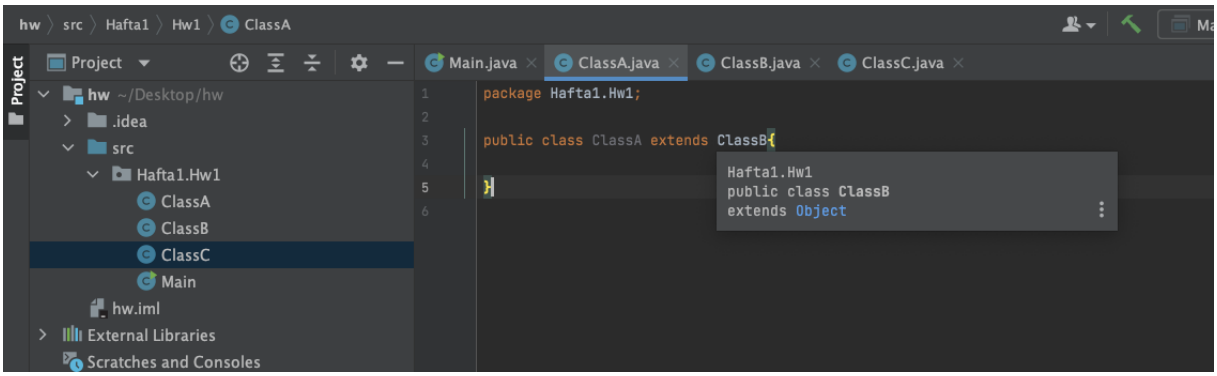
ÇOKLU KALITIM- MULTİPLE İNHERİTANCE

Nesneye yönelik programlama dillerinde kalıtım süper class'ın bir veya daha fazla sub class'a sahip olup özelliklerini alt sınıfa aktarmasıyla gerçekleşir. Böylece alt sınıflar, üst sınıftan miras alma yoluyla süper sınıfın tüm özellik ve metodlarını alır. Bu sayede programımızda yeniden kullanılabilirlik ve anlamlı bir hiyerarşik yapı oluşur.

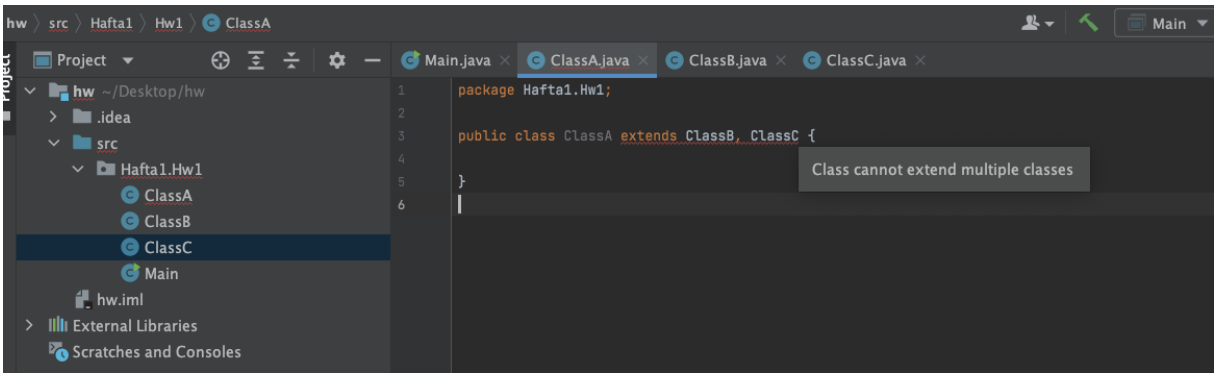
Java'da Çoklu Kalıtım:

Java çoklu kalıtımı desteklemiyor. Bunun nedeni karmaşıklığı azaltmak ve ve dili sadeleştirmektir. Aynı zamanda derleme hatalarından dolayı çoklu kalıtım desteklenmiyor. Çoklu kalıtım bir sınıfa birden fazla sınıf davranışı eklenmesiyle Single Responsiblity ihlal edebilir.

Aşağıdaki örnekte ClassB 'nin ClassA 'e miras verdiği görülmektedir. ClassA böylece ClassB özelliklerini kullanabilir.



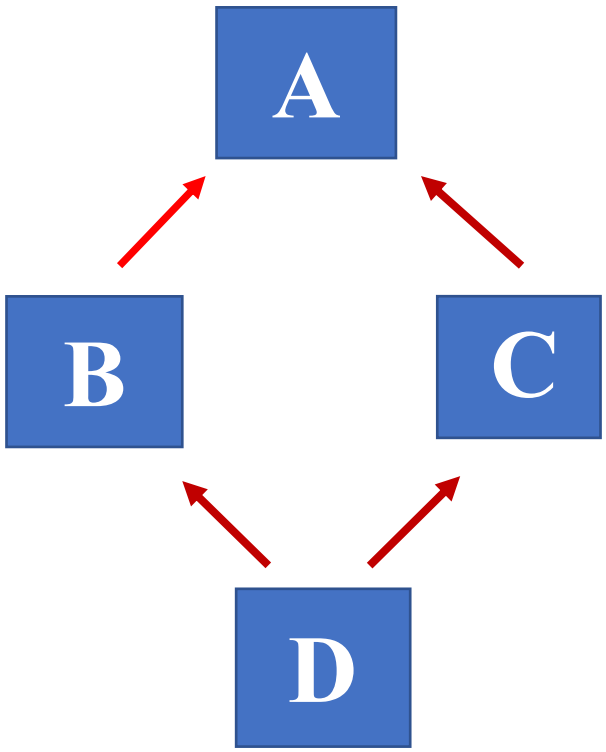
Eğer Multiple Inheritance yapmak istersek ClassA, ClassB ile beraber bir class'dan ClassC daha miras almak isteyince compile error ile ile karşılaşıyoruz.Ekranda gösterilen hata:Class cannot extend multiple classes.



ClassA ‘e birden fazla class’dan miras vermemiz gerekiyorsa bu işlemi interfaceler aracılığıyla yapabiliriz.

Çoklu kalıtımda, süper classta ad çakışmaları ve belirsizlik sorunları ortaya çıkıyor. Aynı ada sahip methodlar içeren üst sınıfla karşılaşıldığında hangi metoda erişebileceğini veya çağıracağını derleyiciler bilemiyor. Bu sebeble interfaceler aracılığla Java’da çoklu kalıtımı destekleyebiliriz.

Java’da çoklu kalıtım yapılmak istendiğinde diamond problemi oluşuyor.



Diamond Problem Şeması

Yukarıdaki şekilde B ve C sınıfı A sınfından kalıtım almış, D sınıfıda B ve C sınıfından kalıtım almıştır. B ve C sınıfı A sınıfındaki bir metodu override edip kullanabilir ve D sınıfıda A sınıfından aynı methodu override etmeden kullanmaya çalışırsa ortaya çıkan karışılığa diamond problem denir.

Çoklu kalıtım gereken yerlerde ve DRY (Don’t Repeat Yourself) prensibini kullanarak Java çoklu kalıtımı interfaceler üzerinden yapabiliriz. Interfaceler tip tanımlamak için kullanılır. İçerisinde default ve static metodlar barındıran fakat metod implementasyonları ve state bulunmayan yapılardır.Birden fazla class tarafından implement edilebilirler ve birden fazla interface extend edebilirler. Çoklu kalıtım gereken yerlerde interface ya da encapsulation kullanarak sorunumuzu çözebiliriz.Interfaceler sayesinde kod üzerinde okunabilirlik ve esneklik sağlanır.

İnterfaceler için küçük bir örnek:

Çalışan sınıfımız;

```
package Hafta1.Hw1;


public interface Calisan {
    public boolean calis(int calisanId);

}
```

Alt sınıflar interface'den kalıtım alırlar.

```
package Hafta1.Hw1;

public class Akademisyen implements Calisan{
    @Override
    public boolean calis(int calisanId) {
        return false;
    }

}
```

Çoklu kalıtımı destekleyen diller Python, C ++ , Perl, Lisp gibi diller desteklemeyen diller ise C#, Java vb.

Kaynakça:

- https://docs.oracle.com/javase/tutorial/java/IandI/multipleinheritance.html
- https://beginnersbook.com/2013/05/java-multiple-inheritance/
- https://denizirgin.com/yaz%C4%B1l%C4%B1m-dillerinde-%C3%A7oklu-kal%C4%B1t%C4%B1m-ve-tekli-kal%C4%B1t%C4%B1m-e952680db24e