
Interfaces, Enumerations, Boxing, and Unboxing

9/8/05

CS360 Windows Programming

1

Last Week

- We
 - Started looking at the different types in C#
 - Introduced the concept of the stack and heap
 - Looked at the components of classes
 - Covered the difference between reference and value types

9/8/05

CS360 Windows Programming

2

Interfaces

- Classes that have no fields and the methods have no implementations
- The implementations appear in the classes that inherit from the interface
- When a class inherits from an interface it **must** implement all *methods* and *properties*

9/8/05

CS360 Windows Programming

3

Interface Example

```
interface INode
{
    string Text
    {
        get;
        set;
    }
    object Tag
    {
        get;
        set;
    }
    int Height
    {
        get;
        set;
    }
    int Width
    {
        get;
        set;
    }
    float CalculateArea();
}
```

9/8/05

CS360 Windows Programming

4

Interfaces Example

- Write a class (Node) that inherits from INode
- Class Node must contain the following *fields*
 - `m_text`
 - `m_height`
 - `m_width`
- Write the code needed to instantiate your Node class and test all properties and methods

9/8/05

CS360 Windows Programming

5

Enumerations

- Enumerations define literals that are then used as constants for their corresponding values

```
public enum DAYS
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
    Sunday
}
```

9/8/05

CS360 Windows Programming

6

Enumerations Cont.

- By default, the values in an enumeration are
 - 0 for the first item
 - 1 larger for every subsequent item

9/8/05

CS360 Windows Programming

7

Enumerations Cont.

```
class EnumTest
{
    public enum DAYS
    { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday }

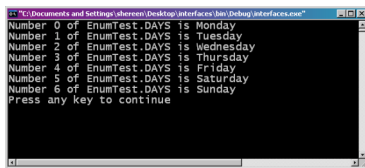
    static void Main(string[] args)
    {
        Array dayArray = Enum.GetValues(typeof(Class1.DAYS));
        foreach (DAYS day in dayArray)
            Console.WriteLine("Number {1} of EnumTest.DAYS is {0}",
                              day, day.ToString("d"));
    }
}
```

9/8/05

CS360 Windows Programming

8

Enumerations Cont.



9/8/05

CS360 Windows Programming

9

Enumerations Cont.

```
public enum DAYS
{
    Monday = 1,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday,
    Sunday
}
```

9/8/05

CS360 Windows Programming

10

Delegates

- We will skip delegates for today

9/8/05

CS360 Windows Programming

11

Boxing and Unboxing

- Why do we have reference and value types?
- Why not just stick with reference types?
- Where are each of these created in memory?
- What happens when we pass a value type to a method that expects a reference type?

9/8/05

CS360 Windows Programming

12

Boxing and Unboxing

- Boxing
 - Creates a copy of the value type on the managed heap
- Unboxing
 - Duplicates a reference type on the stack

9/8/05

CS360 Windows Programming

13

C# Default Values

- The following are automatically initialized to their default values:
 - Static variables
 - Instance variables of class instances
 - Array elements
- For value types, the default value is 0
- For reference types, the default value is null
- Note that local variables are not considered to be initially assigned

9/8/05

CS360 Windows Programming

14

C# Built-in Classes

- String
- ArrayList
- Stack
- Queue
- Hashtable
- These are found in the namespace
 - System.Collections

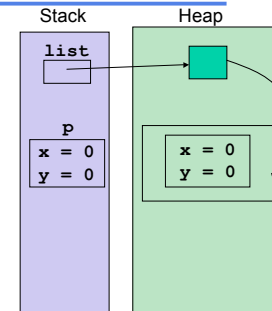
9/8/05

CS360 Windows Programming

15

Example

```
struct Point
{
    public int x, y;
}
. . .
ArrayList list =
    new ArrayList();
Point p;
for (int i = 0; i < 2;
    i++)
{
    p.x = p.y = i;
    list.Add(p);
}
```



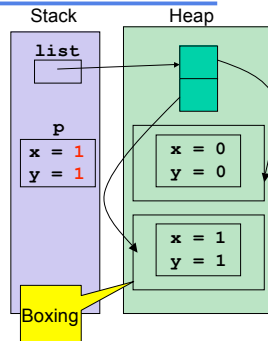
9/8/05

CS360 Windows Programming

16

Example

```
struct Point
{
    public int x, y;
}
. . .
ArrayList list =
    new ArrayList();
Point p;
for (int i = 0; i < 2;
    i++)
{
    p.x = p.y = i;
    list.Add(p);
}
```



9/8/05

CS360 Windows Programming

17

Boxing and Unboxing

- C# hides the boxing from us
- We don't have to change the syntax of our programs
- However, when unboxing, an explicit type cast is needed

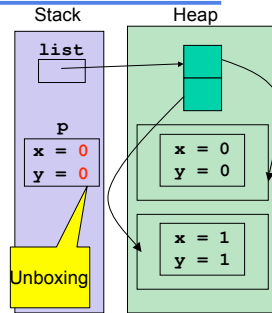
9/8/05

CS360 Windows Programming

18

Example

```
struct Point
{
    public int x, y;
}
. . .
ArrayList list =
    new ArrayList();
Point p;
for (int i = 0; i < 2;
    i++)
{
    p.x = p.y = i;
    list.Add(p);
}
p = (Point) list[0];
```



9/8/05

CS360 Windows Programming

19

Summary

- Completed p. 27 - 38
- Next time we will complete chapter 2
- We will talk about exception handling

9/8/05

CS360 Windows Programming

20