C#
for Programmers

Hello World

```
using System;

class Hello {
   static void Main() {
     Console.WriteLine("Hello world");
   }
}
```

import System;

System.out.println

More Choices

```
public static void Main() {
   ...
}

public static int Main() {
   ...
   return 0;
}

public static int Main(string[] args) {
   ...
   return 0;
}
```

Command Line Compiler

- Create source file: `Hello.cs.`
- Invoke compiler: `csc Hello.cs`
- If OK, `Hello.exe` is created.
- Run executable: `Hello`

C# Program Structure

- Namespaces
  - Types, namespaces
- Type declarations
  - Classes, interfaces, structs, enums, delegates
- Members
  - Fields, methods, constants, indexers, properties events, operators, constructors, destructors

Preprocessor

```
#define Dutch

using System;

public class Preprocessor {
  public static void Main() {
    #if Dutch
      Console.WriteLine(“Hallo Wereld");
    #else
      Console.WriteLine(“Hello World");
    #endif
  }
}
```

Namespace = Packets Made Easy

```
namespace N1 {              // N1
  class C1 {                // N1.C1
    class C2 {}             // N1.C1.C2
  }
  namespace N2 {            // N1.N2
    class C2 {}             // N1.N2.C2
  }
}
```

Creating DLLs

```
using System;

namespace MyMethods {
  public class AddClass {
    public static long Add(long i, long j) {
      return(i+j);
    }
  }
}
```

```
csc/target:library
/out:MyLibrary.DLL
Add.cs Mult.cs
```

```
using System;

namespace MyMethods {
  public class MultiplyClass {
    public static long Multiply(long i, long j) {
      return(i*j);
    }
  }
}
```

Using DLLs

```
using System;
using MyLibrary;
class MyClient {
  public static void Main() {
    long sum = AddClass.Add(10, 10);
    Console.WriteLine(sum);
    long product =
      MultiplyClass.Multiply(10, 10);
    Console.WriteLine(product);
  }
}
```

```
csc /reference:MyLibrary.DLL
        MyClient.cs
```

Value Types

- primitives
  - int i;
- enums
  - enum State { Off, On }
- structs
  - struct Point { int x, y; }

Enums

```
enum Suit {
  Clubs = 0;
  Diamonds = 1;
  Hearts = 2;
  Spades = 3;
}


…
Suit s = Suit.Clubs;

Console.WriteLine (s);

…
```

Typesafe Enums in Java

- Define a class representing a single element of the enumerated type without public constructors; that is, provide public static final fields for each constant in the enumerated type.

  Remember that most programmers are lazy …. (and it is inefficient as well).

Typesafe Enums in Java

```
public class Suit {
  private final int s;
  private Suit(int s) {
    this.s = s;

  }

  public int toint() {
    return s;

  }
  public static final Suit CLUBS = new Suit(0);
  public static final Suit DIAMONDS = new Suit(1);
  public static final Suit HEARTS = new Suit(2);
  public static final Suit SPADES = new Suit(3);
}
```

Exceptions

```
try {
  throw new Exception(“Oops!”);
} catch (Exception e) {
    … Handle exception ……;
} finally {
    … clean up, even if no exception occurred…;
}
```

Does not show up in type of Methods (no “throws” declaration)

Interfaces and Classes

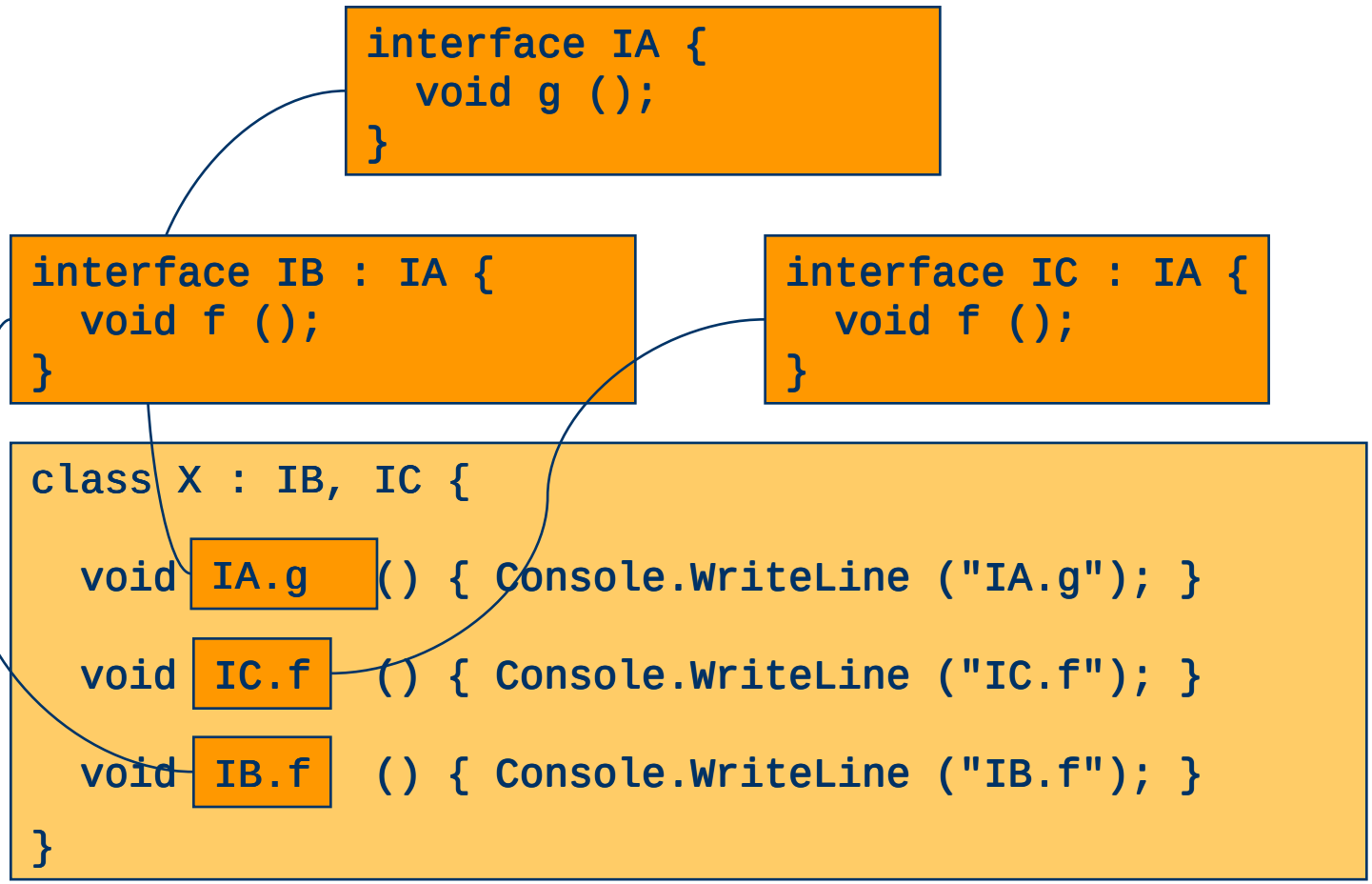
```
interface IFigure {
  int Area ();
}

class Square : IFigure {
  private int side;

  public Square (int side) {
    this.side = side;
  }

  public int Area () {
    return (side*side);
  }
}
```

Constructors

```
class B : A {
  public B (int x) : base (…,…,…) {
    ……
  }

  public B (bool b) : this (…,…,…) {
    ……
  }

  public B (char c) {
    ……
  }
}
```

The optional constructor-initializer Is invoked before executing the constuctor Body (default is `base ( )`).

Interfaces


interface IA {
void g ();
}
interface IB : IA {
void f ();
}
interface IC : IA {
void f ();
}
class X : IB, IC {
void IA.g () { Console.WriteLine ("IA.g"); }
void IC.f () { Console.WriteLine ("IC.f"); }
void IB.f () { Console.WriteLine ("IB.f"); }
}

Querying Interfaces

```
class Test {
  public static void Main () {
    X x = new X ();
    ((IA)x).g();
    ((IC)x).f();
    ((IB)x).f();
  }
}
```

as/is

```
X x = new X ();


if (x is IB) {
   IB b = (IB)x;
   ……
} else {
}
```

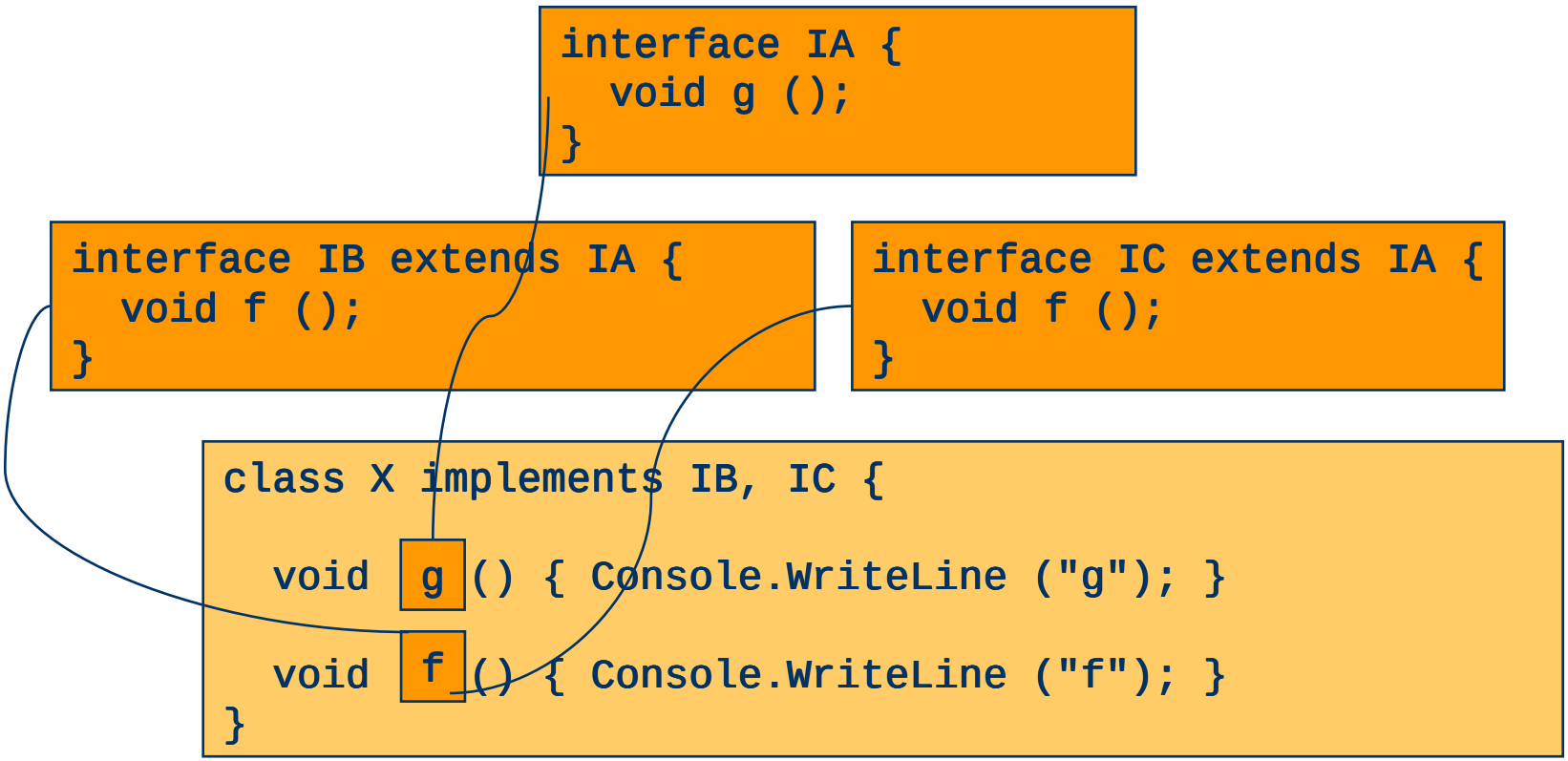
as/is

```
X x = new X ();

IB b = x as IB;
if (b != null) {
   ……
} else {
}
```

as/is

```
X x = new X ();


try {
   IB b = (IB)x;
   ……….
} catch (InvalidCastException e) {
}
```

Interfaces in Java



Virtual Methods

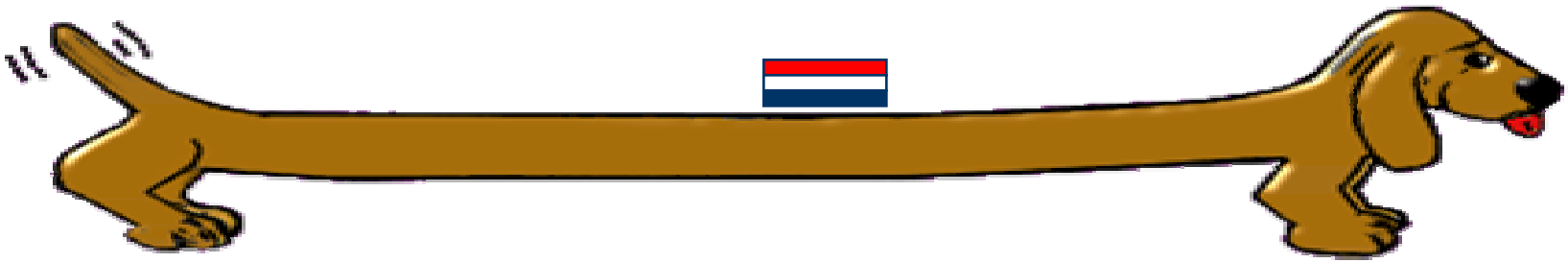
```
public class Dog {
  public virtual void RollOver () {
    Console.WriteLine("Scratch my tummy.");
    Bark();
  }

  public virtual void Bark () {
    Console.WriteLine("WOOF WOOF (Dog)");
  }
}
```

Default in Java

Breeding Dogs

```
using System;

namespace VirtualDog {
  public class Dog {
    public virtual void RollOver () {
      Console.WriteLine("Scratch my tummy.");
      Bark();
    }

    public virtual void Bark () {
      Console.WriteLine("WOOF WOOF (Dog)");
    }
  }
}
```

Breeding Dogs



```
Imports System
Namespace VirtualDog
  Public Class Mopje : Inherits Dog
    Public overrides Sub Bark ()
      Console.WriteLine("WOEF WOEF (Mopje)")
    End Sub
  End Class
End Namespace
```

Breeding Dogs

```
import VirtualDog;

var d = new Dog();
var m = new Mopje();

d.RollOver();
m.RollOver();
```

Properties and Indexers

```
class Party {
  private int start;
  public int Start {
    get {
      return start;
    }
    set {
      start = value;
    }
  }
}
```

You can perform arbitrary computation in get and set blocks

Properties and Indexers

```
class Demo {
   public static Main () {
      Borrel b = new Party ();
      b.Start = 3;
      int x = b.Start;
   }
}
```

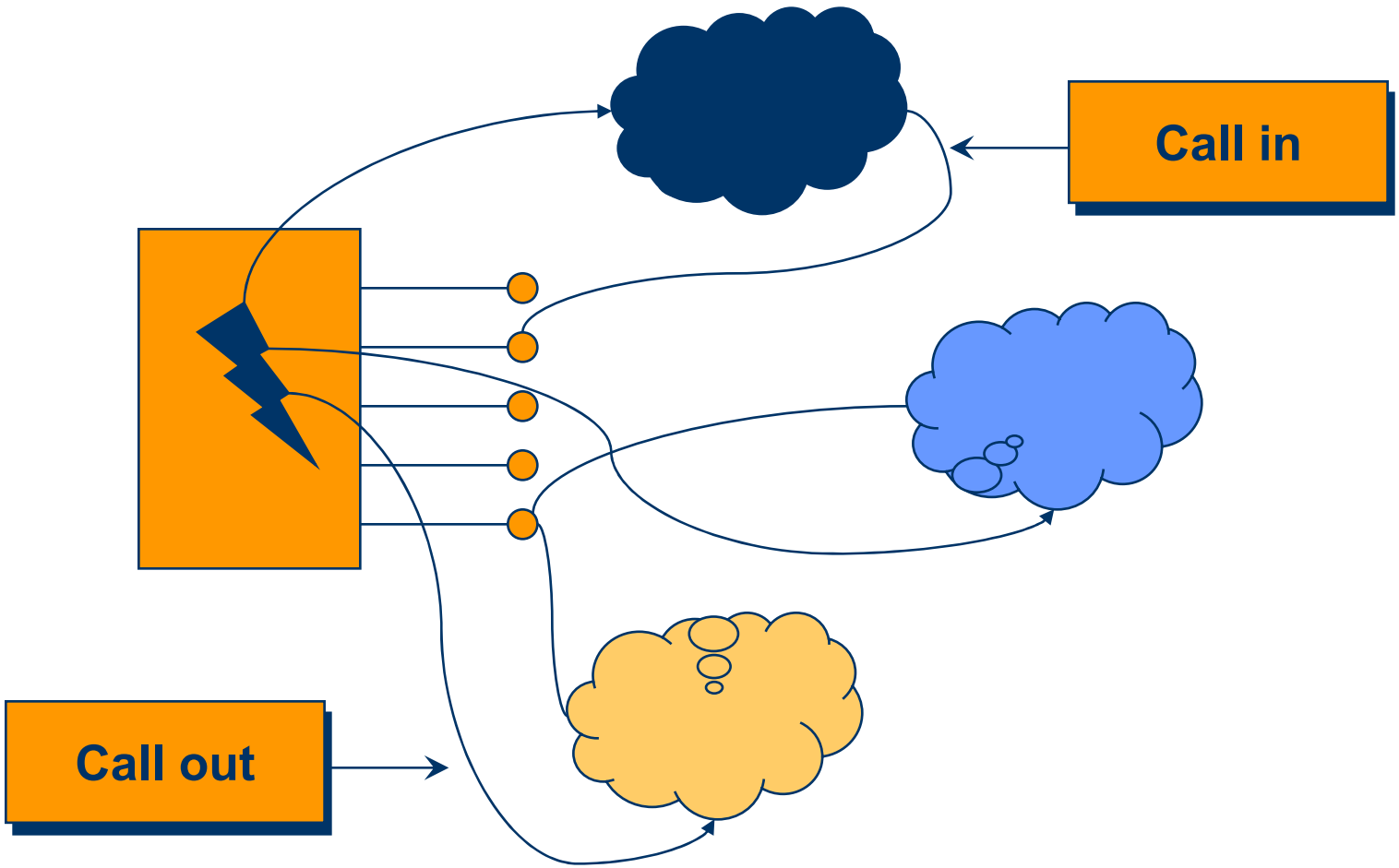
Properties and *Indexers*

```
class Party {
  private Dictionary participants;
  public Borrel() {
    participants = new Dictionary();
  }
  public bool this[String name] {
    get {
      return (participants.Contains(name)
              && (bool)participants[name]);
    }
    set {
      participants.Add(name,value);
    }
  }
}
```

Properties and *Indexers*

```
class Demo {
  public static void Main () {
    Borrel b = new Party ();
    b[“Erik”] = true;
    Console.WriteLine(b[“Bill”]);
  }
}
```

Events



Delegates ↔ Inner Classes

- Declaration

```
delegate void D();
```

- Instantiation & Invocation

```
class Delegates {
  static void F(){System.Console.WriteLine(“F");}
  void G(){System.console.WriteLine(“G”);}

  static void Main() {
    D f = new D(F); D g = new D(G);
    f(); g();
  }
}
```

Delegates in Java

- Declare an interface to represent the delegate type and an (anonymous) class that implements this interface to represent each concrete delegate value:

```
delegate void D();
…
static void F(){
  …
}
…
D f = new D(F);
```

```
interface D {
  void F();
}
…
D f = new D () {
  void F () {
    ….
  }
}
```

Events = Notifications for which Clients Can Attach Event Handlers

```
public delegate void TroubleHandler(Dog sender);

class Dog {

  public event TroubleHandler OnTrouble;

  TriggerTrouble () {
    if (OnTrouble != null) { OnTrouble (this); }
  }
}

Dog d = new Dog();
d.OnTrouble += new TroubleHandler (…);
```

Operator Overloading

```
public static result-type operator
   binary-operator
   ( op-type operand
   , op-type2 operand2
   ) {
  …
}
   public static result-type operator
      unary-operator
      ( op-type operand) {
      …
   }
```

Coercion Overloading

```
public static implicit operator
  conv-type-out
    ( conv-type-in operand ) {
    …
}

public static explicit operator
  conv-type-out
    ( conv-type-in operand ) {
    …
}
```

Attributes

```
namespace List {
  using System.Xml.Serialization;

  [ XmlRoot
    ( "List“
    , Namespace="http://www.meijcrosoft.com"
    , IsNullable=false
    )
  ]
  public class List {
    [ XmlElement("head", IsNullable=false)]
    public string Head;

    [XmlElement("tail", IsNullable=false)]
    public List Tail;
  }
}
```

Attributes

```
csc
  /target:library
  List.cs

Xsd List.dll
```

Attributes

```
<?xml version="1.0" encoding="utf-8"?>
<schema attributeFormDefault="qualified"
elementFormDefault="qualified"
targetNamespace="www.meijcrosoft.com"
xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="List" xmlns:q1="www.meijcrosoft.com"
     type="q1:List" />
  <complexType name="List">
    <sequence>
      <element minOccurs="1" maxOccurs="1" name="head"
         type="string" />
      <element minOccurs="1" maxOccurs="1" name="tail"
            xmlns:q2="www.meijcrosoft.com" type="q2:List" />
    </sequence>
  </complexType>
</schema>
```

Your Own

```
[ AttributeUsage
    ( AttributeTargets.Class
    , AllowMultiple = false
    )
]
public class MyAttribute : Attribute {
  public string msg;
  public MyAttribute (string msg) {
    this.msg = msg; }
 }
```

Where can it occur?

Base class for attributes

Example

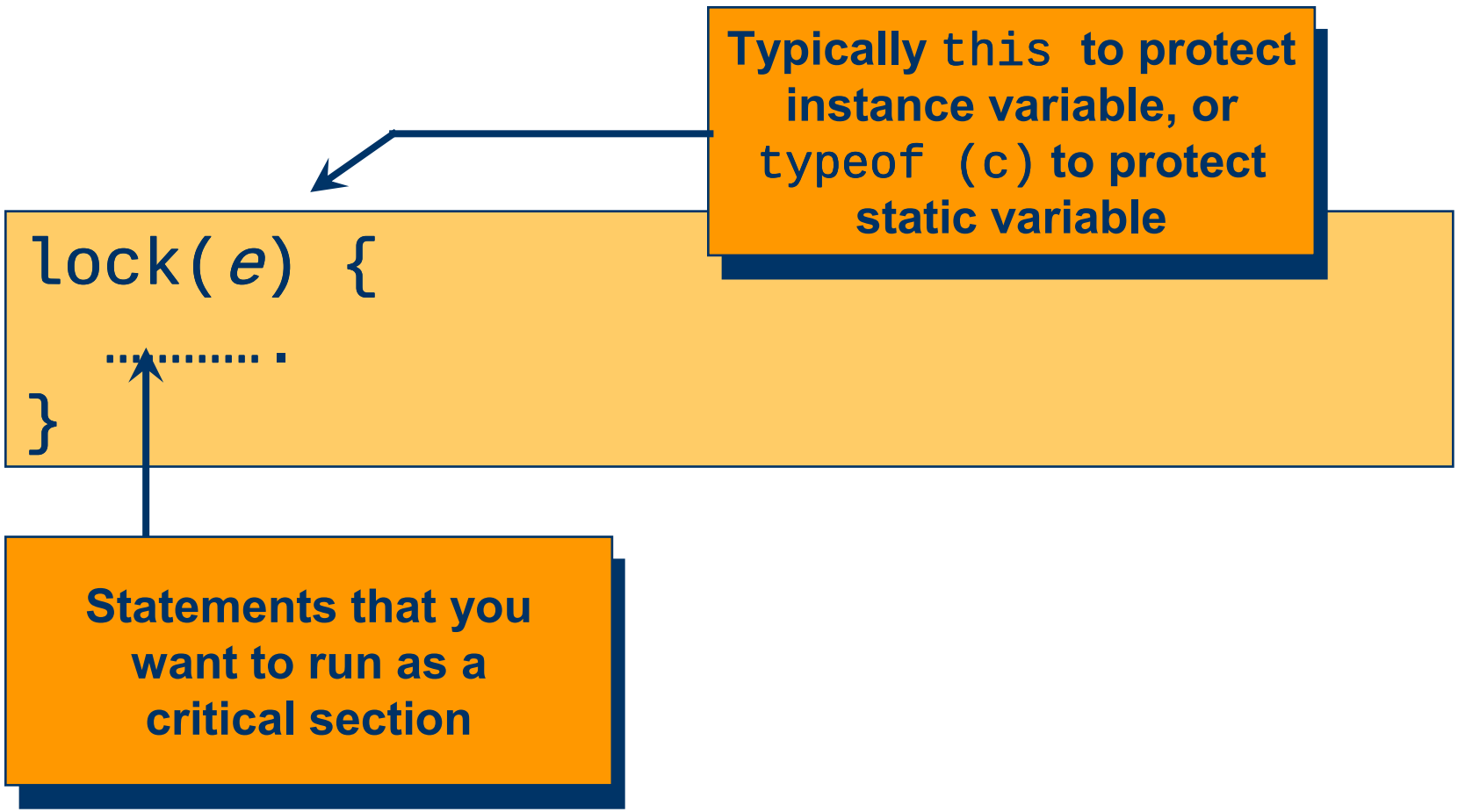
```
[MyAttribute("Hello World")]
class Example {
  public static void Main () {
    object[] Attrs =
     Attribute.GetCustomAttributes
      (typeOf (Example));
   MyAttribute a = (MyAttribute)Attrs[0];
   Console.WriteLine (a.Message);
  }
}
```

Create New Thread

```
using System;
using System.Threading;
class Test {

  static void printA () {
    while (true) { Console.Write("A");}
  }

  static void printB () {
    while (true) { Console.Write("B");}
  }

  public static void Main () {
    Thread a = new Thread(new ThreadStart(printA));
    Thread b = new Thread(new ThreadStart(printB));
    a.Start(); b.Start();
  }
}
```

```
AABBBBAAAABBBBBBAAAAAABBBABA
BBBAABABBBABBAAAABABABABABBB
ABBBABBBABBBBBBBABBABBBBBAAA
AAAABBBABBABBBABBBBABABABBBA
BABABBABABBBAABAAABABBBABBBB
```

Locks and Critical Sections



COM ⇒ .NET

- Type Library Importer (TlbImp.exe)
  - Converts a COM type library into equivalent .NET DLL

```
tlbimp ComComponent.tlb NetComponent.dll
```

.NET ⇒ COM

- Type Library Exporter (TlbExp.exe)
  - Converts a .NET assembly to a COM type library

```
TlbExp
  NetComponent.dll ComComponent.tlb
```

WinForms

```
using System;
using System.Window.Forms;
using System.Drawing;

public class MyForm : Form {
  public MyForm() {
    this.Text = “Hello World”; }
}

public class Demo {
  public static void  Main() {
    Application.Run(new MyForm());
  }
}
```

Controls

- Control component on a form that displays information or accepts user input

No notion of layout manager

```
Button b = new Button ();
b.Location = new Point (256,64);
b.Text = “Click Me”;
this.Controls.Add(b);
```

Example

```
public class HelloWorldForm : Form {
  private Button b = new Button() ;

  private void OnClick
                  (object sender, EventArgs evArgs) {
    b.Text = ”Ouch!”;
  }


  public HelloWorldForm() {
    b.Location = new Point(20, 10);
    b.Text = "Click Me!";
    b.Click += new EventHandler(OnClick);
    this.Controls.Add(b);
  }
}
```