JAVA Metodlar

Metodlar

Metodlar JAVA programlarının ana parçalarıdır. Metodlar sınıfların (class) içinde yer alan küçük program parçacıklarıdır. Metodların çoğunda değişken parametreler metotlar ve sınıflar arasında iletişimi sağlarlar. Ayrıca her metodun kendine özgü değişkenleri de vardır. Metod yapısının ana sebebi programları modüler hale getirmektir. Aynı zamanda aynı program parçacığının tekrarlanmasını önlemeyi de sağlar. Her metod çağrıldığı program parçacığına belli bir değişkeni aktarabilir. Metodların tanımlarında aktardıkları değişken türü tanımlanır. Eğer metod hiçbir değişken aktarmıyorsa **void** sözcüğüyle tanımlanır.

Metodlar

Metodların özellikleri şunlardır;

- Class içerisinde tanımlanmalıdır.
- Modifier bulunmalıdır. (private,public,protected yada belirtmezsek default)
- **Return type** herhangi bir veri tipi yada **void** olabilir.
- Metod ismi belirtilmelidir.
- Metod parametreleri parantez içerisinde belirtilmelidir . Eğer parametre almayacaksa parantez açılıp kapatılır.
- Metod içerisinde kod blokları **{** işaretiyle başlar **}** işaretiyle sonlandırılır.
- Metod parametrelerini ayırmak için**,**işareti kullanılır.
- Exceptionlar metodların yanında **{** işaretinden önce belirtilebilir.

Metodlar

Metotlar oluşturulma şekillerine göre ikiye ayrılırlar:

- ***Önceden Tanımlanmış Metotlar:*** *JAVA kütüphanelerinde bulunan hazır metotlardır.*

Örneğin: *pow(x,y) metodu. Math sınıfına ait olan bu metot, x ve y olarak iki değer alır ve bir değer döndürür. Görevi ilk değerin (x), ikinci değer (y) üssünü almaktır.*

- ***Kullanıcı Tanımlı Metotlar:*** *Bizim yazdığımız ve oluşturduğumuz metotlardır.*

Metotların genel tanımı aşağıdaki gibidir:
Öncelikle bir Main metoduna bakarak hatırlayalım. Nasıldı o main metodu:

Public Static Void Main(String[] args){} Şimdi bunu genel bir ifadeyle yorumlarsak aşağıdaki olay çıkar…

```
erişim_tipi (public,private) static (olabilir, olmayabilir)
dönüş_tipi(void ,int vb) foksiyon_ismi(parametreler){

  // Gövde kısmı

}
```

Metodlar

Erişim Tipleri:

1. public:

public damgası bir değişkeni, metodu ya da sınıfı niteleyebilir. Niteledikleri öğeler herkes tarafından kullanılabilir. Başka pakette olsa bile, program içindeki her kod onlara erişebilir. public damgalı bir sınıfın değişkenlerine ve metotlarına kendi alt-sınıfları ve dışarıdaki başka sınıflar kısıtsız erişebilir. public damgalı değişkenler ve metotlar için de kısıtsız erişim vardır. Uygulama programlarında main() metodunun public damgalı olmasının nedeni budur.

2. private

Bazı değişken, metot ya da sınıflara başka sınıftaki kodların erişmesini engellemek isteyebiliriz. Bunun için private nitelemesini kullanırız. Private damgalı öğelere yalnız aynı sınıftaki kodlar erişebilir, başka sınıftaki kodlar erişemez. Kendi alt-sınıfları bile erişemez. Bir alt-sınıf, atasının public ve ön-tanımlı öğelerine erişebilir, ama private öğelerine erişemez. Onlara erişebilmesi için, super class interface-fonksiyonu kullanılır.

Metodlar

Erişim Tipleri:

3. protected

Bir sınıf içindeki değişken ve metotlara alt-sınıfların erişebilmesini, ama paket içindeki ya da program içindeki başka kodların erişmesini engellemek isteyebiliriz. Bunun için söz konusu öğeyi, protected damgası ile nitelemek yetecektir.

Modifier	Class	Package	Subclass	World
`public`	✔	✔	✔	✔
`protected`	✔	✔	✔	✘
`no modifier*`	✔	✔	✘	✘
`private`	✔	✘	✘	✘

Metodlar

Static Durumu:

Static anahtar kelimesi kullanılarak oluşturulan değişkenler nesne değişkeni değil “sınıf değişkeni” olarak adlandırılırlar. Bu değişkenler nesneye ait değil, sınıfa ait bilgileri taşırlar. Sınıf değişkenleri içinde tanımlandığı sınıftan hiçbir nesne oluşturulmamış olsa bile bellekte yer kaplarlar. Nesne değişkenleri ise ancak bir nesne tanımlandığında bellekte yer kaplarlar. Bu iki tür değişkenin ayrıldığı bir başka nokta da sınıf değişkenlerinin sadece bir örneğinin olmasıdır. Yani o sınıftan kaç tane nesne oluşturulursa oluşturulsun, bellekte tek bir tane sınıf değişkeni vardır ve ne şekilde erişirsek erişelim, aynı sınıf değişkenine erişiriz.

Dönüş Tipi:

Void: Geri dönüş değeri almaz.

int, **String**, **Float**, **Double:** Return değeri almaktadır.

Metodlar

Parametre:

Parametre, malzeme olarak örneklenebilir. Siz metoda malzeme verirsiniz ve o da size onu işleyerek geri döndürür veya döndürmeden kendi işlemlerinde kullanır. Bazı metotlar dışarıdan parametreye ihtiyaç duyarken, bazıları duymaz.

Metod Çağırma (Calling Methods):

Metodları ana mainde iki şekilde çağırabiliriz;

Eğer metod static değilse:

```
SınıfIsmi class = new SınıfIsmi();
class.metod();
```

Eğer metod static ise :

```
metod();
```

THANKS!

Any questions?

Garry T.
Full-Stack Automation Engineer
garry@clarusway.com