→ ABSTRACTION ←

(SOYUTLAMA)

* OOP'nin en temel prensiplerinden biridir.

* Abstraction yapıldığında detayları görmezden gelip detaylara odaklanmış oluruz.

Mesela, bir öğretmen bir öğrenciye "ders çalış diyebilir" veya "okuduğunun özetini çıkar, notlar tut, renkli kalemler kullan" vb diyebilir.

Detayları konuşursak Abstraction yapmamış oluruz. Sadece yapılması gereken şeyi söylüyorsanız abstraction yapıyorsunuz demektir.

Yani öğretmen'in öğrenciye sadece ders çalış demesi Abstraction yaptığı anlamına gelir. Detaylarıda konuşursa Abstraction yapmadığı anlamına gelir.

↓

Yani Abstraction, Programcının neye odaklanmasını sağlar nasıldan çok

↳ Öğrenci ders çalışmalı, peki nasıl çalışmalı?

O konuyu öğrenciye bırakıyoruz. Yani Abstraction yapmış oluyoruz.

Java'da Abstraction iki farklı şekilde olabiliyor;

1) Abstract classes

2) Interfaces

1) Abstract Classlar :

* Abstract Classlardan obje üretilemez. (Abstract classlar içinde Abstract yani body'si olmayan methodları barındırır. Nesneye yönelik programlamada body'si olmayan methodları yarım veya eksik olarak nitelendirebiliriz. Nesneye yönelik programlamanın temel gayesi mükemmel objeler üretmek olduğu için Abstract Classlarla obje üretmenize müsade etmez.

Kısacası Java'da nesnelerin tam olması ve eksiksiz çalışması gerekir ancak bu durum Abstract Classlar için geçerli değil.

* Abstract methodlarda Abstract Methodlarda olabilir Concrete Methodlarda olabilir. İkisini de içerebilir.
(Ama tabii ki abstract methodlar sadece Abstract Methodlar içine konulabilir, normal classın içine koyarsanız tabii ki hatayla karşılaşırsınız :D)

* Abstract classlarda oluşturduğunuz abstract methodun mutlaka bir yerlerde tamamlanması gerekiyor. Yoksa niye abstract method yazalım :D), bu abstract methodlar concrete childlarda tamamlanır. Bu sebeple body'si olmayan methodlar (abstract methodlar) tamamlanır ve concrete hale getirilir.

* Abstract classın içinde variableler olabilir, Constructorlar olabilir ve normal (regular) methodlar olabilir.

↳ Peki bu abstract ~~method~~ classların içindeki normal (riguar) methodlar nedir? Bütün childların yapabileceği ortak işleri belirler.

* Bir abstract classta bodysi dolu bir method (concrete method) diyebiliriz. Mesela, Matematik dersi.

↳ Bu dersi sadece, isteyen öğrenciler alabilir.

Ancak bir de abstract bir method diyebiliriz, Mesela Resim dersi, bu dersi classı miras alan her öğrenci almak zorundadır.

* Abstract Class oluşturulan class kalıtımsın önüne Abstract yazıyoruz.

* Abstract Classlardan obje üretemeyiz.

↳ Bundan Abstract Classların Constructorı yoktur gibi bir sonuç çıkartılmasın. Javada her classın constructorı mutlaka vardır, ne Abstract classlarda bir classtır.

* Abstract Classın içinde Final method yazılabilir. (Subclassın bahsi geçen methodu değiştirememesi için)

~~2) Inter~~ 2) Interfaces :

* Java'da Interfacelerin içindeki tüm methodlar abstracttır denilebilir.

↳ Ancak bir istisnası var. Java, Developerlara Interfaceler içinede bodysi olan Methodlar koyma hakkı vermiş. Ancak bu bir istisnadır.

↳ "Static" keywordu kullanarak Interfacelerin içine bodysi olan method koyabilirsiniz.

↳ "default" keywordu kullanarak Interfacelerin içine bodysi olan method koyabilirsiniz.

* Interface'den obje üretilemez. (Interfacelerin içinde bodysi olmayan yani eksik ve yarım methodlar var, eğer bir tamamlanmamış bir ifadeden obje üretmeye kalkarsak o obje de eksik olur. Tabii ki Java buna izin vermez.)

* Interface bir class olmadığı için tabii ki içinde bir constructor bulunmaz.

Java'da Abstract Classların içine Abstract Method zaten koyabiliyorduk, Java neden Interfaceleri tanımlama gereği duymuş?

Java'da multiple inheritance mümkün değil, yani bir classın birden fazla parenti olamaz. Ama biz bazen bir class için birden fazla parenta ihtiyaç duyarız. İşte Java interfacelere bu yüzden ihtiyaç duymuştur.

Bir childa birden fazla classı parent yapamazsınız ama bir childa birden fazla interface'i parent yapabilirsiniz.

↳ Mesela bisınıf öğrencileri bir child class, bunların Öğretmen diye bir parenta ihtiyacı var, Dersler diye bir parenta ihtiyacı var, Aileler diye bir parenta ihtiyacı var.

* Interfacelerin içindeki methodlar kesinlikle public ve abstracttır. Interfacelerin içindeki methodlar kesinlikle publictir ve kesinlikle abstracttır.

↳ Eğer Java bir şeyi biliyorsa sizin onu söylemenize gerek yok. Java biliyor ya interface'in içindeki methodlar public o yüzden siz method oluştururken public diye yazmaya gerek yok. Abstract diye yazmaya da gerek yok. Java bunu zaten farkında.

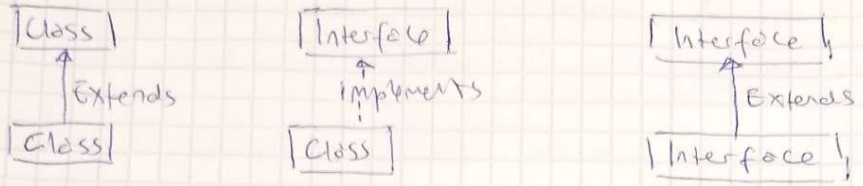
~~Zaten Methodun~~

* Bir interface başka bir interface'in childı olabilir. *

Doğal olarak ilk değer atanması yapmıyoruz zorundayız ve o değeri değiştiremeyiz.

* Interfacedeki variableler Finaldır. Eğer ilk değer atamazsak ve variable'ın değerini değiştirmek istersek ne oluyor?

↳ Hata alıyoruz :D !



Abstract Class ve Interfaceler Arasındaki farklar

* Abstract Classlar ve Interface Java'da ABSTRACTION için kullanılır. *

→ Abstract classlarda hem abstract hem concrete method olur ama interfacelerde sadece abstract methodlar olur.
NOT: Java interfacelerin içine "static" ve "default" keywordler kullanarak bodysi olan methodlar koyduğumuzda mümkün eder.

→ Abstract classın içinde her türlü method olabilir, Ama interfacelerin içindeki variableler

* public * static * final

olmak zorundadır.

→ Abstract class bir classtır oyüzden bir child'ın bir tane abstract class parenti olabilir. Çünkü Java multiple inheritance'ı desteklemez. Ama interfacelerde bir childın birden fazla parent koyabiliriz. (Zaten Java'nın Interfaceleri oluşturma sebebi bu.)

→ Abstract classlar bir çok access modifier kullanabilir. Ancak Interfaceler kesinlikle publicdir ve fieldlar public, static ve final olmak zorundadır.

→ Abstract classlar kullanılması isteğe bağlı olan ve kullanılması zorunlu olan methodları içerirler. Eğer bu tarz bir parent gerekiyorsa Abstract class kullanımı düşünülebilir ancak bu durumda child classın sadece bir tane parente sahip olabileceği akıldan çıkartılmamalı.

Interfaceler Multiple Inheritance'a ihtiyaç duyduğumuzda devreye giriyor. Ancak isteğe bağlı method koyamıyoruz.