# variabelen

### Voorbeeld

### Te gebruiken bij:

-

### Eigenschappen:

-

### Info

#### Overzicht

public class VariableExamples {

public static void main(String[] args) {

// Voorbeeld van verschillende variabelen

// Integer variabele

int age = 25;

// De variabele "age" is van het type int en slaat een geheel getal op, in dit geval 25

// Floating-point variabele

double pi = 3.14159;

// De variabele "pi" is van het type double en slaat een kommagetal op, in dit geval 3.14159

// Booleaanse variabele

boolean isActive = true;

// De variabele "isActive" is van het type boolean en kan de waarden true of false bevatten

// Karakter variabele

char grade = 'A';

// De variabele "grade" is van het type char en slaat een enkel karakter op, in dit geval 'A'

// String variabele

String name = "John Doe";

// De variabele "name" is van het type String en slaat een reeks tekens op, in dit geval "John Doe"

// Array variabele

int[] numbers = {1, 2, 3, 4, 5};

// De variabele "numbers" is van het type int[] (integer array) en slaat een verzameling gehele getallen op

// Constante variabele

final int MAX\_VALUE = 100;

// De variabele "MAX\_VALUE" is een constante van het type int en kan niet worden gewijzigd nadat deze is toegewezen

// Null variabele

String city = null;

// De variabele "city" heeft de waarde null, wat aangeeft dat er geen object aan is toegewezen

// Uitvoer van variabelen

System.out.println("Leeftijd: " + age);

System.out.println("Pi: " + pi);

System.out.println("Actief: " + isActive);

System.out.println("Cijfer: " + grade);

System.out.println("Naam: " + name);

System.out.println("Getallen: " + Arrays.toString(numbers));

System.out.println("Maximale waarde: " + MAX\_VALUE);

System.out.println("Stad: " + city);

}

}

#### Verandering van type

##### Impliciete typecasting:

Impliciete typecasting treedt op wanneer je een gegevenstype naar een ander gegevenstype converteert dat een groter bereik of nauwkeurigheid heeft. Dit typecasting wordt automatisch uitgevoerd door de Java-compiler.

Hier is een voorbeeld van impliciete typecasting:

java

Copy code

int num1 = 10;

double num2 = num1;

In dit voorbeeld wordt de integer num1 impliciet gecast naar een double num2. Omdat een double een groter bereik heeft dan een int, kan de waarde van num1 zonder verlies van gegevens worden toegewezen aan num2.

##### Expliciete typecasting:

Expliciete typecasting treedt op wanneer je een gegevenstype naar een ander gegevenstype converteert dat een kleiner bereik of nauwkeurigheid heeft. Bij expliciete typecasting moet je het doelgegevenstype expliciet aangeven door gebruik te maken van haakjes.

Hier is een voorbeeld van expliciete typecasting:

java

Copy code

double num1 = 3.14;

int num2 = (int) num1;

##### parse

Bij het gebruik van de parse-methode moet je ervoor zorgen dat de tekenreeks een geldige representatie heeft van het beoogde gegevenstype. Anders kan er een NumberFormatException optreden als de tekenreeks niet correct kan worden geconverteerd.

###### String naar int:

java

String numStr = "10";

int num = Integer.parseInt(numStr);

In dit voorbeeld wordt de tekenreeks "10" geconverteerd naar een integer met behulp van Integer.parseInt(). De resulterende waarde wordt toegewezen aan de variabele num.

###### String naar double:

java

String doubleStr = "3.14";

double num = Double.parseDouble(doubleStr);

Hier wordt de tekenreeks "3.14" geconverteerd naar een double met behulp van Double.parseDouble(). De geconverteerde waarde wordt toegewezen aan de variabele num.

###### String naar boolean:

java

String boolStr = "true";

boolean boolValue = Boolean.parseBoolean(boolStr);

Dit voorbeeld toont de conversie van de tekenreeks "true" naar een boolean-waarde met behulp van Boolean.parseBoolean(). De resulterende boolean-waarde wordt toegewezen aan de variabele boolValue.

Merk op dat de parse-methode specifiek is voor elk gegevenstype (bijv. Integer, Double, Boolean) en dat de naam en het gebruik kunnen variëren afhankelijk van het gegevenstype dat je wilt converteren. Raadpleeg de Java-documentatie voor een volledige lijst van parse-methoden die beschikbaar zijn voor elk gegevenstype.

##### Cijfers achter de comma

import java.text.DecimalFormat;

Maak een instantie van de DecimalFormat-klasse aan:

java

Copy code

DecimalFormat decimalFormat = new DecimalFormat("#.##");

Gebruik de format-methode om de double-waarde af te ronden:

java

Copy code

double getal = 3.14159;

String afgerondGetal = decimalFormat.format(getal);

In dit voorbeeld zal afgerondGetal de waarde "3.14" bevatten.

Hier is een compleet voorbeeldprogramma om dit te demonstreren:

java

Copy code

import java.text.DecimalFormat;

public class Main {

public static void main(String[] args) {

DecimalFormat decimalFormat = new DecimalFormat("#.##");

double getal = 3.14159;

String afgerondGetal = decimalFormat.format(getal);

System.out.println(afgerondGetal);

}

}

Als je dit programma uitvoert, zou de uitvoer "3.14" moeten zijn.

Met behulp van de DecimalFormat-klasse kun je de gewenste precisie specificeren door het formaatpatroon aan te passen. In het voorbeeld hierboven wordt het formaatpatroon "#.##" gebruikt, wat betekent dat we twee cijfers achter de komma willen behouden. Je kunt dit patroon aanpassen aan je specifieke behoeften. Bijvoorbeeld, "#.###" zou drie cijfers achter de komma behouden.