class Baglanti

{

static Baglanti b;

private Baglanti()

{

}

static Baglanti()

{

b=new Baglanti();

}

public static Baglanti BaglantiDonder()

{

return b;

}

}

} SİNGELTON

--

PROPERTY

public int Kemal {

get{}

set{}

}

--

class Kesir

{

public int Pay;

public int Payda;

public Kesir(int p,int pd)

{

Pay = p;

Payda = pd;

}

public void Yaz()

{

Console.WriteLine("{0}/{1}", Pay, Payda);

}

public static explicit operator int(Kesir k)

{

//donusumun nasil yapilacagi tarif edilir

int sonuc = k.Pay / k.Payda;

return sonuc;

}

public static implicit operator Kesir(int sayi)

{

Kesir k = new Kesir(sayi, 1);

return k;

}

public static Kesir operator +(Kesir k1, Kesir k2)

{

int yp = k1.Pay \* k2.Payda + k2.Pay \* k1.Payda;

int ypd = k1.Payda \* k2.Payda;

Kesir k = new Kesir(yp, ypd);

return k;

}

public static bool operator ==(Kesir k1, Kesir k2)

{

double sonuc1 = k1.Pay / k1.Payda;

double sonuc2 = k2.Pay / k2.Payda;

return sonuc1 == sonuc2;

}

public static bool operator !=(Kesir k1, Kesir k2)

{

return !(k1 == k2);

} OPERATOR OVERLOADING

--

OPEN-CLOSED UYAN

namespace GenericClass

{

class Islemler<T>

{

T[] dizi = new T[10];

int ESay = 0;

public void elemanEkle(T eleman)

{

dizi[ESay] = eleman;

ESay++;

}

public T elemanDonder(int indis)

{

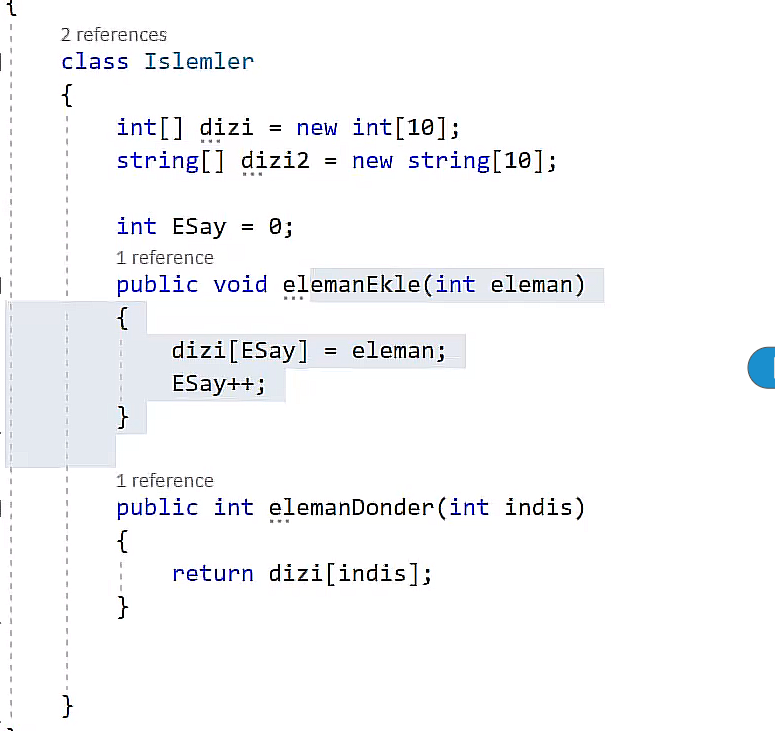
return dizi[indis];

}

}

} GENERİC CLASS

OPEN-CLOSED UYMAYAN



--

POLIMORPHISM

VARLIK SINIFINDA

public virtual void yuru()

{

Console.WriteLine("Varlik gibi yuruyorum");

}

KEDİ SINIDINDA

public override void yuru()

{

Console.WriteLine("kedi gibi yuruyorum");

}

class Sekil

{

public virtual void Ciz()

{

Console.WriteLine("sekil cizildi");

}

}

class Daire : Sekil

{

public override void Ciz()

{

Console.WriteLine("daire cizildi"); //daire çizildi

}

}

class Ucgen : Sekil

{

public new void Ciz()

{

Console.WriteLine("ucgen cizildi"); //sekil çizildi

}

}

class Kare : Sekil

{

public override void Ciz()

{

Console.WriteLine("kare cizildi"); //kare çizildi

}

}

Kalıtımda ata sınıf eğer virtual değişkeni ile yazılmışsa; değiştirilmesine, üstüne yazılmasına izin veriliyor demektir.

Çocuk sınıfta override değişkeni kullanılarak üstüne yazılabilir.

ABSTRACT CLASS

abstract class SoyutSinif

{

abstract public void birMetot(int a);

public void metot2()

{

}

//public int absProp { get; set; }

} Ata Sınıf,

class BirSinif:SoyutSinif

{

public override void birMetot(int a)

{

Console.WriteLine("deneme");

}

} Çocuk Sınıf,

Abstract class’lar çocuk sınıflar için değişken veya metot tanımlayıp içini boş bırakırlar. Başlarına Abstract gelir ve çocuk sınıfların içi boş bırakılan değişken veya metotların gövdelerinin doldurması beklenir. Doldurulmazsa hata oluşur. NESNE TÜRETEMZLER.

SEALD CLASSLAR (KALITIMA KAPALI SINIF)

class A

{

public virtual void metot()

{

Console.WriteLine("Anin metodu");

}

}

class B:A

{

public override void metot()

{

Console.WriteLine("Bnin metodu");

}

}

class C:B

{

public sealed override void metot()

{

Console.WriteLine("Cnin metodu");

}

}

class D:C

{

public new void metot();

{

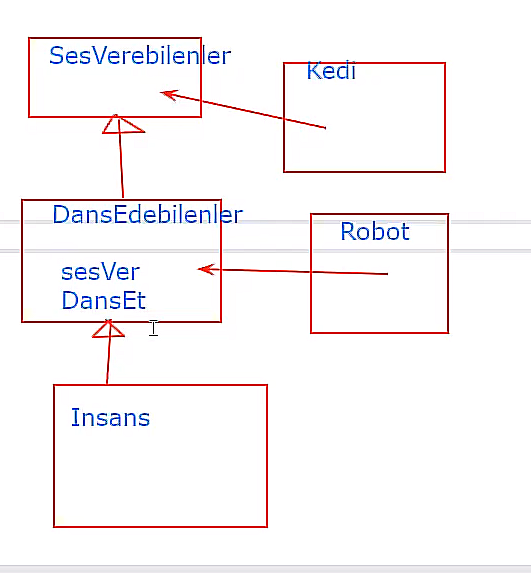
Console.WriteLine("Dnin Metodu")

}

}

Şimdi burada Seald metotlara örnekler var. Seald metotlar kısaca kalıtıma kapalı ya da kalıtıma kapatmak şeklinde tanımlanabilir. Virtual şekilde yazılmış ata sınıf, çekirdekte değişikliğe izin veriyor demektir. B ve C sınıfları bu değişimden yararlanabilirler ama C sınıfındaki sealed değişkeni, C sınıfından sonra ovirride etmeyi kapatır. Yani kısaca kalıtıma kapatır. Abstract classların tam tersidir. NESNE TÜRETEBİLİRLER.

INTERFACE LER



İNTERFACE SEGRETİON BURADA İHLAL EDİLİR ÇÜNKÜ ROBOTUN SES VERMESİ BEKLENMEZ