# scope

### Te gebruiken bij:

-

### Eigenschappen:

-

### Info

Binnen Java (en overigens bijna iedere programmeertaal) zijn onze variabelen gebonden aan **scope**. Je kunt scope zien als de context waarin variabelen beschikbaar of toegankelijk zijn.  
Het biedt een stukje veiligheid.  
Daarom zorgen we dat iedereen alleen toegang heeft tot de functionaliteit die ze nodig hebben. Dit noem je *The Principle of Least Access* Scoping doet dit ook.

Scope is echter iets anders dan het gebruik van access modifiers. Access modifiers kunnen we zelf controleren en toepassen, maar scope is simpelweg een programmeerprincipe

##### Acces-modifiers

* Public: Deze modifier maakt een element openbaar en toegankelijk vanuit elk deel van het programma. Openbare elementen kunnen worden geaccesseerd door andere klassen, methoden en objecten.
* Private: Met de private modifier wordt een element alleen toegankelijk binnen dezelfde klasse waarin het is gedeclareerd. Het is niet toegankelijk buiten die klasse, inclusief subklassen.
* Protected: De protected modifier maakt een element toegankelijk binnen dezelfde klasse, evenals binnen subklassen. Het is echter niet toegankelijk buiten de klassehiërarchie.
* Default (geen expliciete modifier): Als er geen access modifier wordt gespecificeerd, wordt het element toegankelijk gemaakt op het niveau van het pakket. Het element is alleen zichtbaar binnen hetzelfde pakket (directory of module).
* Internal: Deze modifier wordt vaak gebruikt in talen zoals C# en Kotlin. Een intern element is toegankelijk binnen hetzelfde assembly of module, maar niet buiten dat assembly of module.
* Protected Internal: Deze modifier combineert de functionaliteit van de protected en internal modifiers. Een protected internal element is toegankelijk binnen dezelfde assembly of module, evenals binnen subklassen, zowel binnen als buiten het assembly of de module.

Access modifiers kunnen we zelf controleren en toepassen, maar scope is simpelweg een programmeerprincipe

##### Wanneer we variabelen declareren:

Declareer je een variabele in een class? Dan kun je de variabele overal in de class bereiken;

Declareer je een variabele in een methode? Dan kun je de variabele alleen binnen de methode bereiken;

Declareer je een variabele binnen een lus? Dan kun je die variabele alleen binnen de lus bereiken;

Declareer je een variabele binnen een statement die {} gebruikt, zoals een if-else-statement? Dan kun je de variabele alleen binnen die statement gebruiken.

Scope beperkt toegankelijkheid dus van buiten naar binnen, maar niet van binnen naar buiten.

Afbeelding met tekst, schermopname, diagram, lijn

Automatisch gegenereerde beschrijving

### Voorbeeld

**Edhub**

we hebben studenten-accounts, docenten-accounts en admin-accounts. Als student kun je geen aanpassingen maken in de content van cursussen. Docenten kunnen op hun beurt geen wachtwoorden resetten voor andere gebruikers. Dat mogen alleen de admins doen.

**creditcard**

variabele expiresInYear hebben gedeclareerd. Deze variabele staat op het hoogste niveau (in de class) dus alle methoden en {}-statements kunnen deze variabele bereiken. De variabele die we binnen de methode isExpired hebben gedeclareerd, difference, is echter alleen beschikbaar *binnen* deze methode. Wanneer we nog een andere methode zouden schrijven, kan deze niet bij de variabele difference. Probeer je dit wel, dan krijg je een foutmelding.

public class CreditCard {

private int expiresInYear;

public boolean isExpired(int currentYear) {

int difference = expiresInYear - currentYear;

if (difference > 0) {

return true;

} else {

return false;

}

}

public void thisWontWork() {

System.out.println(difference); // geeft een foutmelding

}

}

Ditzelfde geldt wanneer we een loop schrijven. Bij het declareren van een for-loop, gebruiken we altijd een variabele die we i noemen. Deze i is echter alleen beschikbaar binnen de {} van de lus. We kunnen de i daarbuiten niet printen of gebruiken.

for (int i = 0; i < 10 ; i++) {

System.out.println(i); // Werkt!

}

System.out.println(i); // Geeft een foutmelding!

### aantekening