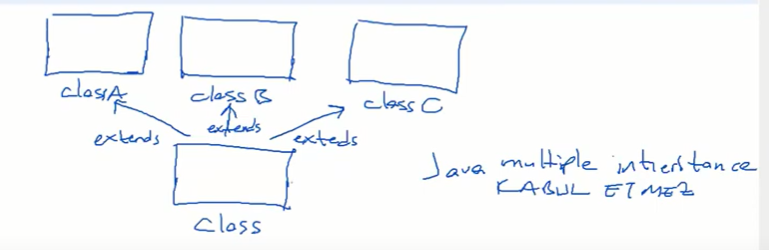
İnterface

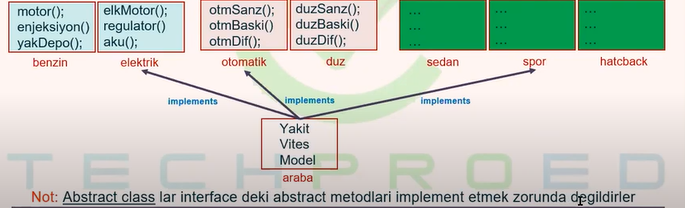
Interface bir class değildir. Interface interfacedir.(Arayüz demektir)

Interface içinde sadece kendinden türüyen sınıfların, içini doldurmak zorunda olduğu body’si olmayan methodlarin oluşturduğu bir yapıdır.

Interface bir ceşit “to do list” yani yapılacak listesidir. Nasıl yapılacağına değil ne yapılacağına odaklanır.



Böyle bir durum söz konusu değildir.Bir class birden fazla class’a inherit edilemez.Bir çocuk 3 baba olmaz



Fakat bir class birden fazla interface ‘e implement edilebilir.Bu bağlanma “implements” ile bağlanabilir.

İnterface nedir neden ihtiyaç vardır?

İnterface içinde mutlaka override etmemiz gereken metotların oldugu arayüzdür

Birden fazla class parents olamaz ama birden fazla interface ‘i parents olarak kabul ederiz

Ekstra bilgiler:

Interface içinde abstract methodlar ve özel concrete methodlar olabilir.

Fakat bir çok kaynakta sadece abstract method vardır diye geçer böyle kabul edilir.

İnterface de body yoktur.

Interface de sadece ABSTRACT, PUBLIC methodlar olur. Bu iki keyword i yazsak da yazmasakta java kabul eder.

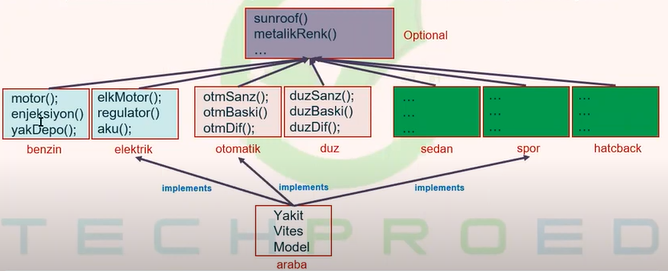
* public void motor();
* void vites();
* abstract void kasa();
* public abstract void teker();

Normal bir classta access modifier belirtmezsek java default kabul eder ama interface de ise public olarak kabul edilir.

Javada interface olusturmak istiyorsak bastan o şekilde olusturulmalıdır. Sonradan kod yazarak ekleme yaparak yapamayız.

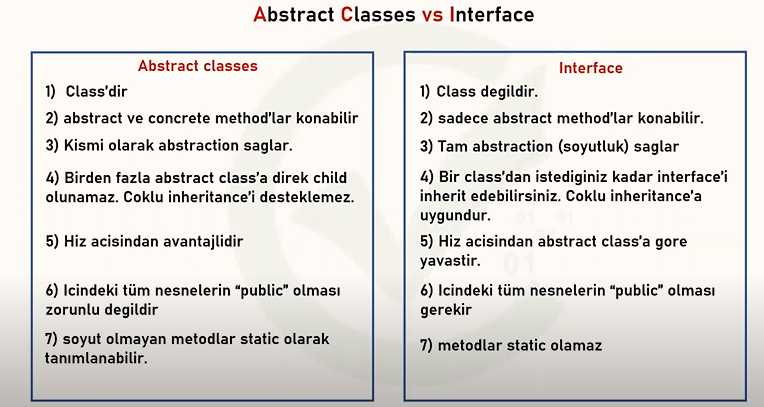
Package 🡺New🡺Class 🡺🡺 Bu standart bir class oluşturma.

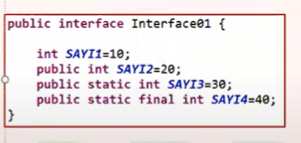
Package 🡺New🡺Interface 🡺🡺 Bu interface oluşturma.



Opsiyonel kısım en tepedeki concrete classtır.

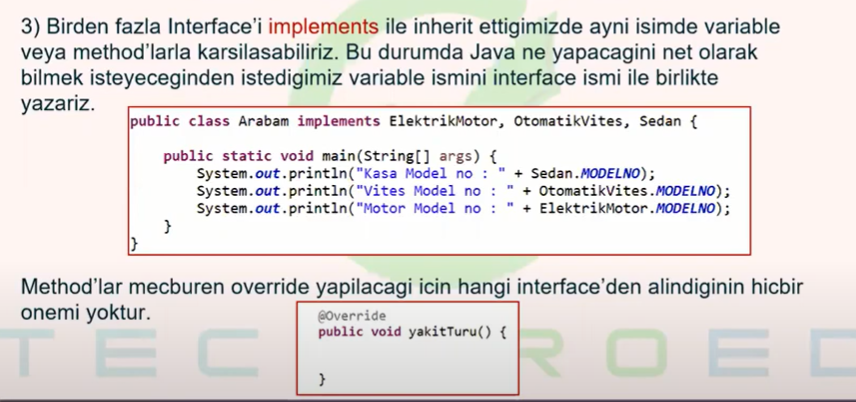
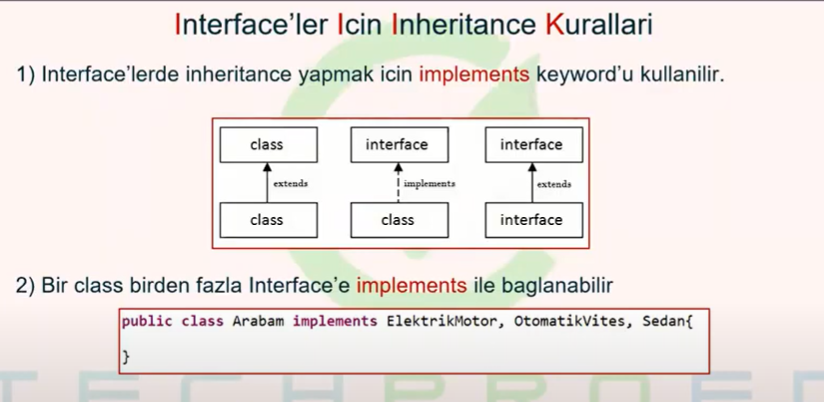
Tüm interfaceler concrete classın child olabilir



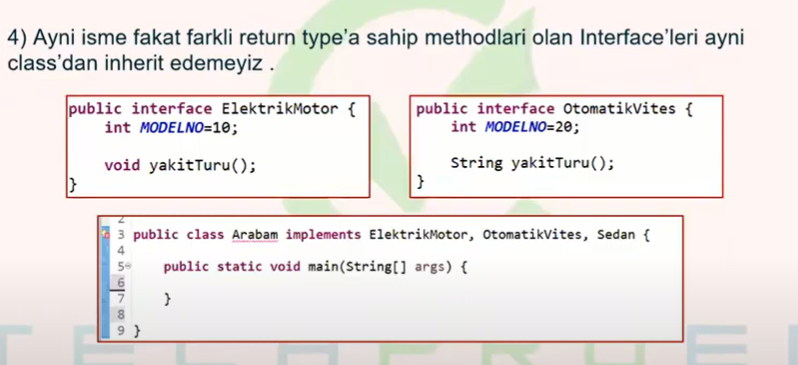


İnterface içerisindeki variableler mutlaka public static ve final olmalidir.

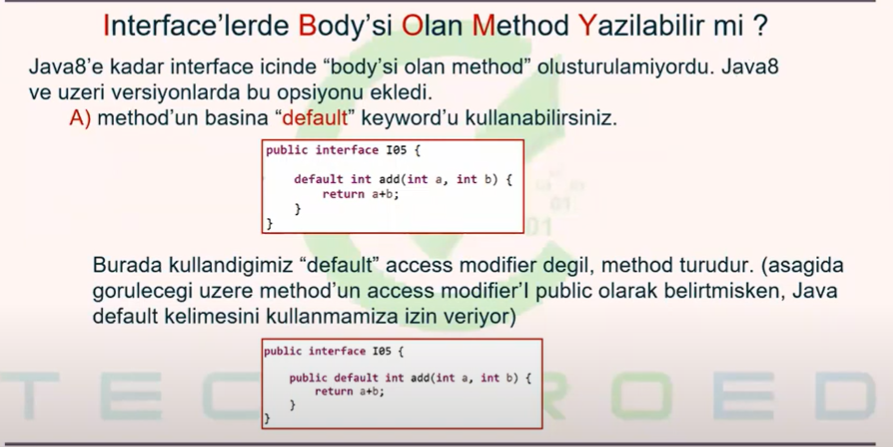
Private veya protected variabler compile time error verir



Variablelar aynı olursa interface.variable şeklinde yazmalıyız.



Metotların isimleri aynı return typeları farklı buna java izin vermez ya o ya bu seçim lazım.



Bu çok özel bir metottdu.

İnterface de default keyword olusturularak concrete metod olusutrabiliriz.

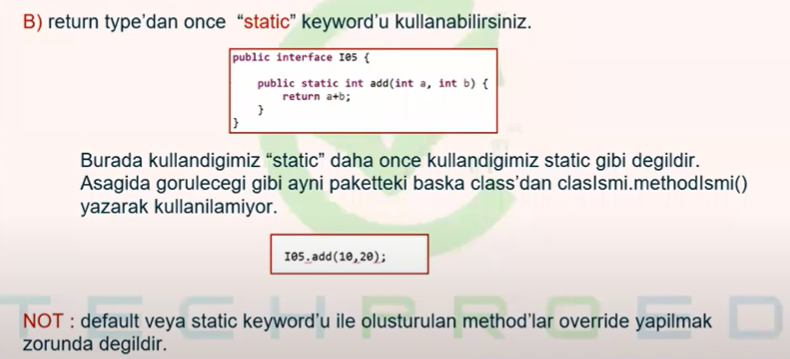
Method un concrete oldugunu belirtir.

Burada ki default access modifer değil javanın özel bir çözümüdür.

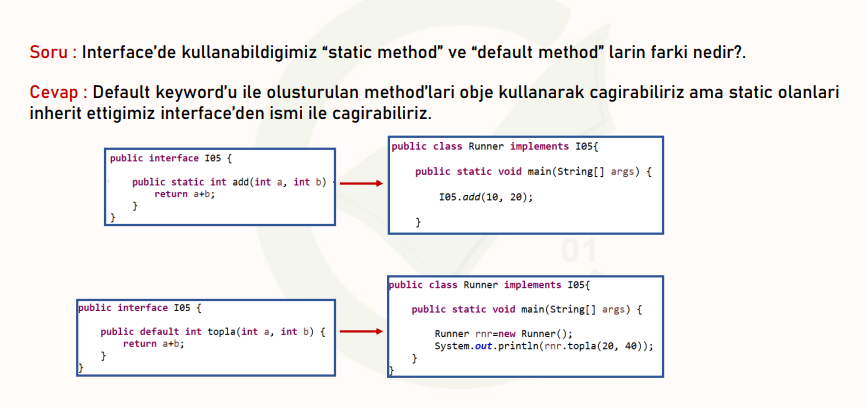
Kullanılmaz genellikle. Çok zorunlu hallerde çıkış kapısıdır .%90 kullanılmaz

Aynı şekilde static de içinde geçerlidir.

Static ve default keywordlarla olusturdugumuz methodlar override edilmek zorunda değildir. Çünkü concrete.NOT : derste denendiğinde metot çağrıldı.



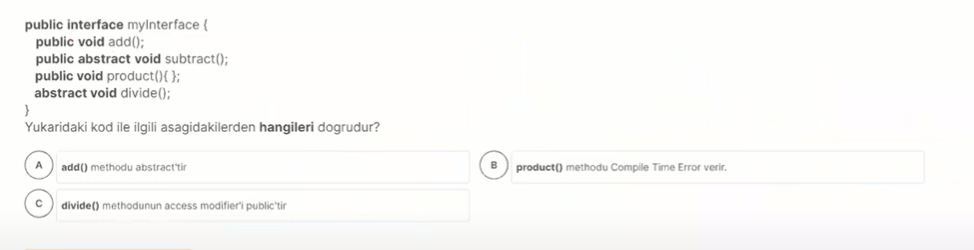
classIsmi.methodIsmi şeklinde çağrıldı ve ulaşılabildi.



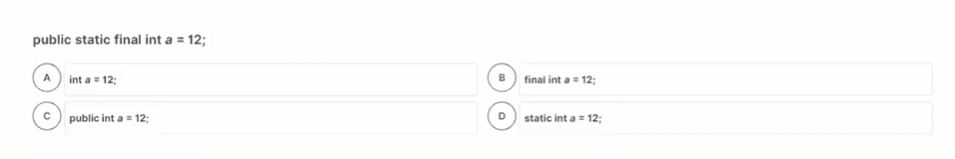
Sorular

Aşağıdakilerden hangisi yanlıştır?

1. interface class abstractır Yanlış
2. interface hem abstract hem concrete metotlardan olusur Yanlış
3. interface de absract method üretilirken abstract yazma zorundalığı yoktur. Doğru
4. interface de metot üretirken concrete keyword u kullanılır. Yanlış

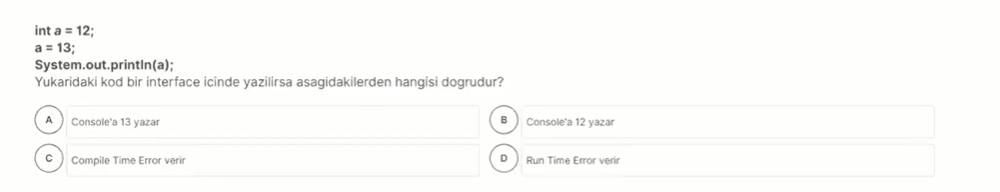


Cevap hepsi

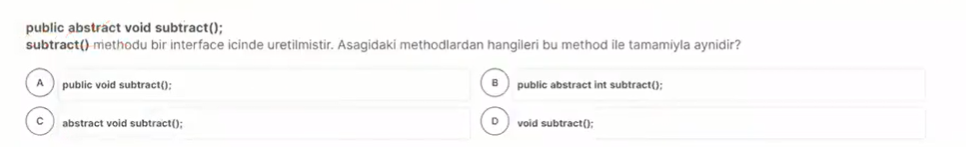


Hangileri aynıdır ?

Public static final yazsakta yazmasakta hepsi



Tekrar değer ataması yapılamaz cevap c



Cevap acd

B de return type değiştirilmiş