Invoer en uitvoer in Java

K. Coolsaet

2018–2019

Vooraf

Java-bibliotheken

- Oud: `java.io`
- Nieuw: `java.nio.file`
- Gebruik `Path` i.p.v. `File`

Wanneer gebruiken...

Bijna nooit rechtstreeks, er zijn handige alternatieven.

Zie cursusnota's hoofdstukken 3, 6 en 9.

- Configuratiebestanden: gebruik 'properties'-bestanden of XML-bestanden
- Gestructureerde gegevens: gebruik een *databank*.

Overzicht

Van ingewikkeld (laag niveau) tot gemakkelijk (hoog niveau)

- (Channels en buffers (`java.nio`))
- (Directe toegang tot bestanden)
- (I/O van binaire gegevenstypes)
- Binaire streams: `InputStream` en `OutputStream`. Werken met *bytes* (`byte`).
- Standaard invoer- en uitvoerkanalen zijn binaire streams. `System.out`, `System.err`, `System.in`
- Tekst-streams: `Reader` en `Writer`. Werken met *lettertekens* (`char` en `String`).
- Gebufferde invoer (`BufferedReader`)
- Kleine bestanden als geheel (in `Files`-klasse)

(Byte) streams

- Niet alleen bestanden. Ook: netwerkverbindingen, standaard invoer- en uitvoerkanalen, arrays als streams, ...
- Leest/schrijft bytes *in volgorde*!
- Niet *tegelijk* lezen/schrijven van/op dezelfde stream
- Specifieke klassen zijn uitbreidingen van `InputStream` en `OutputStream` (zie API)
- Eerst stream *openen* met constructor
- Dan lees- of schrijfoperaties
- Altijd stream *sluiten* op het einde. (Ev. met `finally` in `try-catch`-blok.)

Readers en writers

- Specifieke klassen zijn uitbreidingen van `Reader` en `Writer` (zie API)
- Gebruik in een programma is analoog aan byte streams.
- Één letterteken kan worden voorgesteld als meerdere bytes.
- Dit hangt af van welke *codering* wordt gebruikt
- Klasse `Charset`
  - Indien niet gespecificeerd: `UTF-8`
  - Default voor machine: `Charset.defaultCharset()`
  - Opvragen op naam: `Charset.forName("ISO-8859-1")`
  - Standaardsets: `StandardCharsets.UTF_8`, `...ISO_8859_1`

Conversies

Een input stream omzetten naar een reader:

```
InputStream in = ... ;
Reader reader = new InputStreamReader (in, Charset...);
```

Een outputstream omzetten naar een writer:

```
OutputStream out = ...;
Writer writer = new OutputStreamWriter (out, Charset...);
```

Sluiten van de *laatste* stroom, sluit ook de vorige.

Praktisch

`BufferedReader`

- 'Buffert' invoer
- Heel handig: lees één lijn tegelijk
- Gebruik `Files.newBufferedReader` om te lezen van een (tekst)bestand

`PrintWriter`

- `print`-methoden gooien geen uitzonderingen op
- Mogelijkheid tot 'formatteren' van uitvoer
- Gebruik `Files.newBufferedWriter` om te schrijven
- Zet om met `new PrintWriter(...)`

`System.out` is een `Print`*`Stream`*, maar `println` gebruikt defaultcodering.

Padnamen

Interface `Path` stelt een *padnaam* voor van een bestand.

- Gebruikt waar een bestand moet opgegeven worden in `java.nio`
- = 'lijst' van namen van bovenliggende mappen en de bestandsnaam zelf.
- *Absoluut* pad vs. *relatief* pad
- Omzetten van en naar string. Java-notatie = UNIX-notatie $\neq$ Windows-notatie

Padnaamobjecten aanmaken met methoden uit `Paths`:

```
Path path
  = Paths.get ("src/prog2/nio/gedicht.txt");
  = Paths.get ("src", "prog2", "nio", "gedicht.txt");
```

(In `java.io` wordt hiervoor `File` gebruikt.)

Opvangen van uitzonderingen

Vóór Java 7

```
try {
  Writer writer = new FileWriter ("bestandsnaam");
  try {
     writer.write ("Hallo!\n");
  } catch (IOException ex) {
     ... // verwerk de uitzondering
  } finally {
     if (writer != null) {
        writer.close ();
     }
  }
} catch (FileNotFoundException fnfe) {
  ... // als het bestand niet kon worden geopend
}
```

Opvangen van uitzonderingen

Vanaf Java 7

Try-met-bronnen (*try with resources*)

```
try (Writer writer =
           new FileWriter ("bestandsnaam")) {
   writer.write ("Hallo!\n");
} catch (IOException ex) {
   ... // verwerk de uitzondering
}
```

Voor elke klasse die *AutoCloseable* implementeert.

Extra

Een bestand lezen uit het class path:

- Gebruik `getClass().getResourceAsStream(...)`
- Resultaat: `InputStream`

Lezen van een URL:

- Gebruik `URL.openStream()`

Een inputstream aanmaken uit een `File`.

- Gebruik `file.toPath()` en `Files.newInputStream(...)`

(Verbetert *JavaFX zelfstudienota's* 4.4.)