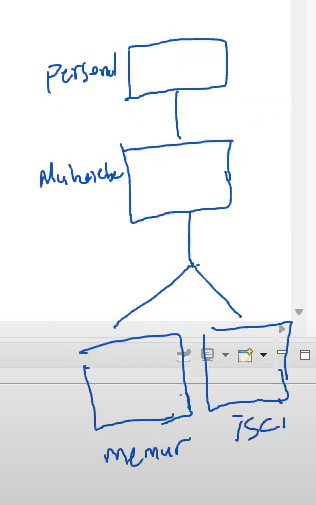
INHERITANCE



**+** Child classlar, parent class tan 🡪 bazı şeyleri inherit ederler

🡪bazı şeyleri değiştirerek kendilerine uygularlar

🡪bazı şeyleri extradan kazanabilirler

🡺**Parent classtaki tüm özellikleri taşırlar**

**+** Bir obje, farklı classlarla karşılaştırıldığında farklı görev ve özellikler alabilir.

Örneğin Ali Canın teyzesi, Veli Canın halası olabilir.

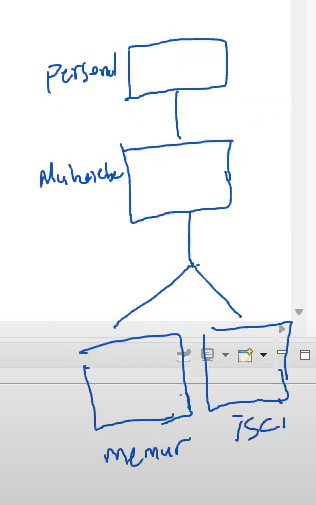
**+** Inheritance ın arkasındaki fikir daha önceden oluşturulmuş classların üzerine yeni classlar oluşturabilmemizdir ki classlar daha derli-toplu olsun.

🡺Böylelikle yeni oluşturduğumuz class, var olan class ın tüm method ve veriable larını kullansın.

🡺Inheritance sayesinde child class parent classtaki public veya protected dataları, objeleri ve methodları problemsiz kullanabilir --> ***REUSABILITY* tekrar tekrar kullanma demektir.**

\* **Genel** özellikler/bilgiler **parent** class ta oluşturulur

\* **Spesifik** özellikler/bilgiler **child** class ta oluşturulur



NOT 1 : Child class lar public ve protected datalari inherite edebilirler.

NOT 2 : **Private datalar inherite edilemez 🡪 Babanin kasası…**

NOT 3 :Default datalar, child ve parent classla ayni package da ise inherite edilebilir ama farklı Packageda ise inherite edilemez.

NOT 4 : Static method ve veriablelar inherit edilemez.

Static method inherit edilmeden de görülebilir. İnherit etmeye gerek yok, class ismi. yaz gelir.

Soru: Parent classtaki variablelara acces modifier olarak *protected yazmayı tercih etmemizin nedeni*ni söyleyiniz.

Cevap: Eğer parent classtaki class üyelerine acces modifier atanmazsa

**Java, default acces modifier** atar.

🡺 Böyle olursa sadece o package altındaki child classlar, parent classı inherit edebilir.

🡺Başka packagelardaki child classların da parent classı inherit

edebilmesi için class üyelerinin access modifierını **PROTECTED** yaparız.

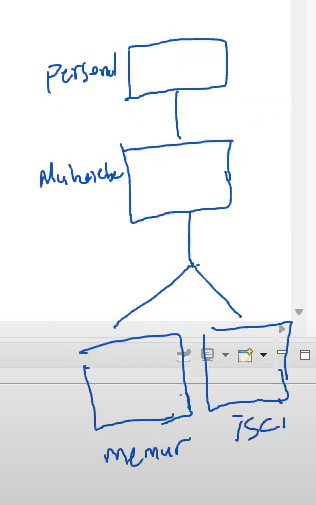
🡺 Public yaparsak da inherit edilebilir. Ama biz herkesin erişmesini

İstemezsek; sadece child classların erişmesini istersek **PROTECTED**’i

tercih ederiz.

**İnheritance** - Video 36 -

Miras-Kalıtım



Parent ve Child Classlar

Bir Classı child yapmak için Class isminin sonuna “extends” keyword ve “parent class” ismi yazılır:

public class Memur extends Muhasebe { }

Memur class ında obje oluşturduktan sonra obje üzerinden parent classlardaki class üyelerine ulaşabilir ve değiştirebiliriz:

**public** **class** Memur **extends** Muhasebe{

**public** **static** **void** main(String[] args) {

Memur memur1=**new** Memur();

memur1.isim="Hasan";

memur1.saatUcreti=16;

memur1.maas=memur1.maasHesapla();

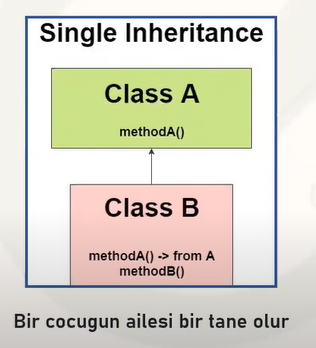
System.***out***.println(memur1.isim + "'in maasi : " + memur1.maas);

}

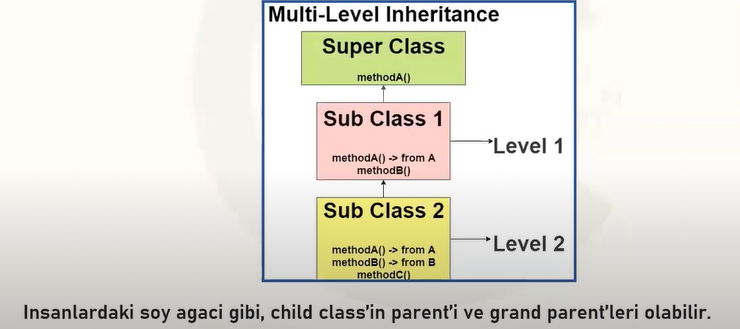
}

Inheritance Türleri

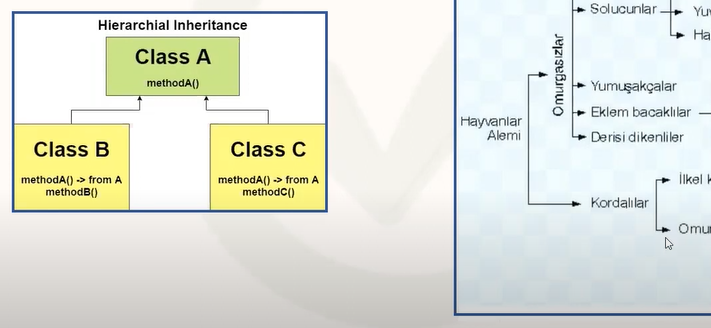
1-Single inheritance 🡪 1 child class 1 parent class



2-Multilevel inheritance 🡪 inheritance zinciri



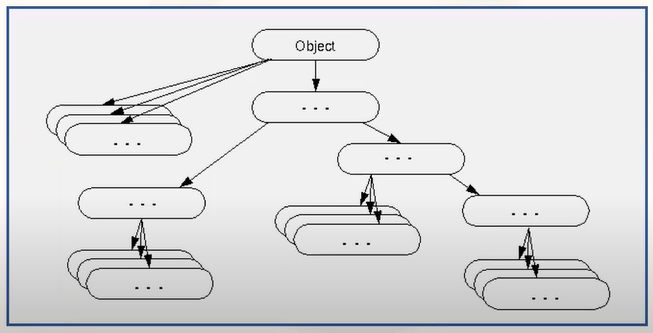
3-Hierarchical inheritance 🡪 1den fazla class 1 parent class



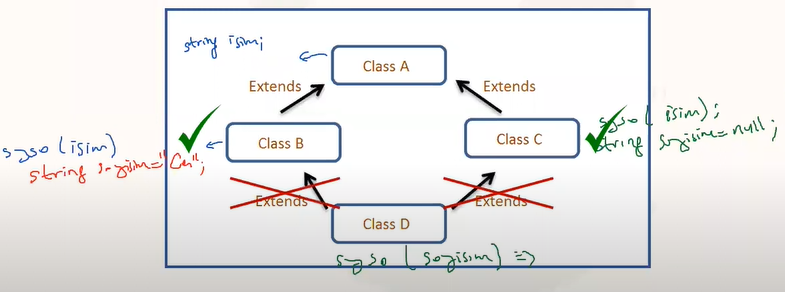
Javada bütün classlar **OBJECT CLASS** ın altındadır.

**OBJECT CLASS** bütün classların parent classıdır.

**OBJECT CLASS**, parent classı olmayan tek classtır.



Multiple inheritance



Java bu yapıyı kabul etmez**!!!**

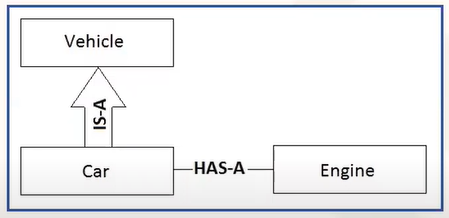
1 parent classın birden fazla child classı olabilir AMAAA

1 child classın birden fazla parent classı **OLAMAZ ! ! !**

**public** **class** Memur **extends** Muhasebe, C01 {}

Java, C01 in altını çizer, kabul etmez!

IS-A () ve HAS-A ()

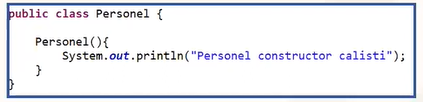


**Child** classtan **parent** classa doğru gidiliyorsa 🡪 Bmw IS-A Car

**Parent** classtan **child** classa doğru gidiliyorsa 🡪 Apartman HAS-A Daire

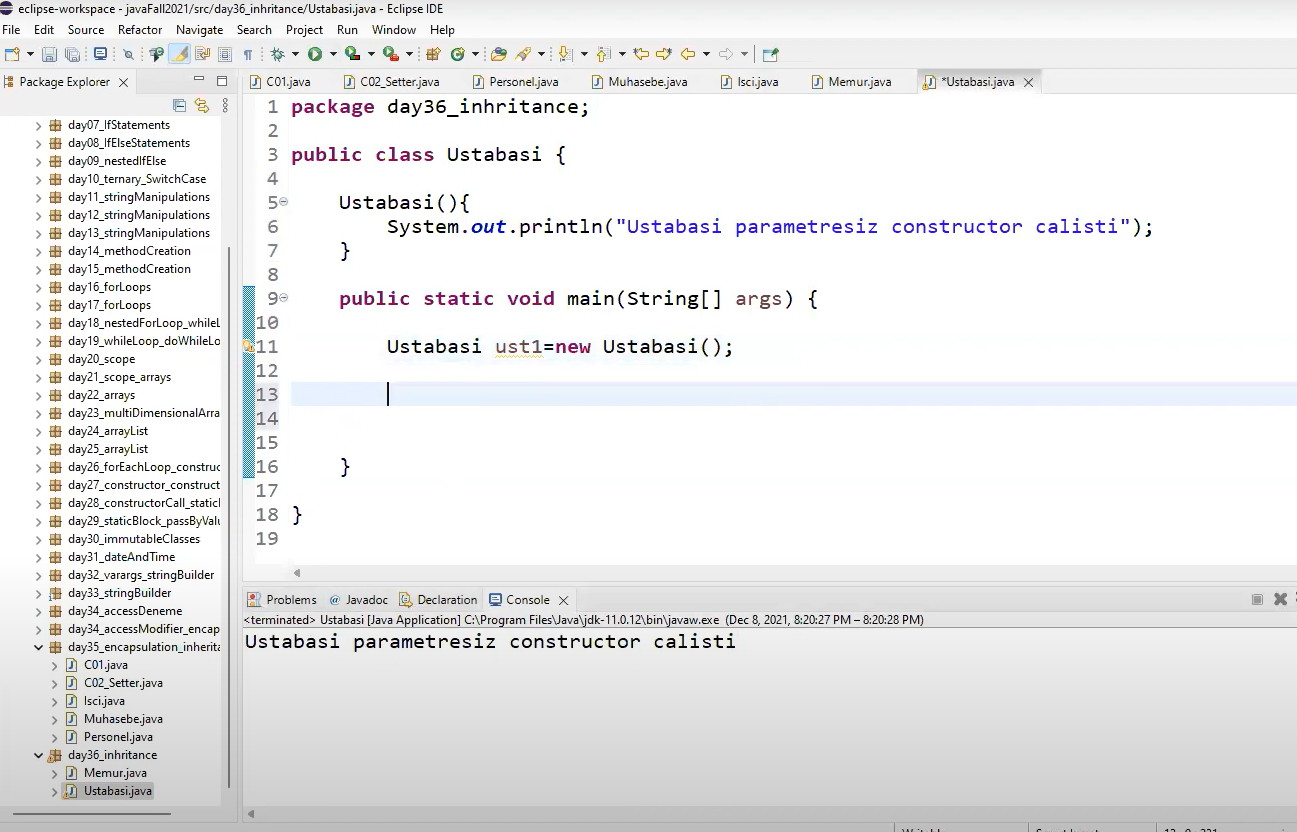
**İnheritance’da Constructor Çağırma**

Constructor 🡪 **Class ismiyle aynı** isimdedir ve **Return Type** yoktur:

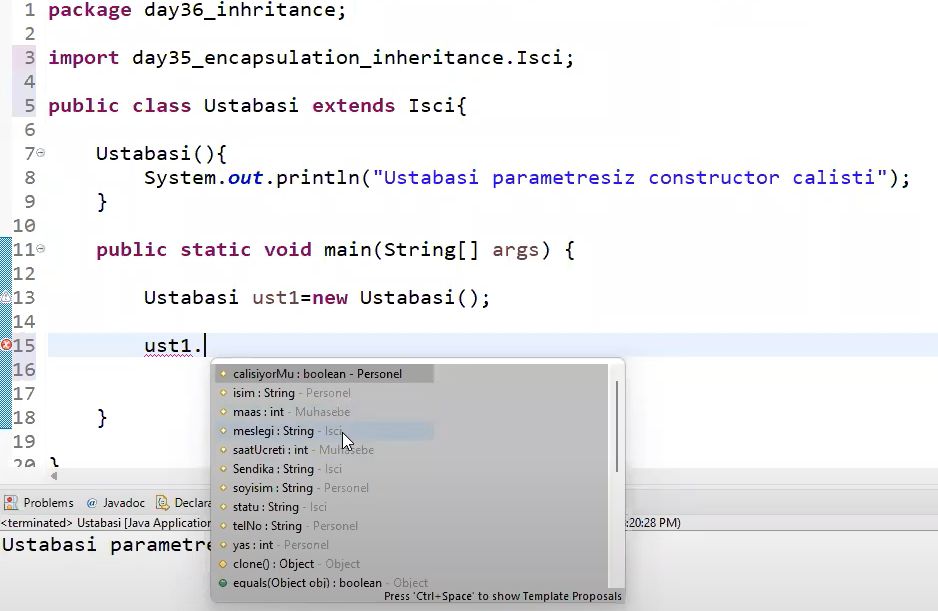


**Manuel olarak parametreli veya parametresiz 1 constructor yazınca *default constructor ölür*.**

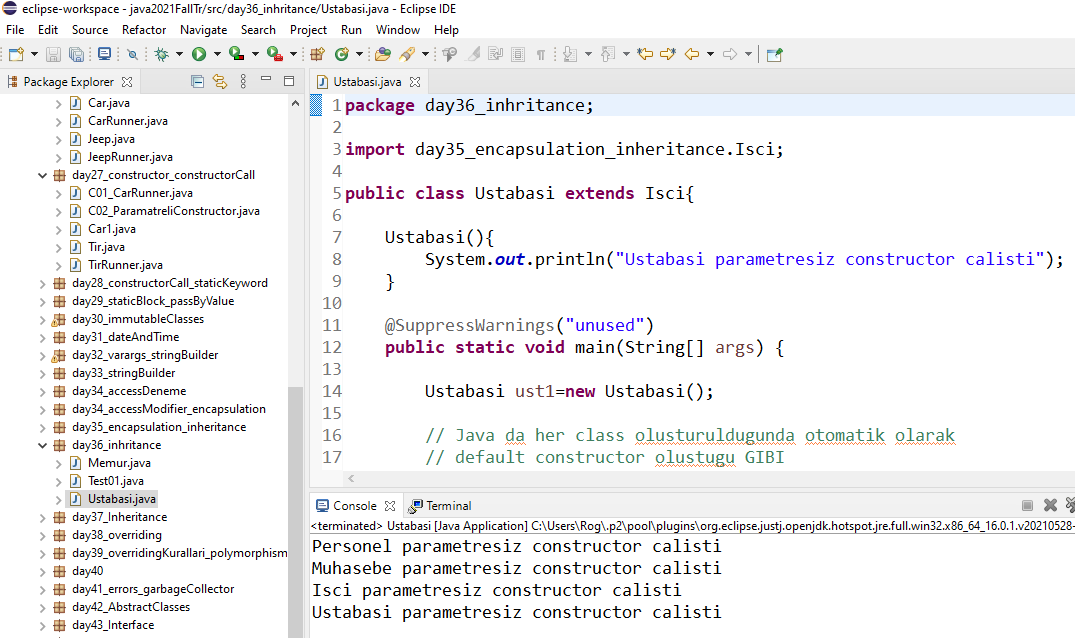
Bir obje constructor çalışmadan oluşamaz**=**Bir obje oluşturulurken ilk önce constructor çalışır:

****

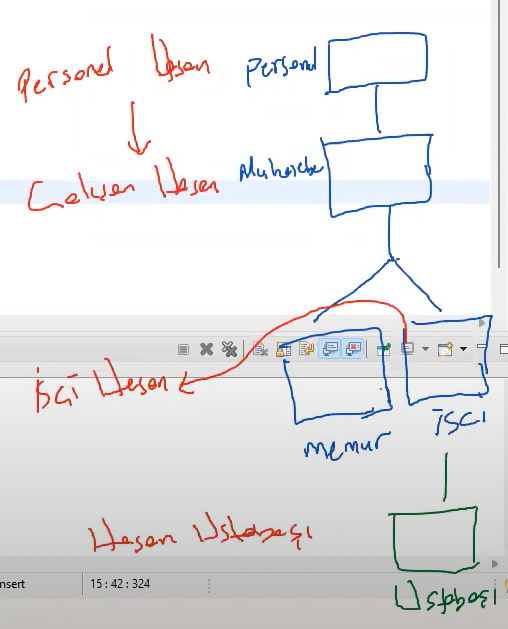
**J**ava, child classta bir obje oluşturulduğunda bu obje üzerinden parent classlardaki class üyelerine ulaşabilmemizi sağlar :



**Ustabaşı classından Personel classına kadar gidip bütün parent classlardaki class üyelerinin getiren ama bizim görmediğimiz bir kod var:**

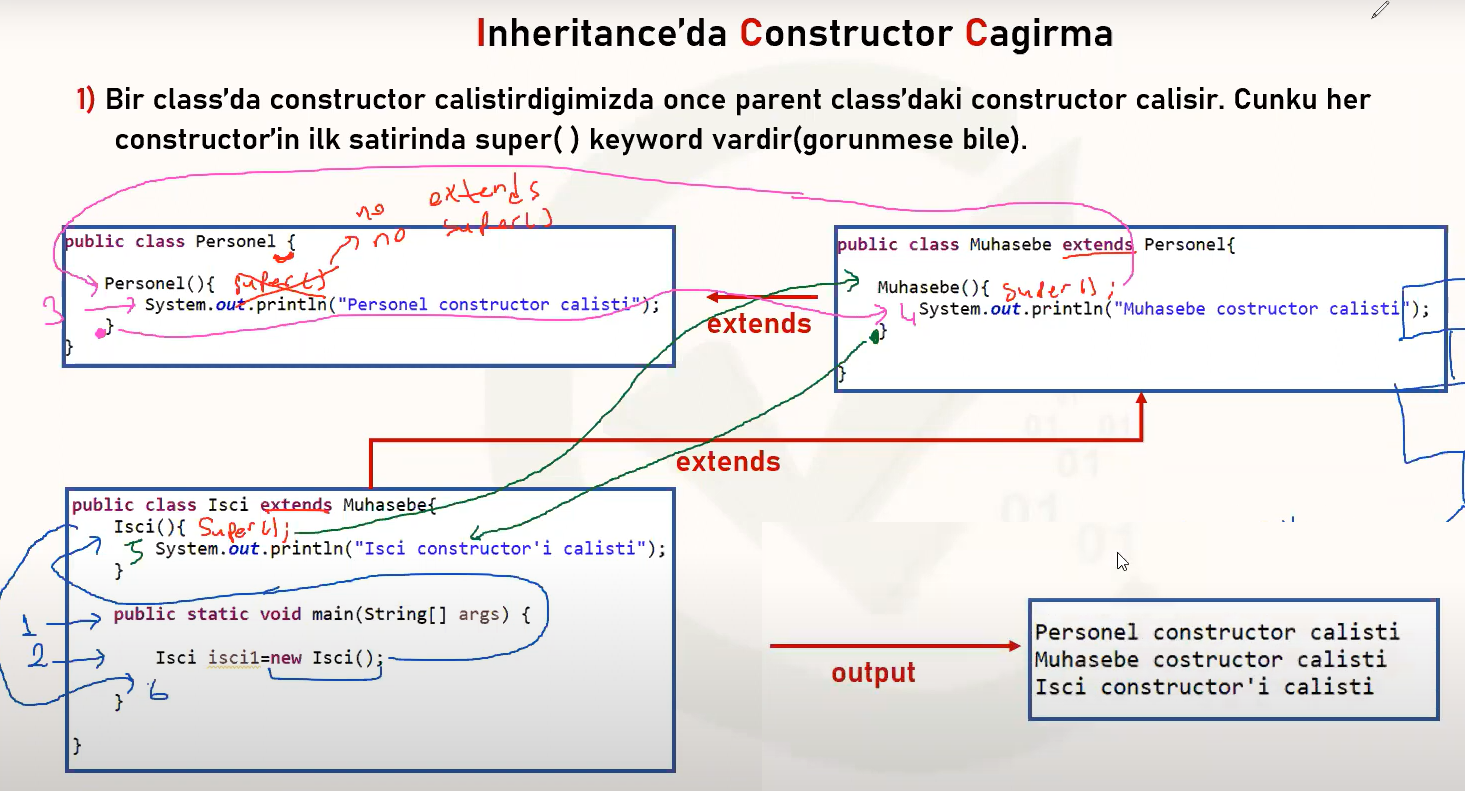
****

**Javada her class oluşturulduğunda otomatik olarak default constructor oluştuğu gibi child classda oluşturulan her bir constructorın ilk satırına gizli bir *super()*** **olusturur/vardır, görünmese bile!!! *Extends* varsa *super()* vardır!!!**

****

**this() 🡪 constructor call : içinde bulunduğu classtaki constructorı çağırır**

**super() 🡪 constructor call : parent classtaki constructorı çağırır**

****

Aşağıdaki 3 class birbiriyle aynıdır, ***default constructor*** ve ***super()*** vardır:

1. **public class Zebra extends Hayvanlar** {

}

1. public class Zebra extends Hayvanlar {

Public Zebra(){

}

}

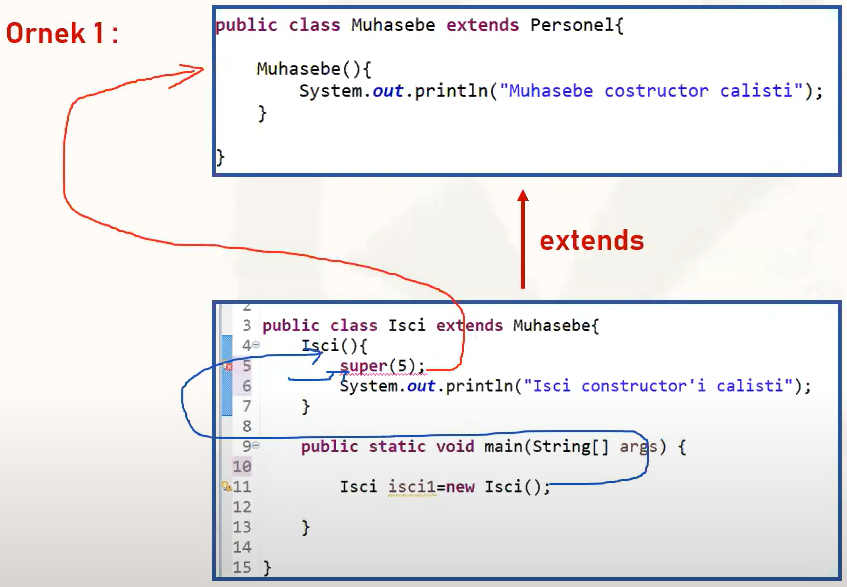
1. **public class Zebra extends Hayvanlar** {

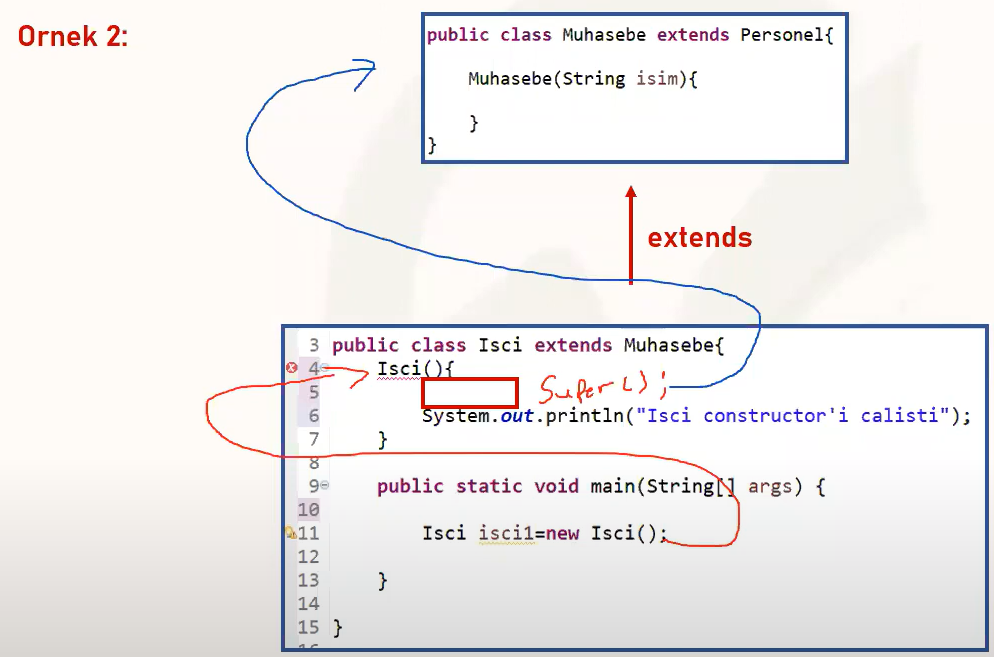
Public Zebra(){

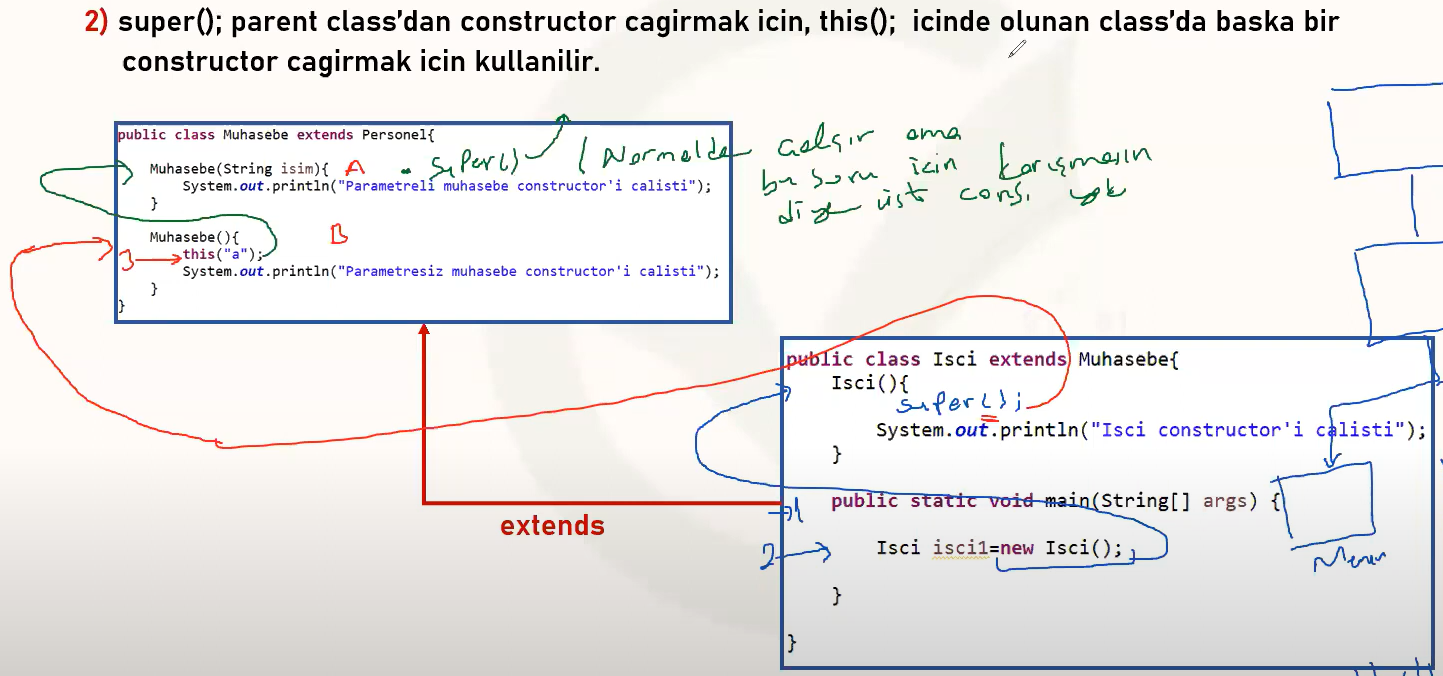
Super();

}

}

**Eğer super classta istenilen özellikte/nitelikte(parametre!!) bir constructor yoksa java CTE verir :**

****



3. sırada çalışan this(“a”) den önce süper() var mı?

OutPut 🡪 Parametreli muhasebe constructor’i calisti

Parametresiz muhasebe constructor’i calisti

Isci constructor’i calisti