Masaüstü Uygulama Geliştirme

Hafta-14

Öğr. Gör. Zafer SERİN

Soyutlama(Abstraction)

- Soyutlama bir mantıktır, bir davranıştır.
- Tanım olarak soyutlama bir iş yaparken sadece ihtiyaç duyulan yapıların bulunması durumudur.
- Programlama açısından kullanıcı işlemleri için kullanılacak bir classta yüzlerce member olabilir, buna karşın sadece kullanıcı kayıt etmek isteyen bir kişiye bu metodun gösterilmesi yeterli olacaktır. İşte bu soyutlamadır.
- Bir başka deyişle soyutlama gerekli olanları göster, gereksiz olanları gösterme demektir.

Soyutlama(Abstraction)

- Soyutlama ile bir sınıfın memberlarından ihtiyaç noktasında alakalı olanları gösterip, alakalı olmayanları göstermemektir.

Abstract Class

- Abstract Class kullanılarak abstraction uygulanabilir.
- Abstraction = abstract class şeklinde bir düşünce yanlıştır. Bunlar aynı kelime kökenine sahip olmasına karşın abstraction interface ile de uygulanabilir. Burada dikkat edilmesi gereken abstraction davranışının abstract class ile uygulanabilecek olmasıdır.
- Bir sınıfın uyması gereken temel yapıyı tanımlamak için abstract class yapısı kullanılabilir ve gerekli modellemeyi gerçekleştirebilirsiniz.

Abstract Class

- Abstract classlardan normal şartlarda nesne üretilemez!

Interface(Arayüz) Nedir?

- Programlama süreçlerinde Interface yapılanması, nesnelere direkt olarak bir arayüz/şablon oluşturulmasını ve bu arayüz üzerinden geliştirici ile nesne arasındaki etkileşimin daha da kolaylaştırılmasını sağlayan bir araçtır.
- Hatta sadece geliştirici ile nesne arasındaki süreci kolaylaştırmamakta, ayrıca bir programın farklı bir programla veya bileşenle etkileşimini de kolaylaştırmaktadır.

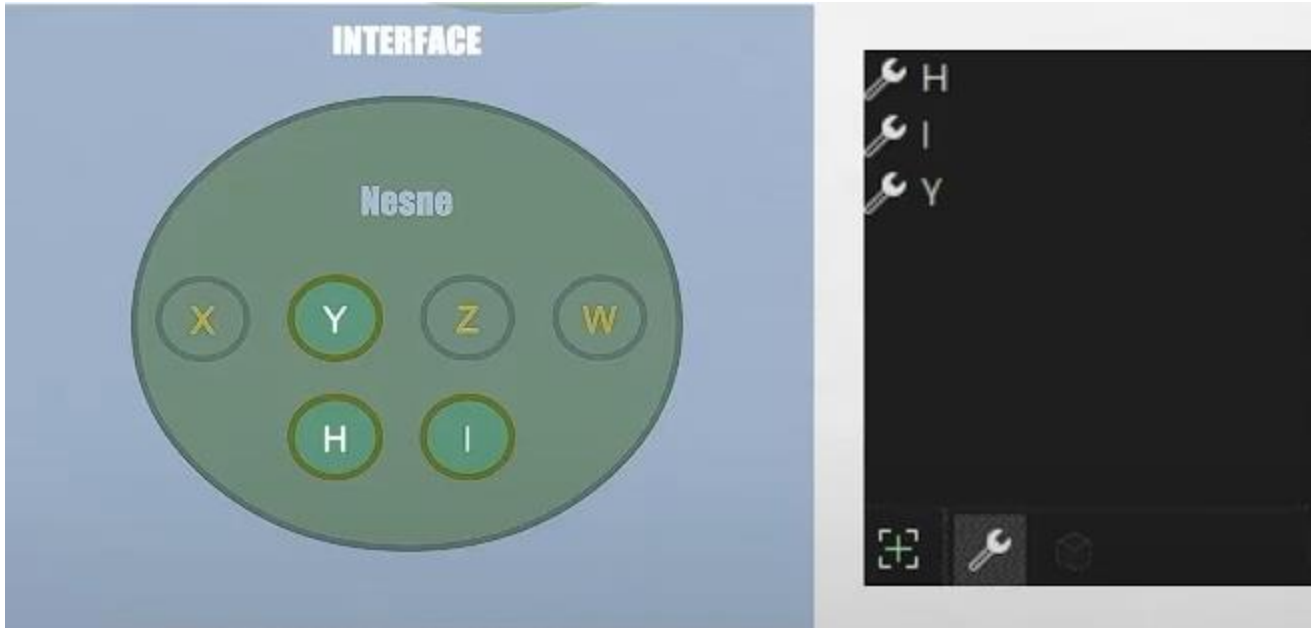
Interface(Arayüz) Nedir?

- Interface'in nesneye bir arayüz sağlaması, kullanıcı açısından ilgili nesnenin nasıl çalıştığına dair ayrıntılı bilgiye ihtiyaç duyulmaksızın, sadece arayüzün sunduğu fonksiyonların veya propertyleri kullanarak etkileşime girilmesini sağlar.
- Interface ilgili nesneye abstraction(soyutlama) uygulayarak belirlenmiş interface üzerinden çalışılmasını ve böylece ilgili nesne ile geliştirme sürecinin kolaylaştırılmasını sağlamaktadır.

Interface(Arayüz) Nedir?

- Abstract classlarda bir nesnenin içerisindeki imzaları modellemenin dışında farklı işlemlerde gerçekleştirilebilirken interfaceler sadece imzalara odaklanan bir yapıdır denilebilir.
- **Interface, abstract classın sadece imzalara konsantre olmuş halidir! O yüzden abstraction(soyutlama) davranışı açısından abstract classa nazaran daha elverişlidir!**

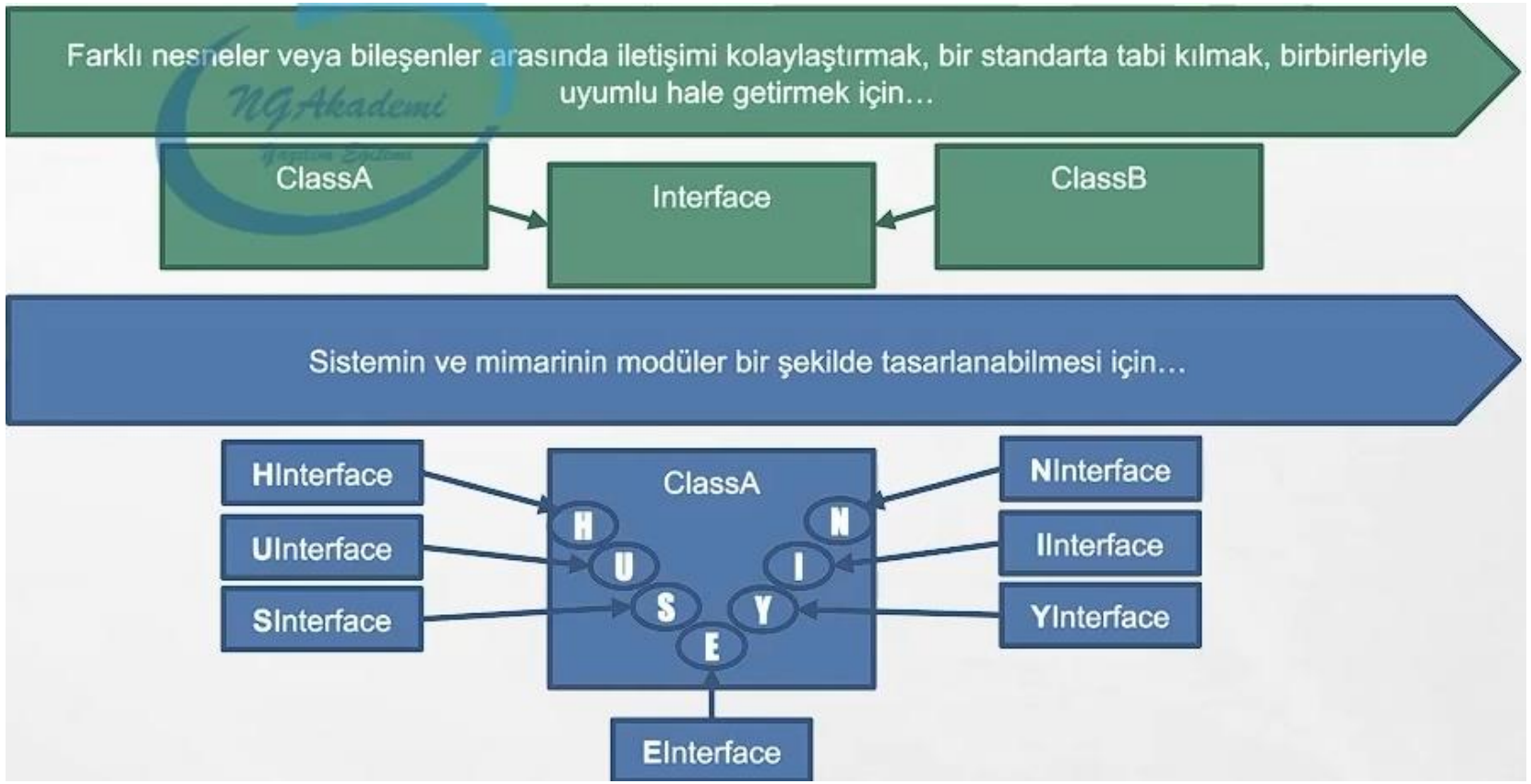
Interface(Arayüz) Nedir?



Interface(Arayüz) Nedir?

- Bizler bir sınıfta(classta) olmasını istediğimiz memberları oluşturmadan önce bir interfacete bu memberların imzalarını tanımlayarak bir şablon oluşturuyoruz.
- Ardından bu interface i kullanarak sınıfta memberları tanımlıyoruz.
- Bir sınıf, bir interface kullanıyorsa eğer aynen abstract classlardaki abstract elemanlarda olduğu gibi içerisindeki memberların tanımlanmasını zorunlu kılar!

Neden Interface Kullanırız?



Interface Nasıl Tanımlanır?

- Interface tanımlayabilmek için 'interface' anahtar sözcüğünden faydalanırız.
- Interface'ler referans türlü değişkenlerdir.
- C# programlama dilinde bir Interface'e isim verirken 'I' ile başlaması bir gelenektir.
- Bir class nerede tanımlanabiliyorsa interface te orada tanımlanabilir.

Interface Nasıl Tanımlanır?

```
interface IKullanici
{

}
```

Interface İçerisinde İmzaların Oluşturulması

- Interface içerisinde metot ve property imzaları tanımlanabilmektedir.
- İmzalar tanımlanırken erişim belirteci kullanılmaz, gövdeler tanımlanmaz sadece imzalar tanımlanacaktır.
- Abstract classlarda olduğu gibi imzalarda 'abstract' keywordü kullanılmaz! Çünkü interfaceler sadece sınıflara uygulatılacak imzaları barındırır abstract classlarda olduğu gibi normal memberlara sahip değillerdir!
- Interface içerisinde field tanımı mümkün değildir!

Interface İçerisinde İmzaların Oluşturulması

```
interface IKullanici
{
    void Dogrula();
    int Hesapla();
    int Yas { get; set; }
}
```

Interface Bir Class'a Nasıl Uygulatır

- **Interfacei kullanabilmek için kalıtım sürecinde kullanılan ':' operatörü kullanılır, ancak bu işlemin adı artık kalıtım değil implementasyon olur! Aynı durum abstract class içinde geçerlidir.**
- Interfacelerden yapılan implementasyon sonucu ilgili class interfaceteki imzaları uygulamak zorundadır ancak abstract classlarda olduğu gibi bu işlem için 'override' anahtar sözcüğü kullanılmaz!
- **Interfaceler sınıfların imzasıdır!**

Interface Bir Class'a Nasıl Uygulatır

```
interface IPersonel
{
    void Dogrula();
}

class Personel : IPersonel
{
    public void Dogrula()
    {
    }
}
```

Interfacelerde Çoklu Kalıtım Durumu

- C#'ta bir class sadece tek bir classtan kalıtım alabilmektedir!
- Buna karşın bir class birden fazla Interfacei aynı anda uygulayabilmektedir.
- Ayrıca ilgili classın nesnesi tüm bu interfaceler ile refere edilebilmekte ve soyutlama davranışı gerçekleştirilebilmektedir.

Interfacelerde Çoklu Kalıtım Durumu

```
interface ITopla
{
   int Topla();
}

interface ICarp
{
   int Carp();
}

class Islem : ITopla, ICarp
{
   public int Topla()
   {
      return 5;
   }
   public int Carp()
   {
      return 5;
   }
}
```

Interfaceler Birbirinden Kalıtım Alabilir mi?

- Interfaceler kendi aralarında kalıtım alabilir. Bu işlem implementasyon olarak değil normal kalıtım olarak isimlendirilir. Burada standart kalıtım kuralları geçerli olacaktır. Buradaki en önemli fark Interfaceler kendi aralarında kalıtım uygulasa bile yatayda yine birden çok Interface uygulanabilir!

Interfaceler Birbirinden Kalıtım Alabilir mi?

```
interface ITopla
{
    int Topla();
}

interface ICikar
{
    int Cikar();
}

interface ICarp : ITopla, ICikar
{
    int Carp();
}
```

Hem Kalıtım Hem Implementasyon Uygulanabilir mi?

- C#'ta bir class, farklı bir classtan kalıtım alırken hem de bir yandan interfacei implemente edecekse eğer önce classtan kalıtım almalı, sonrasında da interfaceler tanımlanmalıdır. Buradaki öncelik önemlidir aksi taktirde derleyici hata verecektir.

Hem Kalıtım Hem Implementasyon Uygulanabilir mi?

```
class Canli
{

}

interface ISolunum
{
   void SolunumYap();
}

interface IDolasim
{
   void DolasimYap();
}

class Insan : Canli, ISolunum, IDolasim
{
   public void DolasimYap()
   {
   }
   public void SolunumYap()
   {
   }
}
```