Lecture # 4

Namespaces

Engr. Fiaz Khan

اَلْحَمْدُ لِلّٰهِ رَبِّ الْعٰلَمِيْنَ وَ الصَّلٰوةُ وَالسَّلَامُ عَلٰى سَيِّدِ الْمُرْسَلِيْنَ

اَمَّا بَعْدُ فَاَعُوْذُ بِاللّٰهِ مِنَ الشَّيْطٰنِ الرَّجِيْمِ ؕ بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ ؕ

اَللّٰهُمَّ افْتَحْ عَلَيْنَا حِكْمَتَكَ وَانْشُرْ

عَلَيْنَا رَحْمَتَكَ يَا ذَا الْجَلَالِ وَالْاِكْرَام

Translation

O Allah عَزَّوَجَلَّ! Open the door of knowledge and wisdom for us, and have mercy on us! O the One Who is the Most Honourable and Glorious! *(Al-Mustatraf, vol. 1, pp. 40)*

Lets Start Learning

Whoever recites Salat upon me one time, Allah عَزَّوَجَلَّ sends ten blessings upon him. *(Sahih Muslim, pp. 216, Hadees 408)*

جس نے مجھ پر ایک بار دُرُودِ پاک پڑھا اللہ عَزَّوَجَلَّ اُس پر دس رَحمتیں بھیجتا ہے۔

(مُسلِم ص۲۱٦ حدیث٤٠٨)

C# - Namespaces

- A **namespace** is designed for providing a way to keep one set of names separate from another.
- The class names declared in one namespace does not conflict with the same class names declared in another.

Defining a Namespace

- A namespace definition begins with the keyword namespace followed by the namespace name as follows −

```
namespace namespace_name {
   // code declarations
}
```

To call the namespace-enabled version of either function or variable, prepend the namespace name as follows −

```
namespace_name.item_name;
```

Example

```
using System;
using first_space;
using second_space;

namespace first_space {
    class abc {
        public void func() {
            Console.WriteLine("Inside first_space");
        }
    }
}
```

Example - Conti

```
namespace second_space {
    class efg {
        public void func() {
            Console.WriteLine("Inside second_space");
        }
    }
}
```

Example - Conti

```
class TestClass {
    static void Main(string[] args) {
        abc fc = new abc();
        efg sc = new efg();
        fc.func();
        sc.func();
        Console.ReadKey();
    }
}
```

Example - Output- Conti

```
Inside first_space
Inside second_space
```

Nested Namespaces

- You can define one namespace inside another namespace as follows −

```
namespace namespace_name1 {

   // code declarations
   namespace namespace_name2 {
      // code declarations
   }
}
```

Note: You can access members of nested namespace by using the dot (.) operator.

Nested Namespaces - Example


```
using System;
using first_space;
using first_space.second_space;

namespace first_space {
    class abc {
        public void func() {
            Console.WriteLine("Inside first_space");
        }
    }
```

Nested Namespaces – Example (Conti)


```
    namespace second_space {
        class efg {
            public void func() {
                Console.WriteLine("Inside second_space");
            }
        }
    }
}
```

Nested Namespaces – Example (Conti)


```
class TestClass {
    static void Main(string[] args) {
        abc fc = new abc();
        efg sc = new efg();
        fc.func();
        sc.func();
        Console.ReadKey();
    }
}
```

Nested Namespaces – Example (Output)


```
Inside first_space
Inside second_space
```

What is Boxing and Unboxing?

- The conversion of value type to reference type is known as boxing and converting reference type back to the value type is known as unboxing.

What is Boxing and Unboxing?

- C# has two kinds of data types, value types and reference types.
- Value type stores the value itself, whereas the reference type stores the address of the value where it is stored.
- Some predefined data types such as int, float, double, decimal, bool, char, etc. are value types and object, string, and array are reference types.

What is Boxing and Unboxing?

- While working with these data types, you often need to convert value types to reference types or vice-versa.
- Both have different characteristics and .NET stores them differently in the memory, it must do some work internally to convert them from one type to another.
- These conversion processes are called boxing and unboxing.

What is Boxing and Unboxing?

- When the common language runtime (CLR) boxes a value type, it wraps the value inside a System.Object instance and stores it on the managed heap.
- Unboxing extracts the value type from the object.
- Boxing is implicit;
- unboxing is explicit.

What is Boxing?

- Boxing is the process of converting a value type to the object type or any interface type implemented by this value type.
- Boxing is implicit.

Example: Boxing

```
int i = 10;
object o = i; //performs boxing
```

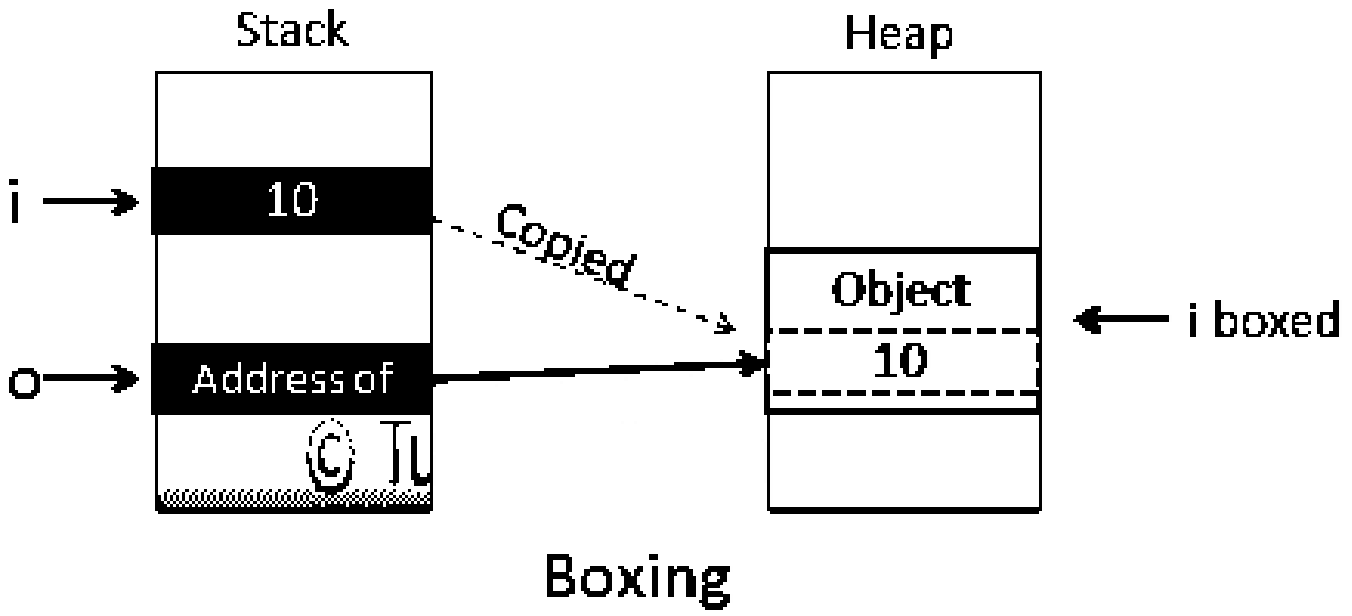
Example – Boxing Explanation

- In the above example, the integer variable i is assigned to object o. Since object type is a reference type and base class of all the classes in C#, an int can be assigned to an object type. This process of converting int to object is called boxing.

Why named boxing?

- As you know, all the reference types stored on heap where it contains the address of the value and value type is just an actual value stored on the stack.
- Now, as shown in the first example, int i is assigned to object o. Object o must be an address and not a value itself.
- So, the CLR boxes the value type by creating a new System.Object on the heap and wraps the value of i in it and then assigns an address of that object to o.
- So, because the CLR creates a box on the heap that stores the value, the whole process is called 'Boxing'.

Why named boxing? - Conti



What is Unboxing?

- Unboxing is the reverse of boxing. It is the process of converting a reference type to value type.
- Unboxing extract the value from the reference type and assign it to a value type.

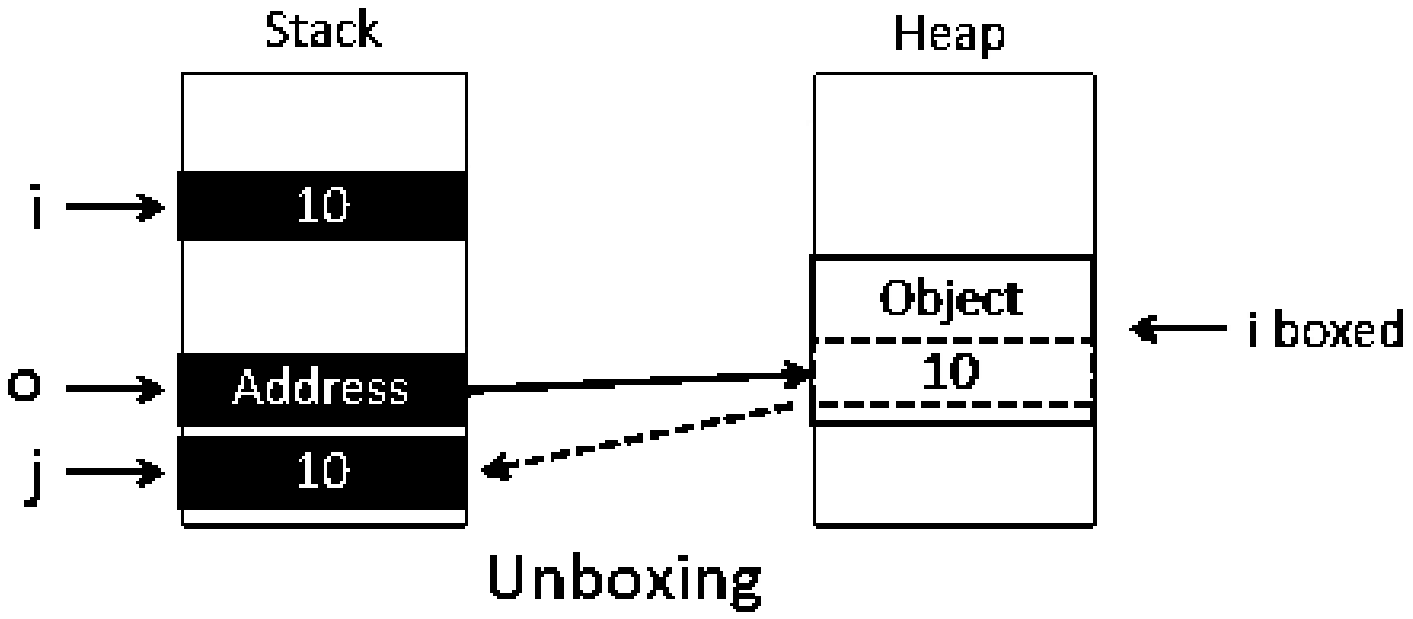
- Unboxing is explicit. It means we have to cast explicitly.

Example

Example: Unboxing

```
object o = 10;
int i = (int)o; //performs unboxing
```

Graphical Representation - Unboxing



Unboxing

- A boxing conversion makes a copy of the value. So, changing the value of one variable will not impact others.

```
int i = 10;
object o = i; // boxing
o = 20;
Console.WriteLine(i); // output: 10
```

The casting of a boxed value is not permitted. The following will throw an exception.

Unboxing

The casting of a boxed value is not permitted. The following will throw an exception.

Example: Invalid Conversion

```
int i = 10;
object o = i; // boxing
double d = (double)o; // runtime exception
```

Unboxing

First do unboxing and then do casting, as shown below.

Example: Valid Conversion

```
int i = 10;
object o = i; // boxing
double d = (double)(int)o; // valid
```

Punch Line of Today's Lecture

- Boxing and unboxing degrade the performance. So, avoid using it.
- Use generics to avoid boxing and unboxing. For example, use List instead of ArrayList

ENGR. FIAZ KHAN

LECTURER COMPUTER SCIENCE

TELF/EF SET / ACEPT CERTIFIED

MICROSOFT ACADEMY TRAINER

ORACLE CERTIFIED PROFESSIONAL

GOOGLE CERTIFIED PROFESSIONAL

Lahore, Pakistan

+92-336-678-2755

muhammad.fiaz@pucit.edu.pk

https://www.linkedin/in/fiazofficials

THANK YOU!

Do you have any questions?