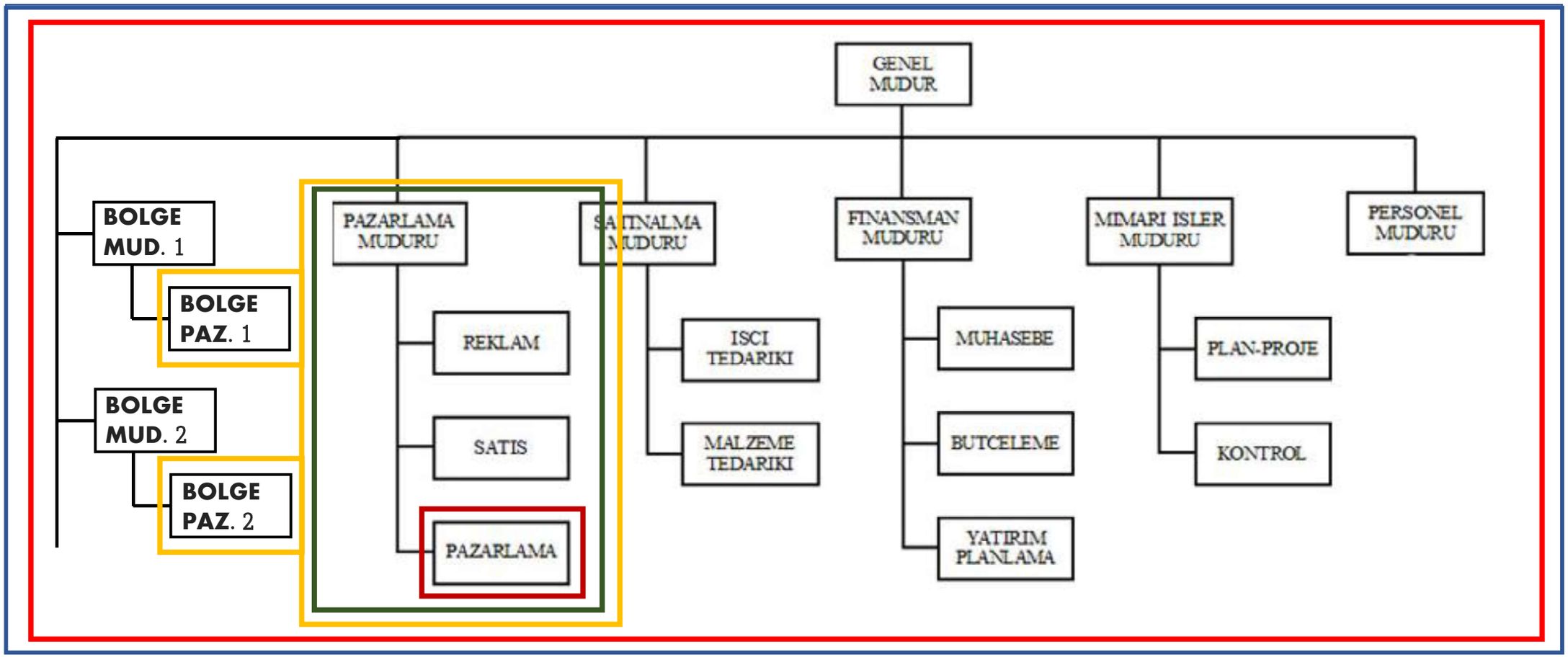
Access Modifier

Java bir method'u, variable'i ya da class'i oluşturulurken bu öğelere kimlerin erişebileceğini belirtme olanağı tanir.



Access Modifier

Java bir method'u, variable'i ya da class'i oluşturulurken bu öğelere kimlerin erişebileceğini belirtme olanağı tanir.

Bu eylemi gerçekleştirebilmek icin kullanilan anahtar kelimelere Access Modifiers (Erişim Belirleyiciler) adını veririz.

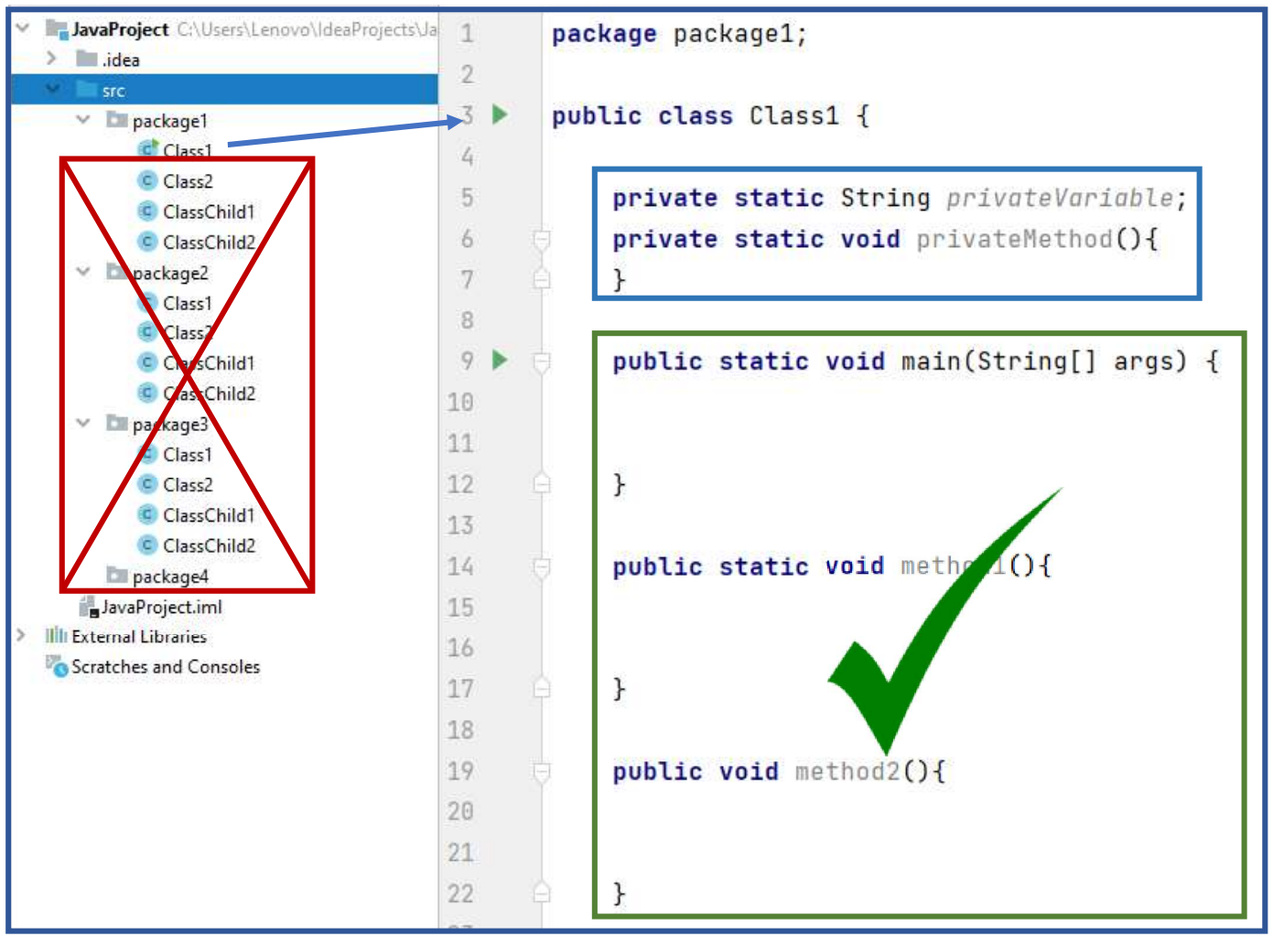
Java'da 4 Farkli access modifier vardir

1) Private
2) Default (Eger herhangi bir Access Modifier yazmazsak, Java Access Modifier'i default olarak kabul eder.)
3) Protected
4) Public

NOT : Class'lar icin sadece public veya default kullanilabilir. Class'lar private veya protected olamazlar.

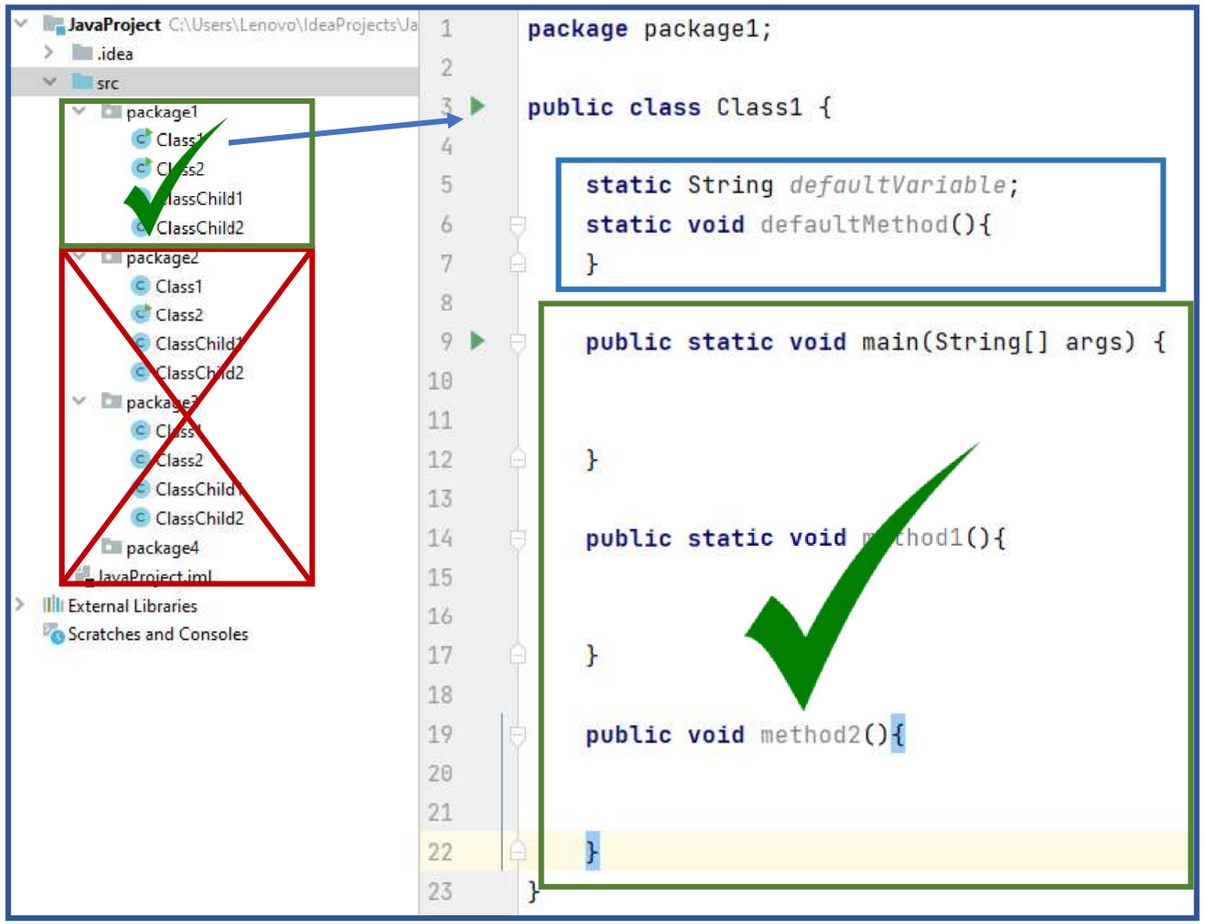
Access Modifier

1) Private : Sadece olusturuldugu Class'da kullanilabilir



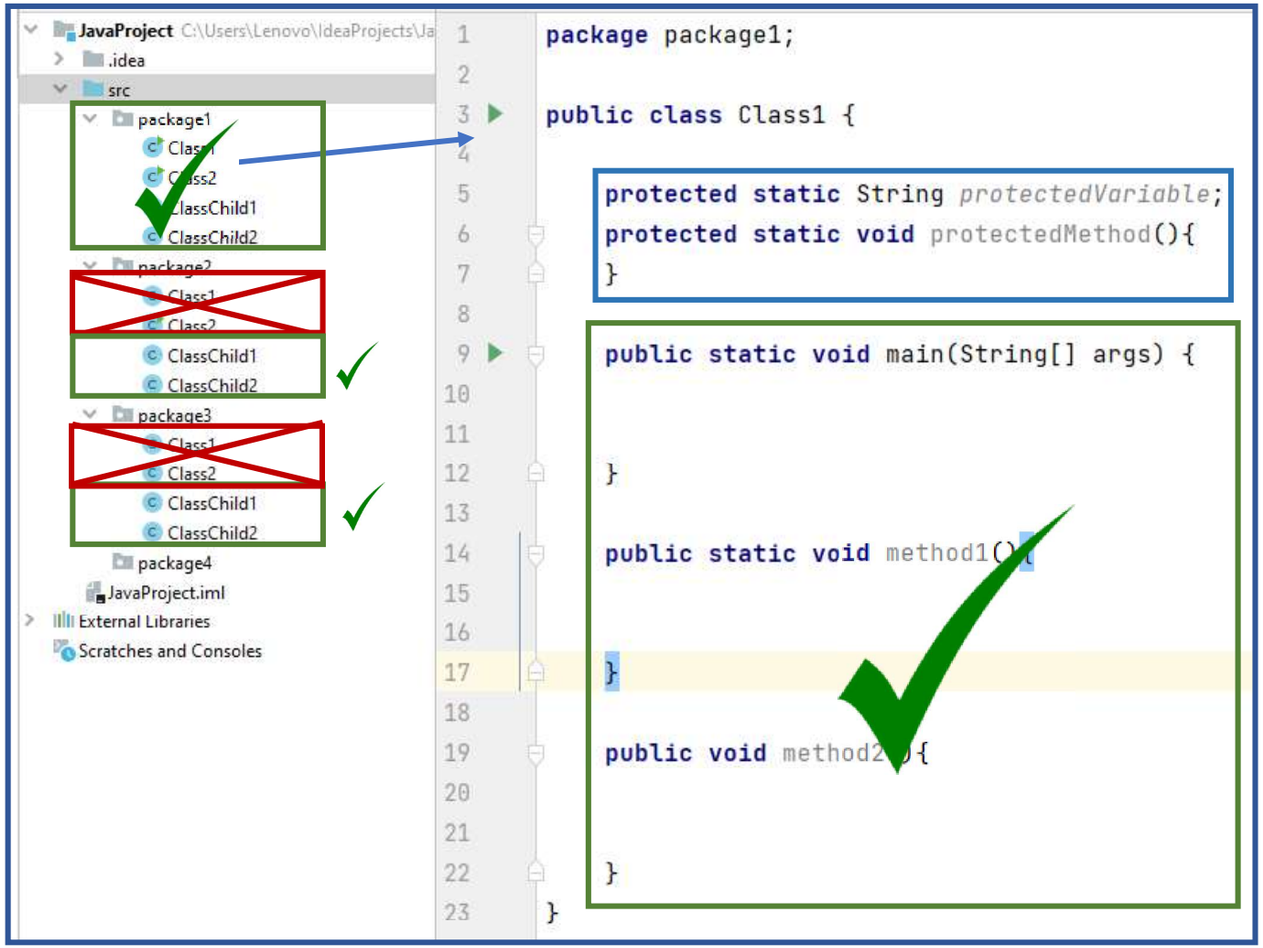
Access Modifier

2) Default : Sadece olusturuldugu Class'in ait oldugu Package icinde kullanilabilir



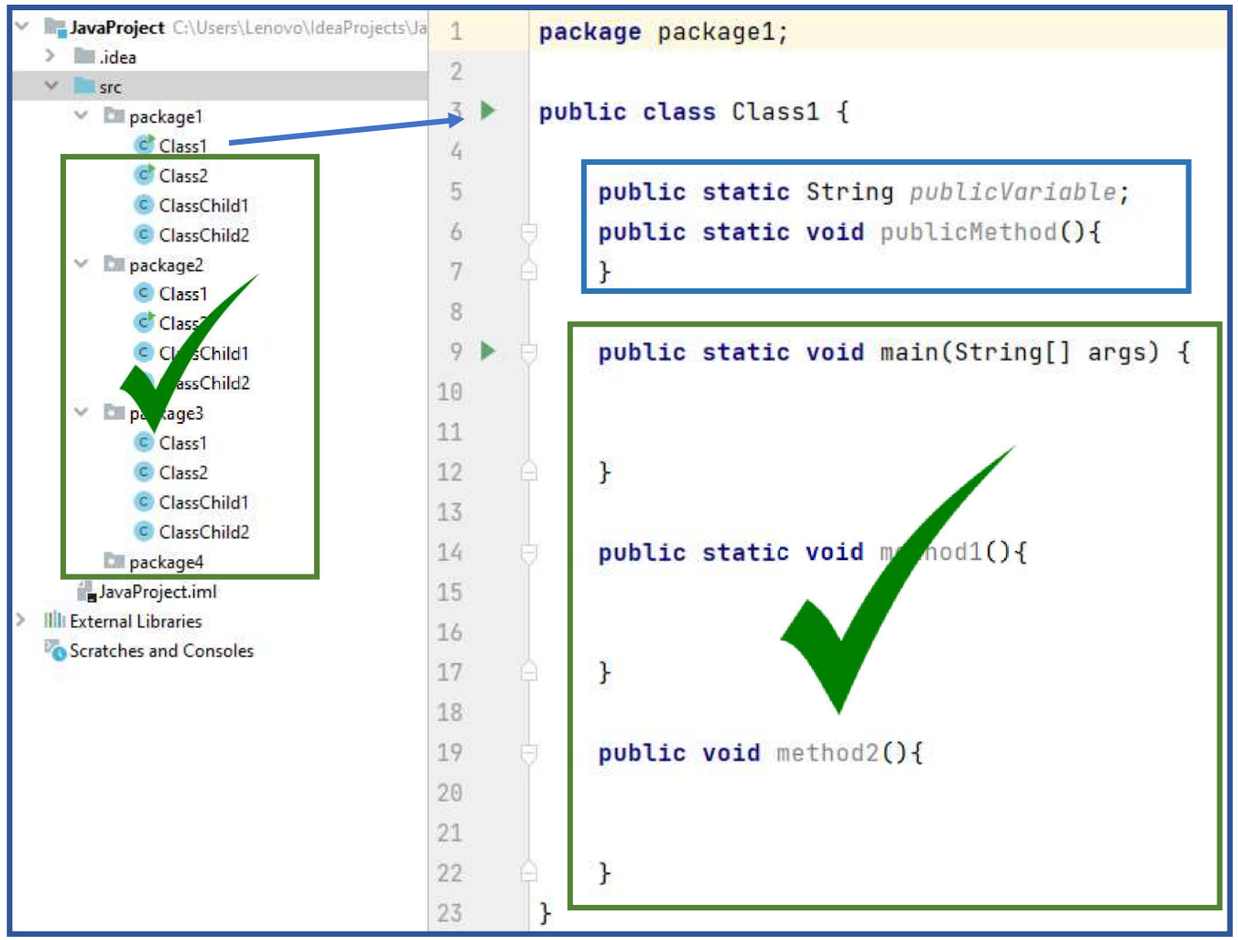
Access Modifier

3) Protected : Olusturuldugu Class'in ait oldugu Package icinde ve baska package icindeki Child Class'larda kullanilabilir

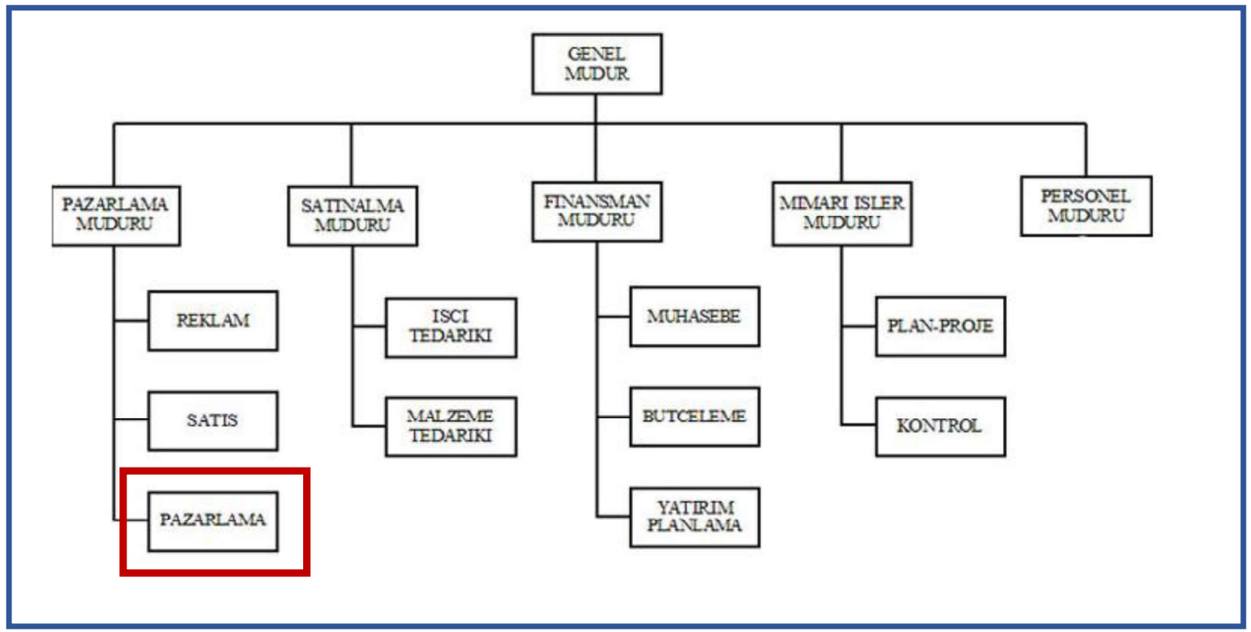


Access Modifier

4) Public : Her yerden erisilebilir. Hicbir sinirlama (restriction) icermez.



Encapsulation (Data Hiding)



Pazarlama birimi olarak rapor duzenliyoruz

Ihtiyaclarimiz

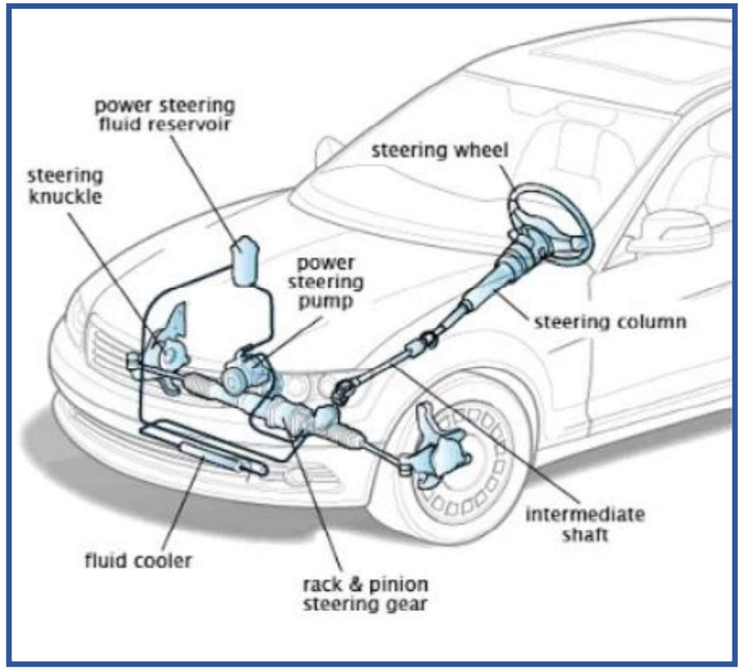
1- Satis bolumundeki personel rapor hazirlayacagimiz verileri girsin ama rapor sonuclarini goremesin

2- Raporu gormesine izin verilenler raporu gorsun ama degistiremesin

Java bir method'u ya da variable'i oluşturulurken class disindan bu öğelere erişimi kisitlama veya belirlenmis datalari degistirebilme olanağı tanir.

Encapsulation (Data Hiding)

Before Encapsulation



After Encapsulation (Data Gizleme)

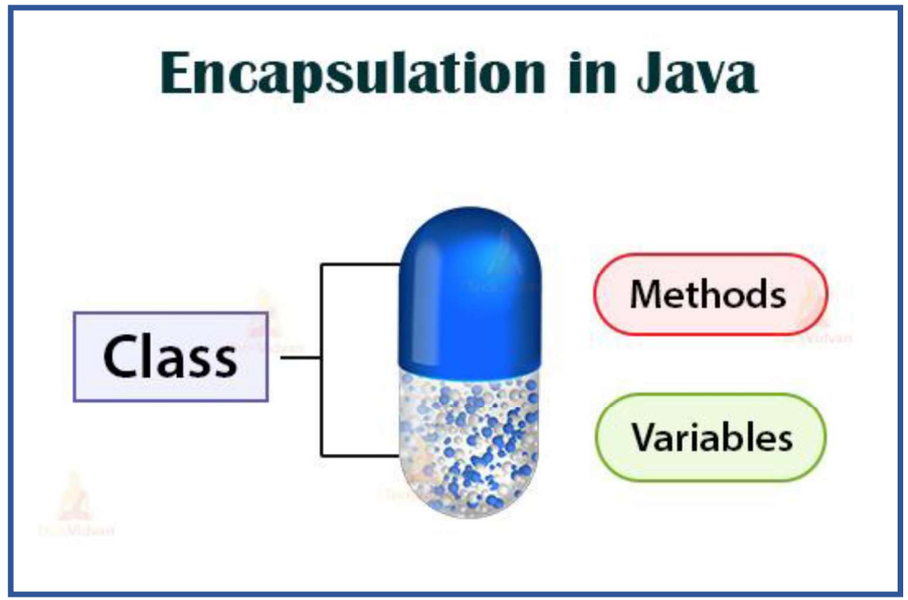
Encapsulation (Data Hiding)

Encapsulation, onemli Class uyelerini korumak icin uygulanan data saklama yontemidir.

Farkli Class'lardan erisilerek ya da yanlış kullanım sonucu kodunuzun veya onemli datalarinizin degismesini istemiyorsaniz Encapsulation ile datalarinizi koruyabilirsiniz.

Encapsule edilen variable ve method'lara sadece sizin verdiginiz oranda erisilebilir.

Encapsule edilen variable ve method'lara izin verdiginiz kisiler ulasabilir ama DEGISTIREMEZ.



Encapsulation (Data Hiding)

Datalarimizi korumak icin data'larimizi private yapabiliriz ama private yaptigimiz datalari baska Class'lar kullanamaz. Bu durum OOP consept'ine uygun olmaz.

Private yaparak KORUMA ALTINA aldigimiz Class uyeleri'ninin sadece OKUNMASINI istiyorsak getter(), DEGER ATANMASINA da izin vermek istiyorsak setter() method'larini olustururuz.

Encapsulation iki adimda yapilir:

1) Class uyelerini *(variable, method)* private yapmalisiniz.

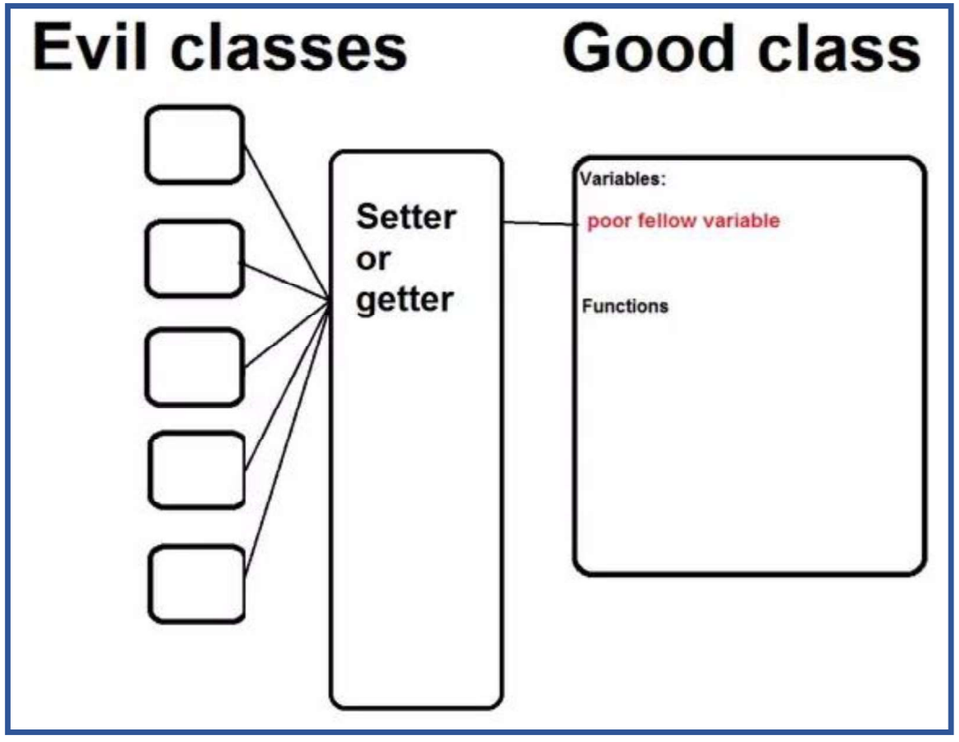
2) public olan getter() ve setter() methodlar uretmelisiniz.

Not: getter() data'yi sadece okumamiza yarar, data'da degisiklik yapamaz.

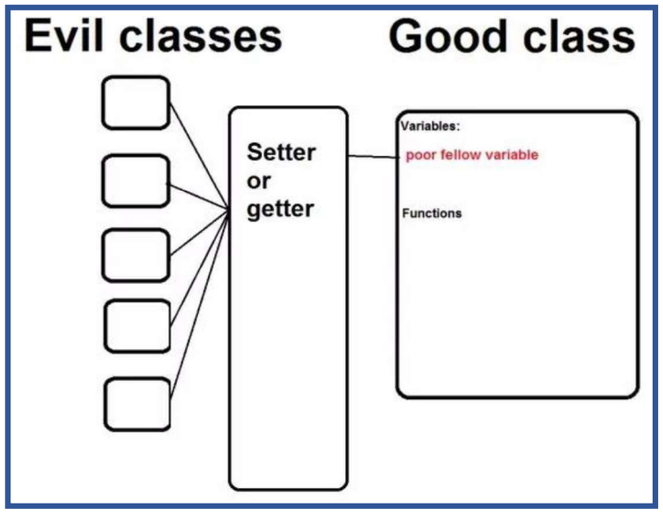
Not: setter() baska Class'larda olusturulan objeler icin data degerini degistirmemizi saglar.

Getters & Setters

Getters and setters
lead to the dark side...



Getters & Setters



Setter Methods : Encapsule ettigimiz class uyelerinin,baska class'lardan obje uretilerek deger atanmasina izin verir.

```
public class Example {

    private String tcNo= "12345678901";

    public static void main(String[] args) {

    }

    public void setTcNo(String tcNo) {
        this.tcNo = tcNo;
    }
}
```

Eger sadece setter method olusturulursa data degerleri degistirilebilir ama ilk atanan deger veya bizim atadigimiz yeni deger baska class'lardan okunamaz.

Getters & Setters (Java Beans)

Getter and setter method'lari "Java Beans" olarak da adlandirilir.

Isim verme kurallari (Naming convention)

1) Data type'lari boolean olan variable'larin getter metod isimleri "is" ile baslar.

```
private boolean happy = true;

public boolean isHappy() {
    return happy;
}
```

```
private int a=10;

public int getA() {
    return a;
}
```

Getters & Setters (Java Beans)

Isim verme kurallari (Naming convention)

```
private int a=10;
private boolean happy;

public void setA(int a) {
    this.a = a;
}

public void setHappy(boolean happy) {
    this.happy = happy;
}
```

Tekrar Sorulari

1) Encapsulation nedir?

Encapsulation, hassas datalari korumak icin kullanilan data saklama yontemidir

2) Datalari nasil saklariz?

Data'lari private access modifier kullanarak saklariz.

3) Saklanan datalara diger class'lardan ulasabiliriz?

Getter ve setter method'larini kullanarak ulasabiliriz.

4) getter( ) method'u ne yapar ?

Saklanan datalari okumamizi saglar.

5) setter ( ) method'u ne yapar ?

Saklanan datalari obje uzerinden update edebilmemizi saglar

6) immutable class nedir?

Encapsule edilen bir class'da sadece getter method'u olusturursak datalari okuyabiliriz ama degistiremeyiz. Bu tur class'lara immutable class denir.

7) setter( ) method'lari icin naming convention nedir?

Tum data turleri icin isimler "set" ile baslar.

8) getter( ) method'lari icin naming convention nedir?

Boolean data turu icin "is" ile, diger data turleri icin "get" ile baslar.

Getters & Setters

Which are methods using JavaBeans naming conventions for accessors and mutators? (Choose all that apply)

A. `public boolean getCanSwim() { return canSwim;}`

B. `public boolean canSwim() { return numberWings;}`

C. `public int getNumWings() { return numberWings;}`

D. `public int numWings() { return numberWings;}`

E. `public void setCanSwim(boolean b) { canSwim = b;}`

Getters & Setters

Which of the following are true? (Choose all that apply)

A. Encapsulation uses package private instance variables.

B. Encapsulation uses private instance variables.

C. Encapsulation allows setters.

D. Immutability uses package private instance variables.

E. Immutability uses private instance variables.

F. Immutability allows setters.