1 Sichtbarkeit von Attributen, Methoden und Klassen – Zugriffsmodifizierer

Bei Klassen, Variablen und Methoden sind folgende Angaben möglich:

Zugriffsmodifizierer	Geltungsbereich
public	Das Element kann überall angesprochen werden: - in der Klasse selbst (also in den Methoden) - in Methoden abgeleiteter Klassen (siehe "Vererbung") - in anderen Klassen
protected	Das Element kann angesprochen werden: - in der Klasse selbst (also in den Methoden) - in Methoden abgeleiteter Klassen (siehe "Vererbung") - in Klassen innerhalb desselben package
(keine Angabe) - Standard - „friendly“ - "package scoped"	Das Element kann angesprochen werden: - in der Klasse selbst (also in den Methoden) - in Klassen innerhalb desselben package
private	Das Element kann angesprochen werden: - in der Klasse selbst (also in den Methoden)

	public +	**protected #**	**default ~**	**private -**
Selbe Klasse	+	+	+	+
Selbes package	+	+	+	-
Unterklasse (extends)	+	+	-	-
Überall	+	-	-	-

2 Anwendung in der Praxis

2.1 Allgemeines

private-Elemente werden immer dann verwendet, wenn implementierungsabhängige Details versteckt werden sollen und die auch in abgeleiteten Klassen nicht sichtbar sein sollen.

protected-Elemente werden dann verwendet, wenn auf sie innerhalb der Vererbungshierarchie zugegriffen werden soll. So können z.B. Attribute aus einer Oberklasse in einer Unterklasse direkt verwendet werden, ohne den entsprechenden Getter oder Setter zu benutzen.

public-Elemente stellen den für alle sichtbaren Teil der Klasse dar. Sie bilden sozusagen ihre Schnittstelle.

friendly-Elemente verhalten sich außerhalb des package wie private und innerhalb des package wie public.

2.2 Geheimnisprinzip

Das Geheimnisprinzip (=Datenkapselung, encapsulation, information-hiding) ist ein zentrales Element der objektorientierten Programmierung.
Dahinter verbirgt sich der Gedanke, dass der *direkte, unkontrollierte* Zugriff auf die Daten (sprich Attribute) eines Objekts von außen verhindert werden soll, um klare Schnittstellen zu erhalten und Seiteneffekte zu vermeiden.
Statt dessen soll der Zugriff nur über Methoden (= definierte Schnittstellen: getter- /setter-Methoden, Accessor- / Mutator-Methoden) erfolgen. Nur wenn eine entsprechende Methode existiert, kann ein Zugriff erfolgen. In den Methoden können Zugriffe an Bedingungen geknüpft werden.

Die Umsetzung des Geheimnisprinzips in Java erfolgt über die Zugriffsmodifizierer private und public. Attribute werden im Allgemeinen als private deklariert, Methoden als public.

unkontrollierter Zugriff auf das Attribut szName der Klasse Kunde. Das soll durch das Geheimnisprinzip **verhindert** werden!

```
public class Kunde
{
1   public String szName;

    public String getName()
    {
      return (szName);
    }

    public void setName(String Name)
    {
      szName = Name;
    }
}
```

```
public class KundeGUI
{

  public static void main (String [] arg)
  {
    Kunde einKunde = new Kunde();
    einKunde.szName = "Maier";
  }

}
```

1	Das Attribut `szName` ist `public` deklariert. Somit ist es für alle Klassen sichtbar und kann von allen Methoden direkt gelesen und geschrieben werden. Das kann zu unerwünschten Seiteneffekten führen und erhöht die Abhängigkeit zwischen Klassen (→ Einschränkung der Wiederverwendbarkeit und Änderbarkeit)

besser **kontrollierter** Zugriff auf das Attribut `szName` der Klasse Kunde über eine Accessor-Methode:

```
public class Kunde
{
1   private String szName;

    public String getName()
    {
      return (szName);
    }

3   public void setName(String Name)
    {
      szName = Name;
    }
}
```

so nicht !

```
public class KundeGUI
{

  public static void main (String [] arg)
  {
    Kunde einKunde = new Kunde();
2   einKunde.szName = "Maier";
4   einKunde.setName("Maier");
  }
}
```

sondern so !

gds2 Gottlieb-Daimler-Schule 2
Technisches Schulzentrum Sindelfingen
mit Abteilung Akademie für Datenverarbeitung

1	Das Attribut `szName` ist `private` deklariert. Somit ist es für alle anderen Klassen *nicht sichtbar* und kann somit *nicht direkt angesprochen* werden.
2	Der direkte Zugriff ist nicht möglich: Compilerfehler: "`szName has private access in Kunde`"
3	Die Methode `setName` ist `public` deklariert. Somit ist sie für alle Klassen sichtbar und kann von allen Methoden direkt angesprochen werden.
4	Der Zugriff auf das Attribut `szName` geschieht über die Accessor-Methode `setName`. Diese könnte nun den übergebenen Namen z.B. auf Gültigkeit prüfen.

Vorteile:

- Schaffung klarer Schnittstellen zwischen den Klassen; dadurch verhindert man eine unnötige Kopplung zwischen Klassen

  → unterstützt die Prinzipien der Softwaretechnik: Strukturierung und Modularisierung

  → erhöht die Wiederverwendbarkeit und Änderbarkeit der Klassen