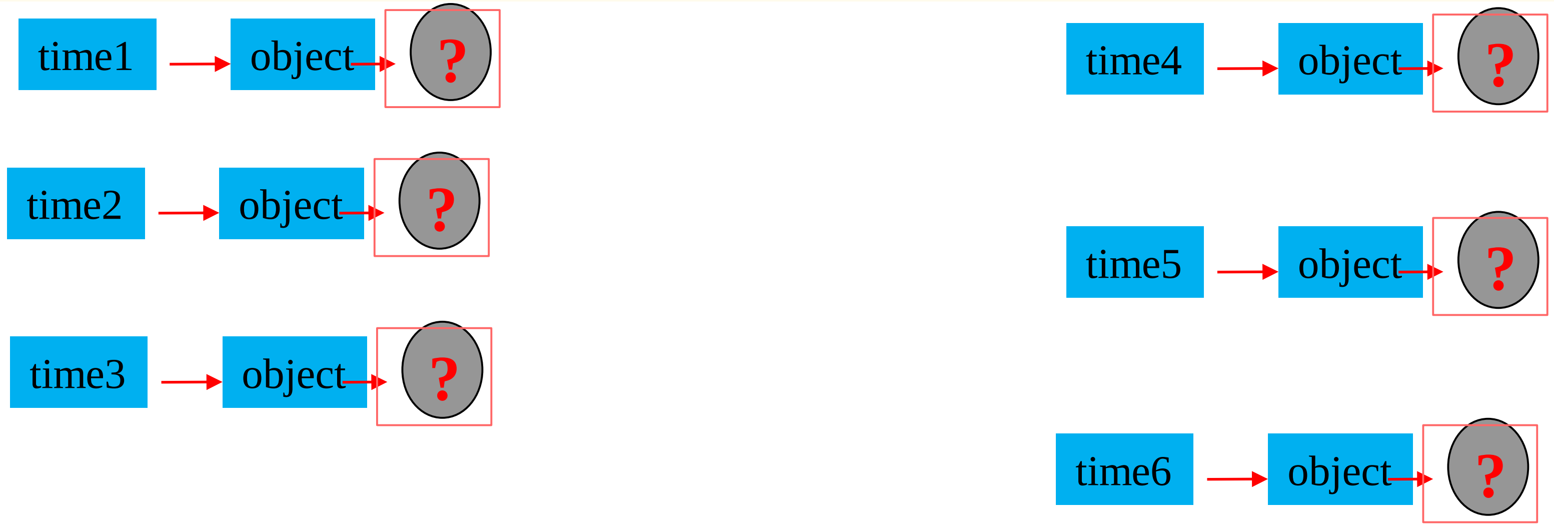
```
'RestrictedAccess.Time1.hour' is inaccessible due to its protection level
'RestrictedAccess.Time1.minute' is inaccessible due to its protection level
'RestrictedAccess.Time1.second' is inaccessible due to its protection level
```

```
using System;
public class Time2  {
   private int hour;    // 0-23
   private int minute;  // 0-59
   private int second;  // 0-59
   public Time2()  {
      SetTime( 0, 0, 0 );
   }
   public Time2( int hour )  {
      SetTime( hour, 0, 0 );
   }
   public Time2( int hour, int minute ) {
      SetTime( hour, minute, 0 );
   }
   public Time2( int hour, int minute, int second ) {
     SetTime( hour, minute, second );
   }
   public Time2( Time2 time ){
     SetTime( time.Hour, time.Minute, time.Second );
   }
```



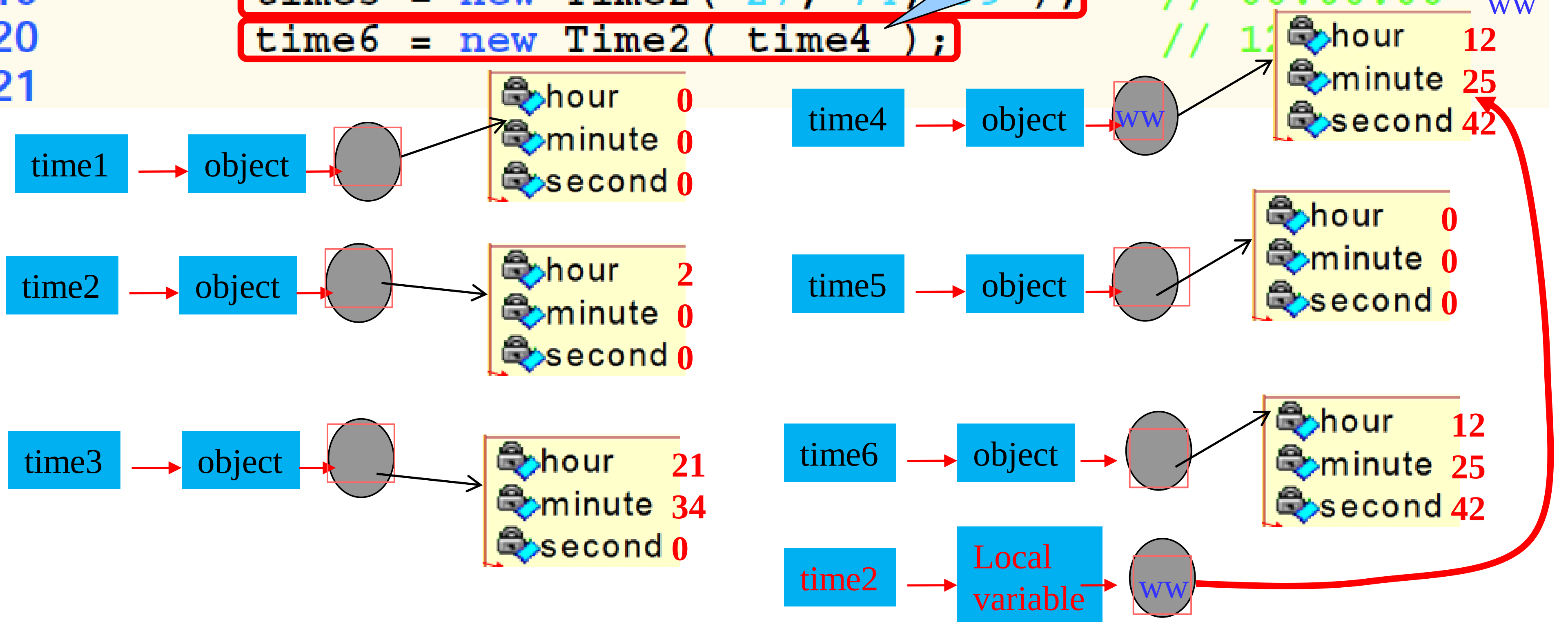


```
class TimeTest2
{
   // main entry point for application
   static void Main( string[] args )
   {
      Time2 time1, time2, time3, time4, time5, time6;

      time1 = new Time2();                    // 00:00:00
      time2 = new Time2( 2 );                 // 02:00:00
      time3 = new Time2( 21, 34 );            // 21:34:00
      time4 = new Time2( 12, 25, 42 );        // 12:25:42
      time5 = new Time2( 27, 74, 99 );        // 00:00:00
      time6 = new Time2( time4 );             // 12:25:42

```



```
class TimeTest2
{
    // main entry point for application
    static void Main( string[] args )
    {
        Time2 time1, time2, time3, time4, time5, time6;

        time1 = new Time2();
        time2 = new Time2( 2 );
        time3 = new Time2( 21, 34 );
        time4 = new Time2( 12, 25, 42
        time5 = new Time2( 27, 74,        );    // 00:00:00
        time6 = new Time2( time4 );              // 12
```

If a variable works as a parameter, its effect is as R-value



8.7 Properties

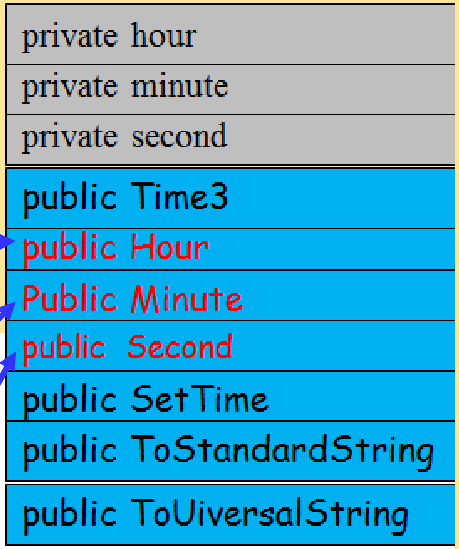
- private data cannot be read/write by other objects
  - The rule can reach the goal of information hiding
  - But it is too strict and inflexible
- Public properties allow program outside to:
  - Get (obtain the values of) private data
  - Set (assign values to) private data
- Get accessor
  - Get value from private value;
  - you can controls transformation (like weight to pound) before yielding the data
- Set accessor
  - Set value to private value
  - You can ensure that check whether the new value is appropriate for the data member (accept the setting or modify the input data)

```
using System;
public class Time3  {
   private int hour;     // 0-23
   private int minute;  // 0-59
   private int second;  // 0-59
   public Time3() {
      SetTime( 0, 0, 0 );
   }
   public Time3( int hour )  {
      SetTime( hour, 0, 0 );
   }
   public Time3( int hour, int minute ) {
      SetTime( hour, minute, 0 );
   }
   public Time3( int hour, int minute, int second ) {
     SetTime( hour, minute, second );
   }
   public Time3( Time3 time ){
     SetTime( time.Hour, time.Minute, time.Second );
   }
```

Private variable

Method overloading

```
public void SetTime(int hourValue, int minuteValue, int secondValue ) {
   Hour = hourValue;
   Minute = minuteValue;
   Second = secondValue;    }
public int Hour   {
   get {
      return hour; }
   set  {
      hour =(( value >= 0 && value < 24 ) ? value : 0 );  }
}
public int Minute{
 get{
    return minute;}
 Set {
    minute =((value >= 0 && value<60 ) ? value:0 );}
}
public int Second {
 get   {
    return second;}
 set {
   second=((value >= 0&& value < 60 ) ? value : 0 );}
}
```



private hour

private minute

private second

public Time3

public Hour

Public Minute

public Second

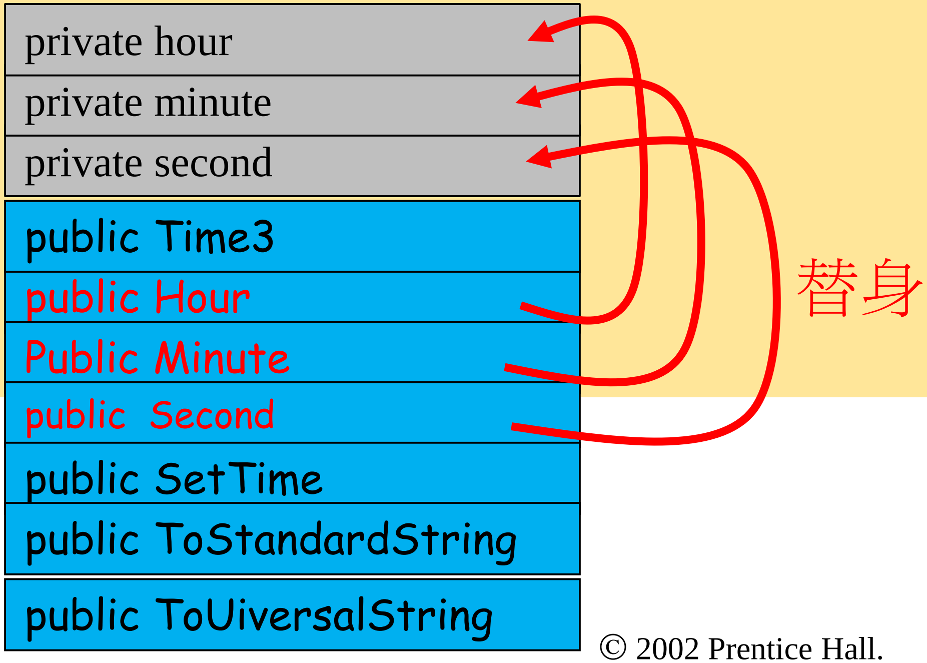
public SetTime

public ToStandardString

public ToUiversalString

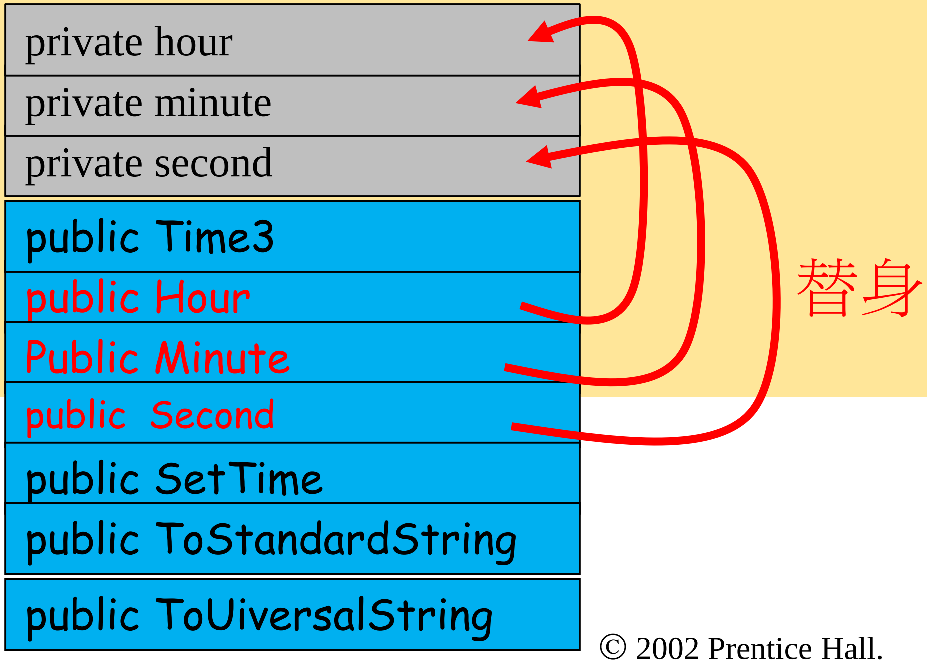
```
public string ToUniversalString() {
   return String.Format(
      "{0:D2}:{1:D2}:{2:D2}", Hour, Minute, Second );
}
public string ToStandardString() {
   return String.Format( "{0}:{1:D2}:{2:D2} {3}",
      ( ( Hour == 12 || Hour == 0 ) ? 12 : Hour % 12 ),
      Minute, Second, ( Hour < 12 ? "AM" : "PM" ) );
}
}
```

Hour, Minute, Second work like function calls



```
   public string ToUniversalString() {
      return String.Format(
         "{0:D2}:{1:D2}:{2:D2}", Hour, Minute, Second );
   }
   public string ToStandardString() {
      return String.Format( "{0}:{1:D2}:{2:D2} {3}",
         ( ( Hour == 12 || Hour == 0 ) ? 12 : Hour % 12 ),
         Minute, Second, (Hour < 12 ? "AM" : "PM" ) );
   }
}
```

Hour, Minute, Second work like function calls



private hour

private minute

private second

public Time3

public Hour

Public Minute

public Second

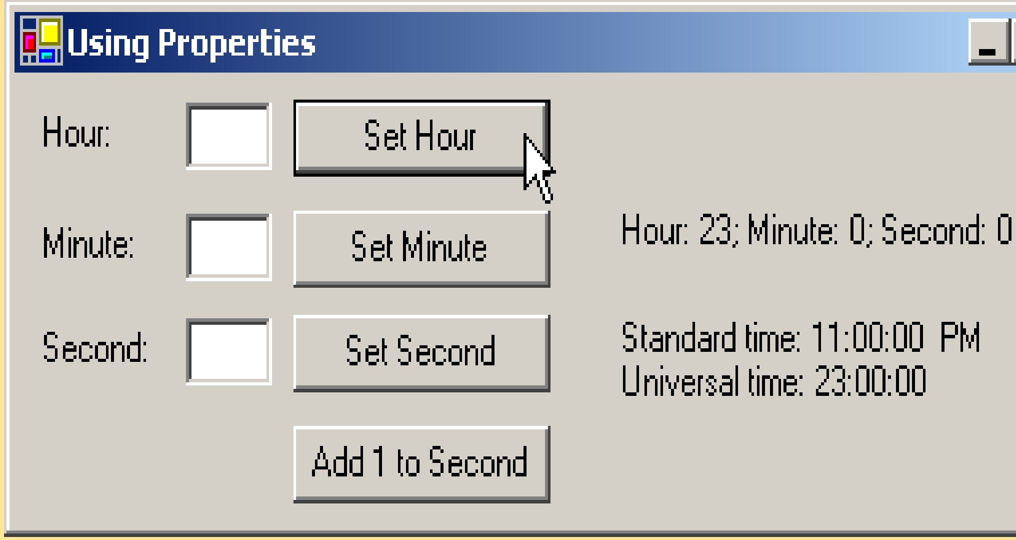
public SetTime

public ToStandardString

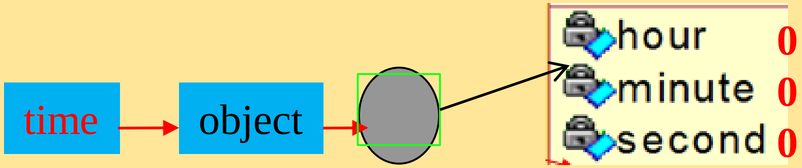
public ToUiversalString

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
public class TimeTest3 : System.Windows.Forms.Form
{
   private System.Windows.Forms.Label hourLabel;
   private System.Windows.Forms.TextBox hourTextBox;
   private System.Windows.Forms.Button hourButton;

   private System.Windows.Forms.Label minuteLabel;
   private System.Windows.Forms.TextBox minuteTextBox;
   private System.Windows.Forms.Button minuteButton;

   private System.Windows.Forms.Label secondLabel;
   private System.Windows.Forms.TextBox secondTextBox;
   private System.Windows.Forms.Button secondButton;

   private System.Windows.Forms.Button addButton;

   private System.Windows.Forms.Label displayLabel1;
   private System.Windows.Forms.Label displayLabel2;
   private System.ComponentModel.Container components = null;

   private Time3 time;

```

```
public TimeTest3()
{
    InitializeComponent();
    time = new Time3();
    UpdateDisplay();
}
[STAThread]
static void Main()
{
    Application.Run( new TimeTest3() );
}
public void UpdateDisplay()
{
    displayLabel1.Text = "Hour: " + time.Hour +
      "; Minute: " + time.Minute +
      "; Second: " + time.Second;
    displayLabel2.Text = "Standard time: " +
      time.ToStandardString() + "\nUniversal time: " +
      time.ToUniversalString();
}
```

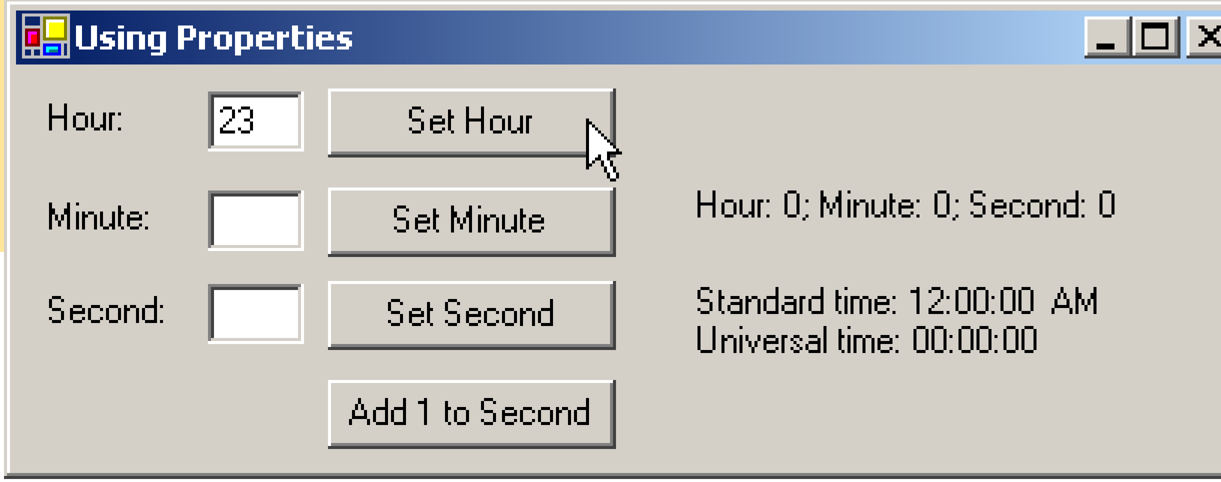


```
Hour: 23; Minute: 0; Second: 0

Standard time: 11:00:00 PM
Universal time: 23:00:00
```



```
private void hourButton_Click(object sender, System.EventArgs e )
{
   time.Hour = Int32.Parse( hourTextBox.Text );
   hourTextBox.Text = "";
   UpdateDisplay();
}
private void minuteButton_Click(object sender, System.EventArgs e )
{
   time.Minute = Int32.Parse( minuteTextBox.Text );
   minuteTextBox.Text = "";
   UpdateDisplay();
}
private void secondButton_Click(object sender, System.EventArgs e )
{
   time.Second = Int32.Parse( secondTextBox.Text );
   secondTextBox.Text = "";
   UpdateDisplay();
}
private void addButton_Click(object sender, System.EventArgs e )
{
   time.Second = ( time.Second + 1 ) % 60;

```

```
        if ( time.Second == 0 )
        {
            time.Minute = ( time.Minute + 1 ) % 60;

            if ( time.Minute == 0 )
                time.Hour = ( time.Hour + 1 ) % 24;
        }
        UpdateDisplay();
    }
}
```

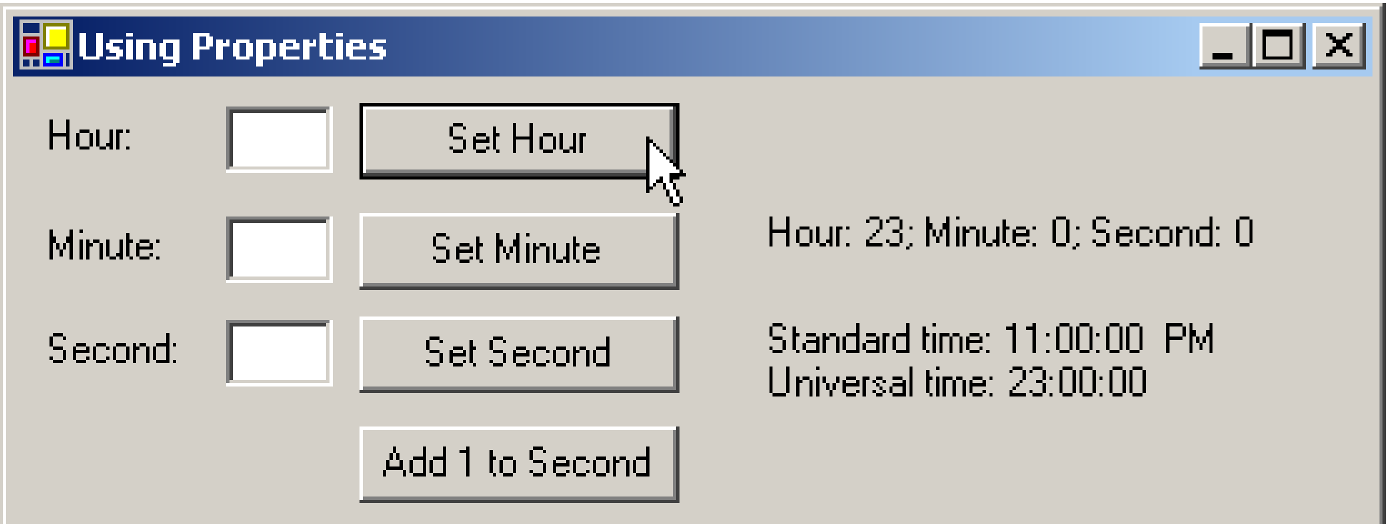


Hour: 23

Minute:

Second:

```
Hour: 0; Minute: 0; Second: 0

Standard time: 12:00:00  AM
Universal time: 00:00:00
```



Hour:

Minute:

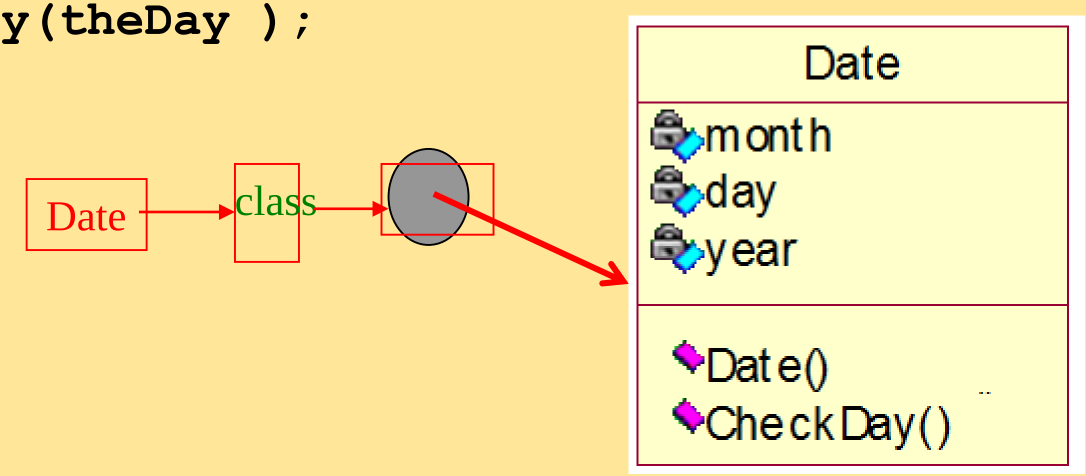
Second:

```
Hour: 23; Minute: 0; Second: 0

Standard time: 11:00:00  PM
Universal time: 23:00:00
```

Composition

- composition (construct a class with other classes)
  - Class contains reference variable of other classes
  - Utilize user-defined types in a class
    - Instances of the user-defined types can appear in host class's attribute or method
      - Host class: a class uses variables of other classes
  - Software Reuse
    - referencing existing object is easier and faster than rewriting the objects' code for new classes

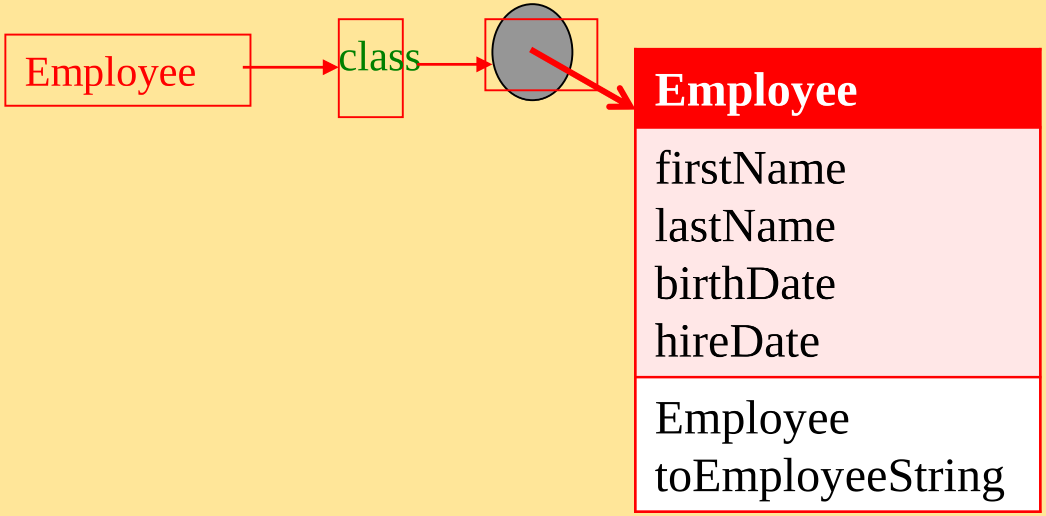
```
using System;
public class Date {
   private int month;  // 1-12
   private int day;    // 1-31 based on month
   private int year;   // any year
   public Date( int theMonth, int theDay, int theYear ){
      if ( theMonth > 0 && theMonth <= 12 )
         month = theMonth;
      else {
         month = 1;
         Console.WriteLine("Month {0} invalid. Set
                            to month 1.", theMonth );
      }
      year = theYear;
      day = CheckDay(theDay );
   }
```



```
        private int CheckDay( int testDay ) {
           int[] daysPerMonth =
              { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
};
           if ( testDay > 0 && testDay <= daysPerMonth[ month ] )
              return testDay;
           if ( month == 2 && testDay == 29 &&
              ( year % 400 == 0 ||
              ( year % 4 == 0 && year % 100 != 0 ) ) )
             return testDay;
           Console.WriteLine("Day {0} invalid. Set to day 1.",
                             testDay );
           return 1;  // leave object in consistent state
        }
        public string ToDateString() {
           return month + "/" + day + "/" + year;
        }
     }  // end class Date
```

```
using System;
public class Employee {
  private string firstName;
  private string lastName;
  private Date birthDate;
  private Date hireDate;
  public Employee( string first, string last,
      int birthMonth, int birthDay, int birthYear,
      int hireMonth, int hireDay, int hireYear ) {
   firstName = first;
   lastName = last;
   birthDate=new Date(birthMonth,birthDay,birthYear);
   hireDate= new Date(hireMonth, hireDay, hireYear );
   }
```

Example 1



Employee

firstName
lastName
birthDate
hireDate

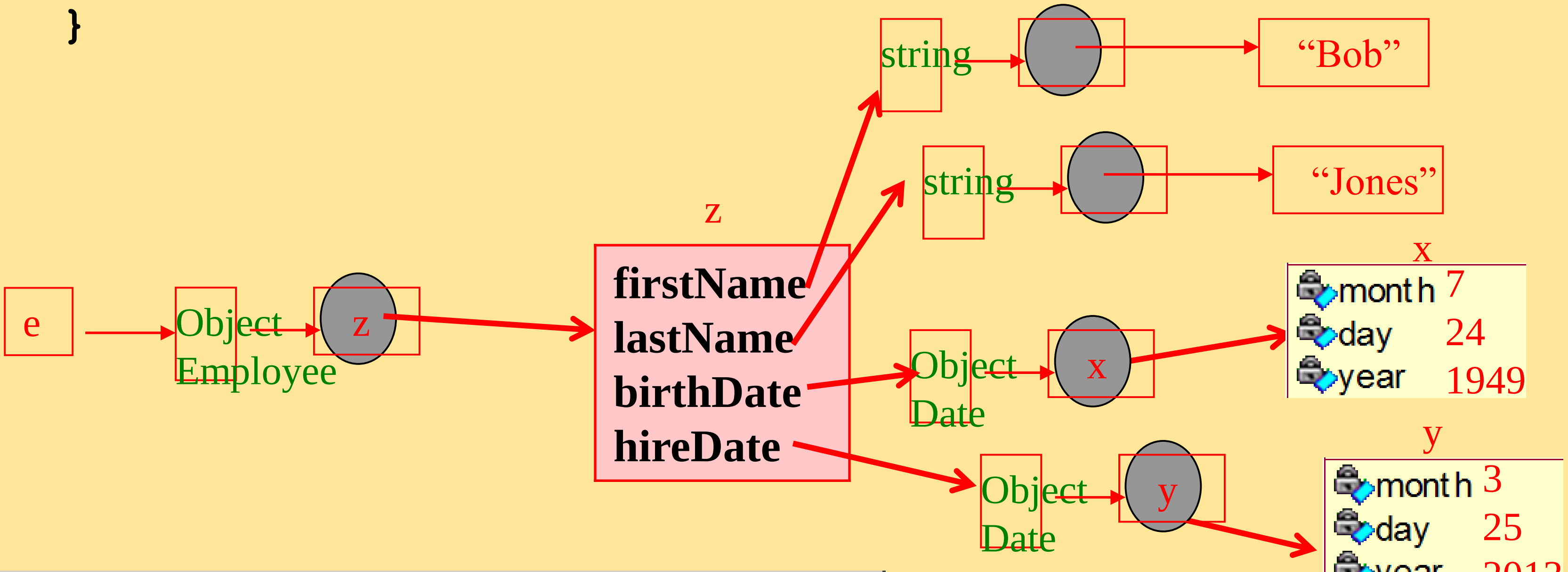
Employee
toEmployeeString

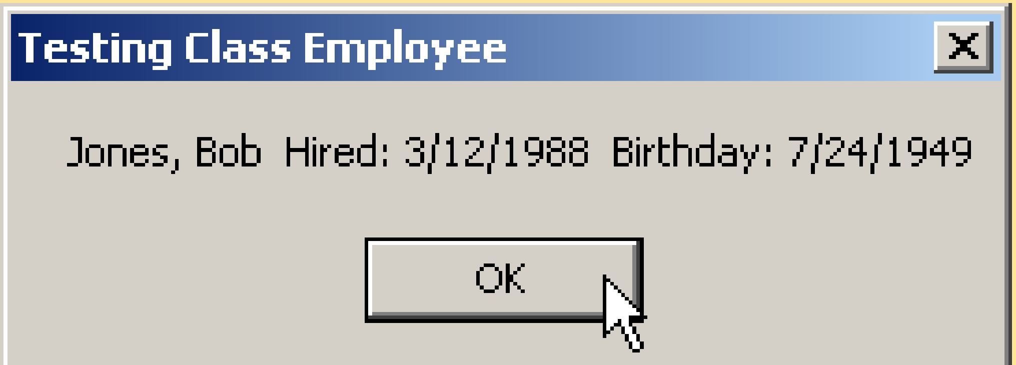
```
   public string ToEmployeeString() {
      return lastName + ", " + firstName +
         "  Hired: " + hireDate.ToDateString() +
         "  Birthday: " + birthDate.ToDateString();
   }
}  // end class Employee
```

Example 1

Example 2

```
using System;
using System.Windows.Forms;
class CompositionTest {
   static void Main( string[] args ) {
      Employee e =
       new Employee("Bob", "Jones", 7, 24, 1949, 3, 12,1988);
      MessageBox.Show( e.ToEmployeeString(),
          "Testing Class Employee" );
   }
}
```

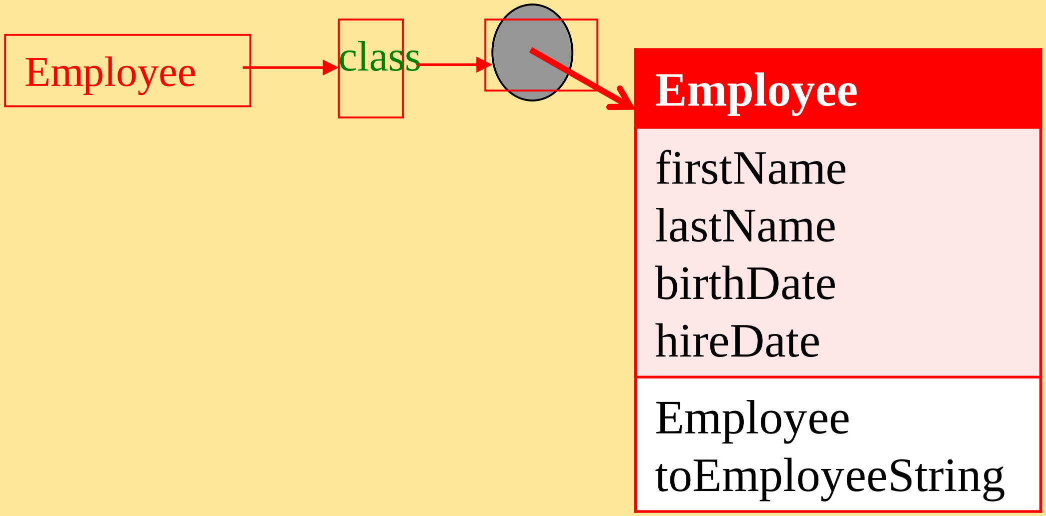



Testing Class Employee

Jones, Bob Hired: 3/12/1988 Birthday: 7/24/1949

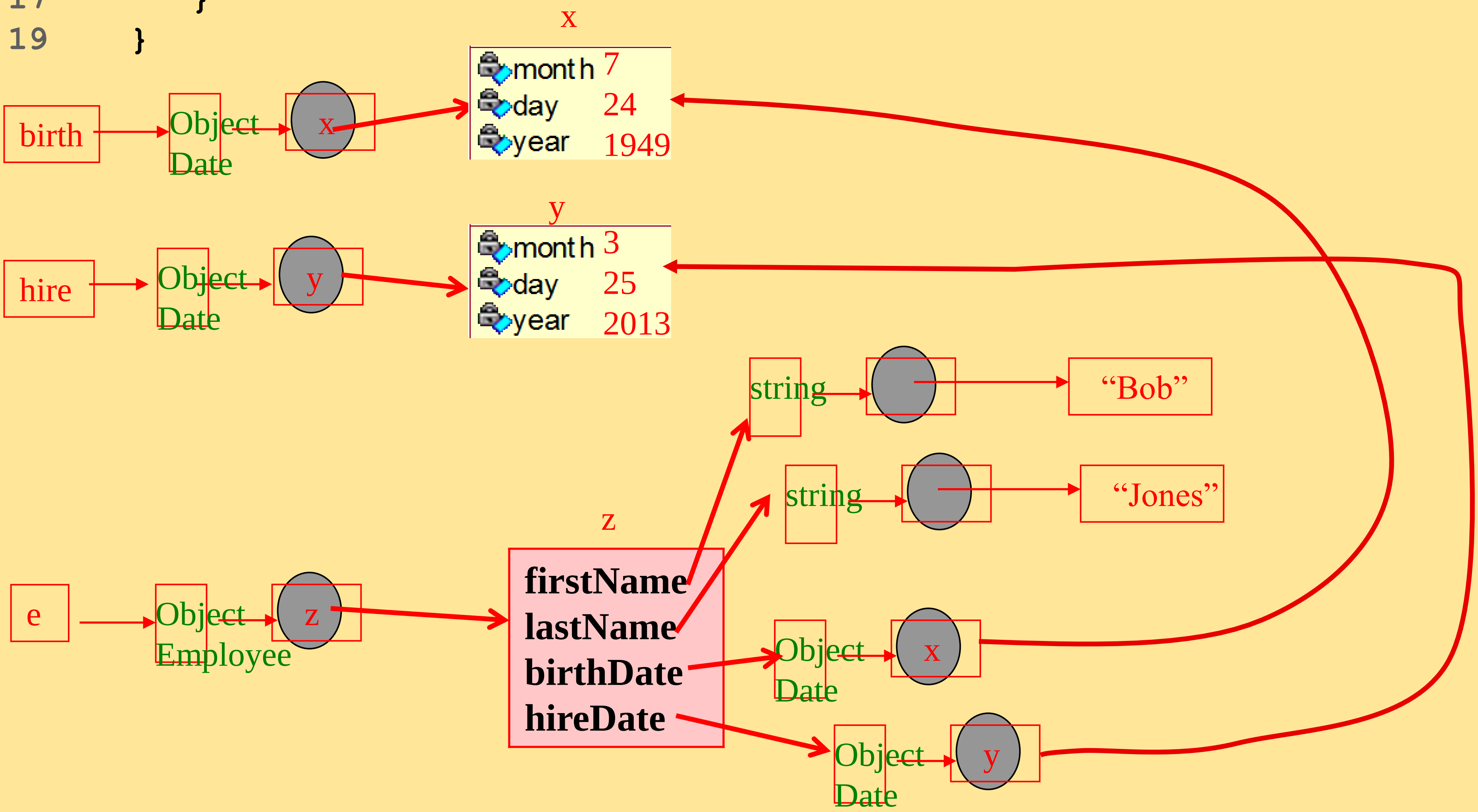
Example 2

```
public class Employee {
   private String firstName;
   private String lastName;
   private Date birthDate;
   private Date hireDate;
   public Employee( String first, String last, Date dateOfBirth,
                    Date dateOfHire ) {
      firstName = first;
      lastName = last;
      birthDate = dateOfBirth;
      hireDate = dateOfHire;
   }
   public String toEmployeeString(){
        ......
   }

} // end class Employee
```



Example 2

```
public class CompositionTest2 {
   static void Main( String args[] ) {
      Date birth = new Date( 7, 24, 1949 );
      Date hire = new Date( 3, 25, 2013 );
     Employee e = new Employee("Bob", "Jones", birth, hire);
       ……
   }
}
```

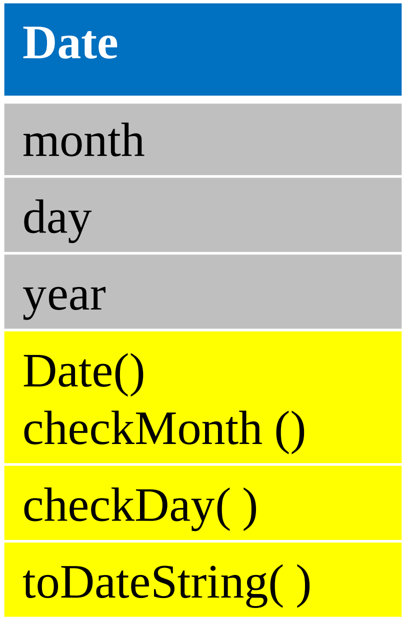


Rehearsal

Example 1 (composition)

◀ ▶

```
public class Date {
   private int month;  // 1-12
   private int day;    // 1-31 based on month
   private int year;   // any year
   public Date( int theMonth, int theDay, int theYear )
   ……… } // end Date constructor
   private int checkMonth( int testMonth ) {
        …}
  private int checkDay( int testDay )    {
          ….. }
  public String toDateString()   {
  …… }
}
```



Date

month

day

year

Date()
checkMonth ()

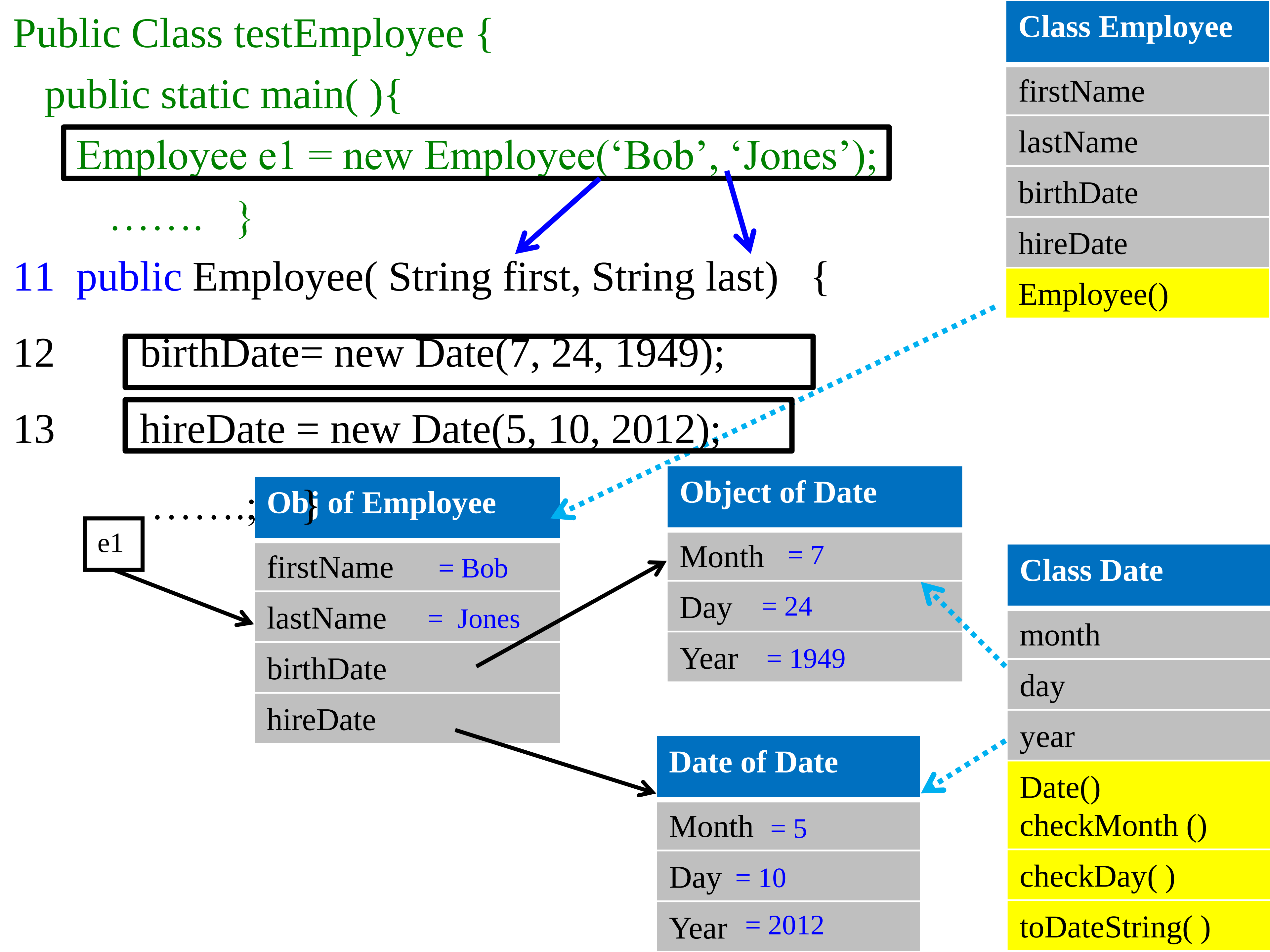
checkDay( )

toDateString( )

```
public class Employee {
      private String firstName;
      private String lastName;
      private Date birthDate;
      private Date hireDate;
 public Employee( String first, String last)   {
    birthDate= new Date(7, 24, 1949);
    hireDate = new Date(5, 10, 2012);
        .......;    }
} // end class Employee
```

```
Public Class testEmployee {
   ......
   public static main( ){
      Employee e1 = new Employee(Jack,
                Wu);
```

Employee
firstName
lastName
birthDate
hireDate
Employee()



```
Public Class testEmployee {
  public static main( ){
   Employee e1 = new Employee(‘Bob’, ‘Jones’);
     …….   }
public Employee( String first, String last)   {
      birthDate= new Date(7, 24, 1949);
      hireDate = new Date(5, 10, 2012);
    ……..;
    }
```

Class Employee

- firstName
- lastName
- birthDate
- hireDate
- Employee()

e1

Obj of Employee

- firstName = Bob
- lastName = Jones
- birthDate
- hireDate

Object of Date

- Month = 7
- Day = 24
- Year = 1949

Date of Date

- Month = 5
- Day = 10
- Year = 2012

Class Date

- month
- day
- year
- Date()
- checkMonth ()
- checkDay( )
- toDateString( )

Rehearsal:

Example 2 (Association)

◀ ▶

```
public class Date {
   private int month;  // 1-12
   private int day;    // 1-31 based on month
   private int year;   // any year
   public Date( int theMonth, int theDay, int theYear )
   ……… } // end Date constructor
   private int checkMonth( int testMonth ) {
         …}
  private int checkDay( int testDay )    {
           ….. }
  public String toDateString()   {
  …… }
}
```

Date
month
day
year
Date() checkMonth ()
checkDay( )
toDateString( )

```
public class Employee {
      private String firstName;
      private String lastName;
      private Date EventDate1, EventDate2;
   public Employee( String first, String last, Date
             dateOfBirth, Date dateOfHire )   {
         firstName = first;
          lastName = last;
          EventDate1 = dateOfBirth;
           EventDate2 = dateOfHire;
…….;   }
} // end class Employee

     …….
```

No new by Employee

Employee

firstName

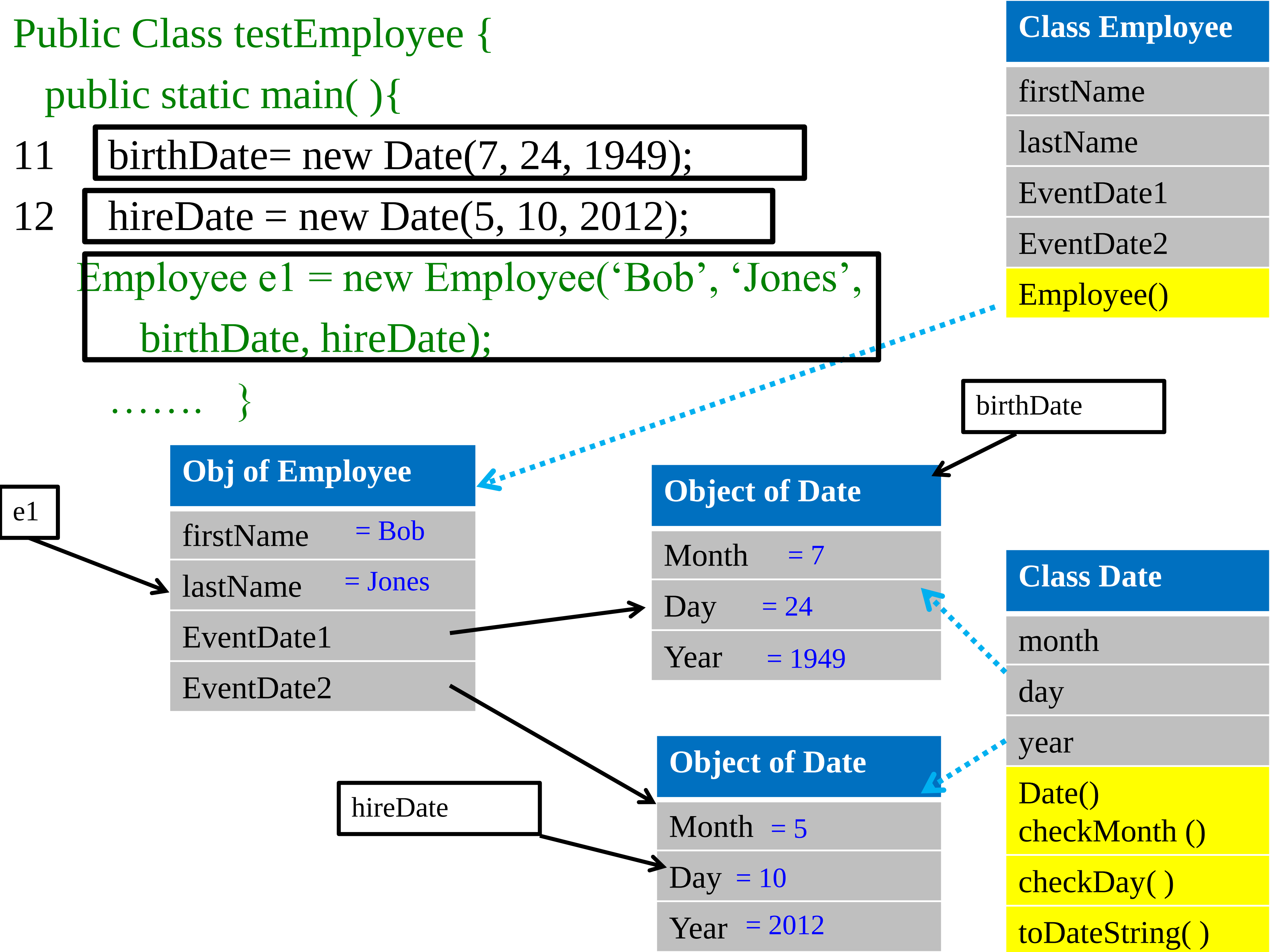
lastName

EventDate1

EventDate2

Employee()

```
Public Class testEmployee {
  public static main( ){
     birthDate= new Date(7, 24, 1949);
     hireDate = new Date(5, 10, 2012);
   Employee e1 = new Employee(‘Bob’, ‘Jones’,
       birthDate, hireDate);
    …….   }
```



Class Employee

firstName
lastName
EventDate1
EventDate2
Employee()

Obj of Employee

firstName = Bob
lastName = Jones
EventDate1
EventDate2

Object of Date

Month = 7
Day = 24
Year = 1949

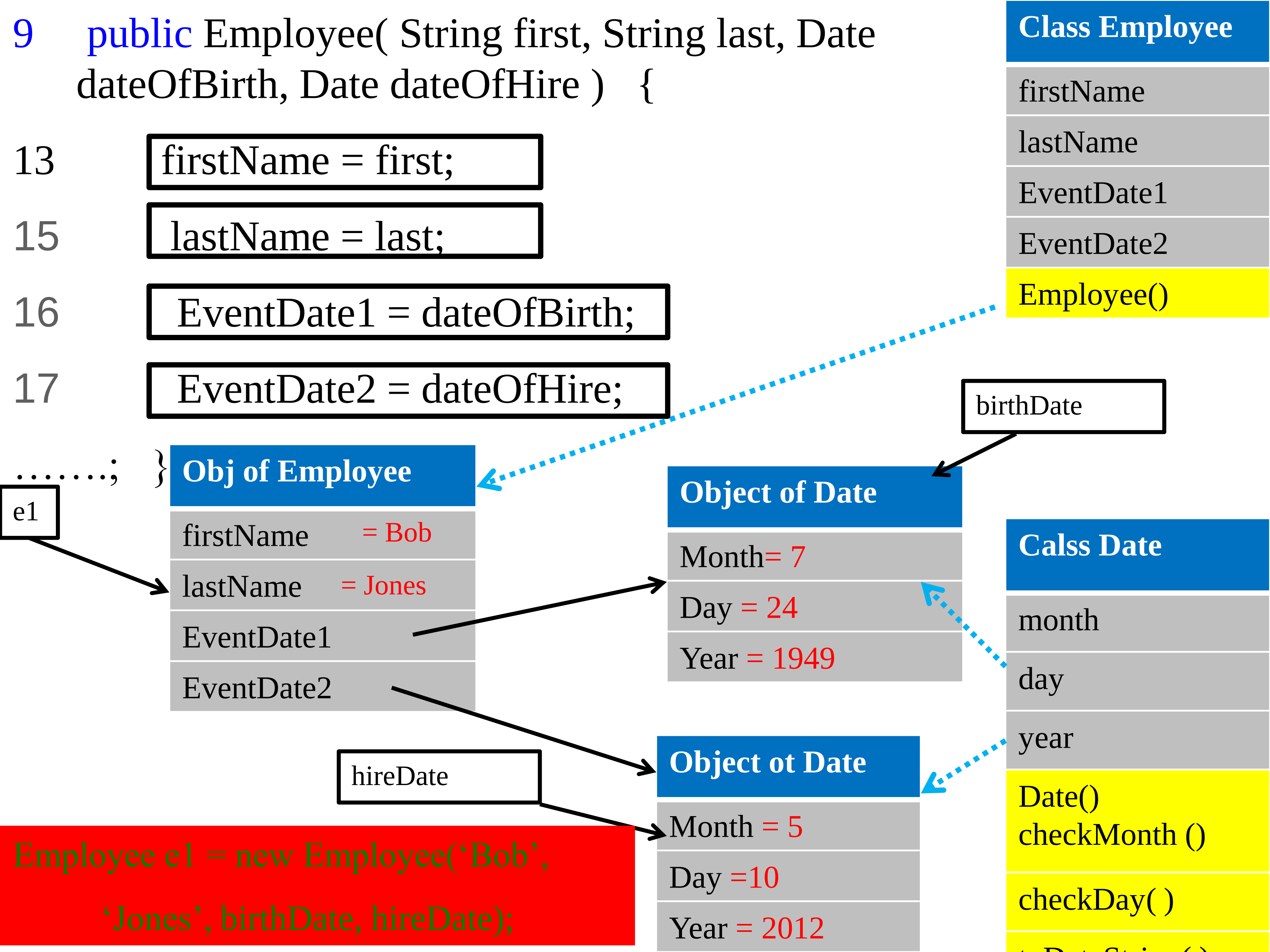
Object of Date

Month = 5
Day = 10
Year = 2012

Class Date

month
day
year
Date()
checkMonth ()
checkDay( )
toDateString( )

```
public Employee( String first, String last, Date dateOfBirth, Date dateOfHire )   {
        firstName = first;
        lastName = last;
        EventDate1 = dateOfBirth;
        EventDate2 = dateOfHire;
…….;   }
```



Class Employee

firstName

lastName

EventDate1

EventDate2

Employee()

Obj of Employee

firstName = Bob

lastName = Jones

EventDate1

EventDate2

e1

birthDate

Object of Date

Month= 7

Day = 24

Year = 1949

Calss Date

month

day

year

Date()
checkMonth ()

checkDay( )

hireDate

Object ot Date

Month = 5

Day =10

Year = 2012

Employee e1 = new Employee(‘Bob’, ‘Jones’, birthDate, hireDate);

Using the this reference

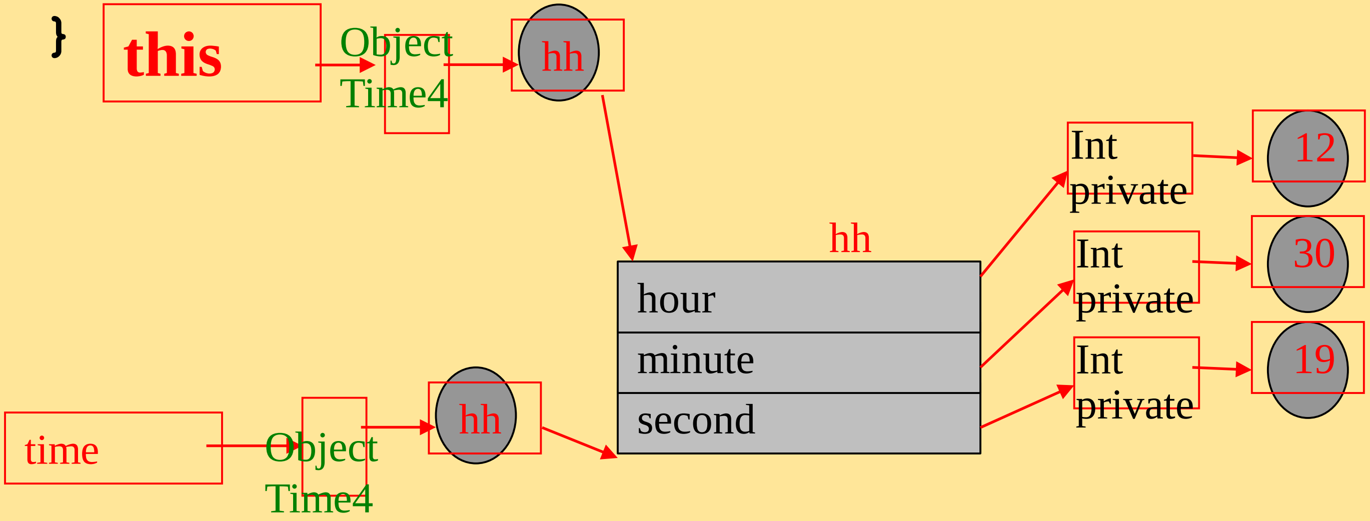
- Keyword this (我)
  - Allows an object to refers to itself
  - “this” explicitly references the object’s own member
  - Often used to distinguish between a method’s parameter and local variables, and the instance variables of an object
  - Note: if “this” is used as a method (with a parenthesis), it means to reference its constructor.

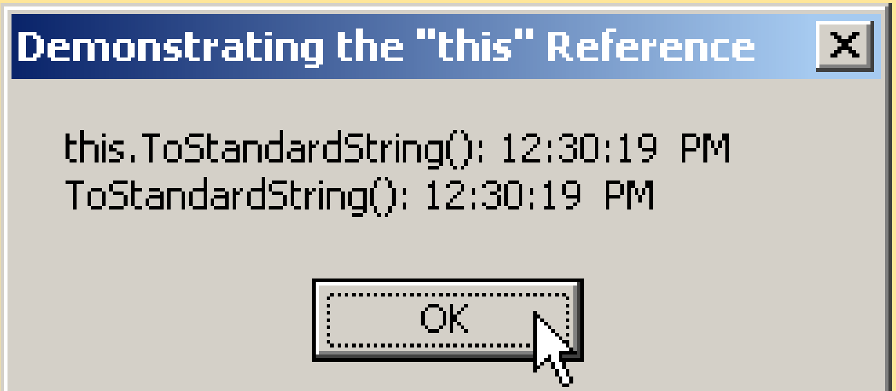
```
using System;
public class Time4 {
   private int hour;
   private int minute;
   private int second;
   public Time4( int hour, int minute, int second ) {
      this.hour = hour;
      this.minute = minute;
      this.second = second;
   }
   public string BuildString() {
    return "this.ToStandardString(): " +
      this.ToStandardString() +
       "\nToStandardString(): " + ToStandardString();
   }
   public string ToStandardString() {
      return String.Format( "{0}:{1:D2}:{2:D2} {3}",
       ((this.hour == 12 || this.hour == 0 ) ? 12 :
        this.hour % 12 ), this.minute, this.second,
        (this.hour < 12 ? "AM" : "PM" ) );
   }
}
```

if the names of arguments of Time4 differs from the member names, “this” can be omitted

“this” can be omitted

```
using System;
using System.Windows.Forms;
class Class1 {
   // main entry point for application
   static void Main( string[] args ) {
      Time4 time = new Time4( 12, 30, 19 );
      MessageBox.Show( time.BuildString(),
         "Demonstrating the \"this\" Reference" );
   }
}
```





Demonstrating the "this" Reference

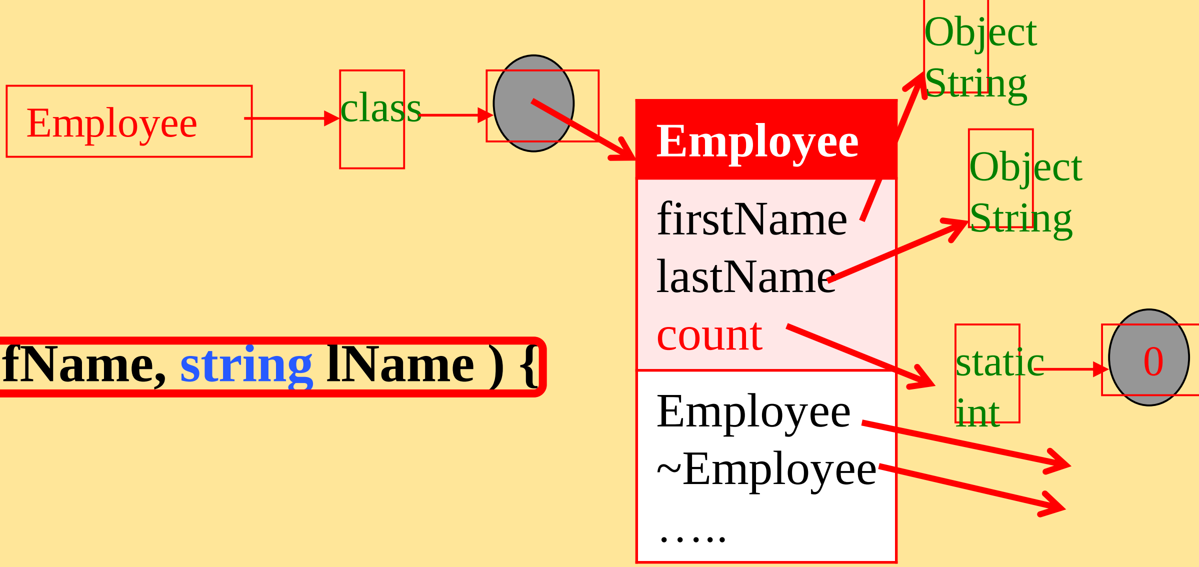
```
this.ToStandardString(): 12:30:19 PM
ToStandardString(): 12:30:19 PM
```

Garbage Collection

- When objects are no longer referenced,
  - the CLR (Common Language Runtime, ie., C# virtual machine) performs garbage collection
- Destructors (or called finalizer)
  - It is a member function of class
  - Functions with the same name as the class but preceded with a tilde character (~)
    - Syntax: ~class_name
  - Cannot take arguments
  - Perform termination, housekeeping before the system reclaims the object's memory
  - Complement of the constructor
  - Only one destructor per class
- Note: Allocation and deallocation of other resources (eq. database connections, file access, etc.) must be explicitly handled by programmers

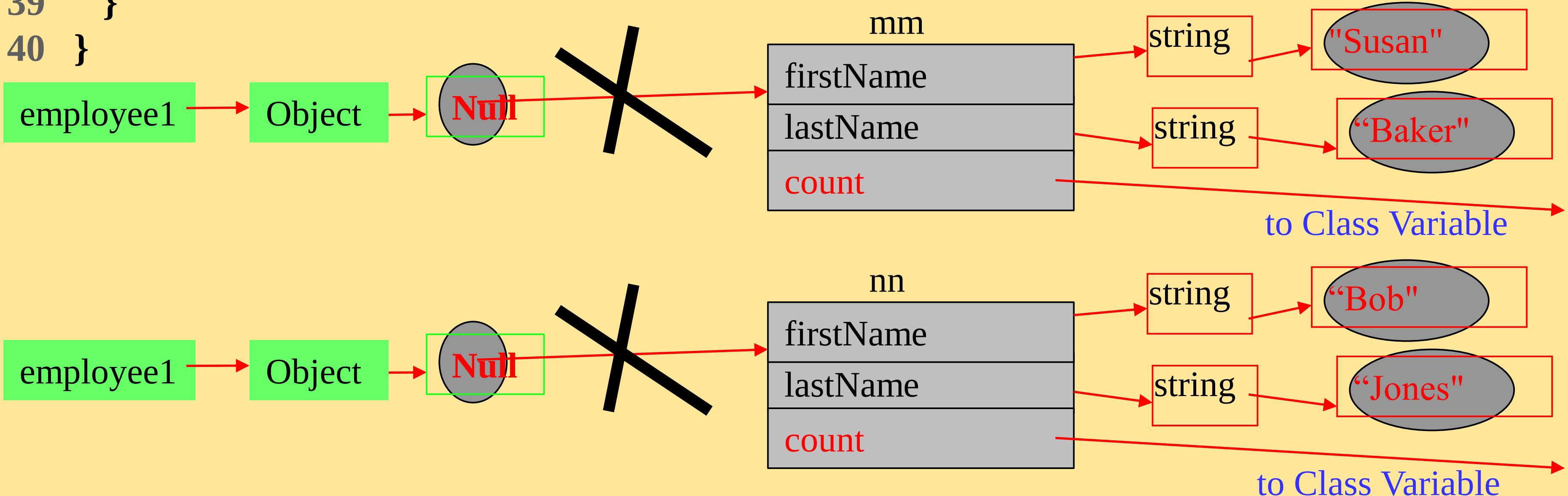
◀ ▶

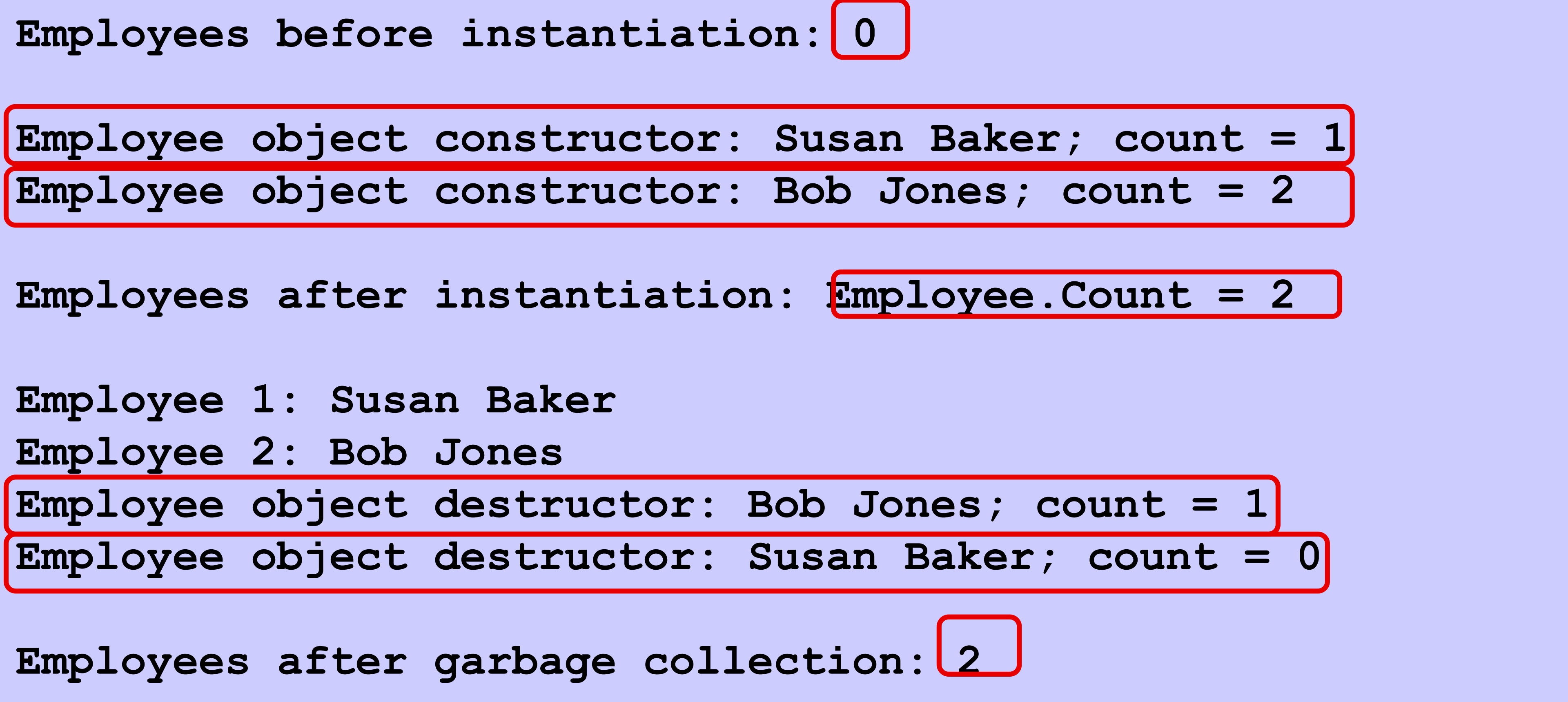
```
using System;
public class Employee  {
  private string firstName;
   private string lastName;
   private static int count;
   public Employee( string fName, string lName ) {
     firstName = fName;
     lastName = lName;
     ++count;

     Console.WriteLine( "Employee object constructor: " +
       firstName + " " + lastName + "; count = " + Count );
   }
   ~Employee()  {
     --count;
     Console.WriteLine( "Employee object destructor: " +
       firstName + " " + lastName + "; count = " + Count );
   }
```



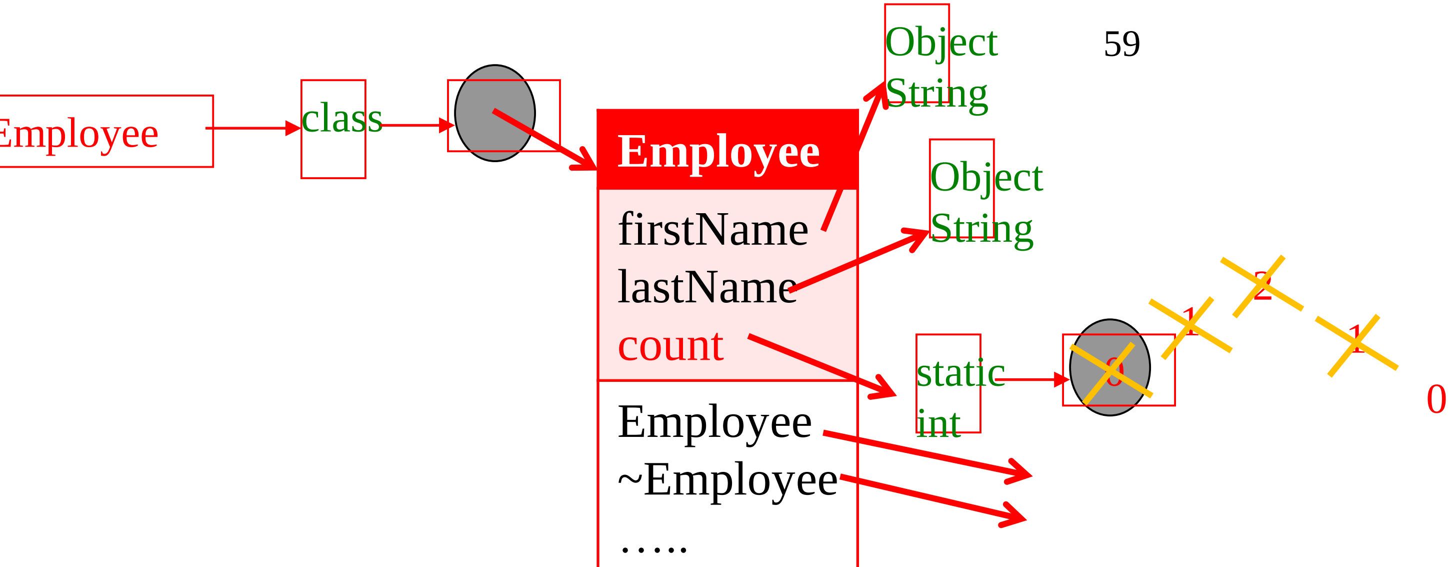
```
   public string FirstName    {
      get  {
         return firstName;
      }
   }
   public string LastName   {
      get   {
         return lastName;
      }
   }
   public static int Count    {
      get      {
         return count;
      }
   }
} // end class Employee
```

```
using System;
class StaticTest   {
   static void Main( string[] args )    {
      Console.WriteLine( "Employees before instantiation: " + Employee.Count + "\n" );
      Employee employee1 = new Employee( "Susan", "Baker" );
      Employee employee2 = new Employee( "Bob", "Jones" );
      Console.WriteLine( "\nEmployees after instantiation: " +
         "Employee.Count = " + Employee.Count + "\n" );
      Console.WriteLine( "Employee 1: " +   employee1.FirstName + " " +
                 employee1.LastName +   "\nEmployee 2: " + employee2.FirstName +
                 " " + employee2.LastName + "\n" );
      employee1 = null;
      employee2 = null;
      System.GC.Collect();
      Console.WriteLine( "\nEmployees after garbage collection: " + Employee.Count );
   }
}
```





```
Employees before instantiation: 0

Employee object constructor: Susan Baker; count = 1
Employee object constructor: Bob Jones; count = 2

Employees after instantiation: Employee.Count = 2

Employee 1: Susan Baker
Employee 2: Bob Jones
Employee object destructor: Bob Jones; count = 1
Employee object destructor: Susan Baker; count = 0

Employees after garbage collection: 2
```



const and readonly Members

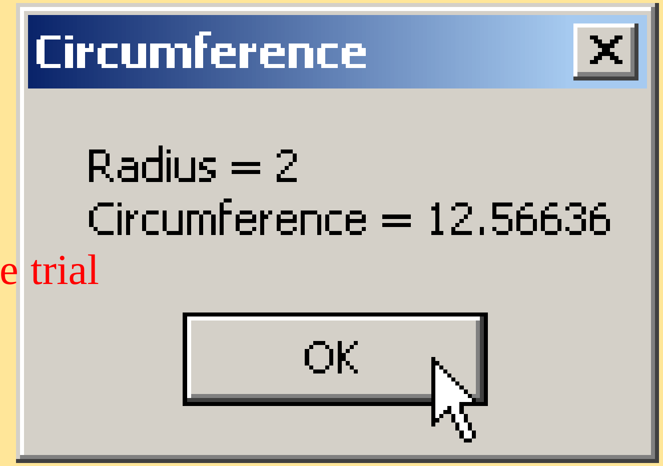
- Declare constant members using the keyword **const**
- **const** members are implicitly **static** (i.e. global variable)
  - Use "static" syntax to access the constant member
- **const** members must be initialized when they are declared

- Use keyword **readonly** to declare members who will be initialized in the constructor but not change after that

```
using System;
using System.Windows.Forms;
public class Constants   {
   public const double PI = 3.14159;
   public readonly int radius;
   public Constants( int radiusValue )  {
      radius = radiusValue;
   }
 }
 public class UsingConstAndReadOnly   {
   static void Main( string[] args )   {
      Random random = new Random();
      Constants constantValues =  new Constants(random.Next( 1, 20 ) );
      MessageBox.Show( "Radius = " + constantValues.radius +
         "\nCircumference = " + 2 * Constants.PI * constantValues.radius,
         "Circumference" );
   }
 }
```

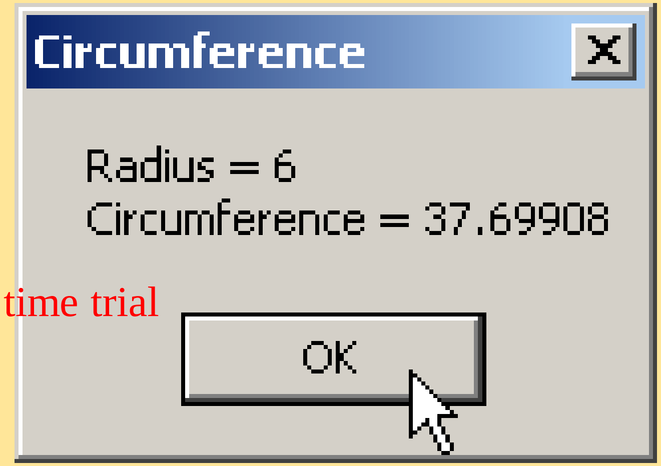
Get a random-number object

Get next random number



```
Circumference

Radius = 2
Circumference = 12.56636
```

First time trial

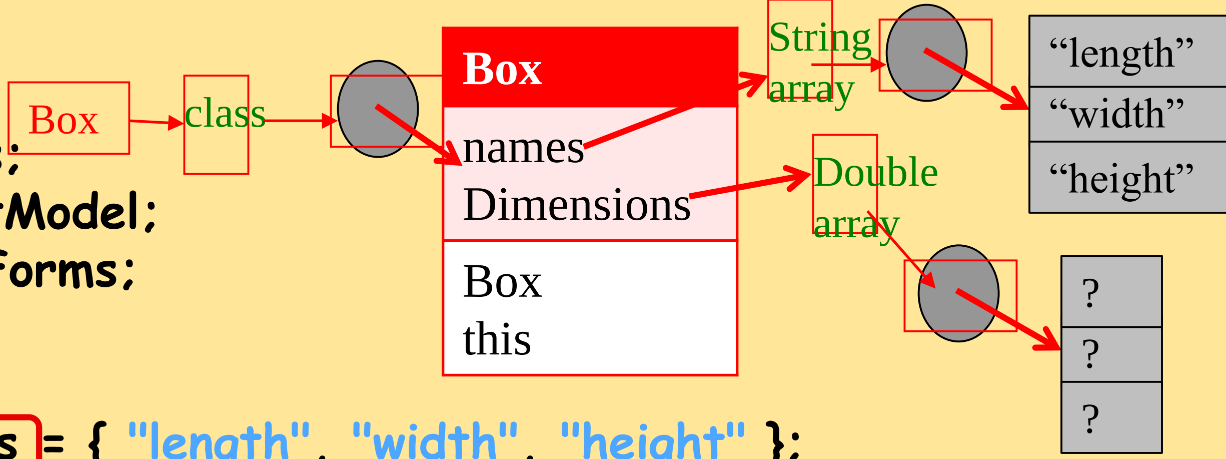


```
Circumference

Radius = 6
Circumference = 37.69908
```

Second time trial

Indexers

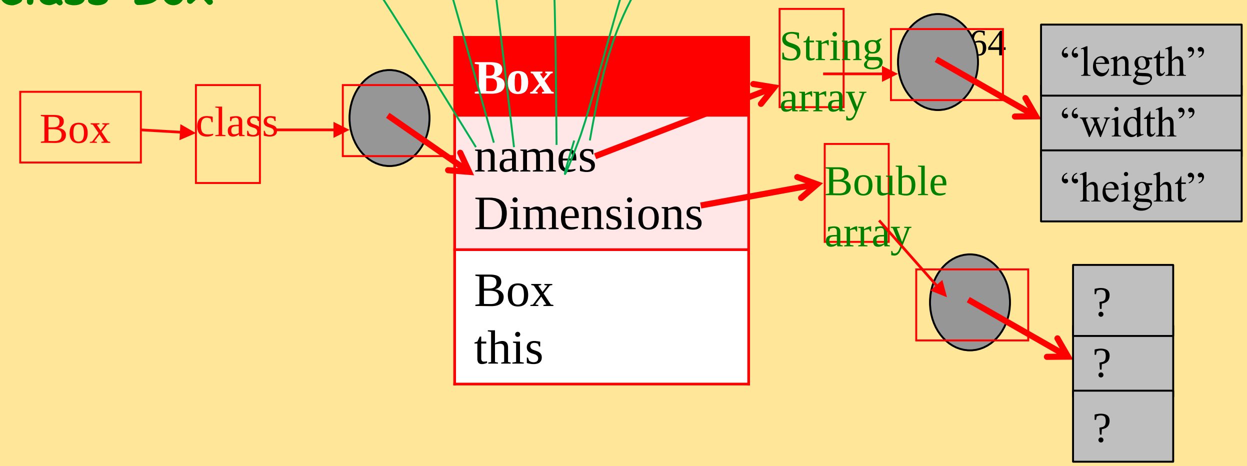
- Sometimes a classes encapsulates data which is like a list of elements (把class當作一串(或陣列)資料使用)
- Indexers are special properties that allow array-style access to the data in the class (此部份由*C#*提供程式架構)
- Indexers can be defined to accept both integer and non-integer subscripts (此部份由程式師配合)
  - By Overloaded properties
- When using indexers, programmers use the bracket ([]) notation, as arrays, for **get** and **set** accessors(get, set 程式內容由程式師提供)



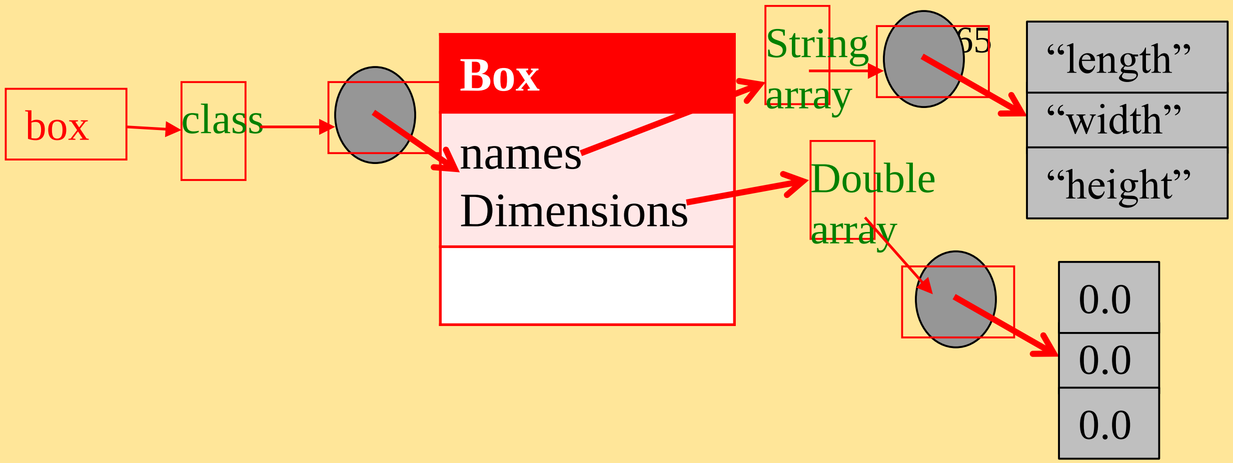
```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
 using System.Data;
 public class Box   {
    private string[] names = { "length", "width", "height" };
    private double[] dimensions = new double[ 3 ];
    public Box ( double length, double width, double height )   {
       dimensions[ 0 ] = length;
       dimensions[ 1 ] = width;
       dimensions[ 2 ] = height;
    }
    public double this[ int index]    {
       get  {
          return ( index < 0 || index > dimensions.Length ) ?
             -1  : dimensions[ index] ;
       }
       set  {
          if ( index >= 0 && index < dimensions.Length )
             dimensions[ index ] = value;
       }
    } // end numeric indexer
```

It is a property function

```
   // access dimensions by their names
   public double this[ string name ]    {
      get       {
         int i = 0;
         while ( i < names.Length && name.ToLower() != names[ i ] )
            i++;
         return ( i == names.Length ) ? -1  : dimensions[ i ];
      }
      set      {
         int i = 0;
         while ( i < names.Length && name.ToLower() != names[ i ] )
            i++;
         if ( i != names.Length )
            dimensions[ i ] = value;
      }
   } // end indexer
} // end class Box
```

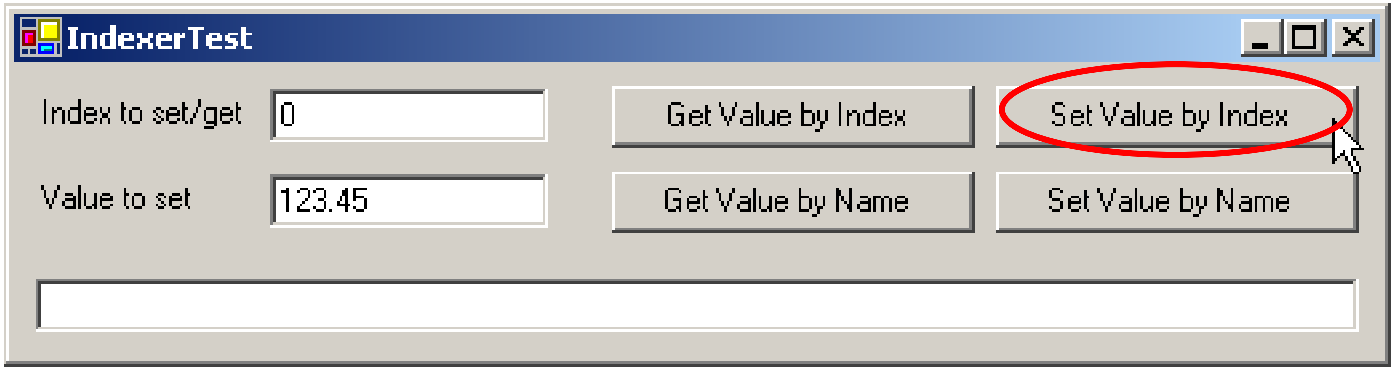


```
public class IndexerTest : System.Windows.Forms.Form {
    private System.Windows.Forms.Label indexLabel;
    private System.Windows.Forms.Label nameLabel;
    private System.Windows.Forms.TextBox indexTextBox;
    private System.Windows.Forms.TextBox valueTextBox;
    private System.Windows.Forms.Button nameSetButton;
    private System.Windows.Forms.Button nameGetButton;
    private System.Windows.Forms.Button intSetButton;
    private System.Windows.Forms.Button intGetButton;
    private System.Windows.Forms.TextBox resultTextBox;
    private System.ComponentModel.Container components = null;
    private Box box;
    public IndexerTest() {
        // required for Windows Form Designer support
        InitializeComponent();
        box = new Box( 0.0, 0.0, 0.0 );
    }
```



```
// main entry point for application
[STAThread]
static void Main()  {
   Application.Run( new IndexerTest() );
}
private void ShowValueAtIndex( string prefix, int index )  {
   resultTextBox.Text =
      prefix + "box[ " + index + " ] = " + box[ index ];
}
private void ShowValueAtIndex( string prefix, string name )   {
   resultTextBox.Text =
      prefix + "box[ " + name + " ] = " + box[ name ];
}
private void ClearTextBoxes()  {
   indexTextBox.Text = "";
   valueTextBox.Text = "";
}
```

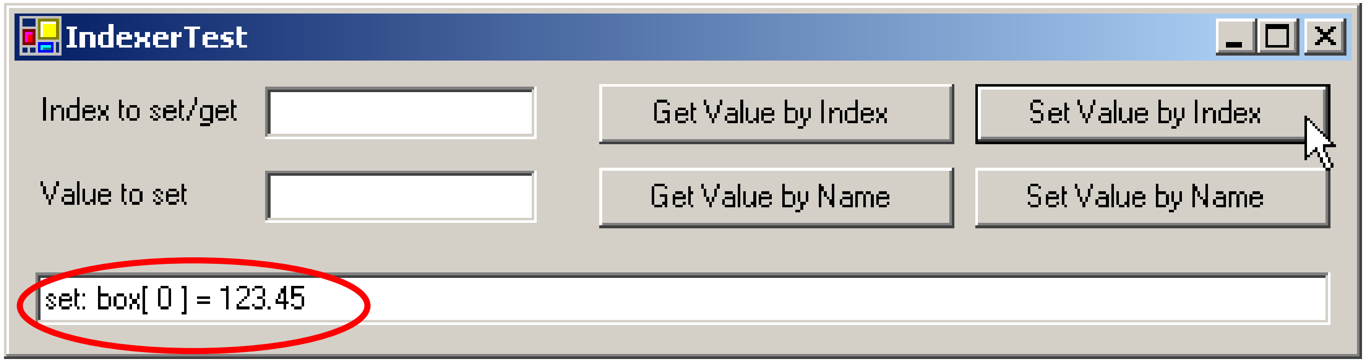
```
    // get value at specified index
    private void intGetButton_Click(object sender, System.EventArgs e ) {
        ShowValueAtIndex("get: ", Int32.Parse( indexTextBox.Text ) );
        ClearTextBoxes();
    }
    private void intSetButton_Click(object sender, System.EventArgs e ) {
        int index = Int32.Parse( indexTextBox.Text);
        box[index] = Double.Parse( valueTextBox.Text );
        ShowValueAtIndex( "set: ", index );
        ClearTextBoxes();
    }
    private void nameGetButton_Click(object sender, System.EventArgs e) {
        ShowValueAtIndex( "get: ", indexTextBox.Text );
        ClearTextBoxes();
    }
    private void nameSetButton_Click(object sender, System.EventArgs e) {
        box[ indexTextBox.Text ] = Double.Parse( valueTextBox.Text );
        ShowValueAtIndex( "set: ", indexTextBox.Text );
        ClearTextBoxes();
    }
} // end class IndexerTest
```



Index to set/get 0

Value to set 123.45

Before setting value by index number

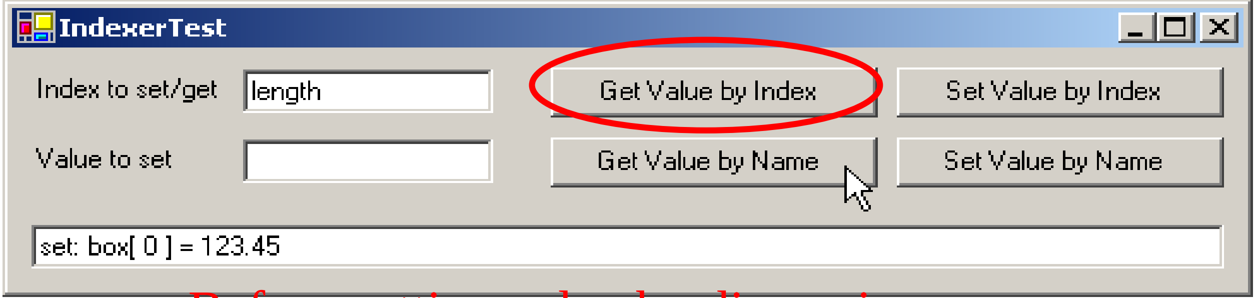


Index to set/get

Value to set

```
set: box[ 0 ] = 123.45
```

After setting value by index number

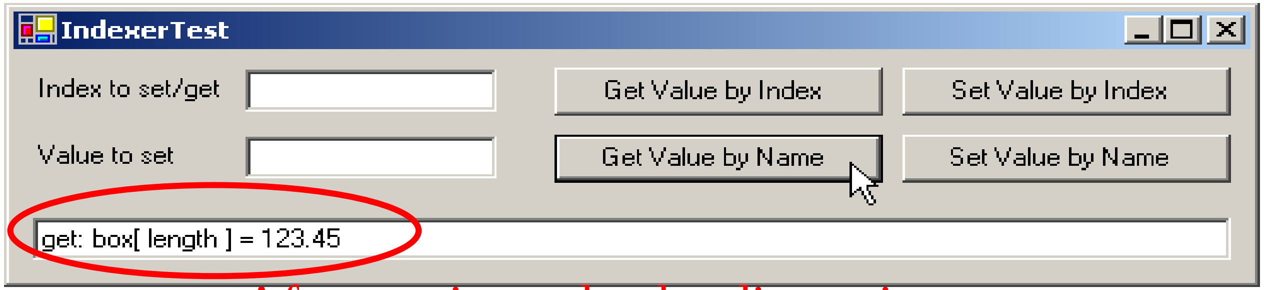


Index to set/get length

Value to set

set: box[0] = 123.45

Before getting value by dimension name

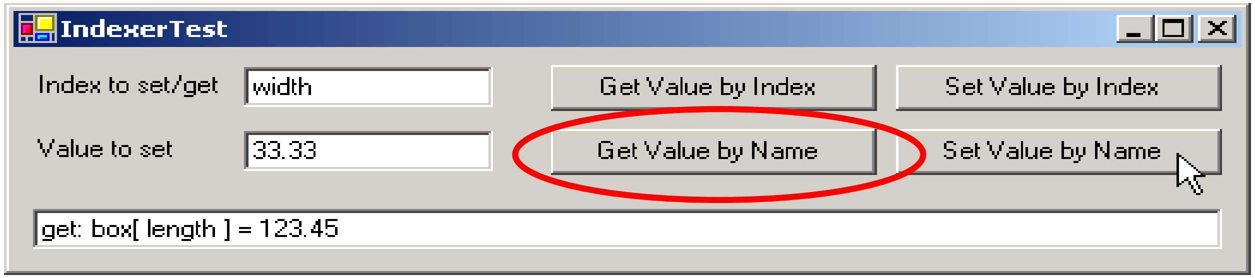


Index to set/get

Value to set

get: box[length] = 123.45

After getting value by dimension name

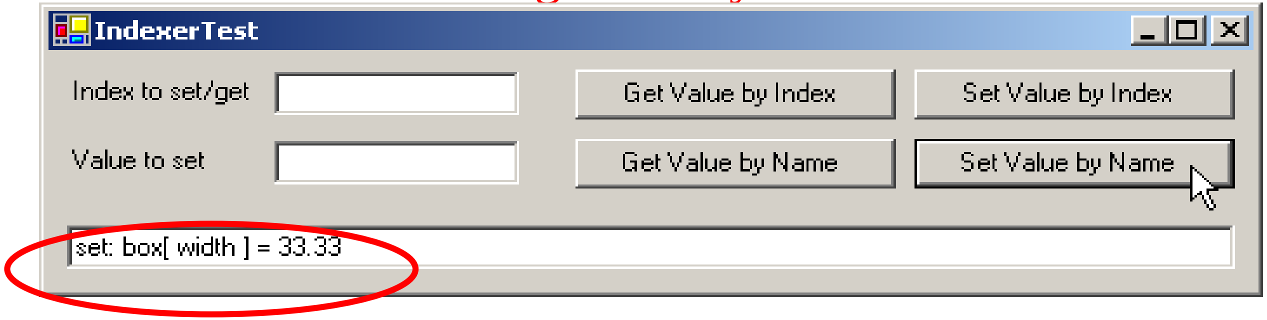


Index to set/get width

Value to set 33.33

get: box[length] = 123.45

Before setting value by dimension name



Index to set/get

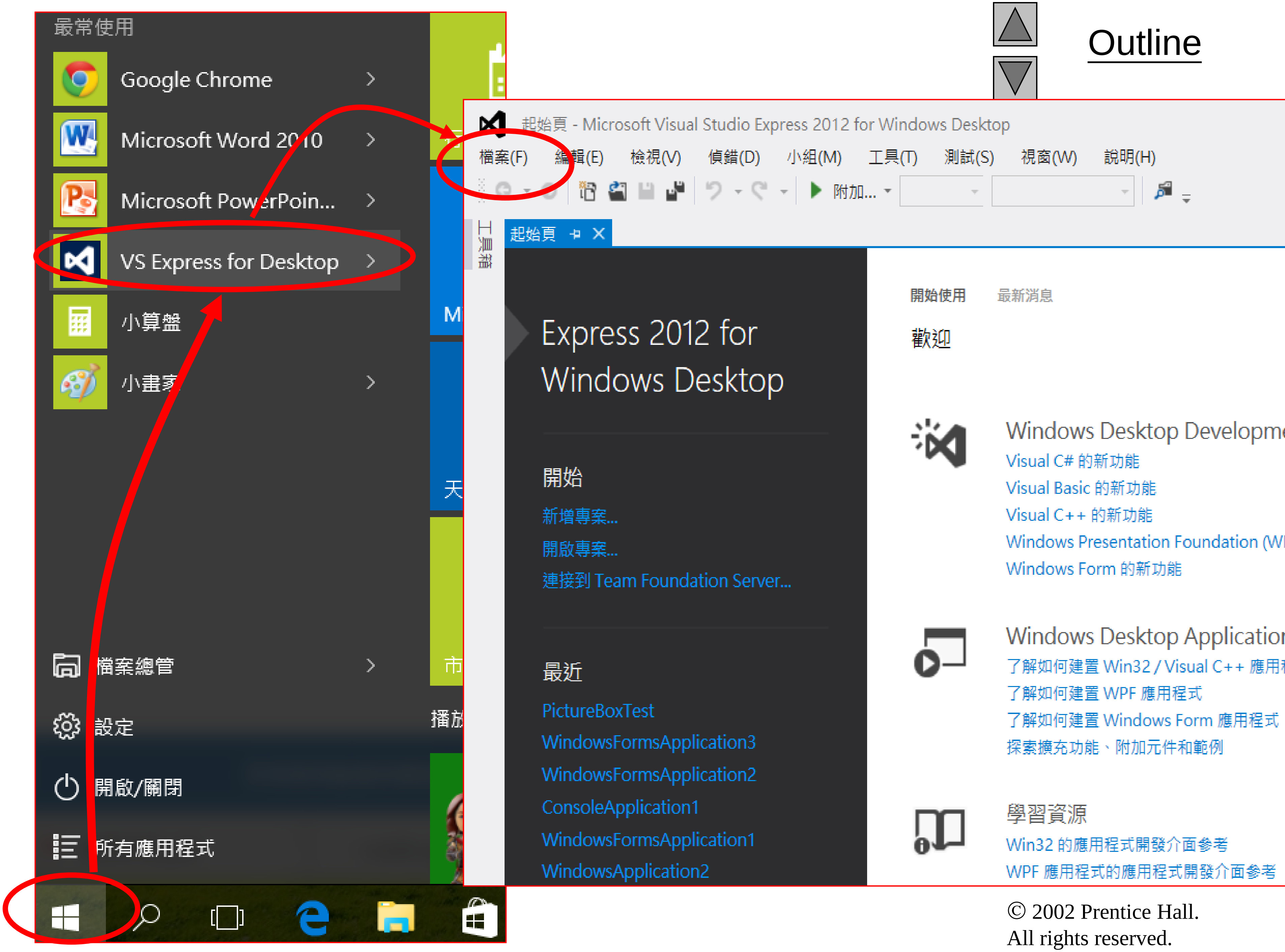
Value to set

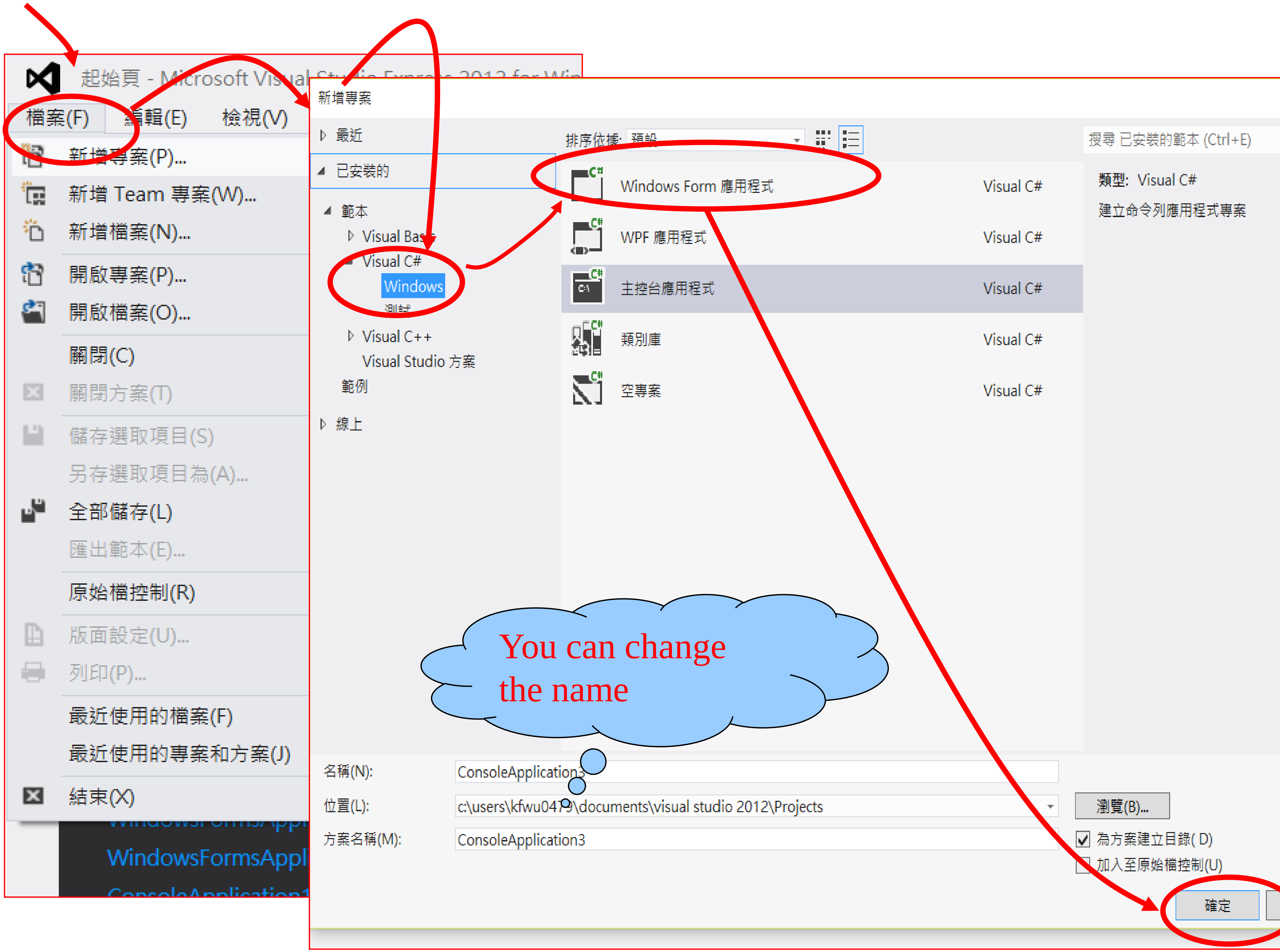
set: box[width] = 33.33

After setting value by dimension name

How to use IDE for Window-form Programming?

Here's what the IDE automates for you...


最常使用
Google Chrome
Microsoft Word 2010
Microsoft PowerPoin...
VS Express for Desktop
小算盤
小畫家
檔案總管
設定
開啟/關閉
所有應用程式
起始頁 - Microsoft Visual Studio Express 2012 for Windows Desktop
檔案(F)
編輯(E)
檢視(V)
偵錯(D)
小組(M)
工具(T)
測試(S)
視窗(W)
說明(H)
附加...
工具箱
起始頁
Express 2012 for Windows Desktop
開始
新增專案...
開啟專案...
連接到 Team Foundation Server...
最近
PictureBoxTest
WindowsFormsApplication3
WindowsFormsApplication2
ConsoleApplication1
WindowsFormsApplication1
WindowsApplication2
開始使用
最新消息
歡迎
Windows Desktop Developme
Visual C# 的新功能
Visual Basic 的新功能
Visual C++ 的新功能
Windows Presentation Foundation (W
Windows Form 的新功能
Windows Desktop Applicatio
了解如何建置 Win32 / Visual C++ 應用和
了解如何建置 WPF 應用程式
了解如何建置 Windows Form 應用程式
探索擴充功能、附加元件和範例
學習資源
Win32 的應用程式開發介面參考
WPF 應用程式的應用程式開發介面參考



You can change the name

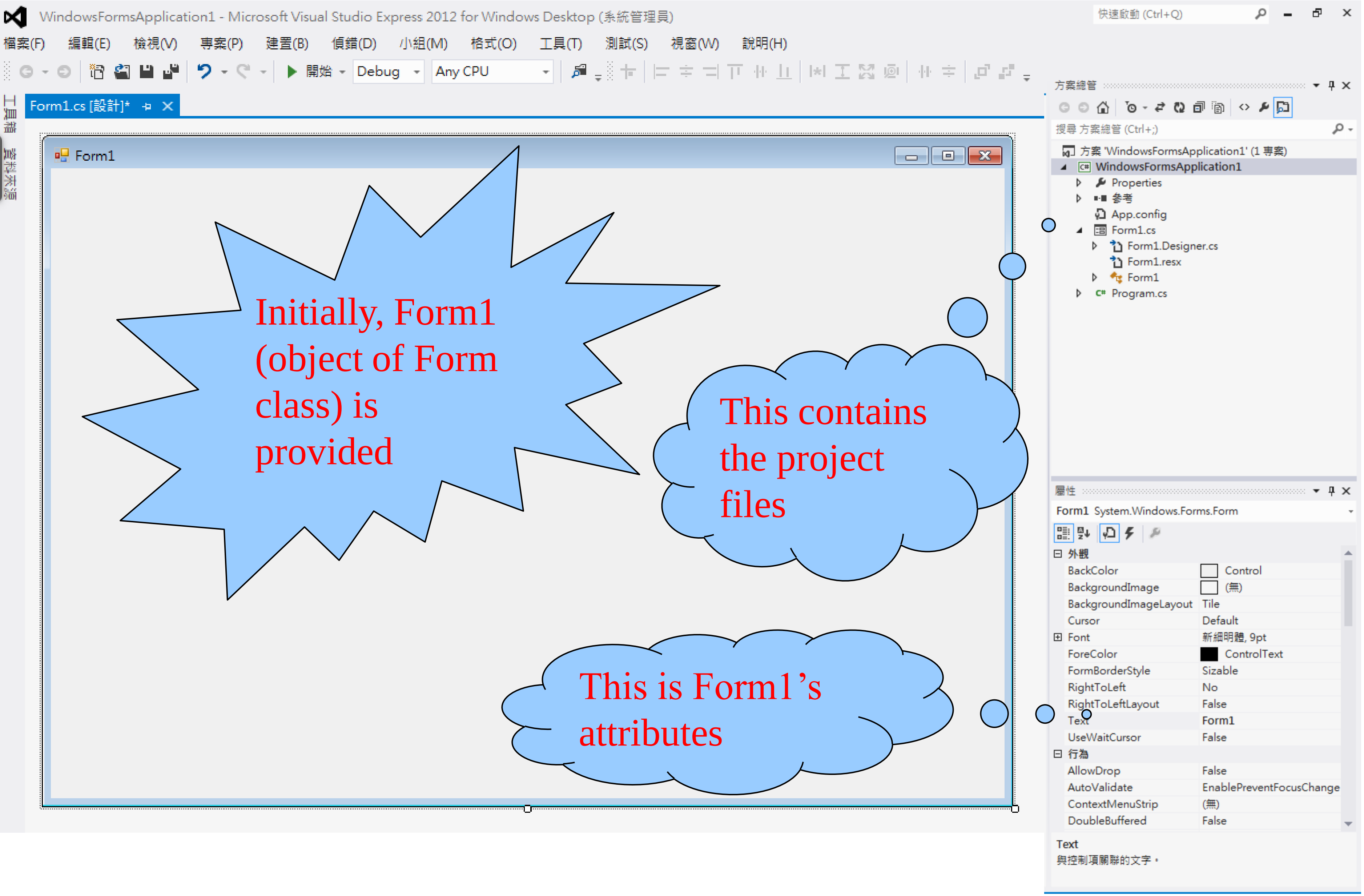
名稱(N): ConsoleApplication3

位置(L): c:\users\kfwu0479\documents\visual studio 2012\Projects

方案名稱(M): ConsoleApplication3

☑ 為方案建立目錄(D)

□ 加入至原始檔控制(U)


WindowsFormsApplication1 - Microsoft Visual Studio Express 2012 for Windows Desktop (系統管理員)
快速啟動 (Ctrl+Q)
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 格式(O) 工具(T) 測試(S) 視窗(W) 說明(H)
開始
Debug
Any CPU
Form1.cs [設計]*
工具箱
資料來源
Form1
方案總管
搜尋 方案總管 (Ctrl+;)
方案 'WindowsFormsApplication1' (1 專案)
WindowsFormsApplication1
Properties
參考
App.config
Form1.cs
Form1.Designer.cs
Form1.resx
Form1
Program.cs
屬性
Form1 System.Windows.Forms.Form

Initially, Form1 (object of Form class) is provided

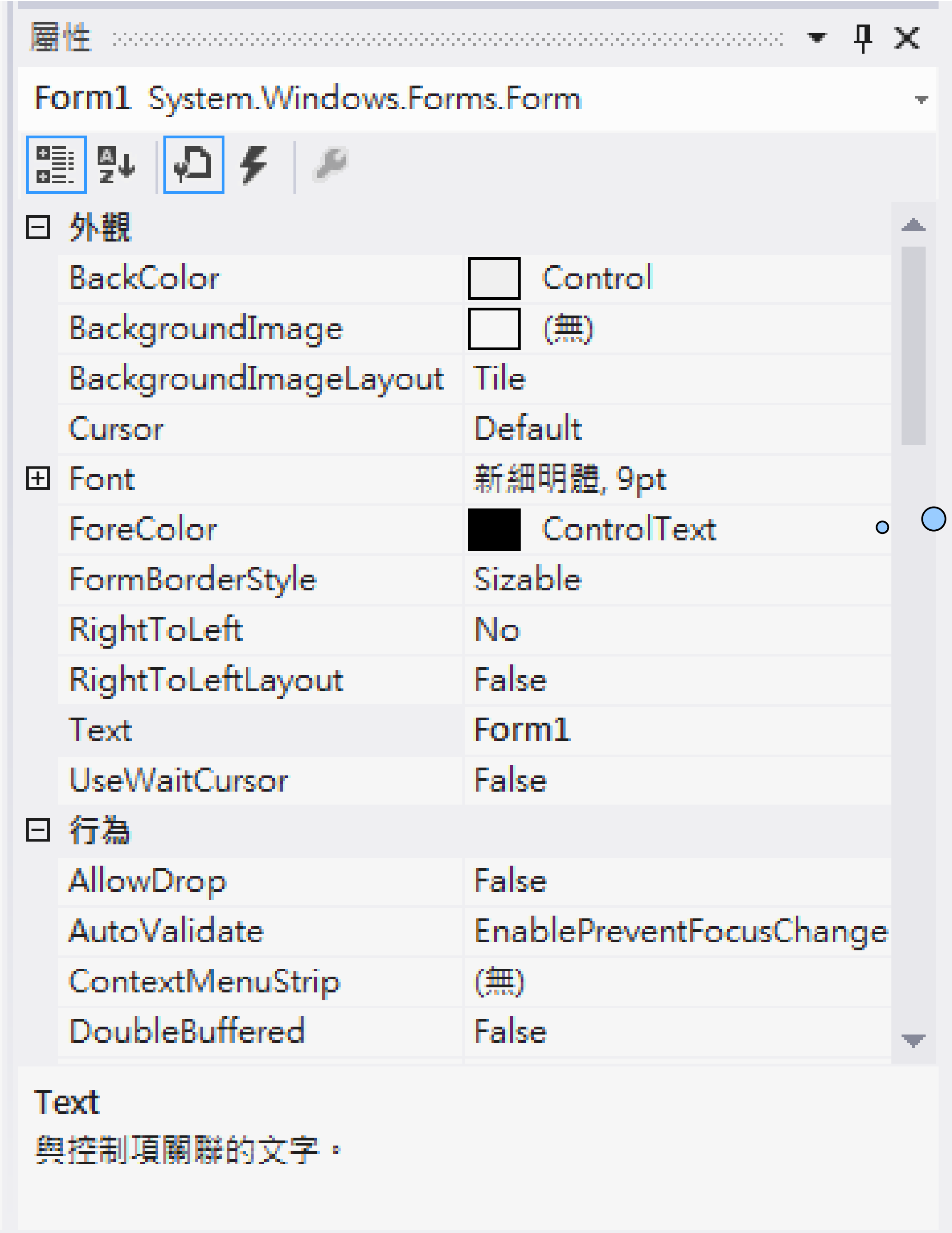
This contains the project files

This is Form1's attributes

```
⊟ 外觀
  BackColor                  □ Control
  BackgroundImage            □ (無)
  BackgroundImageLayout      Tile
  Cursor                     Default
⊞ Font                       新細明體, 9pt
  ForeColor                  ■ ControlText
  FormBorderStyle            Sizable
  RightToLeft                No
  RightToLeftLayout          False
  Text                       Form1
  UseWaitCursor              False
⊟ 行為
  AllowDrop                  False
  AutoValidate               EnablePreventFocusChange
  ContextMenuStrip           (無)
  DoubleBuffered             False
```

Text
與控制項關聯的文字。

◀ ▶



屬性

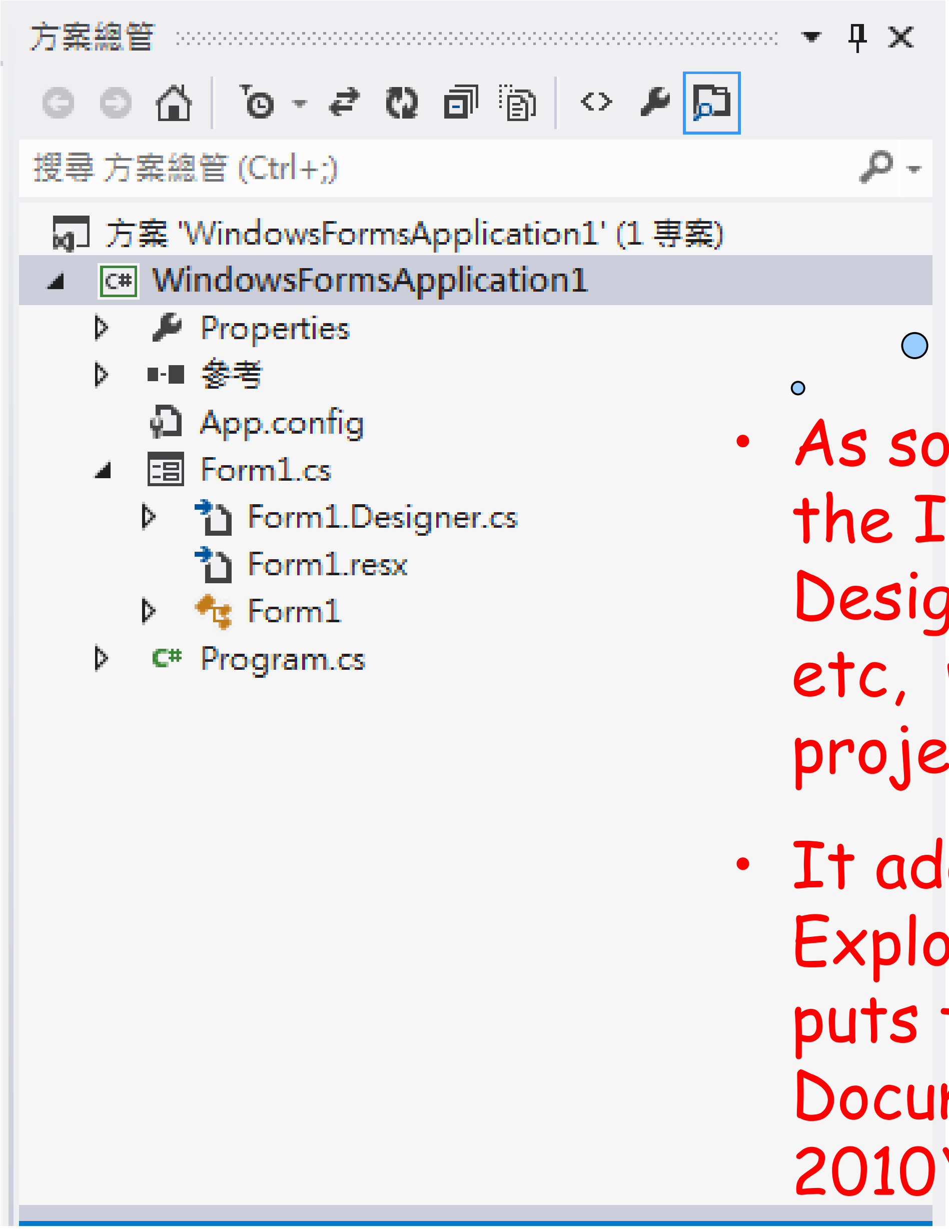
Form1 System.Windows.Forms.Form

⊟ 外觀	
BackColor	Control
BackgroundImage	(無)
BackgroundImageLayout	Tile
Cursor	Default
⊞ Font	新細明體, 9pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Form1
UseWaitCursor	False
⊟ 行為	
AllowDrop	False
AutoValidate	EnablePreventFocusChange
ContextMenuStrip	(無)
DoubleBuffered	False

Text

與控制項關聯的文字。

This is Form1's attributes9



方案總管

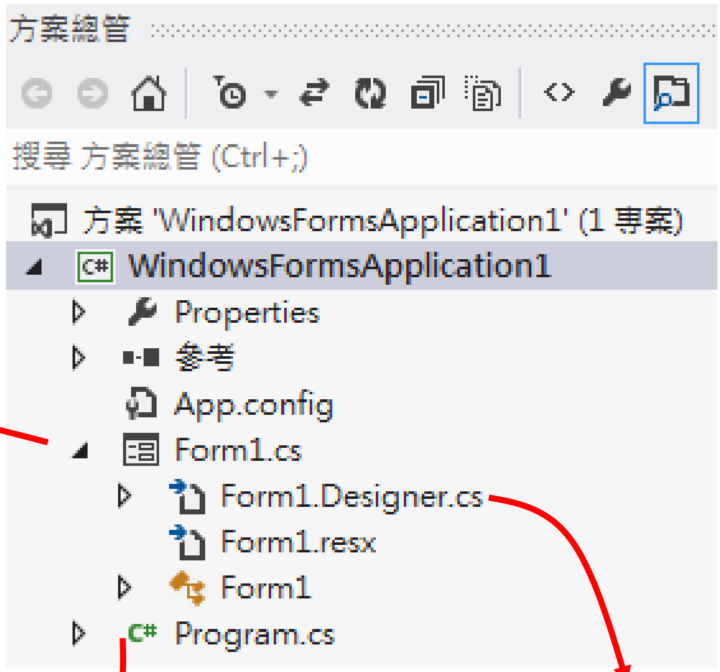
搜尋 方案總管 (Ctrl+;)

方案 'WindowsFormsApplication1' (1 專案)

- WindowsFormsApplication1
  - Properties
  - 參考
  - App.config
  - Form1.cs
    - Form1.Designer.cs
    - Form1.resx
    - Form1
  - Program.cs

This contains the project files

- As soon as you save the project, the IDE creates Form1.cs, Form1, Designer.cs, and Program.cs file, etc, when you create a new project.
- It adds these to the Solution Explorer window, and by default, puts those files in My Documents\Visual Studio 2010\Projects\Contacts\.



方案總管

搜尋 方案總管 (Ctrl+;)

方案 'WindowsFormsApplication1' (1 專案)
- WindowsFormsApplication1
  - Properties
  - 參考
  - App.config
  - Form1.cs
    - Form1.Designer.cs
    - Form1.resx
    - Form1
  - Program.cs

This file contains the C# code that defines the behavior of the form.

This has the code that starts up the program and displays the form.

The code that defines the form and its objects lives here.

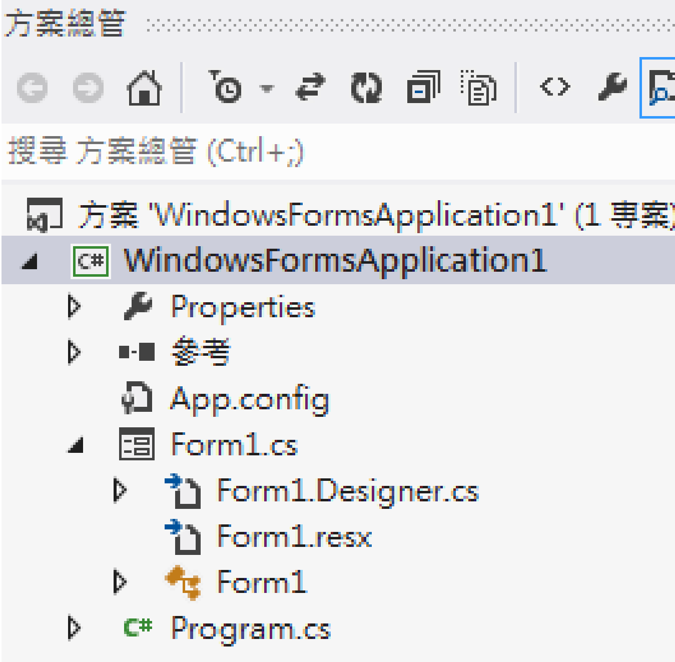
Form1.cs

Program.cs

Form1.Designer.cs

Visual Studio creates all three of these files automatically.

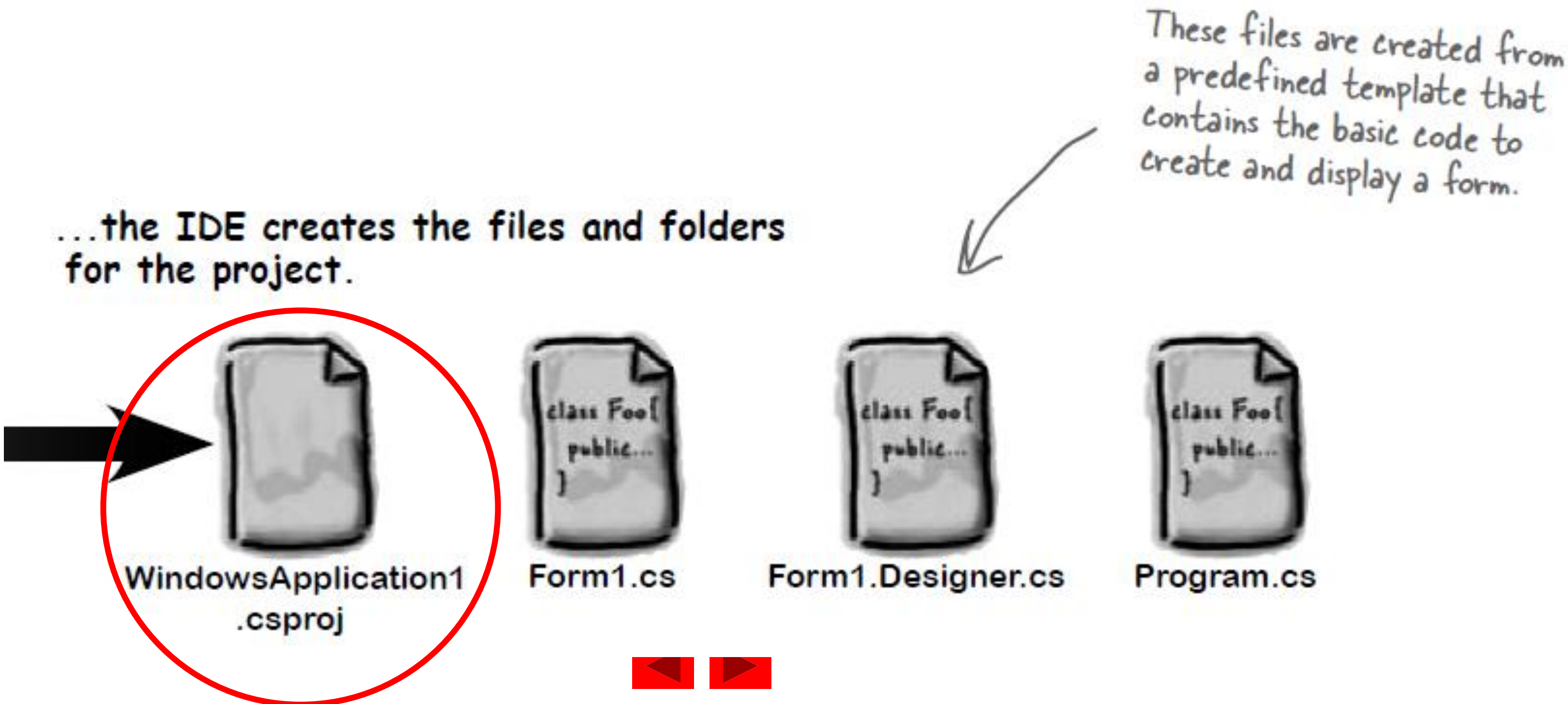
- IDE bundles all of files for your program by creating a solution file and a folder that contains all files
- The solution file has a list of project file (which end in .csproj)



方案總管

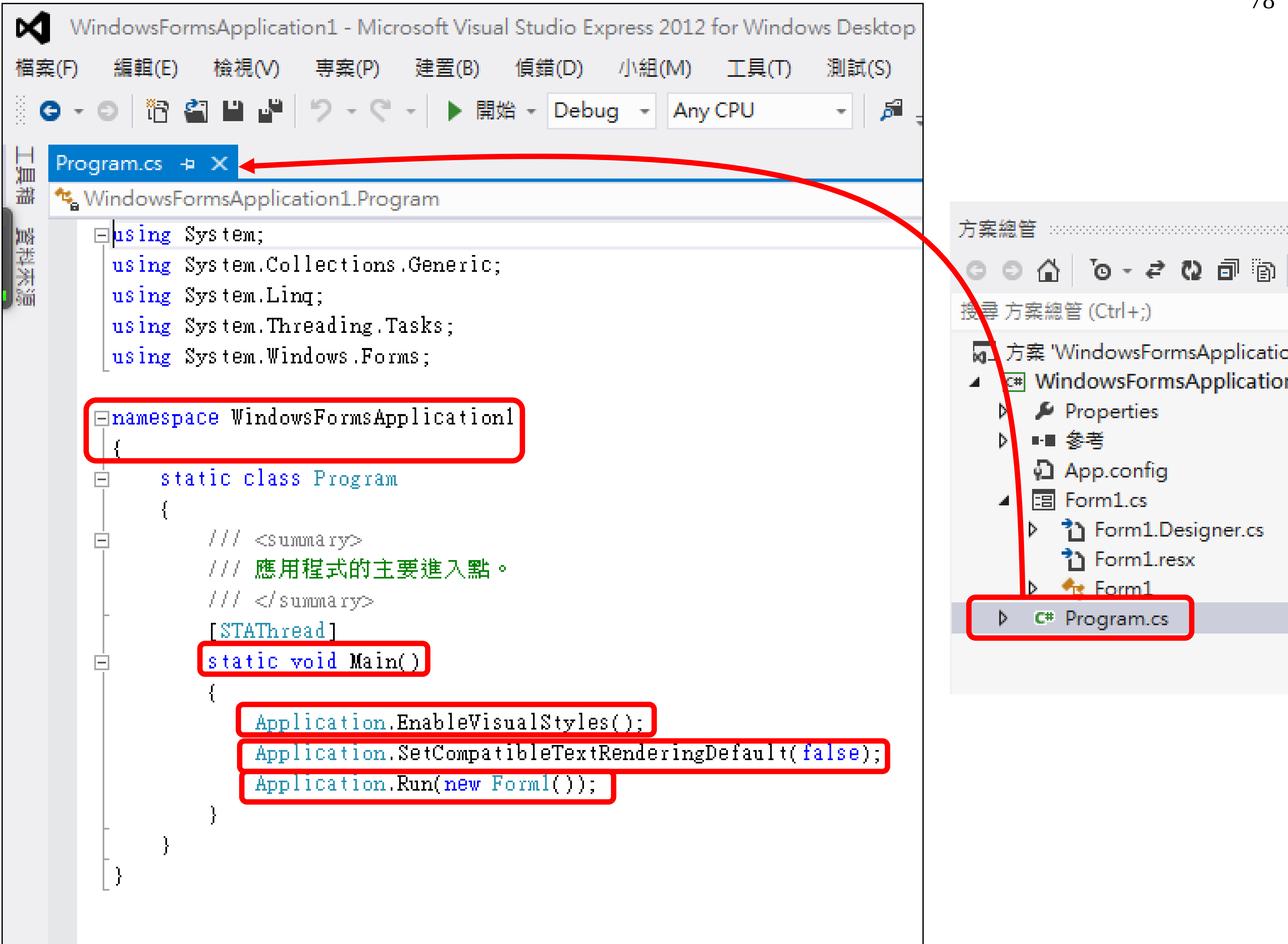
搜尋 方案總管 (Ctrl+;)

```
方案 'WindowsFormsApplication1' (1 專案
◢ WindowsFormsApplication1
    ▷ Properties
    ▷ 參考
      App.config
    ◢ Form1.cs
        ▷ Form1.Designer.cs
          Form1.resx
        ▷ Form1
    ▷ Program.cs
```



These files are created from a predefined template that contains the basic code to create and display a form.

...the IDE creates the files and folders for the project.



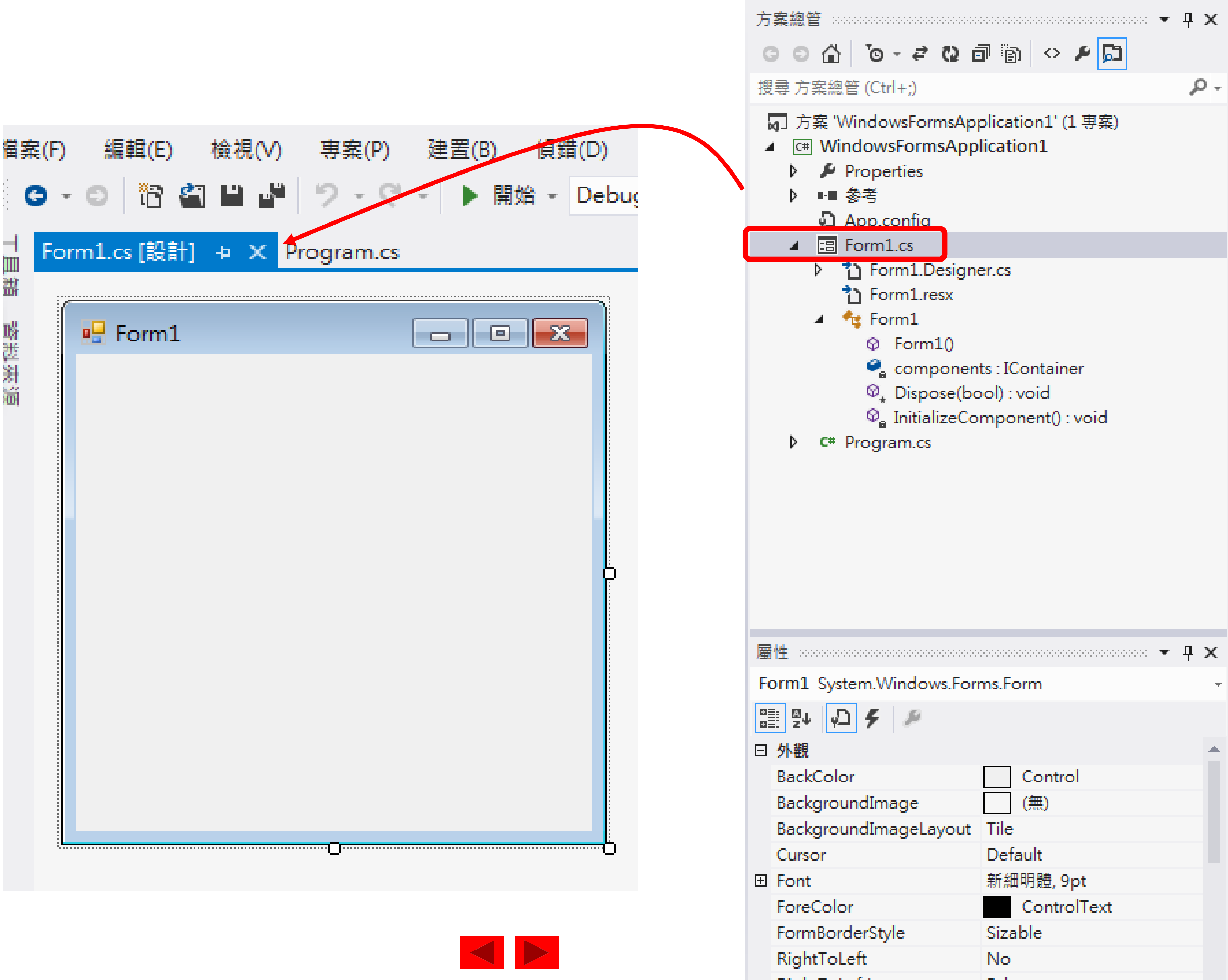
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    static class Program
    {
        /// <summary>
        /// 應用程式的主要進入點。
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Comments

- [STAThread]：Single Thread Apartment (單一執行緒執行區)
- Application Class
  - (You can think it) Application is an process running in a thread.
  - Application Provides **static** methods and properties to manage an application,
    - such as methods to start and stop an application,
    - to process Windows messages, and properties to get information about an application
- Application.run(new Form1)
  - Begins running a standard application message loop on the current thread, and makes the specified form visible.

comments

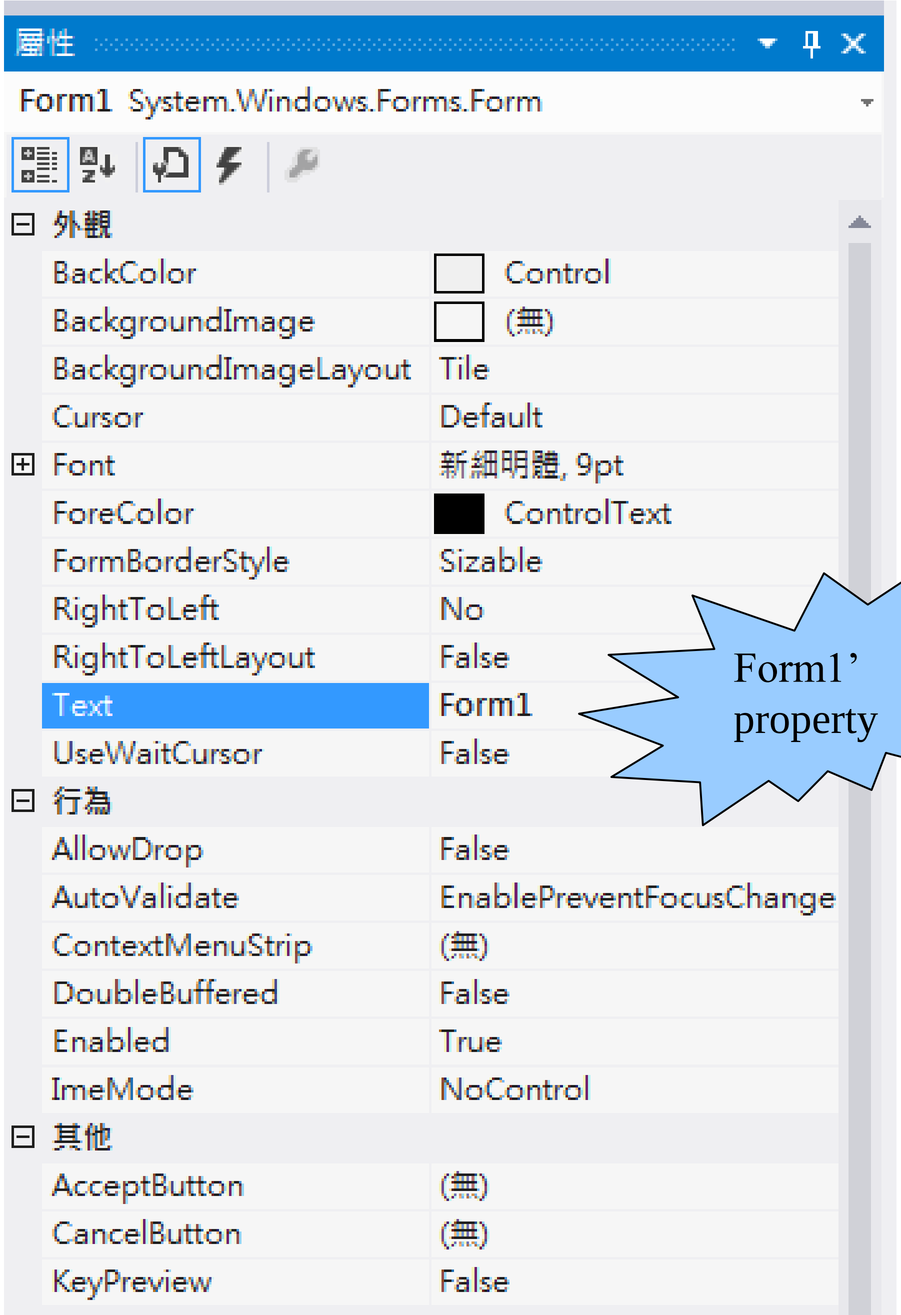
- Application.EnableVisualStyles（）
  - This method is static and enable application visibility.
  - Generally, EnableVisualStyles is the first statement in Main(). That is, the function can be activated before the form usage.



方案 'WindowsFormsApplication1' (1 專案)
- WindowsFormsApplication1
  - Properties
  - 參考
  - App.config
  - Form1.cs
    - Form1.Designer.cs
    - Form1.resx
    - Form1
      - Form1()
      - components : IContainer
      - Dispose(bool) : void
      - InitializeComponent() : void
  - Program.cs

Form1 System.Windows.Forms.Form

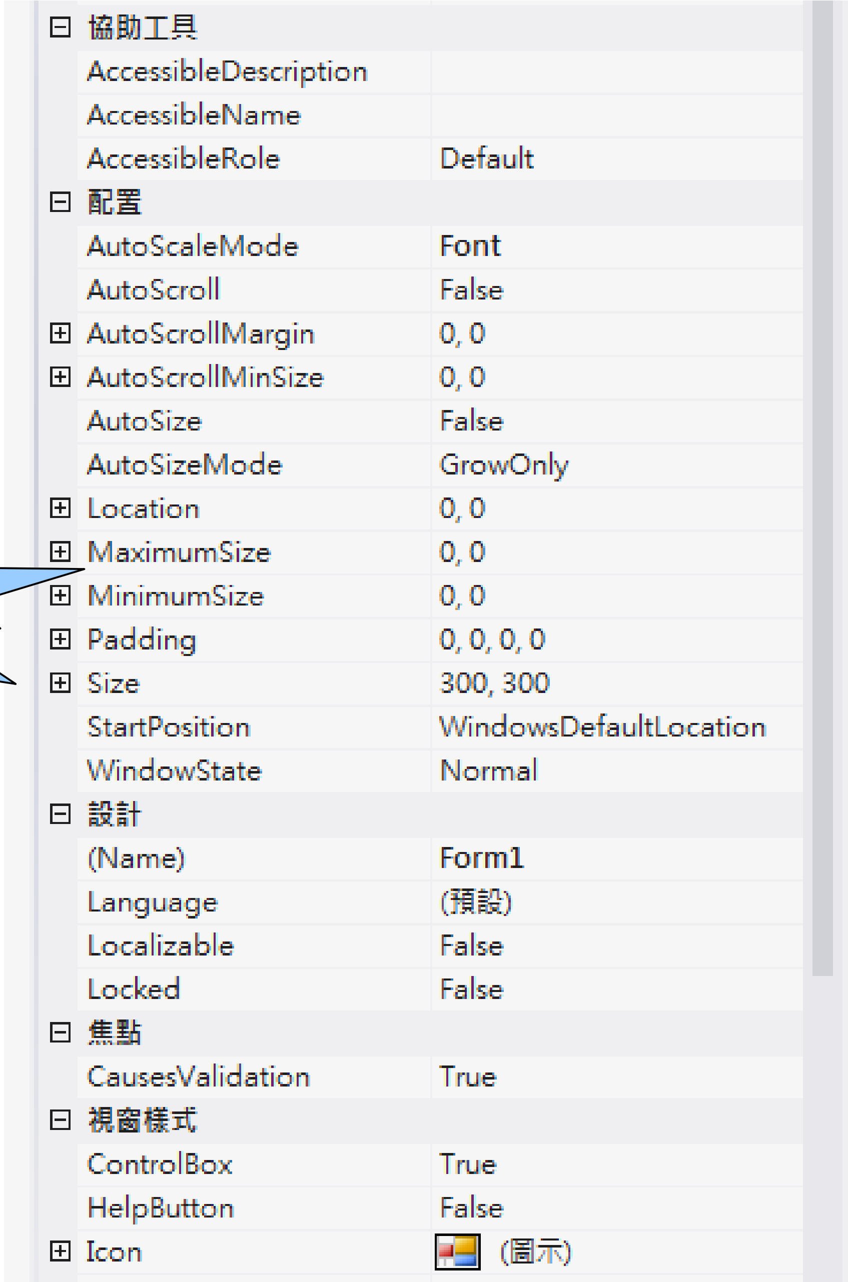
⊟ 外觀	
BackColor	□ Control
BackgroundImage	□ (無)
BackgroundImageLayout	Tile
Cursor	Default
⊞ Font	新細明體, 9pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No



Form1 System.Windows.Forms.Form

⊟ 外觀	
BackColor	□ Control
BackgroundImage	□ (無)
BackgroundImageLayout	Tile
Cursor	Default
⊞ Font	新細明體, 9pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Form1
UseWaitCursor	False
⊟ 行為	
AllowDrop	False
AutoValidate	EnablePreventFocusChange
ContextMenuStrip	(無)
DoubleBuffered	False
Enabled	True
ImeMode	NoControl
⊟ 其他	
AcceptButton	(無)
CancelButton	(無)
KeyPreview	False

Form1’ property

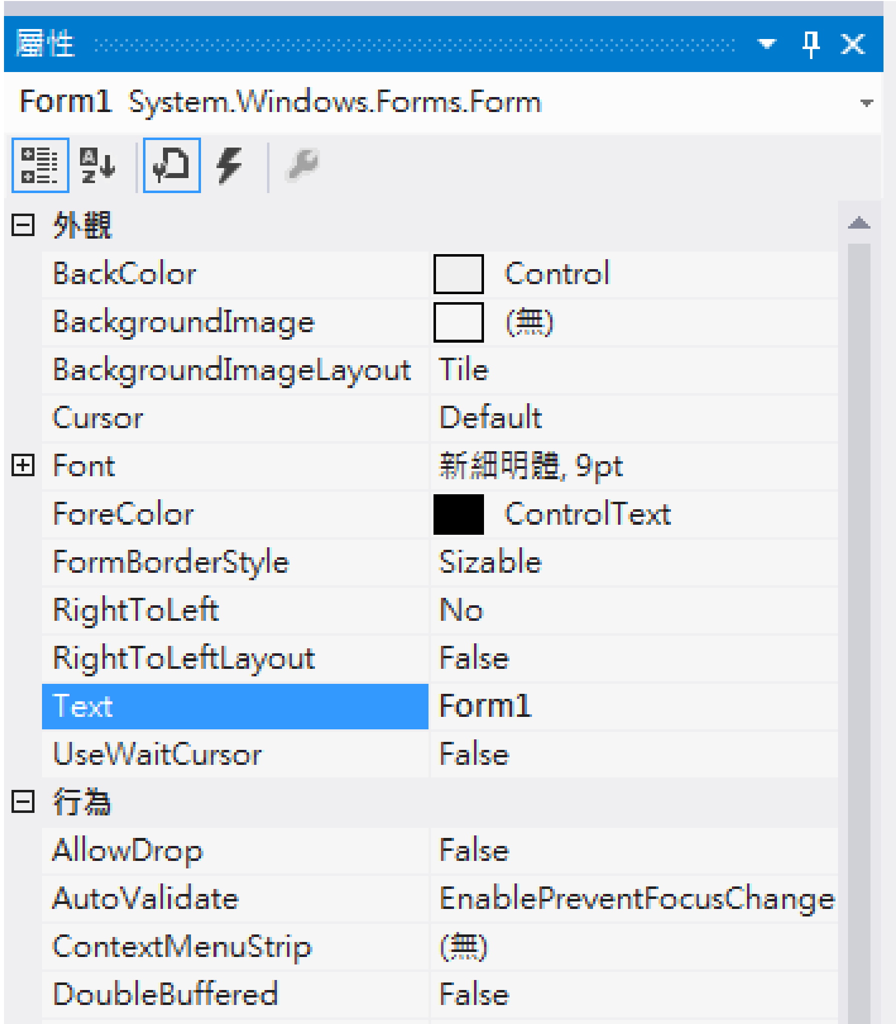


⊟ 協助工具	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
⊟ 配置	
AutoScaleMode	Font
AutoScroll	False
⊞ AutoScrollMargin	0, 0
⊞ AutoScrollMinSize	0, 0
AutoSize	False
AutoSizeMode	GrowOnly
⊞ Location	0, 0
⊞ MaximumSize	0, 0
⊞ MinimumSize	0, 0
⊞ Padding	0, 0, 0, 0
⊞ Size	300, 300
StartPosition	WindowsDefaultLocation
WindowState	Normal
⊟ 設計	
(Name)	Form1
Language	(預設)
Localizable	False
Locked	False
⊟ 焦點	
CausesValidation	True
⊟ 視窗樣式	
ControlBox	True
HelpButton	False
⊞ Icon	(圖示)

◀ ▶

Setting a property on your form

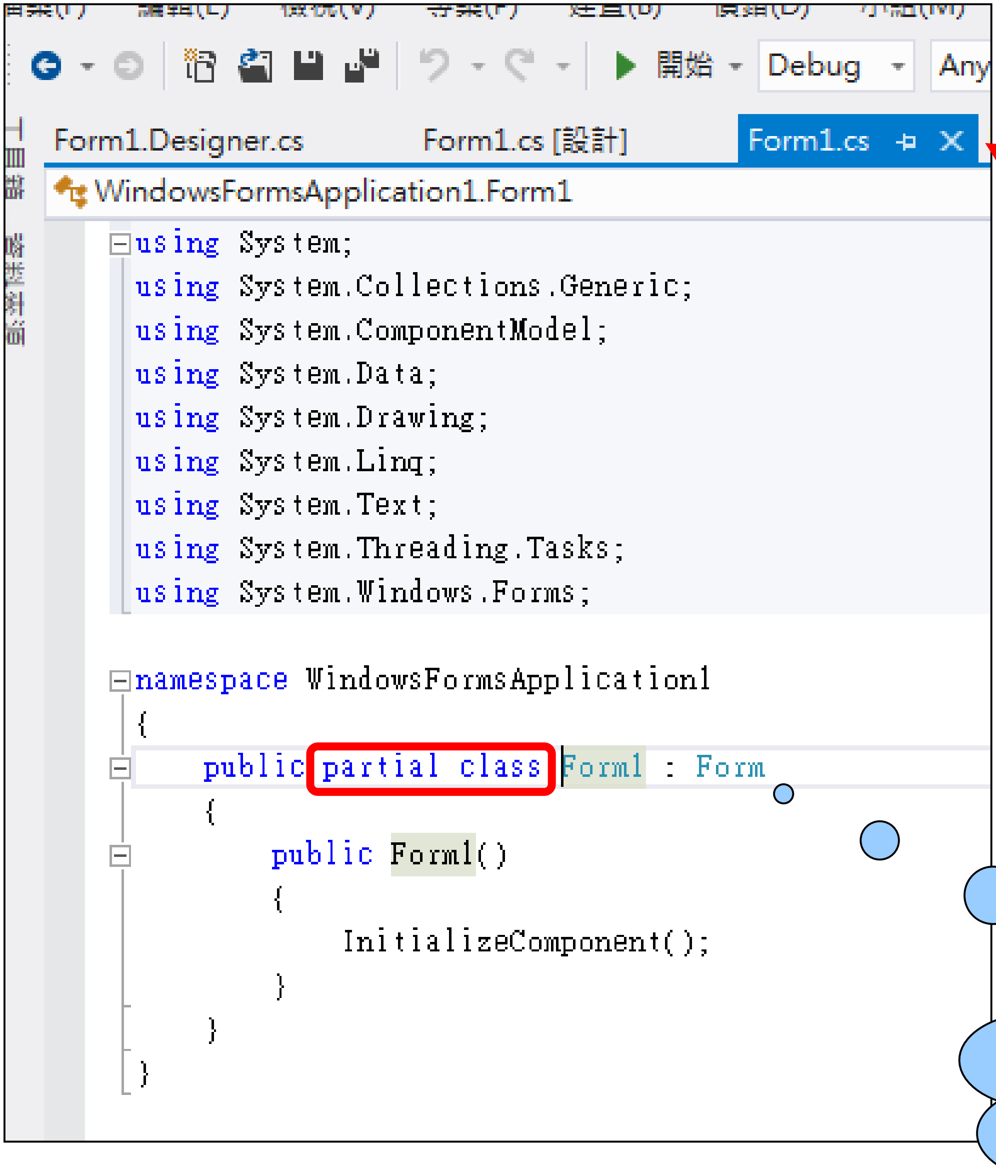
- The property window is a powerful tool that you can use to change all visual and functional properties for the form and the control in the form



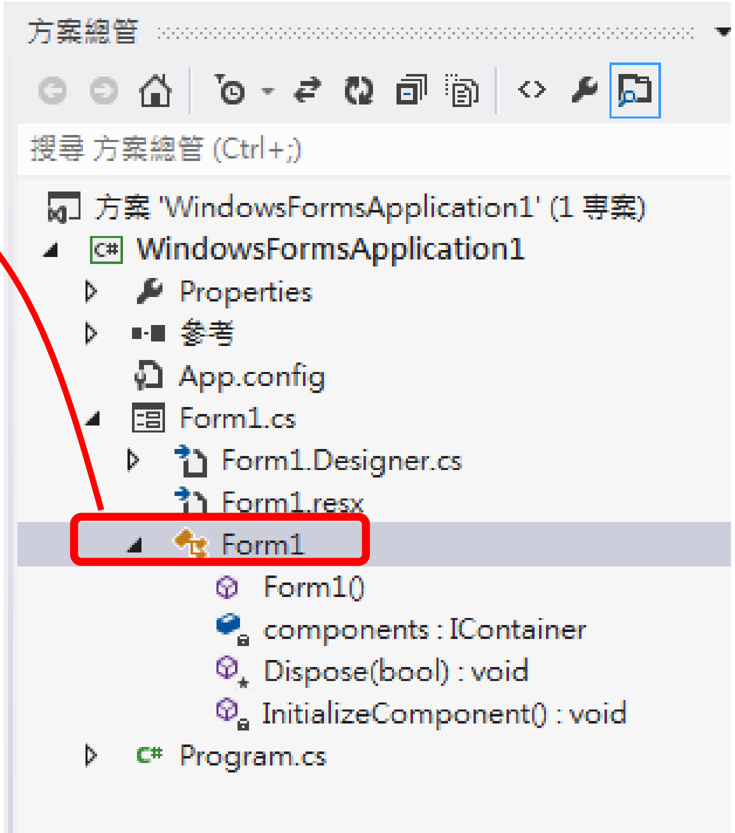
屬性

Form1 System.Windows.Forms.Form

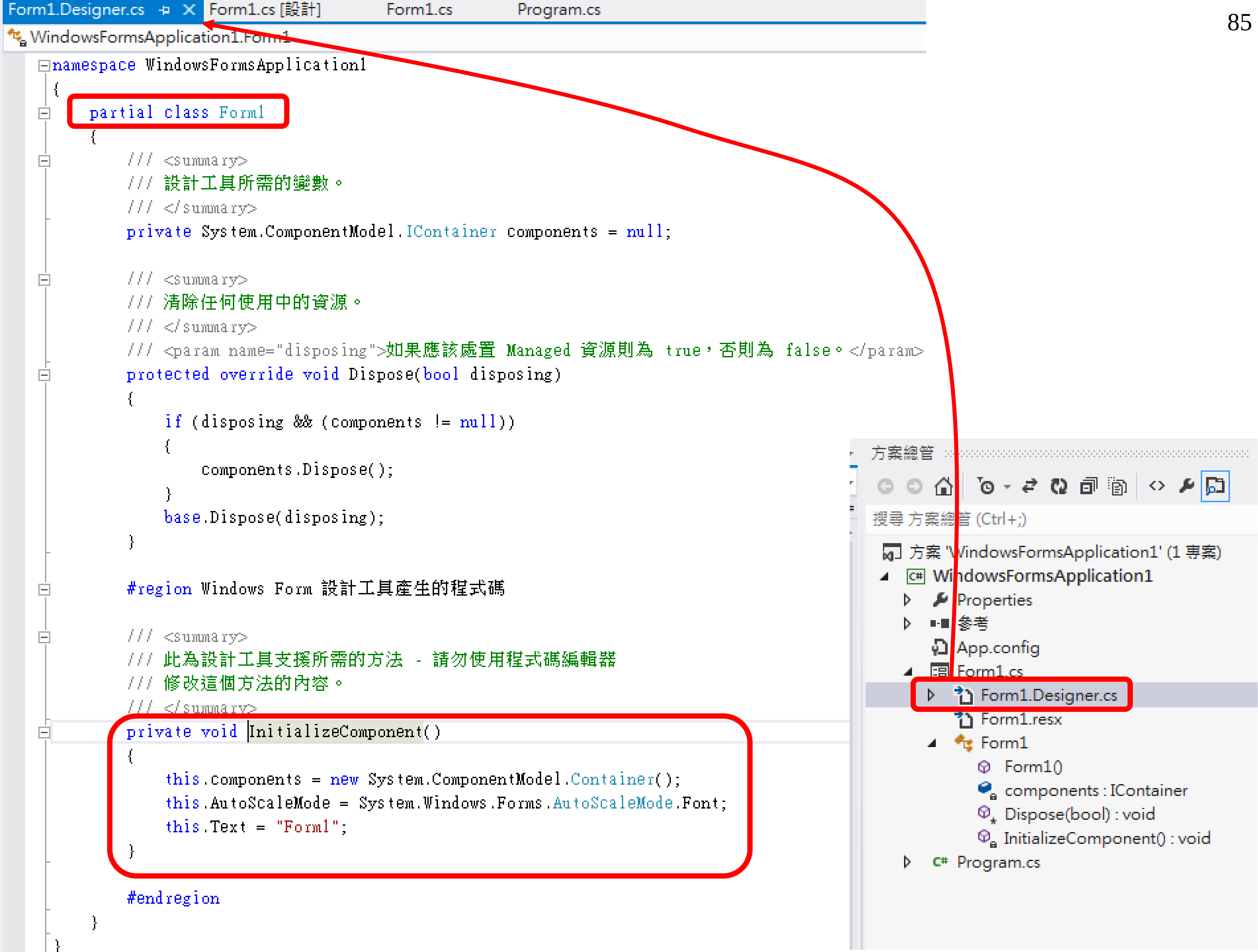
⊟ 外觀	
BackColor	□ Control
BackgroundImage	□ (無)
BackgroundImageLayout	Tile
Cursor	Default
⊞ Font	新細明體, 9pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	Form1
UseWaitCursor	False
⊟ 行為	
AllowDrop	False
AutoValidate	EnablePreventFocusChange
ContextMenuStrip	(無)
DoubleBuffered	False



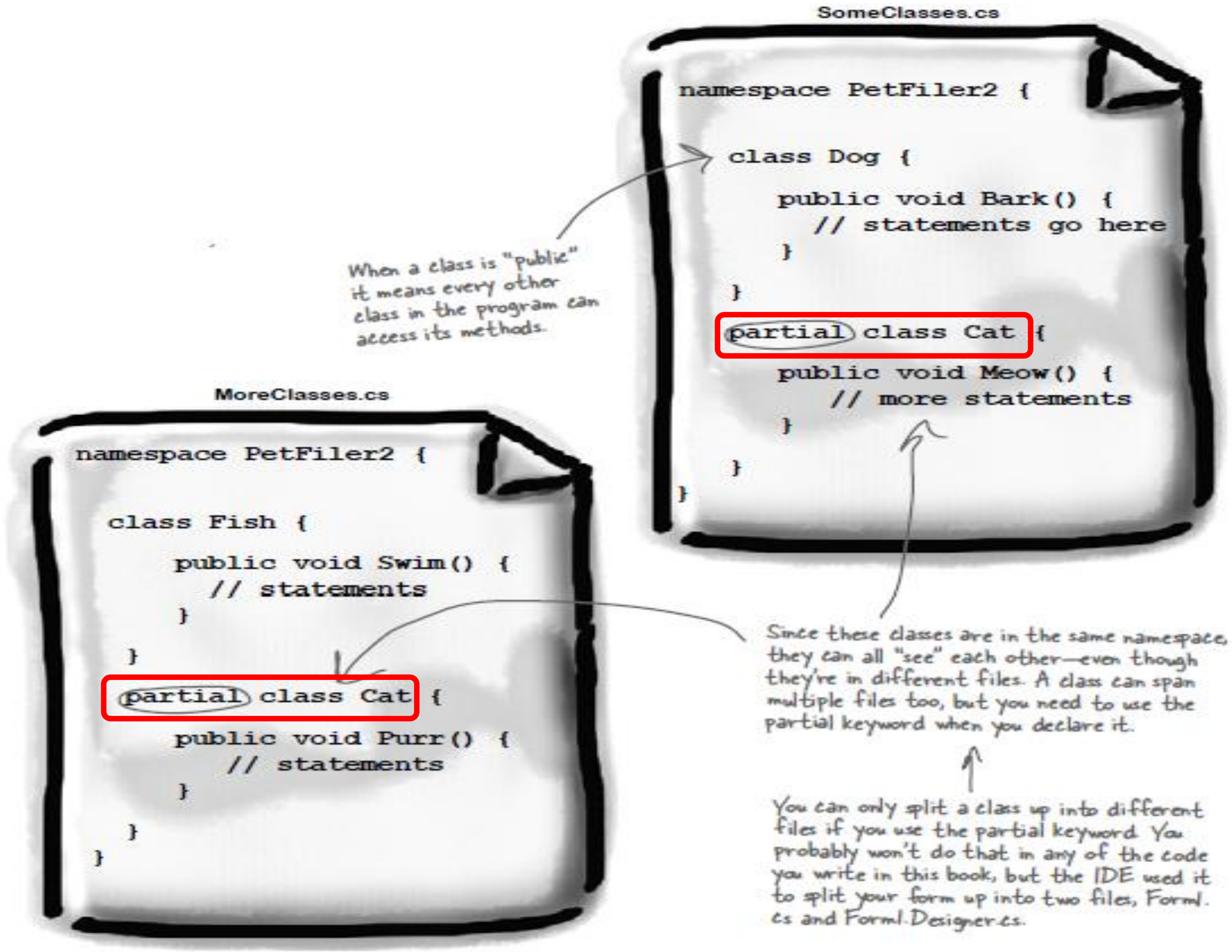
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```



Partial class means a class can be defined in two files (but with the same class name)



```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary>
        /// 設計工具所需的變數。
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 清除任何使用中的資源。
        /// </summary>
        /// <param name="disposing">如果應該處置 Managed 資源則為 true，否則為 false。</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form 設計工具產生的程式碼

        /// <summary>
        /// 此為設計工具支援所需的方法 - 請勿使用程式碼編輯器
        /// 修改這個方法的內容。
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }

        #endregion
    }
}
```



SomeClasses.cs

```
namespace PetFiler2 {

  class Dog {

     public void Bark() {
       // statements go here
     }

  }

  partial class Cat {

     public void Meow() {
       // more statements
     }

  }
}
```

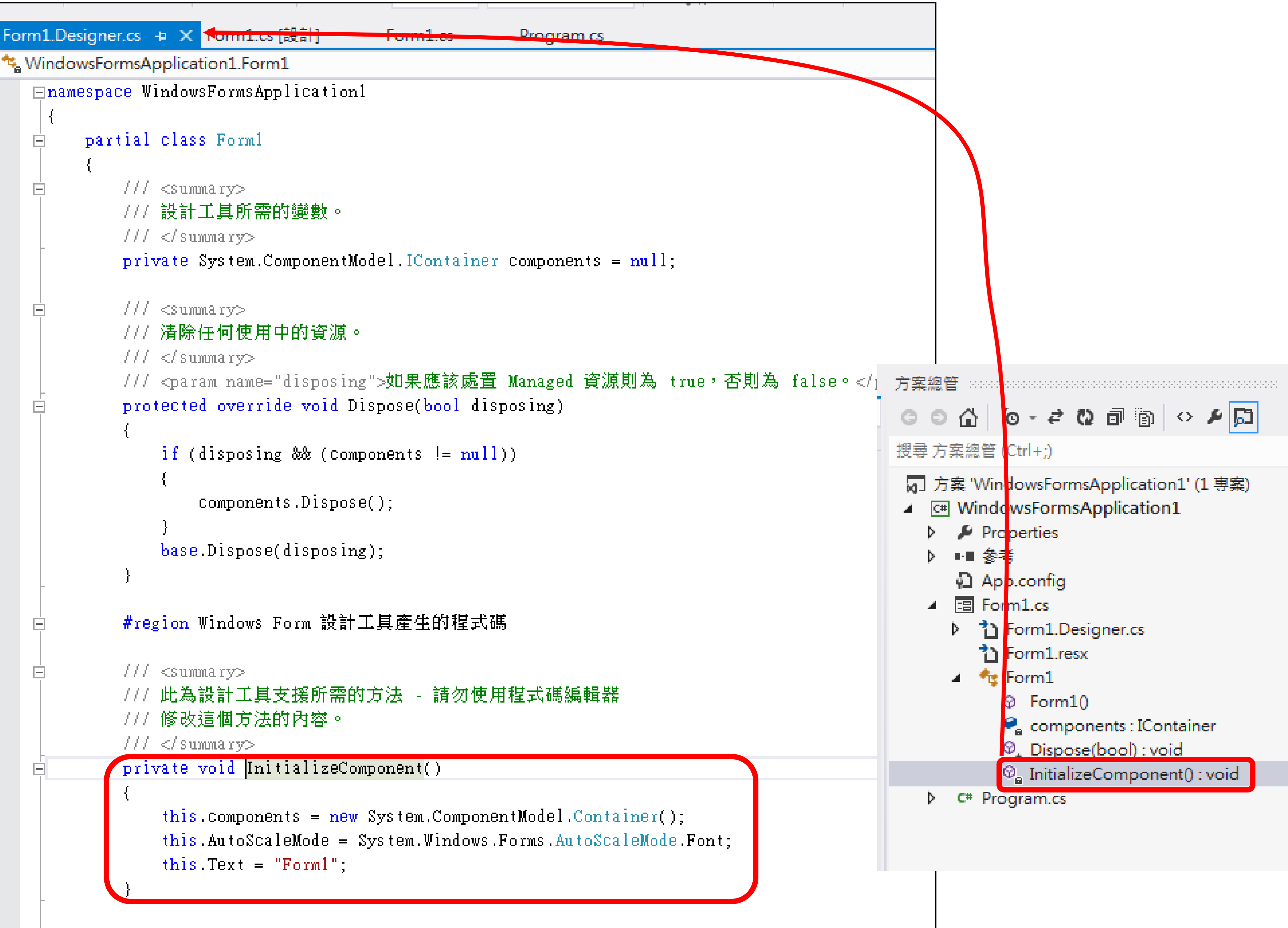
When a class is "public" it means every other class in the program can access its methods.

MoreClasses.cs

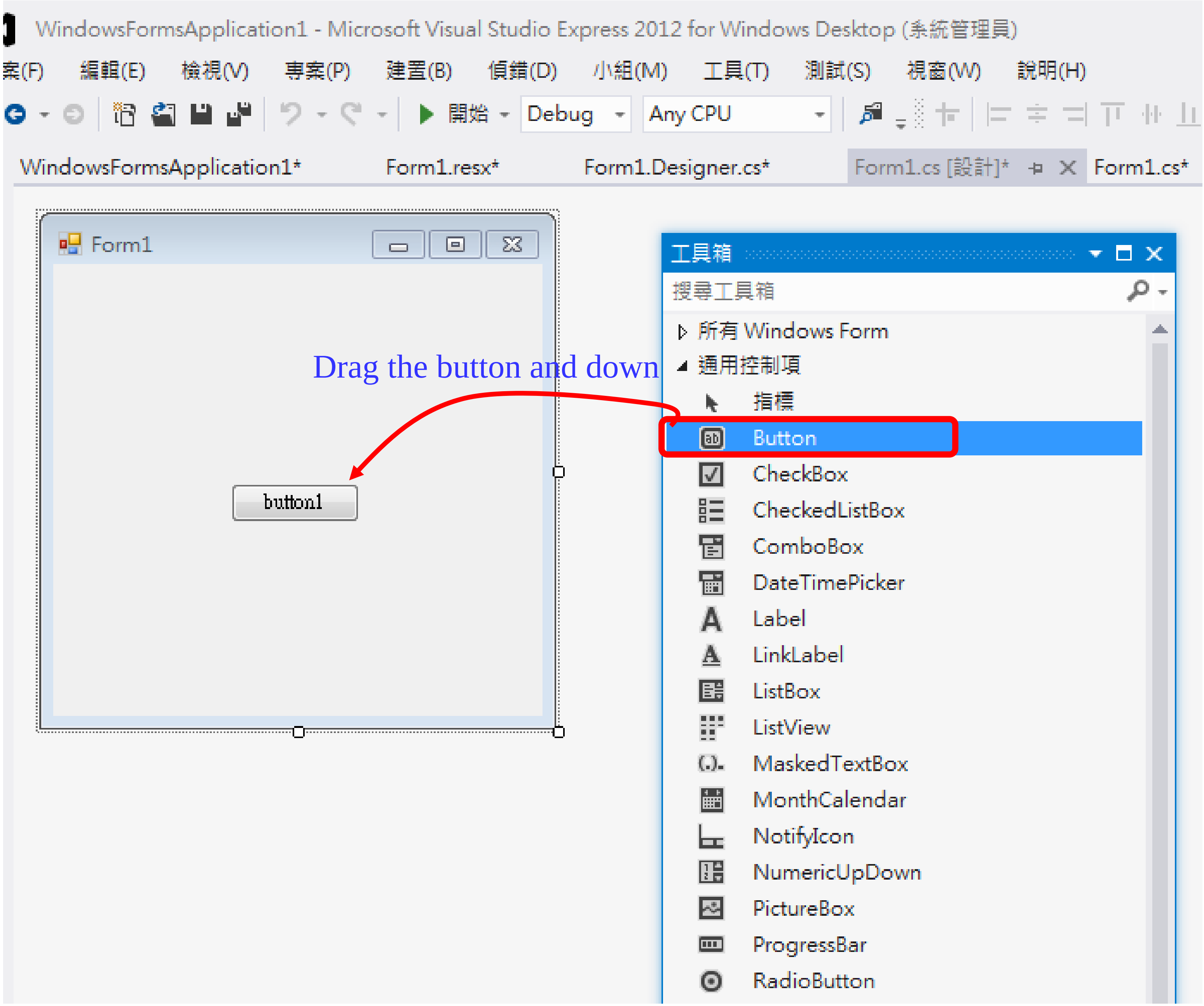
```
namespace PetFiler2 {

  class Fish {

     public void Swim() {
       // statements
     }

  }

  partial class Cat {

     public void Purr() {
       // statements
     }

  }
}
```

Since these classes are in the same namespace, they can all "see" each other—even though they're in different files. A class can span multiple files too, but you need to use the partial keyword when you declare it.

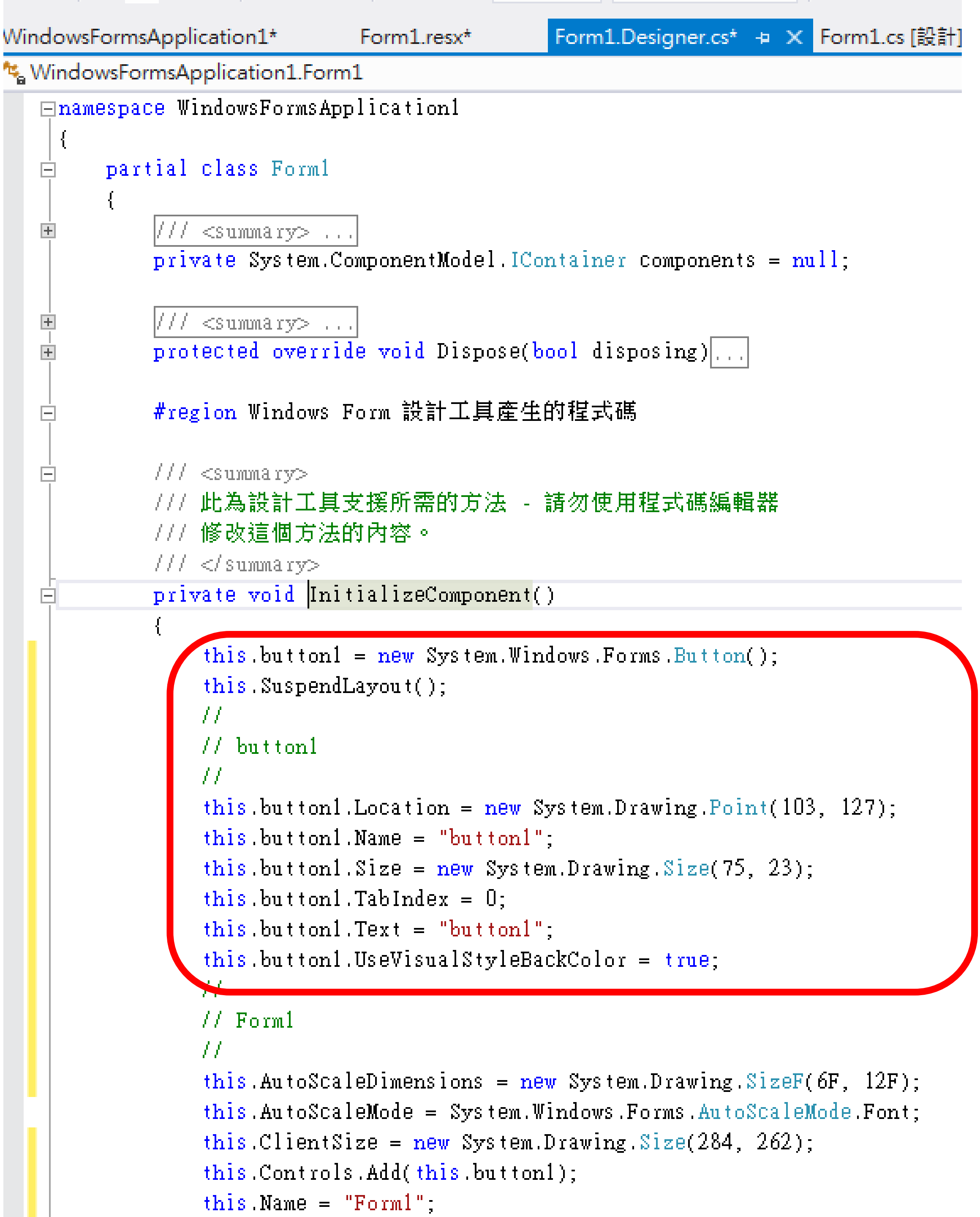
You can only split a class up into different files if you use the partial keyword. You probably won't do that in any of the code you write in this book, but the IDE used it to split your form up into two files, Form1.cs and Form1.Designer.cs.



```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary>
        /// 設計工具所需的變數。
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// 清除任何使用中的資源。
        /// </summary>
        /// <param name="disposing">如果應該處置 Managed 資源則為 true，否則為 false。</
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form 設計工具產生的程式碼

        /// <summary>
        /// 此為設計工具支援所需的方法 - 請勿使用程式碼編輯器
        /// 修改這個方法的內容。
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }
```

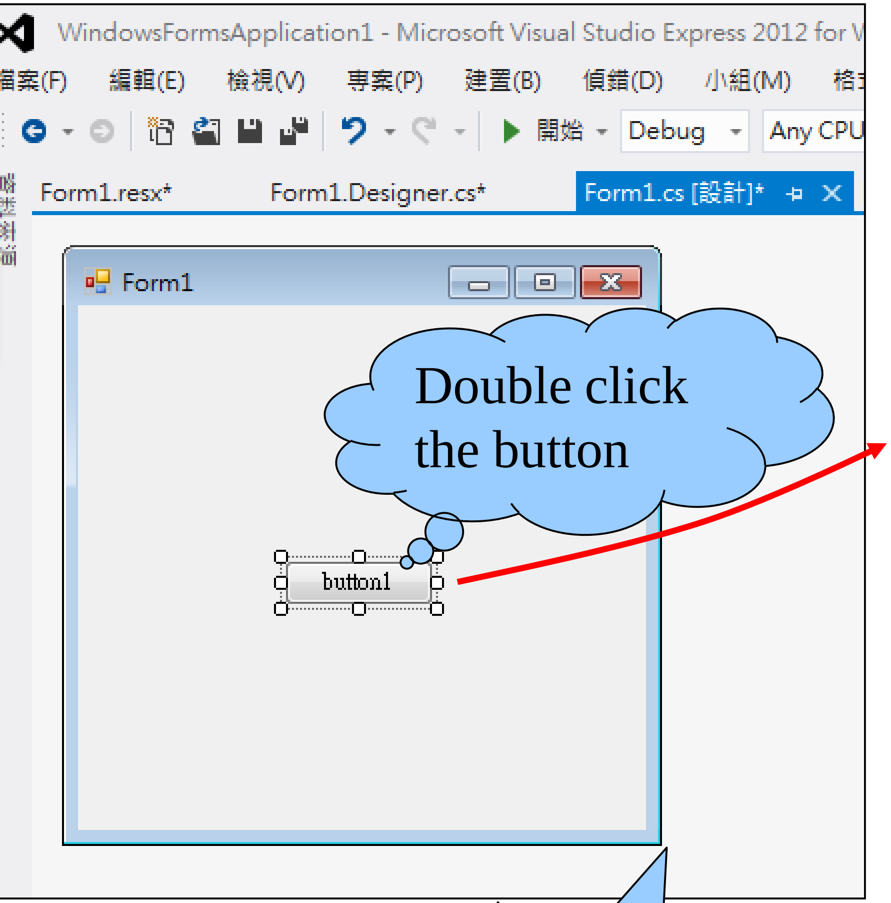


Drag the button and down

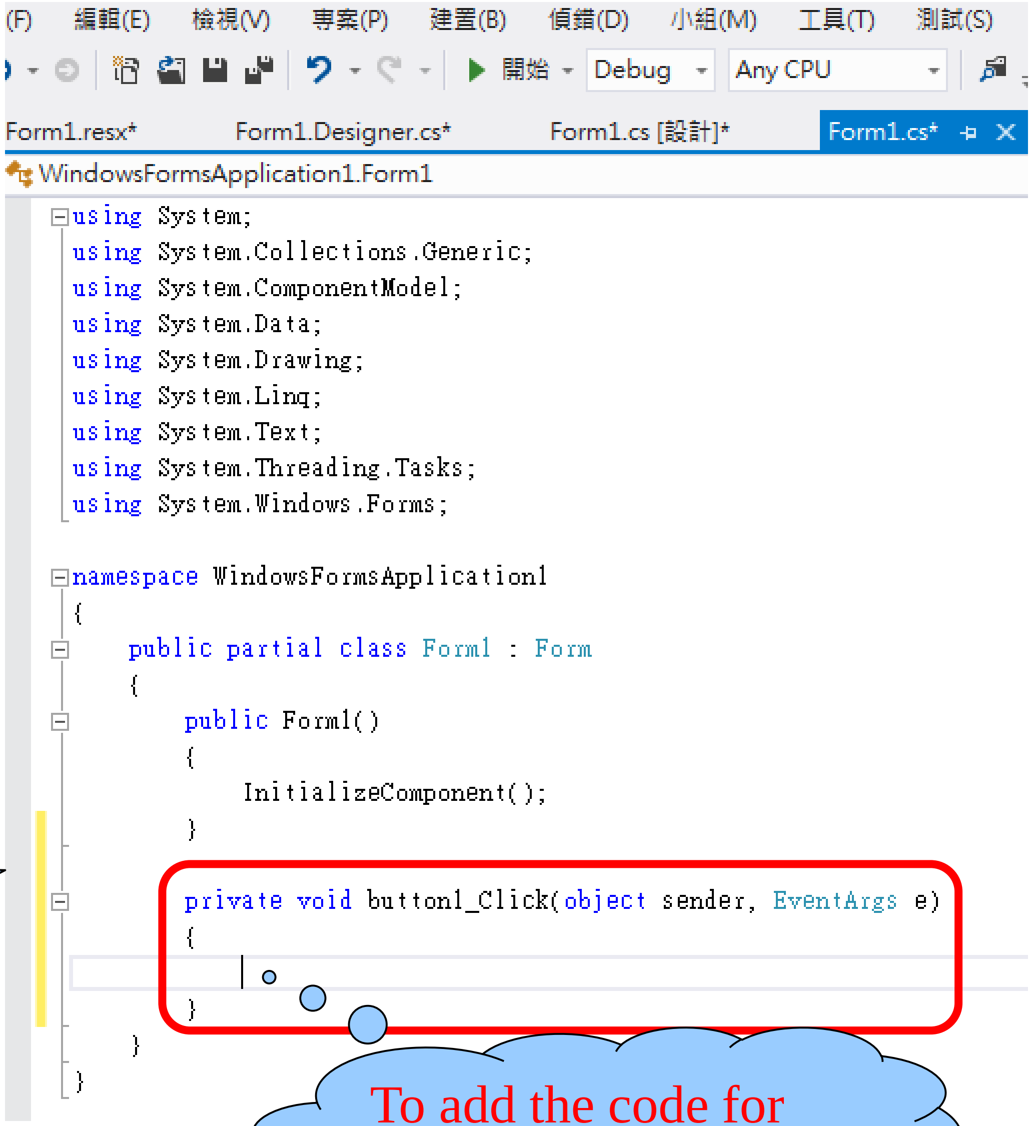


```
namespace WindowsFormsApplication1
{
    partial class Form1
    {
        /// <summary> ...
        private System.ComponentModel.IContainer components = null;

        /// <summary> ...
        protected override void Dispose(bool disposing)...

        #region Windows Form 設計工具產生的程式碼

        /// <summary>
        /// 此為設計工具支援所需的方法 - 請勿使用程式碼編輯器
        /// 修改這個方法的內容。
        /// </summary>
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(103, 127);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(75, 23);
            this.button1.TabIndex = 0;
            this.button1.Text = "button1";
            this.button1.UseVisualStyleBackColor = true;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 12F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(284, 262);
            this.Controls.Add(this.button1);
            this.Name = "Form1";
```

C# IDE automatically reflect the adding component in Form1.Desginer.cs


WindowsFormsApplication1 - Microsoft Visual Studio Express 2012 for W
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 格式
開始 Debug Any CPU
Form1.resx* Form1.Designer.cs* Form1.cs [設計]*
Form1
button1

Double click the button


(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 小組(M) 工具(T) 測試(S)
開始 Debug Any CPU
Form1.resx* Form1.Designer.cs* Form1.cs [設計]* Form1.cs*
WindowsFormsApplication1.Form1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            
        }
    }
}
```

Codes for button click add to Form1. cs

To add the code for click is your job