**EV-RENK PROJESI:**

Bir ev Class ı oluşturalım, renk field ı olsun. Constructor ını, getter ve setter ını koyalım:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Main deyiz: evin constructor ını yazıyoruz. Alt satırda ise, baskaEv nesnesi oluşturur gibi yapıyoruz ancak eşitliğin karşı tarafına üstte oluşturduğumuz ilk nesneyi yazıp, baskaEv i maviEv e atıyoruz. Sonra da her iki nesnenin rengini de getter la çağırıyoruz, hangi renkler dönecektir?

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Tabii ki mavi zira *baskaEv* *maviEv* i referans alıyor.

Şimdi *baskaEv* e sarı rengi set edelim ve yazdıralım. İlginç bir şekilde her iki ev de renk olarak sarı döndü:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Adı maviEv olsa da referans ortak olduğundan baskaEv e sarı rengi set etmemiz, maviEv i de sarı yapıyor.

Şimdi de yesilEv olusturalim: Main de constructor yaparak nesne oluşturalım adı yesilEv olsun.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi de baskaEv i yesilEv olarak atayalim.

metin içeren bir resim

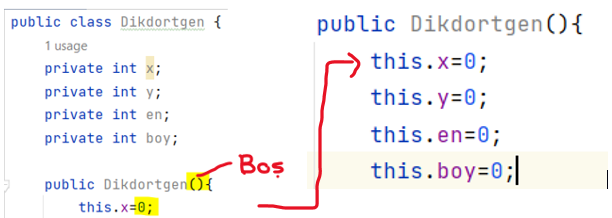
Açıklama otomatik olarak oluşturuldu

Öyleyse, bir evi hangi eve atıyorsak, öncelik o evin sahip olduğu değerdir. Yani eşitliğin karşısındaki referans alınan nesnedir ve onun değeri eşitliğin solundakine atanır.

**DİKDÖRTGEN ÖRNEĞİ:**

Bu örnekteki amacımız constructor lar düzgün oluşturulabilmiş mi, daha düzgün nasıl oluşturulabilir, bunları araştıracağız.

Dikdörtgen Class ı oluşturuyoruz. Dikkat: Constructor da parametre oluşturmuyoruz, boş bırakıyoruz. Öyleyse süslü parantez içinde this ile atamalar yaparken, değerleri kafama göre karşılarına yazabilirim. Örneğin şu an hepsine 0 atıyoruz:



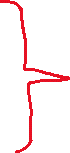
Şimdi iki tane daha constructor daha oluşturacağız. Toplam üç constructor düzgün oluşturulmuş mu?

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin içeren bir resim

Açıklama otomatik olarak oluşturulduŞimdi burada tekrarların olduğu açık, azaltmamız gerekiyor. Şayet, başka bir constructor ilgili field lara zaten atama yapıyorsa, daha fazla parametreye sahip olan da oysa, işi ona yaptırmamız gerekir. Mesela ilk i hiçbir parametre almıyor, üçüncünün olmadığını düşünelim. İkinci de zaten en ve boy, en ve boy a atanmış. Öyleyse birinci costructor da şunu yazardım (ikinciye bakarak);



metin içeren bir resim

Açıklama otomatik olarak oluşturuldusadeleşti

This anahtar kelimesi sadece field lara atama yapmak için değil aynı Class içindeki constructorları çağırmak için de kullanılıyordu. İlk constructor ı bu şekilde sadeleştirdik.

Şimdi ikinciyi üçüncüyle etkileşerek sadeleştirelim:

İlk iki field a 0 değerleri verilmiş zaten aynı this kullanımıyla otomatik atama yaparız. Alttaki en ve boy satırları için ise zaten parametrelerle atama yapılmış parantez içinde. Öyleyse, x ve y için 0 değeri girersem 0 değeri atamış olacağım, en ve boy da değişen bir durum yok. Böylece durumu üçüncü ye yani *bütün field lara atama yapan constructor a* (no 3) bu şekilde havale edeceğim. Öyleyse:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ŞEKİL ÖRNEĞİ:**

Şekil Class ı oluşturuyoruz, iki field ı var: x ve y:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi Sekil Class ından Dikdortgen isimli bir Class türeteceğiz:en ve boy parametreleri var, constructor ını oluşturuyoruz:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Dikkat: generate constructor deyince yukarıda görüldüğü üzere, super class ı belirliyor.

Şimdi aynı constructor ı klonlayıp farklı işlevde bir constructor üretelim:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada şöyle bir durum var: tüm fieldlara atama yapan constructor, super anahtar ı kullanmakla, x ve y değerlerini super class tan alıyor yani bir method yazıp çalıştırdığımızda Şekil class ının sayfasına gidiyor. Ancak ilk constructor ımızda parametre olarak x ve y belirlenmiş, sonrasında da this kelimesiyle oluşturulan ifadede x ve y tekrar düzenlenmiş. Öyleyse super class ın sayfasına gidilmeden x ve y için işlem yapılabilecek. En ve boy a da sabit değerler verilmiş. Bu şekilde de costructor kullanımı olabilir.

**HESAP MAKİNESİ 2:**

**STATIC METOTLARLA INSTANCE METOTLAR ARASINDAKİ FARKLAR**

Metotlar genel olarak 2’ye ayrılmaktadır:

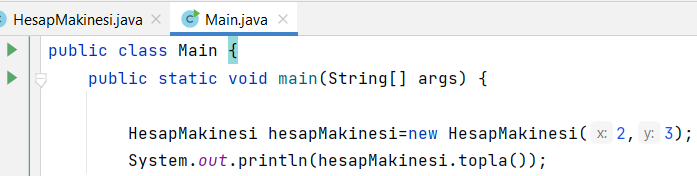
1. Instance metotlar: Nesneye referans verilen metotlar. (instance örnek demek). Nesne oluşturarak oluşturulan tüm metotlar instance. Örnek Halı. Taban isimli class a alanHesapla diye bir metot yazmıştık. Önce taban nesnesini ürettik, bu nesneyi oluşturmasaydık o metodu kullanamazdık. Nesne oluşturarak kullanılan metotlar.
2. Statik metotlar: Herhangi bir nesne üretmek zorunda olunmayan metotlar.

* Bir HesapMakinesi sınıfı oluşturalım, iki tane field ı olsun: x ve y.
* Constructor, getter ve setter oluşturalım.
* Topla isminde bir metodu olsun.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şayet, topla isimli metodu kullanmak istiyorsam Main de nesne oluşturmak zorundayım:



*topla()* metodu, hesapMakinesi nesnesine ait bir metot. Dolayısıyla bir instance metodu.

Şimdi Main de static bir metot oluşturalım:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Üstteki metot Instance iken, alttaki metod; üstte oluşturduğumuz nesneye ihtiyacı olmadan işlemi yapabildiği için bir statik metoddur.

Şimdi bir başka örnek yapalım. Topla isimli bir class oluşturalım:Bunun da toplam isimli bir metodu olsun:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi Main e gelelim ve Topla class ındaki topla metodunu çağıracağız: Metodu topla class ında oluşturduğumuz için burada yeni bir metot yazmadan sadece sout içinde çağırıyoruz:

metin içeren bir resim

Açıklama otomatik olarak oluşturulduBaşka bir class ta oluşturduğumuz metodu nesne oluşturmadan çağırabildiğim için bu metot da static metottur.

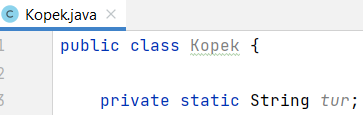
**STATIC VARIABLE:**

Bir de *statik variable* lar var yani statik değişkenler, şimdi ona bir örnek verelim: Kopek isimli bir class oluşturup içine *tur* field ı koyalım:

metin içeren bir resim

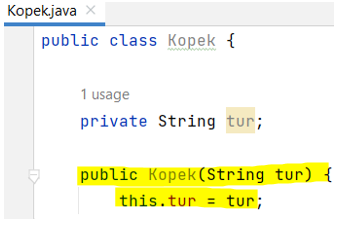
Açıklama otomatik olarak oluşturuldu

Şimdi public ten sonra static yazarak bu değişkeni statik hale getirelim: static anahtar kelimesi koyunca ne olur?



***ÇOK ÖNEMLİ:*** ***Bir class a ait her bir nesne (instance), aynı statik değişkeni paylaşır. Eğer o değişkende bir değişiklik meydana gelirse, bütün instance lar yani bütün nesneler o değişikliği görür.***

Bu statik değişkenler çok kullanılmıyor ama bazen faydalı olabiliyorlar.

Şimdi constructor oluşturalım. Normalde static yazmasaydık şu şekilde oluşacaktı: 

Ancak static anahtar kelimesini kullanırsak bu şekilde constructor formu oluşuyor:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Böyle durumlarda süslü parantez içine this yazmıyoruz, class ın adını yazıyoruz: *Kopek.tur=tur;*

Ayrıca sout yapıp türü çağıralım:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Şimdi main e gelelim: Burada iki tane kopek nesnesi oluşturup, parametrelerini kangal ve karabaş olarak yazıyoruz:

Ve bunlar için turNe metodunu ayrı ayrı çağırıyoruz: Sonuç ilginç, iki ayrı köpek türü dönmedi, son yazılan karabaş olduğu için ekrana her iki çağırmada da karabaş ismi yansıdı. İkinciyi yazmasaydık tek satır tek metot olsaydı bir tane “kangal” dönütü görecektik.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

TAM BİR SINAV SORUSU(!)

Zira, ***Bir class a ait her bir nesne (instance), aynı statik değişkeni paylaşır. Eğer o değişkende bir değişiklik meydana gelirse, bütün instance lar yani bütün nesneler o değişikliği görür.***

Class da fiel dı static olarak belirlemeseydik, this anahtar kelimesini kullanarak atasaydık, bu durumda field static olmadığı için Main de yazdığımız iki metot da ayrı değerler döndürecek, birincisi kangal, ikincisi karabaş dönecekti.