Method Creation / Method Olusturma

Method : Istedigimiz islemi bizim adimiza yapan kod bloklaridir(Is yapmak icin tasarlanmis robotlar gibidirler).

Genelde iki amacla method olustururuz

1- Projemiz icerisinde tekrar tekrar kullanacagimiz bir islem icin her seferinde yeniden kod yasmak yerine bir kere yazip ihtiyacimiz oldukca kullanmak

2- Calistigimiz class'i basit bir yapida tutup, sectigimiz uygun isme sahip method'larla kodumuzu daha anlasilabilir hale getirmek

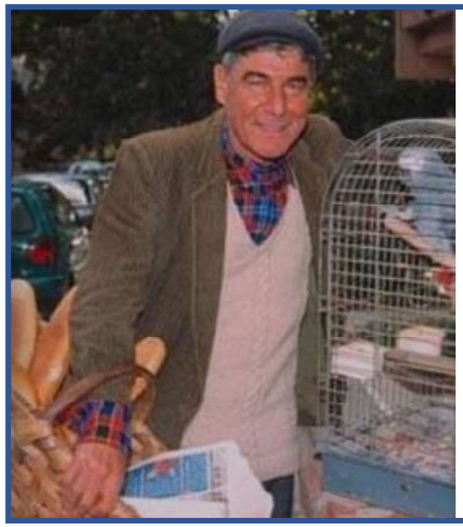
NOT : Bir method'u olusturmak calismasi icin yeterli degildir, method'un calismasi icin mutlaka cagrilmasi(method call) gerekir.

Method Creation / Method Olusturma

Temelde 2 cesit method vardir

1 : Istedigimiz isi yapip bize bir sonuc dondurmeyen veya sadece konsolda yazi yazdiran method'lar. (elektrik faturasini yatiran cocugumuz gibi)

Bunlarin return type'i void olmalidir.



2 : Istedigimiz isi yapip bize bir sonuc dondurmesini istedigimiz method'lar. (bakkaldan alisveris yapip bize getiren kapici gibi)

- Bunlarin return type'i istedigimiz sonuca uygun olmalidir.
- Method'un sonunda return keyword'u ve bize dondurecegi sonuc olmalidir
- Dondurdugu sonucu bir uygun bir variable'a atamaliyiz

Method Creation / Method Olusturma

Method Olusutururken Kullanilan Keyword'ler Nelerdir?

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

1 public : Access Modifier (Erisim duzenleyici):methoda'a kimlerin erisebilecegini belirler
protected : Sadece icinde bulundugu package ve child class'lardan kullanilir
default : Sadece icinde oldugu package
private: Sadece bulundugu class'da kullanilabilir

2 Int : Return Type, methodun ne urettigini ve bize dondurdugunu belirtir

3 myFirstMethod :Olusturdugumuz method'un ismidir. Isim mutlaka kucuk harfle baslar, birden fazla kelimeden olusursa sonraki kelimelerin ilk harfleri buyuk harf yazilir (Camel Case)

4 ( ) parantez: Methodlarda isimden sonra parantez kullanilir ve gerektiginde parantez icinde parametre yazilir.

5 Body (Method Body) : { } arasinda kalan kodlarimizi yazdigimiz bolumdur

Method Creation / Method Olusturma

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

1 Access Modifier (Erisim duzenleyici):

public : methoda'a kimlerin erisebilecegini belirler

protected : Sadece icinde bulundugu package ve child class'lardan kullanilir

default : Sadece icinde bulundugu paket(package)'den kullanilir

private: Sadece bulundugu class'da kullanilabilir

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Method Creation / Method Olusturma

2 static (Ileride detayli anlatilacak)

Bir method olusturulurken static kelimesinin kullanilmasi mecburi degildir.

Main method'umuz static oldugu icin main method'dan cagiracagimiz tum method'lari static yapmamiz gereklidir

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


}
```

Method Creation / Method Olusturma

3 int (Return Type) : methodun ne urettigini ve bize ne dondurdugunu belirtir.

- Return Type, primitive veya non-primitive tum data turlerinden olabilir
- Eger method bir sey dondurmeyecekse (ornegin, sadece bir sey hesaplayip yazdiracaksa) return type olarak void secilir
- Return Type olarak void disinda bir sey yazdiysak, methodun sonunda mutlaka return keyword kullanilmalidir
- Return keyword'den sonra return type'a uygun bir deger veya variable yazilmalidir.
- Return type'a sahip methodlar cagrildiklari satira, return keyword'den sonra yazilan deger veya variable'i dondururler.

```
public static void main(String[] args) {
        int sonuc= topla(15,24);

    }

public static int topla(int num1, int num2) {

    return num1 + num2;
}
```

Method Creation / Method Olusturma

4 myFirstMethod :Olusturdugumuz method'un ismidir. Isim mutlaka kucuk harfle baslar, birden fazla kelimeden olusursa sonraki kelimelerin ilk harfleri buyuk harf yazilir (Camel Case)

5 ( ) parantez : Methodlarda isimden sonra parantez kullanilir ve gerektiginde parantez icinde parametre yazilir.

*** Eger bir Class'da ayni isme sahip birden fazla method olusturmamiz gerekirse parametreleri farkli yapmamiz gereklidir (Overloading)

Method Creation / Method Olusturma

6 Body (Method Body) : { } arasinda kalan kodlarimizi yazdigimiz bolumdur

*** Method nerede olusturulmalidir ?

Method Class body'si icinde Main method disinda olusturulmalidir

```
public class asd {

    public static void main(String[] args) {
        toplama(5,4);


    }

    private static void toplama(int i, int j) {

        System.out.println(i+j);

    }

}
```

Method Creation / Method Olusturma

Method olusturmak method'u calistirmak icin yeterli degildir.

Ihtiyac duyuldugunda daha onceden olusturulmus methodu calistirmak icin Method ismi (parametreler ile birlikte) yazilmalidir.

Bu isleme method cagirma denir

```
public class asd {

    public static void main(String[] args) {

        toplama(5,4);

    }

    private static void toplama(int i, int j) {

        System.out.println(i+j);

    }

}
```

*** Method cagirirken parantez icine yazilan degerlere Arguments (arguman) denir.

*** Method cagirirken kullandigimiz argumanlar ile method parametrelerinin uyumlu olmasi gereklidir.

*** Sayi parametreleri icin char degerler de arguman olarak kullanilabilir

Method Overloading

Interview Sorusu

1) Overloading nedir ? Eger bir Class'da ismi ayni fakat parametreleri farkli olan methodlar olusturursak buna Overloading denir.

2) Overloading nasil yapilir ? Java ayni isim ve ayni parametrelerle birden fazla method olusturulmasina izin vermez. Ayni isimle birden fazla method olusturmak isterseniz method signature (metot imzasi)'nin degistirilmesi gerekir

3) method signature (metot imzasi) nasil degistirilir?

Method signature'i degistirmek icin 3 yontem kullanilabilir

- parametrelerin data tipleri degistirilebilir
- parametrelerin sayisi degistirilebilir
- parametre sayisi ayni olmak zorunda ise farkli data tipindeki parametrelerin sirasi degistirilir

*** method'un return type'ini degistirmek, access modifier'ini degistirmek veya static kelimesi eklemek method signature'i degistirmez