C#'taki Yapı ve Sınıf Nesneleri Nerede Yaratılıyor?

26-Şubat-2010

Pek çok *C#* programcısının sınıf ve yapı kavramlarıyla *stack* ve *heap* kavramlarını yanlış bir biçimde ilişkilendirdiğini görüyorum. Örneğin, "yapı nesneleri *stack*'te sınıf nesneleri *heap*'te tutulur" biçiminde yanlış anlaşılmaya yol açacak bilgiler veren yerli ve yabancı çok sayıda yazı ve makaleyle karşılaştım. Konuya biraz açıklık getirmek istiyorum.

1. İster referans türlerine (*reference types*) ilişkin olsun ister değer türlerine (*value types*) ilişkin olsun tüm yerel değişkenler (yani bildirimleri metodların içerisinde yapılan değişkenler) ve parametre değişkenleri *stack*'te tutulur. Örneğin:

```
public static void Main()
{
    string a;
    int b;
    //...
}

public static void Foo(string c, int d)
{
    //...
}
```

burada *a*, *b*, *c*, *d* değişkenlerinin hepsi *stack*'te tutulur.

2. *new* operatörü ile sınıf, dizi, ya da delege türünden nesneler *heap*'te, yapı ve *enum* türünden nesneler ise *stack*'te yaratılır. *C#* standartlarına göre *new* operatörü yapı ya da *enum* türleriyle kullanıldığında önce *stack*'te geçici olarak yapı ya da *enum* nesnesi yaratılır; sonra yaratılan bu geçici nesne işleme sokulur ve ilgili ifade bittikten sonra da yok edilir. Örneğin (*Point* türünün *X* ve *Y* elemanlarına sahip bir yapı olduğunu varsayalım):

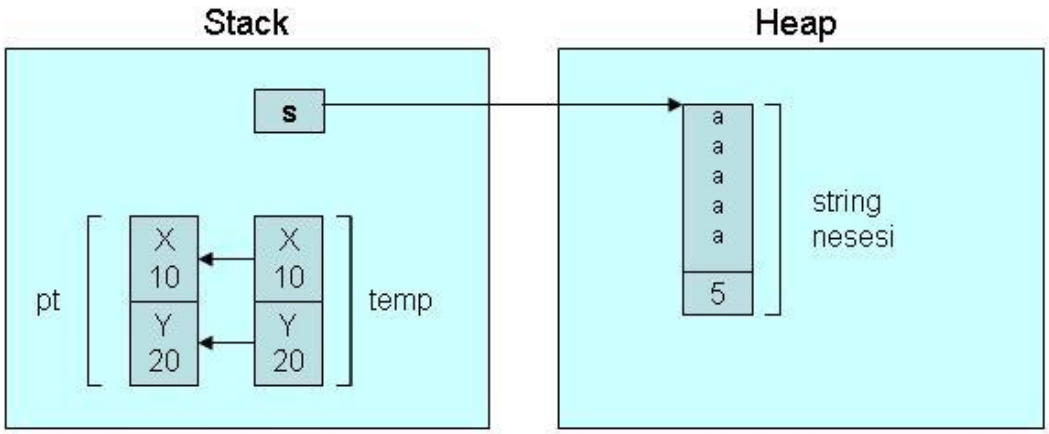
```
public static void Foo()
{
    string s;
    Point pt;

    s = new string('a', 5);
    pt = new Point(10, 20);
    //...
}
```

Burada *string* nesnesi *heap*'te *Point* nesnesi ise *stack*'te yaratılacaktır. Örneğimizdeki *Point* nesnesinin nasıl yaratıldığına dikkat ediniz:

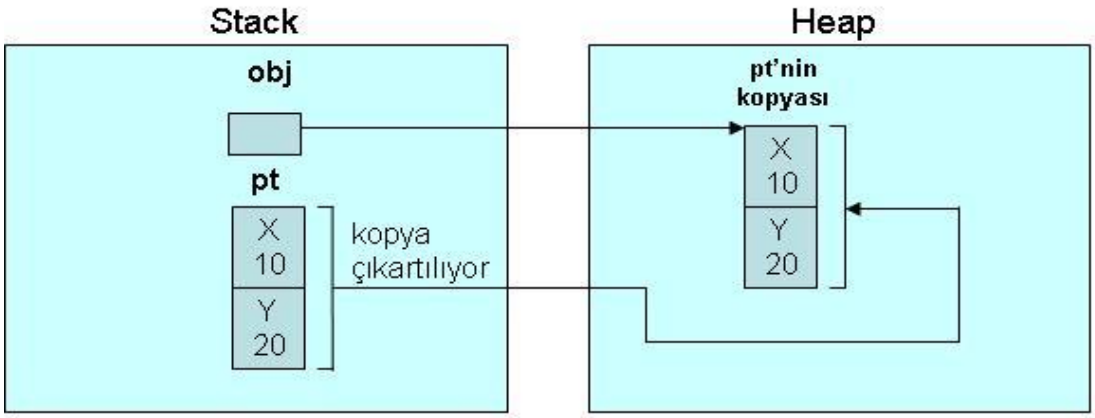
```
pt = new Point(10, 20);
```

Burada *new Point(10, 20)* ifadesi ile birlikte *stack'te* geçici bir *Point* nesnesi tahsis edilecek ve o nesne için yapının başlangıç metodu (*constructor*) çağrılacaktır. Yani bu ifadeyle *stack*'te ilkdeğerlerini almış bir yapı nesnesi elde ediliyor. Sonra onun *pt* nesnesine atandığını görüyorsunuz. Aynı türden iki yapı nesnesi birbirine atandığında yapının karşılıklı elemanlarının birbirine atandığını zaten biliyorsunuz. Örneğimizdeki tahsistaı şekilsel olarak şöyle gösterebiliriz:



3. Bir yapı nesnesi kutulama dönüşürmesi (*boxing conversion*) sonucunda *heap*'te yer alabilir. Bildiğiniz gibi yapı ya da *enum* türünden bir nesne *System.ValueType* ya da *System.Object* türlerine dönüştürüldüğünde kutulama dönüştürmesi gerçekleşir ve nesnenin *heap*'te bir kopyası çıkartılır. Örneğin:

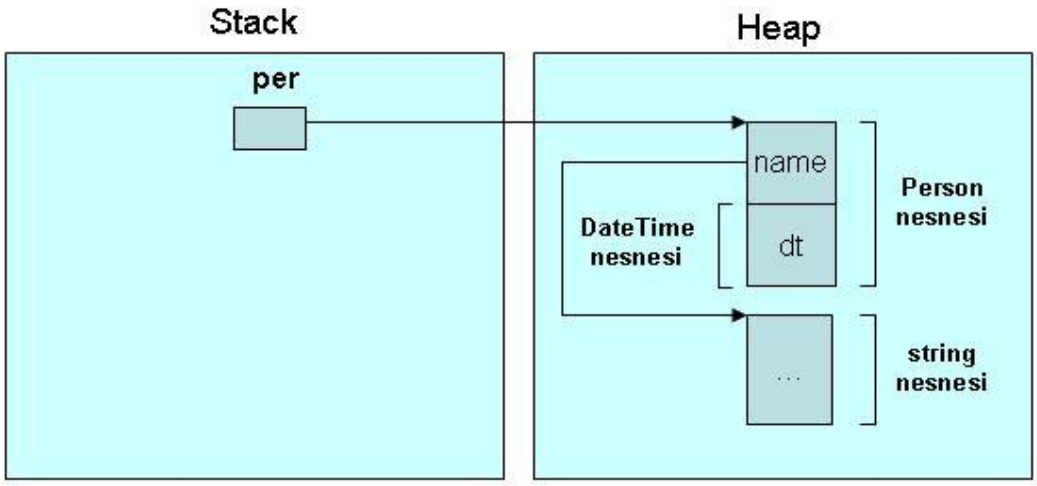
```
object obj;
Point pt = new Point(10, 20);
obj = pt;
```



Burada *obj* referansı *stack*'teki *Point* nesnesini göstermiyor. *Heap*'te kopyası çıkartılmış olan *Point* nesnesini gösteriyor. Kutulama dönüştürmesinden sonra *stack*'teki nesne ile onun *heap*'teki kopyası artık bağımsız iki nesnedir. *Stack*'teki nesne programın akışı bildirimin yapıldığı yerel bloktan çıktığında yok edilir. *Heap*'teki kopya ise nesne seçilebilir (*eligible*) olunca çöp toplayıcı tarafından silinecektir. Bir yapı nesnesi bir sınıf nesnesinin veri elemanı olacak biçimde de *heap*'te bulunabilir. Örneğin:

```
class Person
{
    private string name;
    private DateTime dt;
    //...
}
//...
Person per = new Person();
```

Bu örnekte *Person* nesnesinin bir elemanı *string* sınıfı türünden bir referans diğer elemanı da *DateTime* yapısı türünden bir yapı nesnesidir. *Person* nesnesi *heap*'te yaratılacağına göre *DateTime* yapı nesnesi de *heap*'te bulunacaktır. Ancak yapı nesnesinin bağımsız bir nesne olarak değil, başka bir nesnenin bir bileşeni olarak (*composition*) *heap*'te bulunduğuna dikkat ediniz.



4. Sınıfların ya da yapıların *static* veri elemanları (*static fields*) *.NET* ve *Mono* gibi ortamlarda ilgili türlere ilişkin *Type* nesnesinin içerisinde tutulmaktadır. *CLR*'nin bir tür kullanıldığında o türe ilişkin *metadata* bilgilerinin kökünü tutan bir *Type* nesnesi oluşturduğunu biliyorsunuz. (Türün *Type* nesne referansının *C#*'ta *typeof* operatörüyle ya da *Object* sınıfının sanal *GetType* metoduyla elde edilebileceğini anımsayınız.) O halde *C#*'ta sınıfların ve yapıların *static* veri elemanlarının da dolaylı bir biçimde *heap*'te tutulduğunu söyleyebiliriz.