German Jordanian University الجامعة الألمانية الأردنية

CS355 Web Technologies

Dr. Ismail Hababeh

German-Jordanian University

Lecture 22

.NET Namespaces

- Just like a package in Java technology and the class library is just like the Java API structure.
- Consists of many classes and sub-namespaces.
- Deployed as a component class library itself and is organized in a component–

.NET Namespaces

- It can be deployed as an assembly of binary components.
- In order to use classes in a namespace, a directive using <namespace> in C# or import <namespace> in VB must be included at the beginning of code.
- The system built-in basic class library is deployed in mscorlib.dll file.

DLL and EXE Files

- A .NET component can be:
  - Local component (.dll), which can only be accessed locally (within same application domain), in same machine.
  - Remote (distributed) component (.exe), which can be accessed remotely (across application domains) in the same or different machines.

Local DLL Component Files

- A .NET DLL component can be deployed as:
  - Private component which knows the target client
  - Shared public component which does not know the target client.
- A DLL component can be plugged-in to Windows form, Web form of another DLL or application.

Creating .NET Web Services – C# Program

The following C# program is developed for creating a web service to convert temperatures between Celsius and Fahrenheit.

```
using System;
namespace TempConv
{ public class TempConvComp
{ public double cToF (double c)
    {  return (int) ((c*9)/5.0+32);
    }
  public double fToC (double f)
    { return (int) ((f-32)*5/9.0);
    }
}
}
```

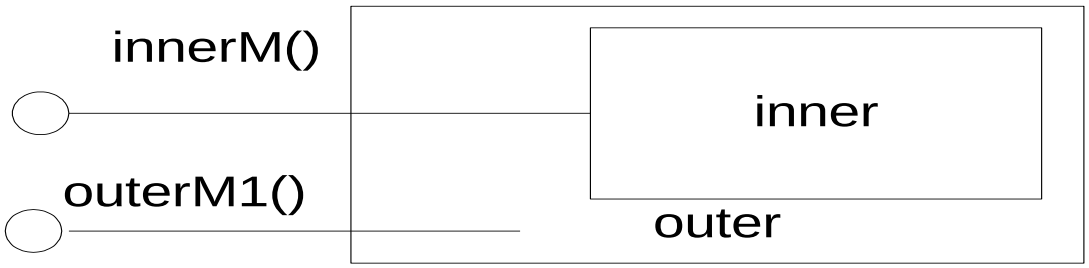
//using directive imports types defined in other namespaces. System is the namespace used to organize too many classes so that it can be easy to handle the application.

The Connection Model of .NET

- .NET compositions enable the reuse of components in either:
  - Aggregation compositions
  - Containment compositions
  - Mix of aggregations and containments in a flat structure or nested compositions in multiple levels in depth.

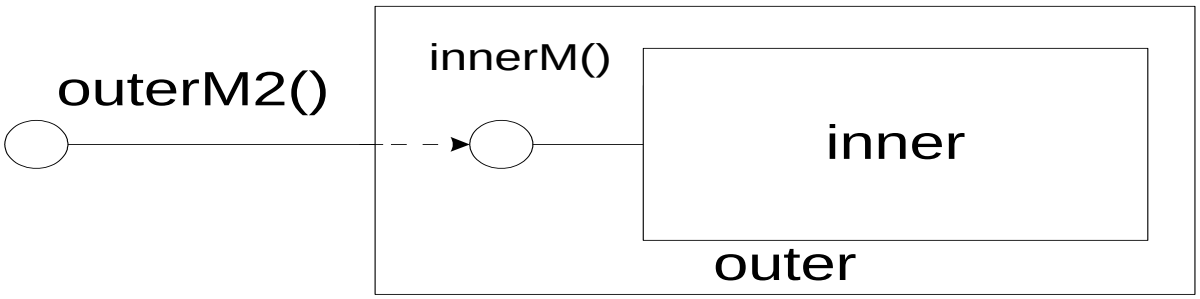
.NET Aggregation Compositions

- In aggregation composition model:
  - The service of inner component hands out its service directly to the client of outer component.
  - The outer component describe the interfaces of inner component.
  - The innerM() method of inner component becomes part of interface to the outer component



.NET Containment Compositions

- In containment composition model:
  - The request of the outer component is forwarded to the inner component.
  - The outer component does not expose the interface of the inner component.
  - The containment is transparent to the client of an outer component.
  - The client is blind of the handler of the request. The outerM2() hands over a request to the innerM() method of inner component

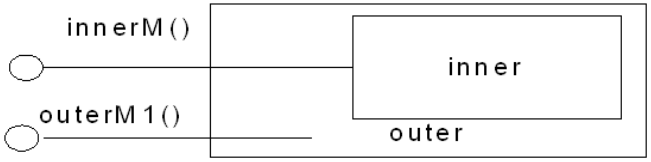


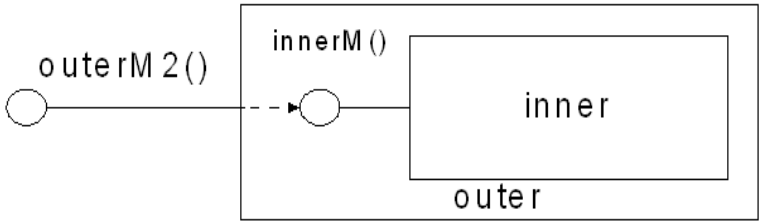
Aggregation and Containment Compositions – Example

```
using System;
namespace NS1
{
    public class Inner
    {
        public void innerM ()
        {
            Console.WriteLine ("I am Inner.")
        }
    }
}
using System;
using NS1;
```

Aggregation and Containment Compositions – Example (cont.)

```
public class Outer
{
    public Inner i = new Inner ();   //aggregation: Outer expose the Inner
    public void outerM1 ()
    {
            Console.WriteLine ("I am outer.");
    }
    public void outerM2()          //delegation in containment
    {
         i.innerM();
        }
    public static void main()
    {
            outer o1 = new Outer();
            Inner i1 =  o1.i;
            i1.innerM();  //interface to  the aggregate
            o1.outerM1();
            o1.outerM2(); // interface to the containment
            Inner i2 = new Inner();
            i2.innerM();
    }
}
```





.NET Components Communication

- Communication between .NET components is accomplished by Delegate and Event.
- The .NET Delegate:
  - A safe and secure method type (a reference to a method) which is similar to function pointer in C++.
  - Passes on a control flow to its registered event handler when the event is raised.
  - It works in the pattern of observer which is kind of event listener in Java.

.NET Delegate

- A Delegate instance can hold:
  - Static method of a class
  - A method of a component
  - A method of object itself
- There are two types of Delegates:
  - SingleCast: delegates one method at a time. References any method with int return type and without any parameter.
  - MultiCast: can bind multiple methods, has void return type, and plays a role of listener.

SingleCast Delegate - Example

```
Delegate int Mydelegate();
public class MyClass
{ public int ObjMethod() { …. }
  public static int  StaticMethod () {  … }
public class Drive { public static void Main()
{ Myclass c = new MyClass();
MyDelegate dlg = new  MyDelegate(c.objMethod());
dlg();
dlg = new MyDelegate (MyClass.StaticMethod());
dlg();
 } }
```

Notes on the SingleCast Delegate Example

- MyDelegate is a Delegate which references any method with int return type and without any parameter.
- The signatures of objMethod and StaticMethod match the Delegate MyDelegate.
- The first dlg() invokes objMethod and second dlg() invokes the class method StaticMethod.

MultiCast Delegate

- A MultiCast Delegate has:
  - Void return type

    Delegate void MultiDelegate();

  - Can bind multiple methods

    It will invoke them in the order of registrations:

    MultiDelegate mdlg = null;
    mdlg += new MultDelegate (Method1);
    mdlg += new MultiDelegate (Method2);

    - Registration is done by += operation and un-registration is done by -= operation.
    - The Delegate plays a role of listener.
    - The event handler must register this listener (Delegate) to be able to handle the event once the event is fired.

.NET Event

- An Event is a message sent by an object to raise an action.
  - The object that raises the event is the source event.
  - The object captures the event and handles it is the target event.
  - There is an event-driven communication model between components or within the same component.
- The Delegate class is the communication channel between source event and target event.
- Event can be:
  - Predefined event such as an event trigger by a Windows Form component.
  - Custom event defined by a developer.

Event Communication - Example

1. Create a delegate.

   Public delegate void DelegateStart();

2. Create a delegate class containing a delegate event field

   public event DelegateStart EventStart { ….. }

3. Define an event handler to register the event listener to be able to handle the event

   public void handleEvent(){ …… }

Event Communication - Example

4. Bind delegate event with event handler via event listener
5. Invoke the event handler
6. Trigger an event

```
Public static void Main()
    { MyClass EventObj = new MyClass();
     EventObj.EventStart += new DelegateStart(handleEvent);
     EventObj.EventStart();
    .....
    }
```