

YMH112
ALGORİTMA ve PROGRAMLAMA II
LAB. KISMI
5. CANLI DERS
GÜNDÜZ (A)GRUBU

27 MAYIS 2021

SAAT =11.00-12:00

ARŞ. GÖR. ALEV KAYA

ENCAPSULATION(KAPSÜLLEME)

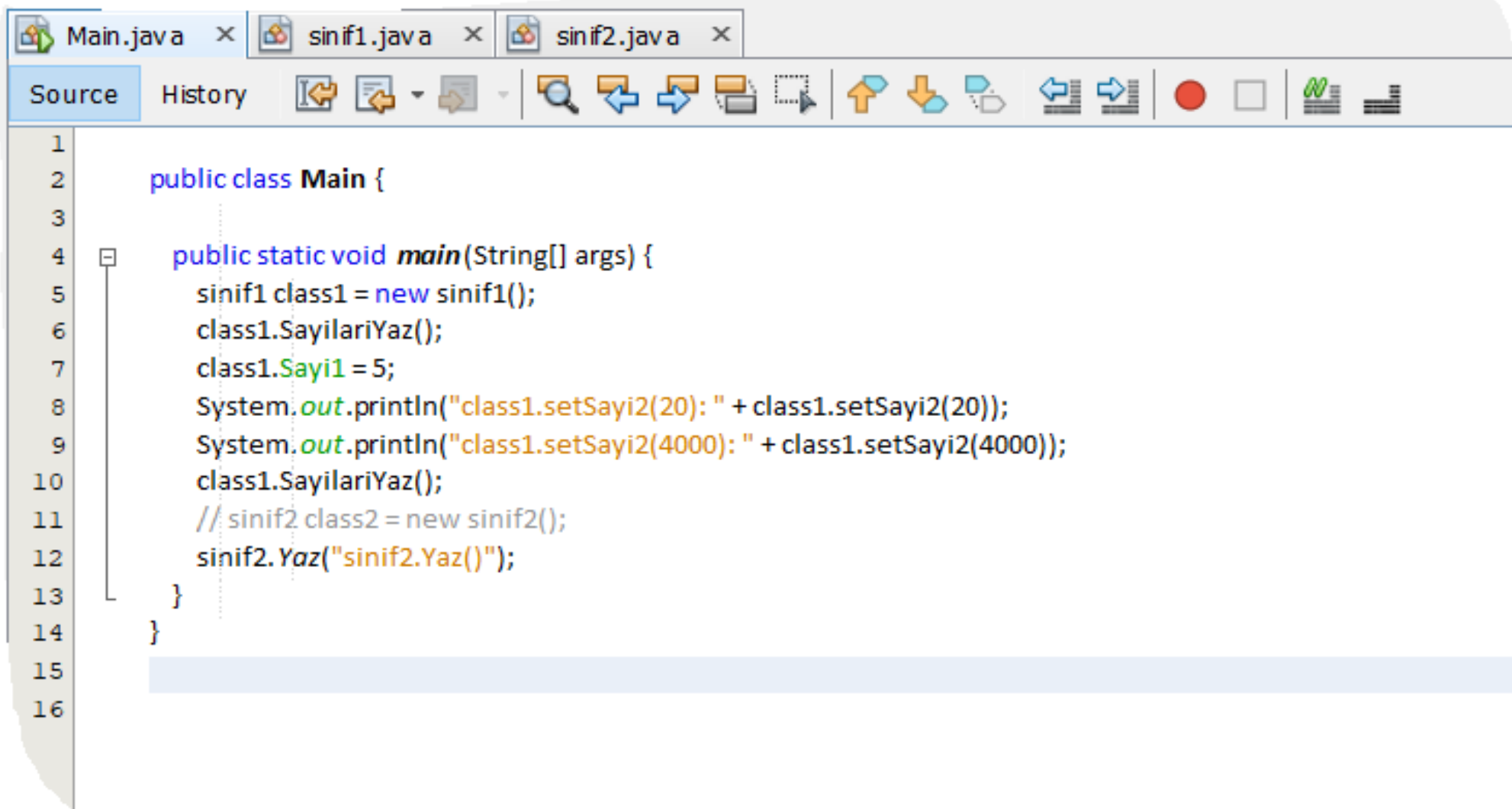
- Çoğu zaman, hayatın her alanında bazı **kısıtlamalar** ile karşılaşırız. Eğer bir şeye sahiplik varsa onun saklanması, korunması da çok doğal bir istektir.
- Herkes, her zaman her şeyini birileri ile paylaşmak istemeyebilir. Daha da ötesi paylaşmasının çok da uygun olmayabileceğini düşünebilir.
- İzinsiz kullanımın, yetkisiz kullanımın önüne geçilmesi için de böyle bir yapı şarttır aslında.
- Programlama dillerindeki **access specifiers(erişim belirleyicileri)** kavramı da tamamen bu doğal dürtünün sonucunda ortaya çıkmıştır.
- Ben bir sınıf yazıp bunu bir paketin içerisinde başkaları ile paylaşabilirim. Başkalarının da yazdığım kod parçacıklarından faydalanmasını sağlayabilirim. Fakat sınıf içerisinde yer alan bazı değişkenlerin kimse tarafından değiştirilememesini, bazı yordamların **override** edilememesini isteyebilirim. Kodun sahibi olarak bunları isteyebilmek benim en doğal hakkımdır aslında.
- Şöyle toparlayabiliriz bu kısmı; **erişim belirleyicileri, programcı tarafından oluşturulan her bir yazılım ögesinin paylaşım sınırlarını belirlemek için kullanılmaktadır...**

- Bizim bu belirteçlerden özellikle üzerinde duracağımız **public, friendly, private, protected** erişim belirleyicileridir.
- İlk önce **friendly erişim belirleyicisi** ile başlamak istiyorum. **Friendly erişim belirleyicisi** java dili için **standart erişim belirleyicisidir**.
- Yani eğer herhangi bir erişim belirteci yoksa friendly olarak kabul edilir. Bu erişim belirleyicisi metodlara, sınıflara ve alanlara uygulanabilmektedir.
- **Özelliği ise aynı paket içerisindeki tüm sınıflardan bu üyelere erişilebilirken, paket dışı üyelere bu üyelere erişilememektedir.**
- Burada enteresan olan ise bu erişim belirleyicisinin kullanılmasının istenmemesidir.
- Yani bir metodun, sınıfın ya da alanın başına **friendly** anahtar sözcüğünün yazılması istenmemektedir. Yazarsanız hata ile karşılaşabilirsiniz.
- **Eğer bir üyeyi friendly olarak tanımlamak isterseniz yapmanız gereken o alanı boş bırakmaktır.**

Modifiers-Elements Matrix in Java

element	Data field	Method	Constructor	Class		Interface	
modifier				top level (outer)	nested (inner)	top level (outer)	nested (inner)
abstract	no	yes	no	yes	yes	yes	yes
final	yes	yes	no	yes	yes	no	no
native	no	yes	no	no	no	no	no
private	yes	yes	yes	no	yes	no	yes
protected	yes	yes	yes	no	yes	no	yes
public	yes	yes	yes	yes	yes	yes	yes
static	yes	yes	no	no	yes	no	yes
synchronized	no	yes	no	no	no	no	no
transient	yes	no	no	no	no	no	no
volatile	yes	no	no	no	no	no	no
strictfp	no	yes	no	yes	yes	yes	yes

- **Public erişim belirleyicisi** tüm üyeler tarafından **herhangi bir sınırlama olmadan** erişilebilmektedir. **Hem sınıflara, hem metodlara, hem de alanlara** uygulanabilmektedir bu erişim belirleyicisi.
- **Public erişim belirleyicisine** sahip bir üyeye hem paket içinden hem paket dışından, hem tanımlandığı sınıftan hem tanımlandığı sınıf dışından, kullanıldığı sınıftan türetilen tüm sınıflardan erişilebilmektedir.
- **Private erişim belirleyicisi** yalnızca metodlara ve alanlara uygulanabilmektedir. Sınıf tanımlamaları için ise bu belirteç kullanılamamaktadır.
- **Private** olarak tanımlanan bir üyeye yalnızca tanımlandığı sınıf içerisinden erişilebilmektedir. **Yani aynı paket içerisindeki bir başka sınıftan ya da farklı bir paketten bu üyelere erişim yapılması yasaklanmıştır.**
- Peki ama bu sınıfların içerisindeki değişkenlere erişilmek ve farklı değerlere setlenmek istendiğinde ne yapılacaktır. Bu değerler tamamen sabit ve değiştirilemez midir? Tabiki değil. İşte tam bu aşamada **encapsulation(kapsülleme)** kavramı karşımıza çıkıyor.
- **Kapsüllemeyi temel olarak dışarıdan erişimi yasaklanmış üyelerin değerlerinin özelleşmiş metodlar yardımı ile değiştirilebilmesi olarak tanımlayabiliriz.** Bu özelleşmiş metodlar içerisinde değeri değiştirilmek istenen üyelerin kontrollü bir şekilde değer değişiklikleri de yapılabilmektedir.



```
1
2 public class Main {
3
4     public static void main(String[] args) {
5         sınıf1 class1 = new sınıf1();
6         class1.SayilariYaz();
7         class1.Sayi1 = 5;
8         System.out.println("class1.setSayi2(20): " + class1.setSayi2(20));
9         System.out.println("class1.setSayi2(4000): " + class1.setSayi2(4000));
10        class1.SayilariYaz();
11        // sınıf2 class2 = new sınıf2();
12        sınıf2.Yaz("sınıf2.Yaz()");
13    }
14 }
15
16
```

Main.java x sinif1.java x sinif2.java x

Source History

1 public class sinif1

2 {

3 public int Sayi1;

4 private int Sayi2;

5 //

6 public sinif1()

7 {

8 System.out.println("sinif1()");

9 this.Sayi1 = 50;

10 this.Sayi2 = 3500;

11 }

12 public boolean setSayi2(int Sayi)

13 {

14 if((Sayi<1000 && Sayi>500) || (Sayi<5500 && Sayi>3000))

15 {

16 this.Sayi2 = Sayi;

17 return true;

18 }

19 else

20 {

21 return false;

22 }

23 }

24 public void SayilariYaz()

25 {

26 System.out.println("Sayi1: "+Sayi1);

27 System.out.println("Sayi2: "+Sayi2);

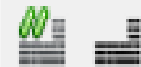
28 }

29 }

Main.java x sinif1.java x sinif2.java x

Source

History



1
2
3
4
5
6
7
8
9
10
11
12
13

```
public class sinif2
{
    private sinif2()
    {
        System.out.println("sinif2()");
    }
    public static void Yaz(String Metin)
    {
        System.out.println(Metin);
    }
}
```

```
compile-single:
run-single:
sinif1()
Sayi1: 50
Sayi2: 3500
class1.setSayi2(20): false
class1.setSayi2(4000): true
Sayi1: 5
Sayi2: 4000
sinif2.Yaz()
BUILD SUCCESSFUL (total time: 0 seconds)
```


- İlk önce sınıfların içeriklerinden başlayalım. **sinif1** sınıfı içerisinde birisi **public birisi ise private erişim belirtecine** sahip iki tane **integer türünden değişken** var.
- **SayilarıYaz** isimli metod bu değişkenlerin o an için sahip olduğu değerleri yazıyor.
- **SetSayi2** isimli metod ise **Sayi2** değişkeninin değerini setliyor.
- **Eğer değerler bizim istediğimiz ve belirttiğimiz koşullara uyuyor ise true uymuyor ise ise false değerini geri döndürüyor bu metod.**
- **sinif2** sınıfının yapıcısı dikkatinizi çekmiş olmalı. Çünkü **private** türden belirtilmiş.
- **sinif2** sınıfı içerisindeki **Yaz** isimli metod **static void** türündendir ve kendisine gönderilen **string** değişkenleri yazdırmaktadır.
- Programın nasıl çalıştığına bakacak olursak; main metodu içerisinde ilk önce **sinif1** türünden *class1* isimli bir değişken oluşturulduğu görülecektir.
- **class1** daha yeni oluşturulurken yapıcısı çağırılır ve **Sayi1, Sayi2** değişkenlerine başlangıç değerleri olan **50 ve 3500** değerleri atanır.

- Program çıktısına bakacak olursanız **yapıcı metod** içerisinde bu değerlerin **50 ve 3500** olarak yazdırıldığını da göreceksiniz.
- **15. satırda *Sayi1*** değişkenine **5** değerini atıyoruz fakat ***Sayi2*** değişkenine bu şekilde bir atama yapmamız mümkün değil. Çünkü **private** olarak tanımlanmış bulunmakta.
- **16. satırda *SetSayi2*** metodu çağırılarak ***Sayi2*'ye 20** değeri atanmak isteniyor fakat bu metod içerisinde izin verilen aralıklar arasında olmadığı için değer setlenemiyor ve **false** döndürülüyor.
- **17. satırda** atanmak istenen **4000** değeri izin verilen aralıklarda olduğu için ***Sayi2 4000*** değerine setleniyor ve değeri değiştirilmiş oluyor.
- Program çıktısına da bakacak olursanız değerini setleyen metodun **true** döndürdüğünü göreceksiniz.

- **Sinif2** sınıfının yapıcı metodu **private** tanımlandığı için bu sınıfa ait bir nesne örneği tanımlanamaz (Az önce yarım bıraktığımız sorunun cevabını da böylece vermiş oluyoruz). Bu sınıf içerisinde **static bir metod** var.
- Yeri gelmişken söylemek gerekir. **Static metodlar ait oldukları sınıfa ait nesne örneğinden çağrılmasına gerek kalmadan nokta vuruşu yaparcasına çağırılabilir.**
- **Protected erişim belirleyicisi ise kalıtım kavramı ile iç içe geçmiştir.**
- **Protected** tanımlanan bir üyeye farklı bir paket içerisinde erişilememekte fakat aynı paket içerisinde erişilebilmektedir.
- **Private** üyelere ne olursa olsun sınıf dışından erişilememekteydi fakat protected üyelere türetildikleri sınıflardan erişilebilmektedir.
- Ayrıca protected erişim belirleyicisi tıpkı **private** gibi sınıflara uygulanamamaktadır.