3b. Afronden van waarden

H4. Werken met Data

1

Convert.ToInt32(value)

- Omzetten naar int door *afronding*!

2

int x = (int)value; // cast

- Omzetten naar int door *afkappen*!

3

Math.Round(value)

- Afronden naar dichtstbijzijnde geheel getal
- Idem Convert.ToInt32
- Opletten voor Banker's rounding

4

Convert.ToDouble(value)

- Omzetten naar double

Hoe afronden?

Over afronden in .NET

- `Math.Round()` en `Convert.ToInt32()`
  - Werkt voor de meeste gevallen zoals je verwacht:

```
double d1 = 4.2;
double d2 = 4.8;
String line = $"afgerond: {Math.Round(d1)} en {Math.Round(d2)} ";


Console.WriteLine(line);
```

Output: `Afgerond: 4 en 5`

Math.Round

- Kan ook extra parameter aanvaarden om aan te geven tot hoeveel cijfers na de komma moet afgerond worden:

```
double d1 = 4.12345;

String line = $"afgerond: {Math.Round(d1,1)} en {Math.Round(d1,4)} ";


Console.WriteLine(line);
```

Output: `Afgerond: 4.1 en 4.1234`

Over afronden in .NET

- `Math.Round()` en `Convert.ToInt32()`
  - Echter soms heel onverwachte resultaten bij afronden op halve waarden

```
double d1 = 4.5;
double d2 = 5.5;
String line = $"afgerond: {Math.Round(d1)} en {Math.Round(d2)} ";


Console.WriteLine(line);
```

Output: `Afgerond: 4 en 6`

Over afronden in .NET

- Verklaring
  - Er wordt hier gebruik gemaakt van een afrondingsmechanisme dat bekend staat als het bankiersalgoritme (“**banker's rounding**”)
  - Regel: een halve waarde wordt afgerond naar het dichtstbijzijnde EVEN getal
    - 4.5 → 4 (en niet 5 zoals je zou verwachten)

Zonder banker's rounding afronden

- Nuttig als je bijvoorbeeld wilt weten hoe je punten zullen zijn

```
double d1 = 9.50; double d2 = 8.50;
String line = $"afgerond: {Math.Round(d1 ,MidpointRounding.AwayFromZero)} en {Math.Round(d2,MidpointRounding.AwayFromZero)} ";
Console.WriteLine(line);
```

Output: `Afgerond: 10 en 9`

- Zoniet kreeg je 10 en 8

Demo time

- Afronden