Inheritance (Kalitim - Miras)

Hayvanlar

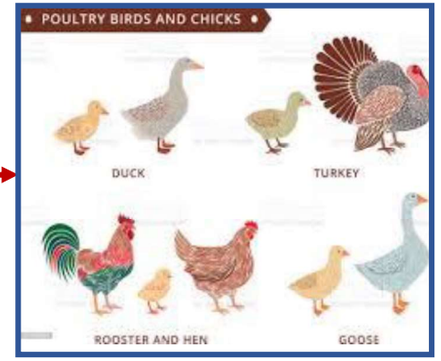
(Hareket eder, nefes alir
Beslenir, Cogalir, ölür)

Baliklar

(Denizde yasar, solungacla nefes alir,
yuzerek hareket eder)

Kuslar

(Kanatlari vardir,
akcigerle nefes alirlar,
gagalari vardir)



Kumes Hayvanlari

(ucamazlar,
yuruyerek har.ederler)

Avci Kuslar

(ucarlar, et yerler,
penceleri vardir)

Inheritance (Kalitim - Miras)

Baba

Anne

Ali Can

Mert

Merve

Kiz Cocugu

Kiz Kardes

Abla

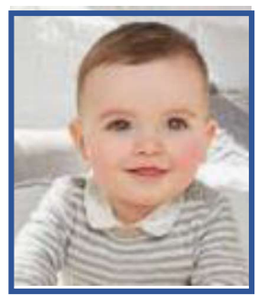
Hala

Anne

Teyze

Can

Ayse



yusuf

Inheritance (Kalitim - Miras)

IK

Personel

Kisisel bil.
Departman
Izin

Calisan

Mesai
Avans
Rapor
Std.Ucret

Muhasebe

- Isciler
  - Usta Basi
    saat ucreti 20
    Mesai : is bitene kadar
  - Surekli isciler
    saat ucreti 15
    Mesai : gunluk 8 saat
  - Gecici isciler
    saat ucreti 12
    Mesai : haftalik 25 saat
- Beyaz Yakalilar
- Yan Hizmetler
  - Disardan hizmet alimi
  - Kendi Personelimiz

Inheritance (Kalitim - Miras)

- Java'da inheritance, bir objenin/class'in baska bir objenin/class'in tüm özelliklerini ve davranışlarını elde ettiği bir mekanizmadır.
- Inheritance,OOP'lerin (Nesne Yönelimli programlama sistemi) önemli bir parçasıdır.
- Java'da Inheritance'in arkasındaki fikir, daha once'den olusturulmus Class'larin üzerine yeni Class'lar oluşturabilmemizdir.
- Inheritance sayesinde yeni olusturdugumuz bir class'in var olan bir class'in tum methodlarini ve variable'larini kullanmasini saglayabiliriz.
- Inheritance bu islemin adidir. Inheritance sayesinde child class, parent class'daki public veya protected primitive datalari, objectleri, veya metodlari problemsiz bir sekilde kullanabilir.

Inheritance (Kalitim - Miras)

Inheritance sayesinde yazılan bir code'un tekrar tekrar kullanılabilmesi (reusability) mümkün olur.

Geneli kapsayan class uyeleri parent class'a, daha spesifik olanlar ise child class'larda olusturulur.

NOT 1: Child classlar public ve protected data'lari problemsiz bir sekilde inherit edebilir.

NOT 2: Private data'lar inherit edilemez.

NOT 3: Default data'lar child ve parent ayni package'da olduklari zaman inherit edilebilirler.

NOT 4: Static Methods veya variable'lar inherit edilemezler.

Inheritance (Kalitim - Miras)

Interview Question

Nicin Inheritance kullaniriz ?

Inheritance sayesinde parent olarak tanimlanan class(ve onun parent class'larindaki) protected/public class uyelerini kullanabiliriz(reusability).

Inheritance'in faydalari nelerdir?

1: Tekrarlardan kurtuluruz

2: Daha az kod yazarak islemlerimizi yapabiliriz

3: Kolayca update yapabiliriz

4: Application'in bakimi ve surdurulmesi (maintenance) kolaylasir

Inheritance (Kalitim - Miras)

```java
public class Personel {

    public static int sayac=1000;
    public int id;
    public String isim;
    public String soyisim;
    public String adres;
    public String tel;

    public int idAtama() {
        this.id=sayac;
        sayac++;
        return id;
    }
}
```

Parent Class (Super)

```java
public class Muhasebe extends Personel {

    public int saatUcreti;
    public String statu;
    public int maas;

    public int maasHesapla() {

        int maas = saatUcreti*8*30;
        return maas;
    }
}
```

Child Class (Sub) Parent Class (Super)

```java
public class Memur extends Muhasebe{

    public static void main(String[] args) {
        Memur memur1=new Memur();
        memur1.isim="Ali";
        memur1.soyisim="Can";
        memur1.tel="5521245789";
        memur1.saatUcreti=20;
        memur1.maas=memur1.maasHesapla();
        memur1.id=memur1.idAtama();

        Memur memur2=new Memur();
        memur2.isim="Aliye";
        memur2.soyisim="Canli";
        memur2.tel="5521545789";
        memur2.saatUcreti=25;
        memur2.maas=memur2.maasHesapla();
        memur2.id=memur2.idAtama();

        System.out.println(memur1.id + " " + memur1.maas);
        System.out.println(memur2.id + " " + memur2.maas);
    }
```

Child Class (Sub)

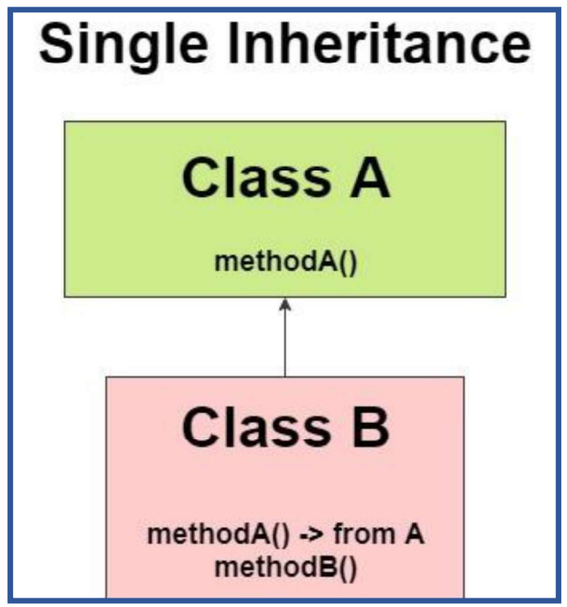
```java
public class Isci extends Muhasebe{

    public static void main(String[] args) {

        Isci isci1=new Isci();
        isci1.isim="Mehmet";
        isci1.soyisim="Bulutluoz";
        isci1.tel="5551234567";
        isci1.saatUcreti=10;
        isci1.maas=isci1.maasHesapla();
        isci1.id=isci1.idAtama();

        Isci isci2=new Isci();
        isci2.isim="Ayse";
        isci2.soyisim="Bulut";
        isci2.tel="5557654321";
        isci2.saatUcreti=15;
        isci2.maas=isci2.maasHesapla();
        isci2.id=isci2.idAtama();

        System.out.println(isci1.id + " " + isci1.maas);
        System.out.println(isci2.id + " " + isci2.maas);
```

Child Class (Sub)

Inheritance Turleri

Single Inheritance

Java Single Inheritance kabul eder. Bir child class'in sadece bir tane parent class'i olabilir .

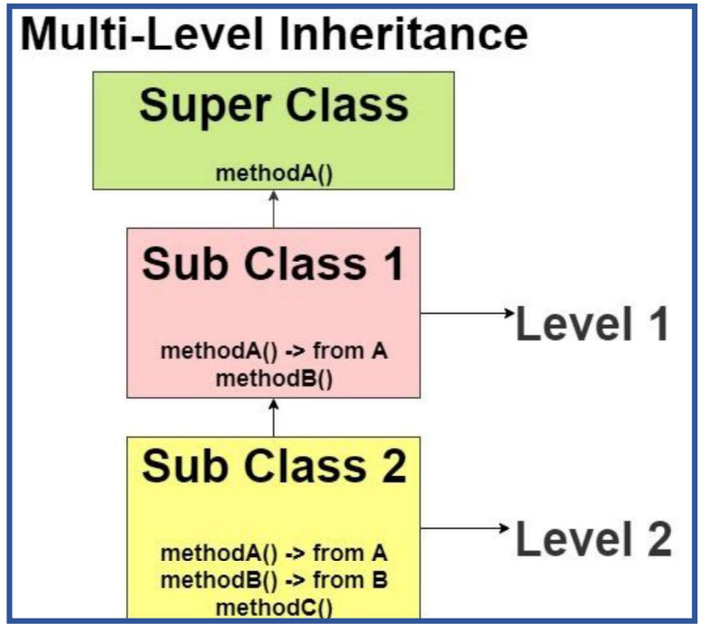


Bir cocugun ailesi bir tane olur

Inheritance Turleri

Multilevel Inheritance

Java Inheritance zincirini kabul eder. Bir child class'in sadece bir tane parent class'i olabilir (ve onun parent class zinciri).

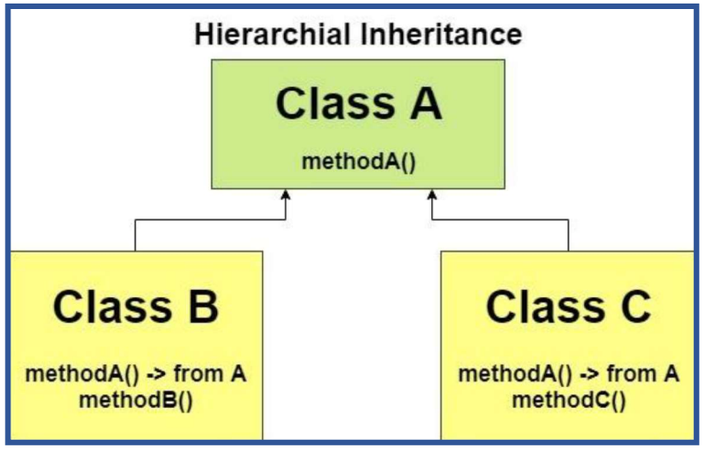


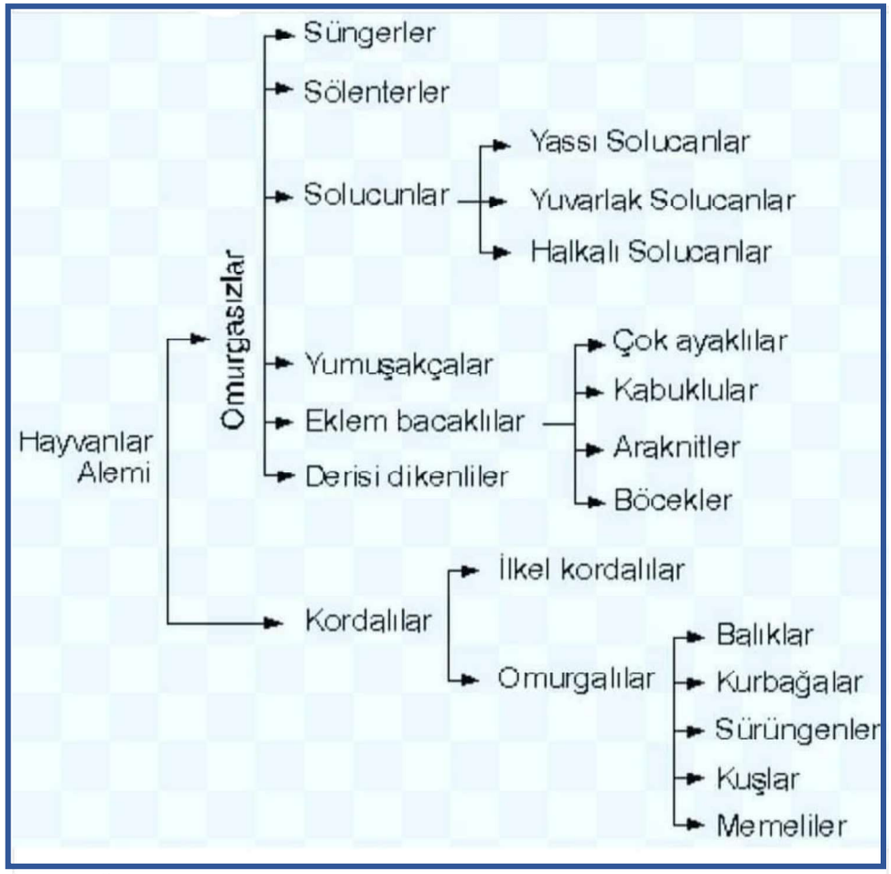
Insanlardaki soy agaci gibi, child class'in parent'i ve grand parent'leri olabilir.

Inheritance Turleri

Hierarchical Inheritance

Birden fazla class ayni class'i parent olarak kullanabilir.

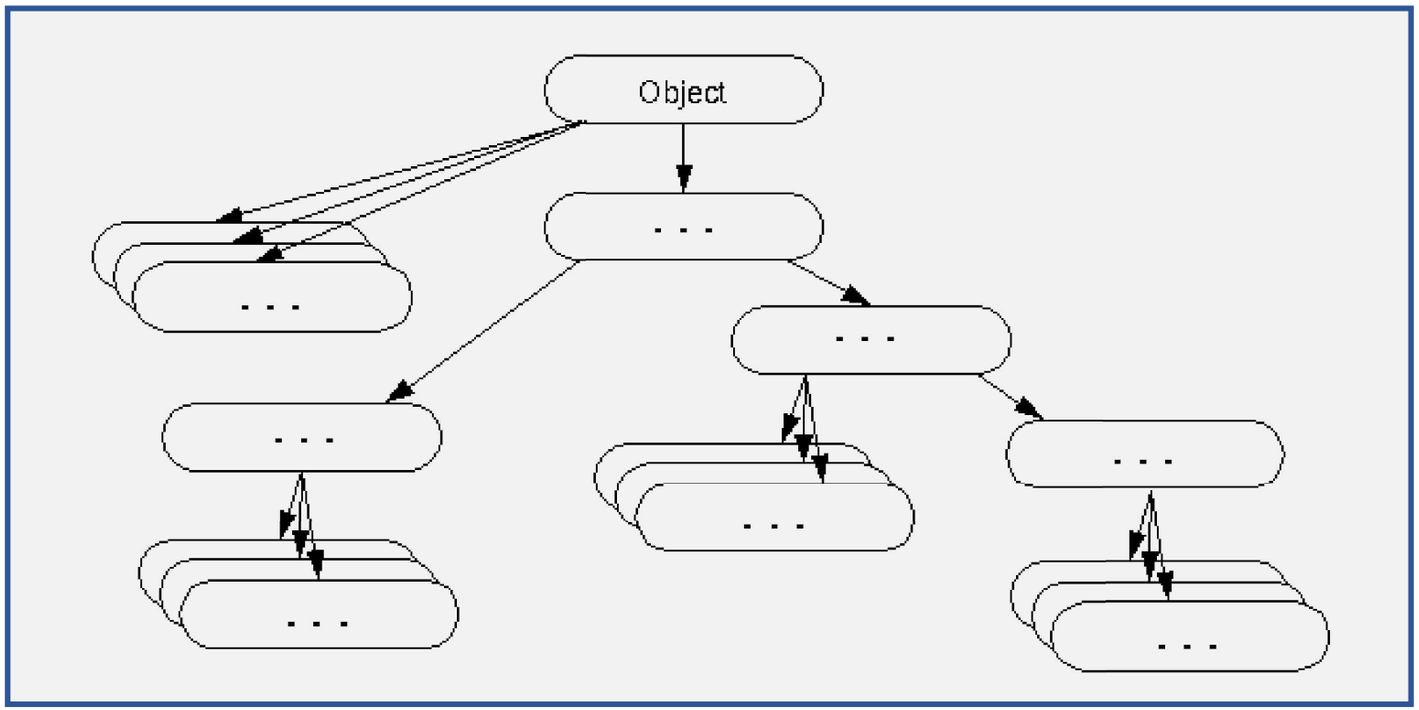




Inheritance Turleri

Java'da, butun class'lar Object Class'dan inherit ederler.

Object Class butun class'larin parent'idir ve Object Class parent'i olmayan tek class'dir.

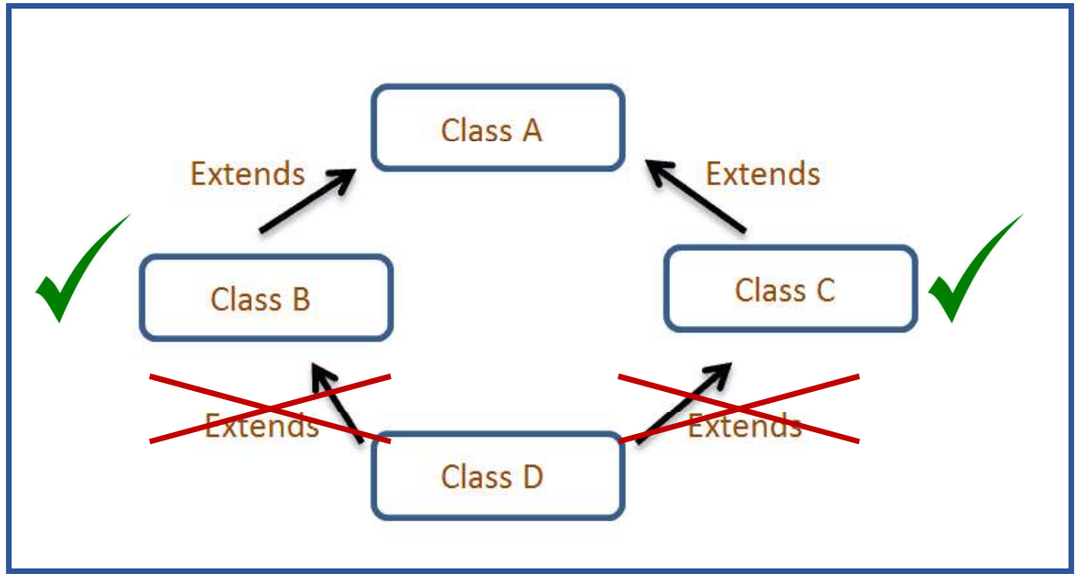


Inheritance Turleri

Multiple Inheritance

Bir class'in birden fazla class parent olarak kabul etmesi demektir,

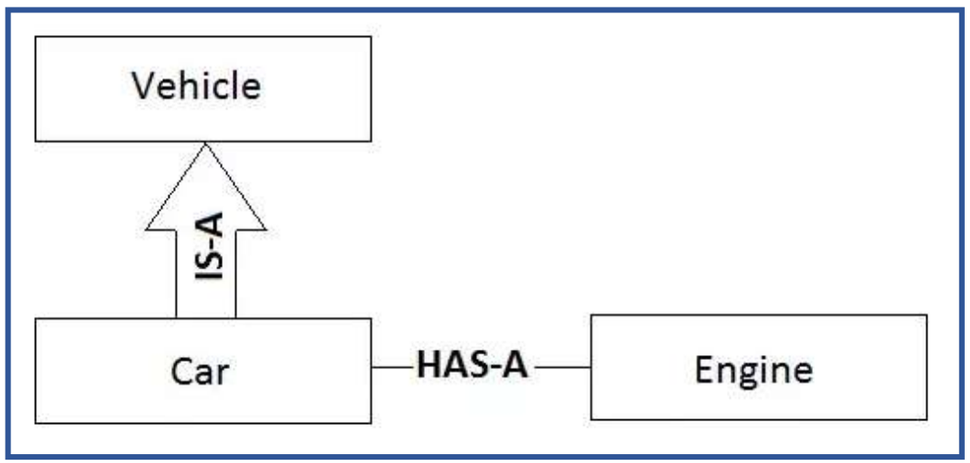
ancak Java multiple inheritance KABUL ETMEZ



IS-A ( ) & HAS-A ( ) Relationship

IS-A ilişkisini kolayca tanımlayabileceğinizi unutmamak önemli bir noktadır. Bir extends anahtar sözcüğünü gördüğünüz her yerde, bu sınıfın IS-A ilişkisine sahip olduğu söylenebilir.

(BMW IS-A Car, Car IS-A Vehicle vb..)



HAS-A iliskisi Java'da kodun yeniden kullanılabilirliği için kullanılır.

Java'da Has-A ilişkisi, basitçe, bir sınıfın bir örneğinin başka bir sınıfın bir örneğine veya aynı sınıfın başka bir örneğine başvurusu olduğu anlamına gelir.

(Apartman HAS-A daire, daire HAS-A mutfak vb..)

Inheritance'da Constructor Cagirma

1) Bir class'da constructor calistirdigimizda once parent class'daki constructor calisir. Cunku her constructor'in ilk satirinda super( ) keyword vardir(gorunmese bile).

```
public class Personel {

    Personel(){
        System.out.println("Personel constructor calisti");
    }
}
```

extends

```
public class Muhasebe extends Personel{

    Muhasebe(){
        System.out.println("Muhasebe costructor calisti");
    }

}
```

extends

```
public class Isci extends Muhasebe{
    Isci(){
        System.out.println("Isci constructor'i calisti");
    }

    public static void main(String[] args) {

        Isci isci1=new Isci();

    }

}
```

output

```
Personel constructor calisti
Muhasebe costructor calisti
Isci constructor'i calisti
```

Inheritance'da Constructor Cagirma

Asagidaki 3 class birbiriyle aynidir.

```
public class Zebra extends Hayvanlar {

}
```

```
public class Zebra extends Hayvanlar{
   Public Zebra(){
   }
}
```

```
public class Zebra extends Hayvanlar  {
   Public Zebra(){
   super();
   }
}
```

Inheritance'da Constructor Cagirma

2) Eger parent (super) class'da super() ile cagirdiginiz constructor yoksa Java Compile time Error verir.

Ornek 1 :

```
public class Muhasebe extends Personel{

    Muhasebe(){
        System.out.println("Muhasebe costructor calisti");
    }

}
```

extends

```
3  public class Isci extends Muhasebe{
4      Isci(){
5          super(5);
6          System.out.println("Isci constructor'i calisti");
7      }
8
9      public static void main(String[] args) {
10
11         Isci isci1=new Isci();
12
13     }
14
15 }
```

Inheritance'da Constructor Cagirma

Ornek 2:

```
public class Muhasebe extends Personel{

    Muhasebe(String isim){

    }
}
```

extends

```
3  public class Isci extends Muhasebe{
4      Isci(){
5
6          System.out.println("Isci constructor'i calisti");
7      }
8
9      public static void main(String[] args) {
10
11         Isci isci1=new Isci();
12
13     }
14
15 }
```

Inheritance'da Constructor Cagirma

2) super(); parent class'dan constructor cagirmak icin, this(); icinde olunan class'da baska bir constructor cagirmak icin kullanilir.

```
public class Muhasebe extends Personel{

    Muhasebe(String isim){
        System.out.println("Parametreli muhasebe constructor'i calisti");
    }

    Muhasebe(){
        this("a");
        System.out.println("Parametresiz muhasebe constructor'i calisti");
    }
}
```

extends

```
public class Isci extends Muhasebe{
    Isci(){

        System.out.println("Isci constructor'i calisti");
    }

    public static void main(String[] args) {

        Isci isci1=new Isci();

    }

}
```

Inheritance'da Constructor Cagirma

Output nedir?

```
public class Okul {
    public Okul() {
        System.out.println("Parent class cons.");
    }

}
```

```
class Sinif extends Okul {
    public Sinif(int age) {
        super();
        System.out.println("child class parametreli cons.");
    }
    public Sinif() {
        this(11);
        System.out.println("child class parametresiz cons.");
    }

    public static void main(String[] args) {
        Sinif sinif1=new Sinif();
    }
}
```

Inheritance'da Constructor Cagirma

1) Asagidaki programdaki CTE'ler nasil duzeltilir ve duzeltilip calistirildiginda konsolda ne yazdirir?

```
 6 class Derived {
 7 public Derived(String temp) {
 8     System.out.println("Derived class " + temp);
 9 }}
10
11 public class Test01 extends Derived {
12 public Test01 (String temp) {
13     System.out.println("Test class " + temp);
14
15 }
16 public static void main(String[] args) {
17     Test01 obj = new Test01();
18
19 }
20 }
21
```

Inheritance'da Constructor Cagirma

2) Asagidaki programdaki CTE'ler nasil duzeltilir ve duzeltilip calistirildiginda konsolda ne yazdirir?

```
class Derived {
public Derived(String temp) {
    System.out.println("Derived class " + temp);
}}

public class Test01 extends Derived {
public Test01 (String temp) {
    super("Hoscakal");
    System.out.println("Test class " + temp);

}
public static void main(String[] args) {
    Test01 obj = new Test01("Merhaba");

}
}
```

```
Derived class Hoscakal
Test class Merhaba
```

Inheritance'da Class Uyelerini Cagirma

- "super." keyword parent class'dan variable cagirmak icin kullanilir. "this." keyword icinde bulunulan class'dan variable cagirmak icin kullanilir.

- Esasinda "this" keyword parent class'dan variable cagirmak icin de kullanilabilir; fakat tavsiye edilmez. Cunku, child ve parent class'larda ayni isimli iki variable varsa, "this" parent class'dan variable cagiramaz.

- super( ) ve this( ) constructor cagirmak icin kullanilirlar ve constructor'in ilk satirinda olmalidirlar. Bu durumda bir constructor'da ikisinin birden olmasi mumkun degildir.

- super. ve this. variable cagirmak icin kullanilirlar. Ilk satirda olma sarti olmadigi icin ikisi birlikte kullanilabilirler.

Inheritance'da Class Uyelerini Cagirma

```java
class Class2 {
    protected int num1=10;
    protected int num2=11;
    protected String name="Ali Can";
    protected String name2="Veli Cem";

    Class2(){

        System.out.println("Parent Class constructor calisti");
    }
}
```

extends

```java
public class Deneme extends Class2{
    int num1=20;
    int num3=21;
    String name2="Hakan San";
    String name3="Kemal";
    Deneme(){
        System.out.println("Child Constructor calisti");

        System.out.println(this.num1);
        System.out.println(super.num1);
        System.out.println(this.num2);
        System.out.println(super.num2);
        System.out.println(this.num3);
        // System.out.println(super.num3);
        super.name1="Hatice Sen";
        System.out.println(this.name1);
        System.out.println(super.name1);
        this.name2="Kadir Naz";
        System.out.println(this.name2);
        System.out.println(super.name2);
        System.out.println(this.name3);
        //System.out.println(super.name3);
    }
    public static void main(String[] args) {
        Deneme deneme=new Deneme();
    } }
```

Parent Class constructor calisti
Child Constructor calisti
20
10
11
11
21
Hatice Sen
Hatice Sen
Kadir Naz
Veli Cem
Kemal

outputs

Inheritance'da Class Uyelerini Cagirma

```
public class Zebra extends Animal {
    public Zebra() {
        System.out.println("Child cons. runs at the end");
        super();
    }
}
```

Does not compile

```
public class Zebra extends Animal { public Zebra() {
        super();
        System.out.println("Child cons. runs at the end");
        }
}
```

compile

Inheritance'da Class Uyelerini Cagirma

Output nedir?

```
class Okul {
    public void getDetails() {
        System.out.println("Derived class ");
    }
}

public class Test03 extends Okul {
    public Test03() {
        System.out.println("Test class ");
        super.getDetails();
    }

    public static void main(String[] args) {
        Test03 obj = new Test03();
        obj.getDetails();
    }
}
```

```
Test class
Derived class
Derived class
```

Tekrar Sorulari

1- Inheritance'in avantajlari nelerdir ?

A) Reusability B) Maintenance C) Less Code

2- Bir Class'a Parent Class olusturmak icin Syntax nedir?

public class ChildClassIsmi extends ParentClassIsmi

3- Hangi access modifier'lar inherit edilebilir ?

public ve protected olanlar heryerden, default olanlar ayni paketten inherit edilebilir.

4- super( ) ile this( )'in farki nedir?

super( ) parent class'dan, this( ) ise icinde bulunulan class'dan constructor cagirmak icin kullanilir

5- super( ) ile super.'nin farki nedir?

super( ) parent class'dan constructor, super. ise variable veya method cagirmak icin kullanilir

6- this( ) ile this.'nin farki nedir?

this( ) constructor, this. ise clss variable veya method'u cagirmak icin kullanilir

Tekrar Sorulari

7- super. ile this.'nin farki nedir?

super parent class'dan variable veya method cagirmak icin kullanilir, this ise icinde bulunulan class'da class level variable veya method'lari cagirmak icin kullanilir.

this ile parent class'dan da variable veya method cagrilabilir ancak ayni isimde bir variable/method hem icinde bulunulan class'da hem de parent class'da olursa this parent class'da olani degil icinde bulunulan class'dakini cagirir.

Emin olmak icin parent class icin super kullaniriz.

8- super( ) ve this( ) bulunduklari constructor'da ilk sirada olmalidir. True / ~~False~~

9- super( ) ve this( ) bir constructor'da sadece 1 kere kullanilabilir. True / ~~False~~

10- super( ) ve this( ) birlikte ayni constructor'da kullanilabilir. ~~True~~ / False

Inheritance'da Data Type Kullanimi

```java
public class Personel {
    public String isim;
    public String soyisim;
    public String statu;

}
```

extends

```java
public class Isci extends Personel{
    String bolum;
    int isBasYili;

    public static void main(String[] args) {

    }
```

extends

```java
public class UstaBasi extends Isci {
    String sorumluOlduguBirim;
    int sorumluOlduguIsciSayisi;

    public static void main(String[] args) {

    }
```

UstaBasi Class'inda 3 data turu ile Usta Basi objesi olusturulabilir

```java
public static void main(String[] args) {
    UstaBasi ub1 = new UstaBasi();
    ub1.sorumluOlduguBirim="tamirhane"; // Ustabasidan
    ub1.bolum="Tamirhane"; // Isciden
    ub1.isim="Mehmet"; // Personelden

    Isci ub2=new UstaBasi();
    ub2.bolum="Atolye"; //Isciden
    ub2.statu="Isci"; //Personelden

    Personel ub3=new UstaBasi();
    ub3.soyisim="Bulut"; // Personelden

}
```

Bir obje olustururken data turunu parent(lar)'dan secebiliriz

Avantaj : Daha genis tanimlama yapilabilir

Dezavantaj : o class ve parent class'lara ait olan variable'lar kullanilabilir.

Ayni isimde iki method varsa Data Turu'ne bakilir.

Inheritance (Kalitim - Miras)

Hayvanlar

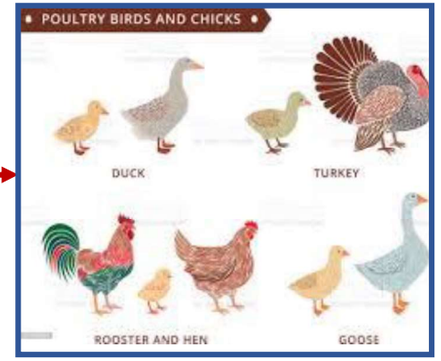
(Hareket eder, nefes alir
Beslenir, Cogalir, ölür)

Baliklar

(Denizde yasar, solungacla nefes alir,
yuzerek hareket eder)

Kuslar

(Kanatlari vardir,
akcigerle nefes alirlar,
gagalari vardir)



Kumes Hayvanlari

(ucamazlar,
yuruyerek har.ederler)

Avci Kuslar

(ucarlar, et yerler,
penceleri vardir)