**====İNHERİTANCE====**

Java da class lar arasında parent child ilişkisi vardır.

Parent class a “süper class” da denir. Child class a “sub class” da denir.

Ortak özellikler parent class a, spesifik özellikler child class a yazılır.

Parent child relationship (inheritance) faydaları:

1. Az kod yazmış olma (less code)
2. Bir class ta yaz tekrar tekrar kullan (reusability)
3. Update etmek istersek paren ta düzeltme yapmamız yeterli (maintenance)
4. Az kod yazdığımız için class larda kod sadeliği (well organization)

Java class larda “Multiple Inheritance” yi desteklemez. Yani java da bir child class ın 1 den fazla parent ı olamaz.

Bir parent class ve 1 den fazla child class ile oluşturulan çok katlı parent child relationship e “Hierarchical Inheritance” denir.

Child-Parent-Grandparent… şeklinde oluşturulan çok katlı relationship e “Multi-level Inheritance” denir.

Object class bütün class ların parent class ıdır. Java da parent class ı olmayan tek class Object dir.

Java da parent child ilişkisi kurmak için “child class isminden sonra extends parent class ismi”

public class X (parent)

public class Y extens X (Y, X nın çocuğu)

şeklinde yazılır ve X HAS A Y (X, Y ye sahiptir.) ya da Y İS A X (Y, X tir) diye söylenir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **Aynı class** | **Aynı paket** | **Paket dışından sadece çocuklar** | **Paket dışı** |
| **Private** | Yes | No | No | No |
| **Default** | Yes | Yes | No | No |
| **Protected** | Yes | Yes | Yes | No |
| **Public** | Yes | Yes | Yes | Yes |

Parent ların class larına veri paylaşabilmeleri için yukarıdaki Access modifier kuralları geçerlidir.

Parent class lar child larına ihtiyaçları olduğunda veri paylaşımı yapıyorlar, bunun karşılığında da sen bir constructor inşaa edeceksen ve sana bir yönlendirme (şu metoda git, şu variable yı al ) yapılmadığı sürece, önce tek tek çıkarak en üste (mesela dedeye) gelip, burada denileni yapıp, geldiğin yoldan denenleri yaparak geri dönmelisin diyor.

public Insanlar () {syso ”insanlar”}

public Amerikan extens Insanlar () {syso “Amerikan”}

public Hollywood extens Amerikan () {syso “Hollywood”}

Hollywood obj = new Hollywood();

class ları ve child class taki obje olduğunda run ettiğimizde, objenin altında bir yönlendirme olmadığı için (gizli super lerle) adım adım dedeye çıkılır, orada istenilen yazılıp babaya, orada da istenen yazılıp en son child a gelinir ve artık oradaki yazılır.

super();----- > parent taki parametresiz metoda git demek

super. ----- > noktadan sonra variable yazıldı diyelim, paren taki o variable a git demek

this(); ----- >bulunduğun class taki parametresiz metoda git demek

this. ----- > bulunduğun class taki variable a git demek

**yönlendirme yapıldığında= gizli super lere gerek yok**

1. **class** A
2. {
3. **int** i;
4. **void** display()
5. {
6. System.out.println(i);
7. }
8. }
9. **class** B **extends** A
10. {
11. **int** j;
12. **void** display()
13. {
14. System.out.println(j);
15. }
16. }
17. **class** Runner
18. {
19. **public** **static** **void** main(String args[])
20. {
21. B obj = **new** B();
22. // B obj = new A(); 2. durum
23. Soutv (obj.i); // variable a git
24. obj.j=2;
25. obj.display(); //metoda git
26. }

Child class tan child class constructor la bir obje oluşturulmuş, altında “i” variable ına git yazdır diyor. Variable yönlendirmesi yapıldığı zaman objenin Data Type ına bakılır, soruda “B” diyor, o zaman B class ındaki “i” variable ına gidilir (B class ında i yok o yüzden babasına gidiyor ve 0 default değerini alıyor alt satırda yazdır dediği için 0 yazdırıyor ).

Objenin bir altında display(); metoduna git diyor, metod yönlendirmesi yapıldığı zaman objedeki constructor a bakılır, soruda “B” diyor, B class ında display metodu var ve “J” yi yazdır diyor (display metoduna gitmeden önce j ye 2 atanmış, o yüzden 2 yazdırır). Eğer 2. Durumdaki gibi constuctor “A” olsaydı, display metodunu A class ında arardık .

Run class ta sout(super.i) gibi bir ifade görmek, parent taki “i” yi yazdır, super.display(); gibi bir ifade görürsekte parent taki display(); metoduna git demektir. Eğer this.display(); görürsek bu class taki display metodu, this.i görürsekte bu class taki i variable ı demektir. Bu arada child class aradığını kendinde bulamayıp babaya gider ve orada da bulamazsa CTE verir.