



# CLARUSWAY

WAY TO REINVENT YOURSELF



clarusway



clarusway



clarusway



clarusway



clarusway

## Method Creation

Lesson:  
JAVA Chapter 10

# Method Creation (Method Oluşturma)

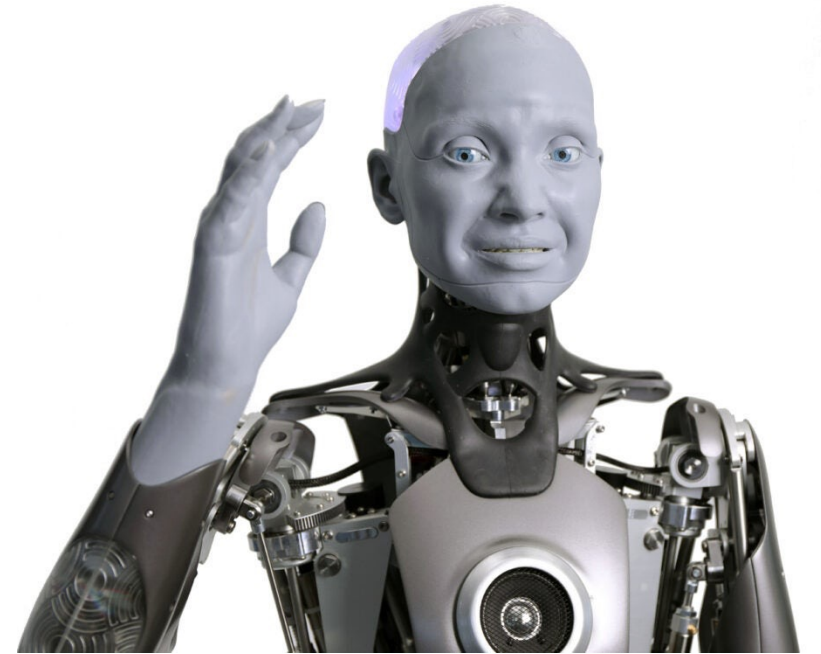
**Method:** İstenilen işlem(aksiyonu) uygulayan code bloğuna **metod** denir. (Belirli bir iş için tasarlanmış robotlar gibi...).

Methodlar ;

**1-** Tekrar tekrar kullanacak bir işlem için her işlemde aynı kodu yazmamak için,

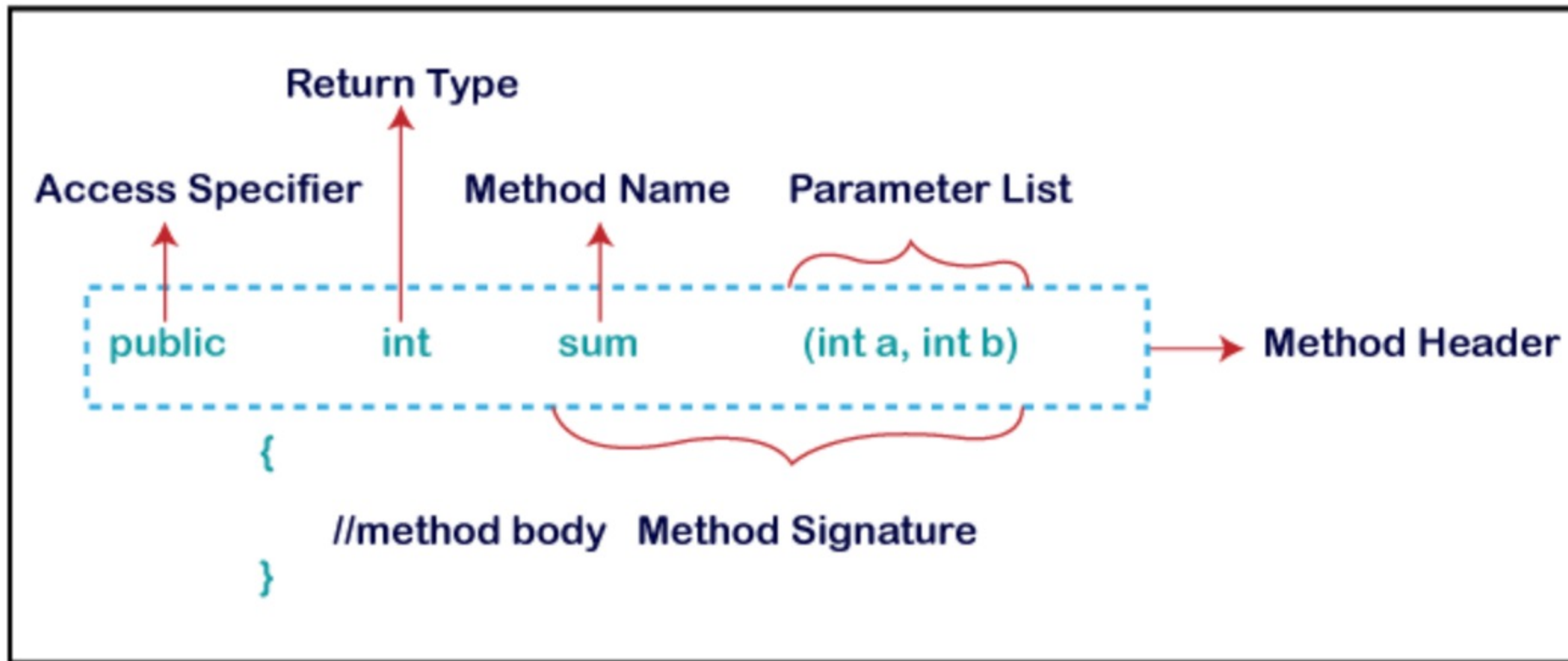
**2-** Çalışılan class'ı basit bir yapıda tutup, kodumuzu daha anlaşılabilir hale getirmek için kullanılır.

**\* TRICK :) Bir method'u oluşturmak çalışması için yeterli değildir, method'un çalışması için mutlaka çağırılması (method call) gerekir.**



# Method Creation (Method Oluşturma)

## Method Declaration





# Method Creation (Method Oluřturma)

Temelde 2 eřit method vardır.

1 : İstedigimiz isi yapıp bize bir sonuc dondurmeyen veya sadece konsolda yazı yazdıran method'lar. ( elektrik faturasını yatıran cocugumuz gibi)

Bunların return type'i void olmalıdır.



2 : İstedigimiz isi yapıp bize bir sonuc dondurmasını istedigimiz method'lar. ( bakkaldan alışveriş yapıp bize getiren kapıcı gibi)

- Bunların return type'i istedigimiz sonuca uygun olmalıdır.
- Method'un sonunda return keyword'u ve bize dondureceği sonuc olmalıdır
- Dondurduğu sonucu bir uygun bir variable'a atamalıyız



# Method Declaration (Method Tanımlama) Keyword'leri

```
public int myFirstMethod () {}  
1      2      3      4  5
```

- 1 **Access Modifier (Erişim Düzenleyici)** : Method'a kimlerin erişebileceğini belirler.  
**public** : Methoda herhangi bir kısıtlama olmadan istenilen yerden erişilebilir.  
**private** : Sadece methodun tanımlandığı class'da erişilebilir.  
**protected** : Sadece methodun tanımlandığı class ve child class'lardan erişilebilir.
- 2 **Int** : Return Type: method aksiyonu sonucu çıktının data type'nı ve döndürdüğünü belirtir.
- 3 **myFirstMethod** : Tanımlana method'un ismidir. Method ismi mutlaka küçük harfle başlar, eğer method name birden fazla kelimeden oluşursa sonraki kelimelerin ilk harfleri büyük yazılır. (camelCase)
- 4 **() parantez** : Methodlarda isimden sonra parantez kullanılır ve gerektiğinde parantez içinde parametre yazılır.
- 5 **Body (Method Body)** : { } Arasında kalan code create edilen bölümdür.



# Method Creation (Method Oluşturma)

## 2 static:

Bir method create edilirken **static** keyword kullanılması zorunlu değildir.

Main method static olduğu için main method'dan çağıracağımız tüm method'ları static olmalıdır.

```
public static void main(String[] args) {  
  
}
```



# Method Creation (Method Oluşturma)

**3 int (Return Type):** Method'un ne ürettiğini ve bize ne döndürdüğünü belirtir.

- Return Type, primitive veya non-primitive tüm data type olabilir.
- Eğer method bir şey döndürmeyecekse (Örneğin, sadece bir değer hesaplayıp print edecek) return type olarak **void** seçilir.
- Return Type olarak void dışında bir data type kullanılırsa, methodun sonunda mutlaka **return** keyword kullanılmalıdır.
- Return keyword'den sonra return type'a uygun bir **değer veya variable** yazılmalıdır.
- Return type'a sahip methodlar çağrıldıkları satıra, return keyword'den sonra yazılan değer veya variable'i döndürürler.

```
public static void main(String[] args) {  
  
    int maas = maasHesapla(yevmiye: 250, gun: 26);  
  
}  
  
public static int maasHesapla(int yevmiye, int gun) {  
    return yevmiye * gun;  
}
```





# Method Creation (Method Oluşturma)

**4 myFirstMethod:** Oluşturduğumuz method'un ismidir. İsim mutlaka küçük harfle başlar, birden fazla kelimeden oluşursa sonraki kelimelerin ilk harfleri büyük harf yazılır. (camelCase)

**5 () parantez:** Methodlarda isimden sonra parantez kullanılır ve gerektiğinde parantez içinde parametre yazılır.

**\* TRICK :)** Eğer bir Class'da aynı isme sahip birden fazla method create edildiğinde parametreleri farklı olmalı. (Overloading)





# Method Creation (Method Oluşturma)

**6 Body (Method Body):** {} arasında kalan aksiyon code create edilen bölümdür.

**\*TRICK :) Method nerede create edilmeli ?**

**Method Class body'si içinde Main Method dışında create edilmeli.**

```
public class ClarusWay {// Class başlangıcı

    public static void main(String[] args) {// main method başlangıcı

        int maas = maasHesapla(yevmiye: 250, gun: 26); // method call

    }// main method sonu

    public static int maasHesapla(int yevmiye, int gun) {

        return yevmiye * gun;

    }

}// Class sonu
```



# Method Creation (Method Oluşturma)

Method create etmek method'u çalıştırmak için yeterli değildir.

**İstendiğinde önceden create edilen method'u çalıştırmak için** method name (parametreler ile birlikte) ile call edilmelidir.

```
public class ClarusWay {// Class başlangıcı

    public static void main(String[] args) {// main method başlangıcı

        int maas = maasHesapla(yevmiye: 250, gun: 26); // method call

    }// main method sonu

    public static int maasHesapla(int yevmiye, int gun) {
        return yevmiye * gun;
    }

} // Class sonu
```

\*\*\* Method call'da parantez içine yazılan değerlere **Arguments (argüman)** denir.

\*\*\* Method call'da kullandığımız argümanlar ile method parametrelerinin uyumlu olması gereklidir.

\*\*\* Sayı parametreleri için char değerler de argüman olarak kullanılabilir.



# Method Overloading

## Interview Soruları

### 1) Overloading nedir?

Eğer bir class'da ismi aynı fakat parametreleri farklı olan method'lar oluşturursak buna **Overloading** denir.

### 2) Overloading nasıl yapılır?

JAVA aynı isim ve aynı parametrelerle birden fazla method oluşturulmasına izin vermez. Aynı isimle birden fazla method oluşturmak isterseniz **Method Signature (method imzası)**'nın değiştirilmesi gereklidir.

### 3) Method Signature (method imzası) nasıl değiştirilir?

Method Signature'ı değiştirmek için 3 yöntem kullanılabilir.

- Parametrelerin data tipleri değiştirilebilir.
- Parametrelerin sayısı değiştirilebilir.
- Parametre sayısı aynı olmak zorunda ise farklı data tipindeki parametrelerin sırası değiştirilir.

\* **TRICK :** Method'un return type'ini, access modifier'ini değiştirmek veya static kelimesi eklemek method signature'ı değiştirmez.



# Method Overloading

```
public static int sumTwoNumbers(int a, int b) {  
    return a + b;  
}  
  
public static int sumThreeNumbers(int a, int b, int c) {  
    return a + b + c;  
}  
  
public static int sumFourNumbers(int a, int b, int c, int d) {  
    return a + b + c + d;  
}
```

**BAD PRACTICE!**

```
public static int sum(int a, int b) {  
    return a + b;  
}  
  
public static int sum(int a, int b, int c) {  
    return a + b + c;  
}  
  
public static int sum(int a, int b, int c, int d) {  
    return a + b + c + d;  
}
```

**GOOD PRACTICE!**





# THANKS!

## Any questions?

HalUk Bilgin | JAVA Backend Developer

BAŞARI GAYRETE AŞIKTIR 😊

