

Inheritance (Kalıtım)

Object Oriented Programming

Equals, GetHashCode ve GetType Metotları
Nesnelerdeki ToString, Nereden Gelmektedir?

??



```
class Program
{
    0 references
    static void Main(string[] args)
    {
        İnsan insan = new İnsan();
        insan.
    }
}

1 reference
class Canlı
{
    0 references
    public int Yasi { get; set; }
}

1 reference
class İnsan : Canlı
{
    ...
}
```

- Equals
- GetHashCode
- GetType
- ToString
- Yasi

Nesnelerin Atası/Ademi Object Türü

- C# programlama dilinde tüm sınıflar **Object** sınıfından türetilir.

Yani kalıtım
alsa da
almasa da
mı?

```
class Canli
{
    public int Yasi { get; set; }
}

class Insan : Canli
```

Object sınıfının
içeriğinde ne var
acaba?

Yani
nihayetinde bir sınıf
ya da
Object sınıfından
türetilmiştir.

Yok eğer bu şekilde herhangi bir sınıftan kalıtım alıyorsa, bir sınıfın aynı anda birden fazla sınıftan kalıtım alamama prensibinden yola çıkarak biryandan da Object sınıfından türemeyecek sadece kalıtım aldığı sınıftan türeyecektir.

Alan sınıf bu şekilde herhangi bir sınıftan kalıtım alıyorsa default olarak **Object** sınıfından türetilen olacaktır.

Tabi burada kalıtım veren sınıf herhangi bir sınıftan türemiyorsa eğer en nihayetinde Object'ten türeyeceği için dolaylı yoldan Insan sınıfı da Object'ten kalıtım almış olacaktır.

Object Sınıfı Memberları

```
namespace System
{
    ...public class Object
    {
        ...public Object();
        ...~Object();
        ...public static bool Equals(Object? objA, Object? objB);
        ...public static bool ReferenceEquals(Object? objA, Object? objB);
        ...public virtual bool Equals(Object? obj);
        ...public virtual int GetHashCode();
        ...public Type GetType();
        ...public virtual string? ToString();
        ...protected Object MemberwiseClone();
    }
}
```

Constructor

Destructor

Members
İşte bu member'lardan public olanlar tüm sınıf nesnelerinde otomatik olarak erişilebilir olmaktadır.

İsim Saklama(Name Hiding) Sorunsalı

- Kalıtım durumlarında atalardaki herhangi bir member ile aynı isimde member'a sahip olan nesneler olabilmektedir.

```
class A
{
    public int X { get; set; }
}
class B : A
{
    public int X { get; set; }
}
```

B b = new B();
b.

Equals
GetHashCode
GetType
ToString
X

Şimdi bu X A'dan mı geliyor, yoksa B'den mi??

Günümüzde buna ihtiyaç yoktur!

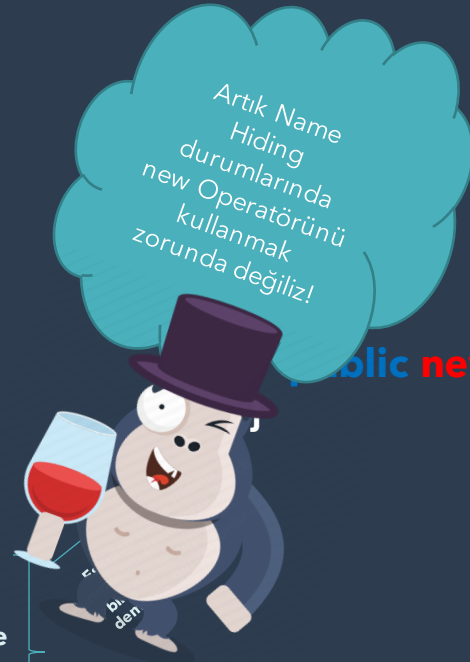
Evet, eskiden name hiding durumlarında alt sınıfın içerisindeki üyeyi new ile işaretliyorduk

İhtiyaç yoktur lakin biz yine de Name Hiding durumlarında new operatörünün kullanımını inceleyelim...

Name Hiding Durumlarında **new** Operatörünün Kullanımı

```
class A
{
    public int X { get; set; }
}
class B : A
{
    public int X { get; set; }
}
```

Yukarıda bir Name Hiding durumu söz konusudur. Günümüzde bu şekilde Name Hiding durumlarında ekstradan bir bildiride bulunmak zorunda değiliz.



```
X { get; set; }
```

```
public new int X { get; set; }
```

Bu şekilde bildiride bulunduğumuzda derleyiciye Name Hiding durumunun olduğunu ve bunu iradeli bir şekilde yaptığımızı bildirmiş bulunuyoruz.

Record'lar da Kalıtım?

- Record'lar sade ve sadece Record'lar dan kalıtım alabilmektedirler.
- Class'lar dan kalıtım alamazlar yahut veremezler!
- Kalıtımın tüm temel kuralları record'lar için geçerlidir;
 - Bir record aynı anda birden fazla record'dan kalıtım alamaz!
 - Record'lar da temelde class oldukları için üretilir üretilmez otomatik olarak Object'ten türerler.
 - base ve this keywordleri aynı amaçla kullanılabilmektedir.
 - Name Hiding söz konusu olabilmektedir.
 - Ve aklıma gelmeyen diğer tüm durumlar da record'lar için geçerlidir.

```
record MyRecord
{
}

record MyRecord2 : MyRecord
{
}
```

Yorulduk müdür!

