Nesne Yönelimli Programlama

1.Hafta

Konu: C#, IDE Kullanımı,Form Yaratma,Toolbox İçeriği

Dr. Öğr. Üyesi Güncel SARIMAN
E-posta: guncelsariman@mu.edu.tr

C# Nedir?

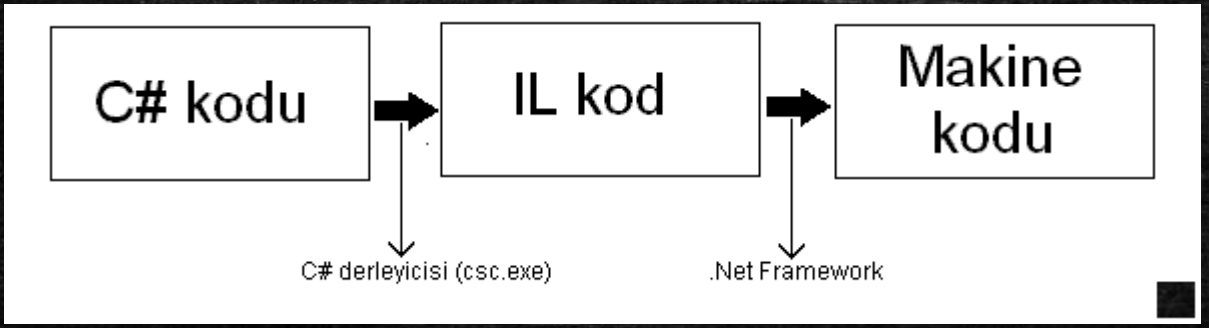
- Microsoft tarafından geliştirilmiş olan bir programlama dilidir. C++ ve Java dillerine oldukça benzer, ancak C#'ın bu dillere benzerliği yanında farkları da vardır.
- Örneğin C#, C++'dan farklı olarak % 100 nesne yönelim tekniğine sahiptir. Java'dan farklı olarak ise C#'ta gösterici (pointer) kullanılabilir. Böylelikle eski yazılım bileşenleriyle uyumlu bir şekilde çalışılabilir.

.NET Framework

- C# kodları, C++ veya Visual Basic'ten farklı olarak direkt makine koduna derlenmez. Önce IL dediğimiz bir ara koda derlenir. Bu derlenen ilk kodun dosyasına assembly denir ve uzantısı exe'dir. Bu dosya çalıştırılmak istendiğinde ise .Net Framework devreye girer ve IL kodu makine koduna dönüştürür.
- Böylelikle artık kodu bilgisayar anlayabilir. İşte bu yüzden de yazdığımız programın bir bilgisayarda çalışması için o bilgisayarda .Net Framework programının kurulu olması gerekir, çünkü .Net Framework IL kodu bilgisayarın anlayabileceği koda çevirir.

.NET Framework

- .Net Framework, oluşturduğu makine kodlarını geçici bir süreliğine belleğe koyar, eğer aynı kodlar tekrar çalıştırılmak istenirse tekrar IL koddan makine koduna dönüşüm yapmak yerine bu belleğe kaydettiği makine kodlarını kullanır.
- Bu yüzden oluşturduğumuz programımızı ilk çalıştırdığımız zaman programımız biraz yavaş çalışabilir, ancak daha sonraki çalışmalarda oldukça hızlanacaktır.



Namespaces and .NET Class Library

- ✓ Programcıların işlerini kolaylaştırmak için bir takım hazır kütüphaneler vardır fakat C# dili ile gelen hazır bir takım kütüphaneleri yoktur.
- ✓ Bunun yerine Framework dediğimiz altyapıda bir takım temel türler ve sınıflar mevcuttur. Bu sınıf ve türleri organize edebilmek için Namespace kavramı kullanılır.

Namespaces and .NET Class Library

```
namespace Test1;
 class testProgram
{ }
namespace Test2;
class testProgram
{ }
```

Kullanımı: Test2.testProgram şeklinde

Namespaces and .NET Class Library

- ✓ System isim alanı : .NET çalışırken gerekli temel sınıfları içerir. Ayrıca diğer tüm sınıf kütüphaneleri de bunu içinde kümelenmiştir. System hiyerarşinin tepesinde bulunur.
- ✓ Örneğin tüm veritabanı işlemleri için kullanılacak sınıf kütüphanesi "System.Data" dır.
- ✓ Bu sınıf kütüphanesi içindeki SQL ile işlemler için "System.Data.SqlClient" isim alanı mevcuttur.

Namespaces and .NET Class Library

- ✓ System.Net : HTTP ve ağ protokolleri için kullanılır.
- ✓ System.Xml : XML verileri ile çalışmak için
- ✓ System.IO : dosyalara bilgi girişi, dosyadan bilgi okuma, I/O işlemleri için kullanılır.
- ✓ System.Windows.Forms: Windows tabanlı uygulamalarda kullanılan zengin grafik arabirimi kontrollerini içerir.

Çoklu Dosyalar

- ✓ C# ' ta çoklu dosya kullanımı ile birden fazla kaynak kod referans edilerek veya DLL dosyası halinde diğer kaynak kodlar tarafından kullanılabilir.
- ✓ Örneğin: Elimizde çeşitli sayı tiplerinin dönüşümünü yapan 'Program.cs' ve 'topla' isimli fonksiyon içeren 'Program1.cs' adlı kod dosyalarımız olsun.

Çoklu Dosyalar

```
using toplama;
namespace kutuphane
{
   class program
   {
      static void Main(string[] args)
      {
```

```
 float a,b;
         string c;
         Console.WriteLine("a sayısı gir");
         a     =     Convert.ToSingle(Console.ReadLine());
//float a çevrim
         Console.WriteLine("b sayısı gir");
         b  =float.Parse(Console.ReadLine());  //  float  a
çevrim

Console.WriteLine("toplam={o}",toplama.program.t
opla(a, b));
         Console.ReadKey();     }
   } //program.cs
```

Çoklu Dosyalar

```
namespace toplama
{
   class program
   {
      public static float topla(float a, float b)
      {
         return (a + b);
      }
   }
}


//program1.cs
```

Veri Türleri

Stack Bölgesi: Tanımlı değişkenlerin tutulduğu bellek alanıdır. Derleyici tarafından değişkenlere yapılacak yer tahsisatı önceden bilinmelidir.

Heap Bölgesi: Stack'ten farklı olarak heap bölgesinde tahsisatı yapılacak nesnenin derleyici tarafından bilinmesi zorunlu değildir. Bu programlarımıza esneklik getirir.

Heap'te bir nesne için yer tahsisatı new kelimesi ile yapılır.

Veri Türleri

Değer tipleri; veriyi taşıyan ve taşıdığı veriye göre bellek üzerinde yer dolduran değişken türleridir.

Bellekte az yer kaplarlar ve hızlı bir şekilde erişilebilirler. Ayrıca belleğin "stack" bölgesinde tutulurlar.

Referans türleri ise, bellek bölgesinde veri yerine adresi tutarlar ve o adresin gösterdiği yerde de veri tutulur.

Başka bir deyişle, bir ifade referans türleri içeriyorsa nesnenin adresi üzerinden işlem yapılmaktadır. Veri taşınmasını gerektiren işlemlerde nesnenin bütün verisi kopyalanmaz.

Veri Türleri

Değişken türleri: "int", "long", "float", "double", "decimal", "char", "bool", "byte", "short", "struct", "enum"

Referans türleri: "string", "object", "class", "interface", "array", "delegate", "pointer"

Veri Türleri

- ✓ new ile tahsis edilen alanlar dinamiktir. Çalışma zamanında tahsisat yapılır, derleme zamanında bir yer ayrılmaz.
- ✓ Stack'e göre daha yavaştır.
- ✓ Değer veri türleri Stack, Referans veri türleri Heap' te tutulurlar.

Değer Tipleri (Value Types)

- ✓ Değer tiplerinde bir nesnenin değeri direkt olarak saklıdır.
- ✓ int a=3,b;
- ✓ b=a;
- ✓ Bu noktada a üzerindeki değişikliklerden b etkilenmeyecektir.
- ✓ Değer tiplerinin tamamı Object denilen bir nesneden türemiştir.

Değer Tipleri (Value Types)

Değer tiplerine ilk değer verme;

```
a=new int(); //yapıcı çalışır.(referans tip)
a=0;
```

Yukarıdaki iki satırda aynı işlemi yapar.

```
float b; //derleyici hatası, atama yapılması gerekir.
Error2  Use of unassigned local variable b'
float b=new float(); //hata vermez yapıcı çalıştı
b=3.21f          //yeni atama yapılıyor
```

Referans Tipleri (Reference Types)

- ✓ Referans tipleri .Net Framework platformunda çok kullanılan tiplerdir. Büyük bir esnekliğe sahip olmakla beraber metodlarda kullanılırken çok iyi performans sağlar.

- ✓ Referans tipleri stack bölgesinde dataların bulunduğu adresleri depolarlar. Pointer olarakta adlandırılabilir. Adreslerin işaret ettiği asıl veriler ise belleğin heap adı verilen bölgesinde depolanmaktadır.
- ✓ Çalışma zamanı (Runtime) garbage collection diye bilinen bir mekanizma sayesinde bellek yönetimini kontrol etmektedir. Garbage collection ise çöp toplayıcı bir mekanizmadır. Yani belleği periyodik olarak kontrol eder ve bellekte artık kullanılmayan öğeleri yok eder.

Veri Tipleri

	Long Form	in Java	Range
sbyte	System.SByte	byte	-128 .. 127
byte	System.Byte	---	0 .. 255
short	System.Int16	short	-32768 .. 32767
ushort	System.UInt16	---	0 .. 65535
int	System.Int32	int	-2147483648 .. 2147483647
uint	System.UInt32	---	0 .. 4294967295
long	System.Int64	long	-2^{63} .. $2^{63}-1$
ulong	System.UInt64	---	0 .. $2^{64}-1$
float	System.Single	float	±1.5E-45 .. ±3.4E38 (32 Bit)
double	System.Double	double	±5E-324 .. ±1.7E308 (64 Bit)
decimal	System.Decimal	---	±1E-28 .. ±7.9E28 (128 Bit)
bool	System.Boolean	boolean	true, false
char	System.Char	char	Unicode character

Tür Dönüşümleri

Bilinçsiz Tür Dönüşümü: Derleyici bizim için yapıyor.

```
int s=10;
float a;
a=s;
Console.WriteLine(a);
```

Tür Dönüşümleri

Küçük türün büyük türe dönüştürülmesi:

Küçük tür büyük türe dönüştürülürken fazla bitler sıfır ile doldurulur.

Örn: byte a=12;

```
int b;
b=a;
```

Tür Dönüşümleri

Büyük türün küçük türe dönüştürülmesi: İstenmeyen durum. Ancak "()" operatörü ile yapılır.

Örn:

```
int b=256;

byte i=(byte) b; //sonuç o 'dır.
```

Referans-Değer Dönüşümleri

Referans ve Değer Türleri Arasındaki Dönüşüm

- ✓ C# dilinde değer tipindeki verileri referans tipine çevirmek önemli bir konudur. Değer veri tipleri Stack, Ref. veri tipleri Heap'te tutulur.

- ✓ C# dilinde herşey nesne(object) referans türünden türetilmiştir. Temelde bir sınıf vardır. Örneğin object sınıfının ToString() metodu bütün temel veri ve referans türlerinde kullanılır.

```
string str = 345.59f.ToString()
```

Referans-Değer Dönüşümleri

Örn:

```
int a=5;
int b=7;
string a1=a.ToString();
string b1=b.ToString();
Console.WriteLine(a+b);
Console.WriteLine(a1+b1);
```

Sonuç: 12

57

Referans-Değer Dönüşümleri

```
int a=0;  int d = (int) 6.0; //float -> integer dönüşüm

object k="merhaba"+15; //object türü, hem karakter hem sayısal

float b=10.5f; //float tanımı

double c=20.1; //double tanımı

Double dd = new double(); //referans olarak double tanımı

const double pi = 3.14; //sabit tanımı

string[] isimler ={ "Ozlem","Nesrin", "Ozge", "Fulya" }; //string dizi tanımı

object[] isim ={ "Ozlem","Nesrin", "Ozge", "Fulya" }; //object dizi tanımı

string s = "true"; //string tanımı

string dd="12.45f";

b= float.Parse(dd); //string tip float'a çevriliyor

b=Convert.ToSingle(dd); //String float'a çevriliyor
```

Referans-Değer Dönüşümleri

```
a =Convert.ToInt32(b + c); //float -> integer

bool cevap = (Convert.ToBoolean(s)); //boolean tanımı

Console.Write((float)a/d+"\n");  //  () operatörü ile float dönüşümü

Console.WriteLine("cevap=" + cevap); // cevap = true  yazar

Console.WriteLine(k.GetType()); //bulunduğu sınıf,alanadını verir.

a = Convert.ToSingle(Console.ReadLine()); //girilen değer float'a çevriliyor

Console.WriteLine("a={0} b= {1} c={2} d={3} ", a, b, c,d);

if (isimler[0].Equals("Ozlem")==true)  //eğer dizinin ilk elemanı Ozlem ise yazar

            Console.WriteLine("birinci isim Ozlem");

foreach (string ss in isimler)  // string dizi içindeki her bir eleman yazdırılıyor

        { Console.WriteLine(ss); }
```

System.Convert Sınıfı (Temel Veri Tiplerinin Dönüşümü)

Convert.ToBoolean(str)
Convert.ToByte(str)
Convert.ToInt32(str)
Convert.ToChar(str)
…

```
int a=50;
byte b=Convert.ToInt32(a);//Yanlış tür dönüşümü
 string c="12.34";
a=float.Parse(c);
```

Referans-Değer Dönüşüm(Boxing)

Günümüzdeki popüler dillerde referans ve değer tipleri arasında dönüşüm yapılmamaktadır.

Böyle bir çevrime ihtiyaç duyulduğunda "Boxing" kutulama yapılır. Bir değer tipini referans tipe atadığımızda stack'teki bilgi bit olarak heap'e kopyalanır ve stack'teki object türünden olan değişken heap'i gösterecek şekilde ayarlanır.

Örn:Bilinçsiz boxing işlemi.

```
int i=50; //değer tipi

object o=i; //boxing
```

Örn:Bilinçli boxing işlemi.

```
int i=50; //değer tipi

object o=(object) i; //boxing
```

Referans-Değer Dönüşüm(Unboxing)

Boxing işleminin tam tersidir. Aşağıdaki koşullara uyularak yapılmalıdır.Bilinçsiz yapılamaz.

Unboxing işlemine tabi tutulacak nesnenin daha önceden boxing işlemine tabi tutulmuş olması.

Boxing işlemine tabi tutulmuş olan bu nesnenin unboxing işlemi sırasında doğru türe dönüştürülmesidir.

Örn: int i=50;

```
object o=i;

int j=(int)o;
```

C# ile Genel Programlama Bilgileri

- ✓ Form kontrolleri
- ✓ Diyalog Pencereleri
- ✓ Şartlı Dallanma Yapıları
- ✓ Diziler
- ✓ Döngüler
- ✓ String İşlemleri
- ✓ I/O İşlemleri
- ✓ Metotlar

Form Kontrolleri

Label Kontrolü

LinkLabel Kontrolü

TextBox Kontrolü

Button Kontrolü

Combobox Kontrolü

ListBox Kontrolü

ListBox Örnekleri

TreeView Kontrolü

GroupBox Nesnesi

Panel Kontrolü

TabControl Nesnesi

CheckBox Kontrolü

CheckedListBox Kontrolü

DataGridView Kontrolü Özellikleri

DataGridView Örnekleri

RadioButton Kontrolü

RadioButton Örnekleri

Timer Kontrolü

Timer Örnekleri

ProgressBar Kontrolü

ProgressBar Rengini Değiştirme

PictureBox Kontrolü

MenuStrip Kontrolü

ContextMenuStrip Kontrolü

ImageList Kontrolü

ToolTip Kontrolü

DateTimePicker Kontrolü

Form kontrolleri

```
comboBox1.Items.Add("Anakart");

comboBox1.Items.Add("İşlemci");

comboBox1.Items.Add("RAM");


comboBox1.Items.Clear();

comboBox1.SelectedItem.ToString()
```

```
listBox1.Font = new Font("Arial", 14);
listBox1.BackColor = Color.Olive;
listBox1.ForeColor = Color.White;

listBox1.Items.Add("ANDROID");
listBox1.Items.Add("ARDUINO");
listBox1.Items.Add(textBox1.Text);

listBox1.Items[listBox1.SelectedIndex].ToString();
listBox1.Sorted = true;
listBox1.Items.Clear();
listBox1.Items.Remove(listBox1.SelectedItem);
```

Listbox içindeki sayıları toplama

```
private void button1_Click(object sender, EventArgs e)
    {
      double toplam = 0;
      for(int i=0;i<listBox1.Items.Count;i++)
      {
        toplam += Convert.ToDouble(listBox1.Items[i]);
      }
      label1.Text = "Toplam : " + toplam;
    }
```

GroupBox-CheckedListbox

GroupBox kontrolü, Windows Form' da alt kontrolleri belirli bir gruba yerleştirmek için bir kapsayıcı olarak kullanılır. Kullanım amacı kontrolleri bir grup altında toplayarak kategorilere ayırmaktır.

CheckedListbox kontrolü içindeki işaretlenmiş yada işaretlenmemiş olan tüm elemanlara index numarası kullanılarak erişilebilir.

```
checkedListBox1.Items.Add("PHOTOSHOP");
```

Tüm listeyi seçmek için;

```
int toplam=checkedListBox1.Items.Count;
for(int i=0;i<toplam;i++)
{
   checkedListBox1.SetItemChecked(i, true);
}
```

Diyalog Pencereleri

MessageBox

MessageBox DialogResult

ColorDialog

FolderBrowserDialog

Font Dialog

OpenFileDialog

OpenFileDialog ile Dosya Açma

SaveFileDialog

Diyalog Pencereleri

MessageBox.Show(Mesaj,Başlık,Buton Tipi,Simge)

Mesaj: Mesaj penceresinde görüntülenecek mesajdır.String veri türünden değer alır.En fazla 1024 karakter olabilir.

Başlık: Mesaj penceresinin başlığını belirtir.

Tip: Mesaj penceresi üzerinde görüntülenecek olan butonları belirtir. MessageBox üzerinde görüntülenecek buton tipleri şu değerleri alabilir.

Simge: Mesaj penceresinde görüntülenecek simgeleri belirlemek için kullanılır. Kullanılacak simgeler şunlardır;

Diyalog Pencereleri

MessageBox DialogResult

Evet yada Hayır seçeneklerinden Evet (Yes) basıldığında Formu kapatan, Hayır (No) butonuna basıldığında ise MessageBox' ta Çıkış yapılmadı uyarılarını veren yapı.

```
DialogResult dialog = new DialogResult();
        dialog = MessageBox.Show("Programdan çıkılsın mı?","ÇIKIŞ", MessageBoxButtons.YesNo);
        if(dialog==DialogResult.Yes)
        {
           this.Close();
        }
        else
        {
           MessageBox.Show("Çıkış yapılmadı");
        }
```

Çalışma Sorusu