Değer ve Referans Tipleri

https://medium.com/@m.baybars.keskin.0/de%C4%9Fer-ve-referans-tipleri-53a35f5ae9e

Herkese merhabalar bu yazımızda yazılım dünyasında karşımıza çıkan değer ve referans tiplerini inceleyeceğiz ve aralarında farkları aktarmaya çalışacağız. Dilerseniz Stack ve Heap kavramlarından kısaca bahsederek başlayalım, bu kısımlar RAM (Random Access Memory)'in mantıksal bölümleridir diyebiliriz. Stack'de değer tipleri, pointer ve adresler saklanırken, Heap'de ise referans değerleri saklanmaktadır.

Stack'e erişim Heap'den daha hızlıdır ve Stack, LIFO (Last-In-First-Out) mantığında çalışmaktadır. Yani son gelen ilk olarak çıkar. Bu sebep ile aradan herhangi bir eleman çıkartamazsınız, birbirleri ile ilişki içerisindedirler.

Struct tipindeki değişkenler değer tipleridir ve Stack içerisinde saklanmaktadır. Class tipindeki değişkenler ise referans tipleridir ve referansları Stack'de kendisi ise Heap'de saklanır.

Örnek bir kod parçası, CustomerManager adlı bir sınıfımız olduğunu düşünelim ve bu sınıfın üç ayrı metodu vardır add, remove ve update. Bu sınıftan iki nesne oluşturularak gerekli kavramlar bu örnek üzerinden aktarılmaya çalışılacaktır.

STACK		HEAP
CustomerManager customerManager1	=	new CustomerManager();
CustomerManager customerManager2	=	new CustomerManager();

Referans Tip (Reference Type)

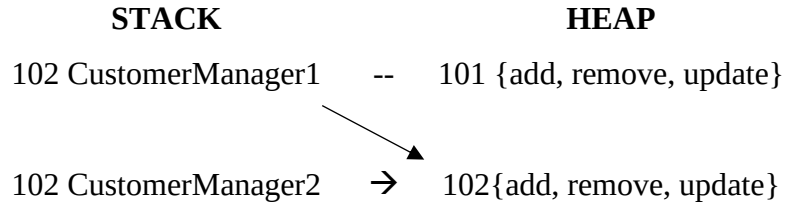
STACK		HEAP
101 CustomerManager1	→	101{add, remove, update}
102 CustomerManager2	→	102 {add, remove, update}

101 numaralı adresin referansı, ok ile gösterilen kısımdır.

102 numaralı adresin referansı, ok ile gösterilen kısımdır.

customerManager1 = customerManager2;

Bu şekilde bir tanımlama şu anlama gelir: customerManager1'in referans numarası, customerManager2'nin referans numarasına eşittir. Yani bu durumda siz customerManager1'i çağırdığınız takdirde yukarıdaki gibi 101 numaralı adresteki öğeler değil de 102 numaralı adresteki öğeler çağrılacaktır çünkü customerManager1'in numarası da 102 olmuştur ve adresi oraya aittir.



CustomerManager customerManager1 = new CustomerManager();

Artık bu ifade anlamını yitirmiş olur çünkü referansı farklı bir yere bağlandı. Bunu birazdan **Garbage Collector (çöp toplayıcı)** Java'nın bellek yöneticisi onu bellekten silecektir. Bir Class'ı oluşturan programcının, onu oluştururken dikkat etmesi gerekmektedir çünkü **new** pahalıdır. Bellekte bir nesneyi new'lemek pahalıdır, o yüzden sadece ihtiyaç duyduğumuz zaman new'lemeliyiz.

Değer Tipi (Value Type)

Primitive types: int, double, float, byte bu sayısal değikenler değer tipleridir. Değer tipini dilerseniz basit bir örnekle açıklamasını yapalım.

Örnek:

int sayi1 = 10; sayi1'in değerini 10 olarak tanımladık.

int sayi2 = 20; sayi2'nin değerini 20 olarak tanımladık.

sayi2 = sayi1; Bu ifade sayi2'nin değeri eşittir sayi1'in değeri anlamına gelir, yani sayi2 artık sayi1'in değerine sahiptir bu bağlamda sayi2 10 değerine sahip oldu.

sayi1 = 30; sayi1'in değerini 30 olarak değiştirdik.

sayi1 ve sayi2 referans tip olmadıkları için, sayi1'in değeri değişse de sayi2'nin değer sabit kalacaktır. Değer tiplerinde bir referans yoktur değişkenler direk tanımlanan öğeyi benimser. Örneğimize gelecek olarak sonuçta sayi2'ni değeri 10 olarak kalacaktır, sayi1 ile bir referans ilişiği bulunmadığından ötürü sayi1'in değeri daha sonra değiştirilmiş olsa da sayi2'i sayi1'in sadece atanan ilk değerini sahiplenmiştir.

STACK	**HEAP**
sayi1 = 10 → 20	
sayi2 = 20 → 30	

Referans tip olmadıkları için Heap bölgesinde herhangi bir şey gerçekleşmemektedir.

Örnek:

Sırdaki örneğimiz ise diziler ile ilgilidir, diziler ne kadar değer tipmiş gibi gözükse de aslıdan bir referans tiptir, başında int yazması sizi lütfen yanıltmasın.

```
int[] sayilar1 =  new int[] {1, 2, 3};
int[] sayilar2 =  new int[] {4, 5, 6};
sayilar2 = sayilar1; sayilar2'nin referans numarası eşittir sayilar1'in referans numarasıdır.
sayilar1[0] = 10;
```

Bu durumda bellekte tek nesne vardır ve iki değişken aynı nesneyi tutmaktadır.

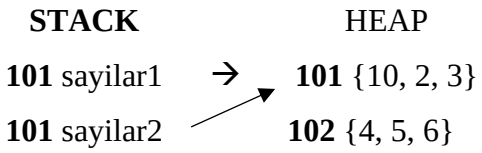
STACK		HEAP
101 sayilar1	→	**101** {1, 2, 3}
102 sayilar2	→	**102** {4, 5, 6}

```
sayilar2 = sayilar1;
```

STACK		HEAP
101 sayilar1	→	**101** {1, 2, 3}
101 sayilar2	↗	**102** {4, 5, 6}

Artık sayilar2 de sayilar1'in karşılığını tutacaktır çünkü referans numaraları eşitlendi.

sayilar1[0] = 10 Bu ifade sayilar1 dizisiniz sıfırıncı index değerine sahip olan değeri 10 olarak değiştirir, o halde tablo şekildeki gibi olacaktır:



sayilar1 de sayilar2 de heap'teki 101 referans nolu {10, 2, 3} tuttuklarından, sayilar2[0] da 10'a eşit olacaktır.

Önemli Not: Bu yazıdaki örnekler ve anlatımlar **Engin Demiroğ'a** aittir, vermiş olduğu eğitimler ile biz öğrencilere tüm katkılarından dolayı minnettarız. Dilerseniz https://www.youtube.com/watch?v=y9vmseRAMiU adresindeki videodan, öğretmenin kendi anlatımı ile konuyu dinleyebilir ve pekiştirebilirsiniz.