Abgrenzung von Paketen

Die Arbeitsweise der Pufferung hängt von der einzelnen Datenquelle beziehungsweise -senke ab. Während allgemeine `OutputStreams` beispielsweise einfach eine gewisse Anzahl Bytes aufsammeln, reagiert `System.out` auf Zeilenwechsel. Das ist der Grund für das zeilenweise Echo in `EndlessConsoleEcho` (Listing 1.1).[23]

Kontrolle der Ausgabepufferung

Für ein Java-Programm ist die Pufferung nicht direkt erkennbar.[24] Die Pufferspeicher selbst sind verborgen. Ausgabeseitig kann ein Java-Programm „auf Verdacht“ die Leerung des Ausgabepuffers mit einem Aufruf der Methode `flush` erzwingen:[25]

`void flush()`
: Entleert die Ausgabepuffer und gibt alle gepufferten Daten sofort aus.

Beispiel: langsame Textausgabe

Das folgende Programm `SlowLetters` gibt endlos die Buchstaben des Alphabets mit einer Rate von 10 Zeichen pro Sekunde aus:[26]

```
import java.io.*;

public class SlowLetters {
    public static void main(String... args) throws IOException, InterruptedException {
        int letter = 'A';
        while(true) {
            System.out.write(letter);
            // out.flush();
            Thread.sleep(100);
            letter++;
            if(letter > 'Z')
                letter = 'A';
        }
    }
}
```

Listing 1.7: Beobachtung der Pufferung der Standardausgabe.

Startet man das Programm, so erscheint zunächst nichts auf dem Bildschirm, weil sich die Buchstaben in einen Ausgabepuffer ansammeln. Nach einigen Sekunden

[23] Ein weiterer Grund ist die Eingabepufferung. Das Betriebssystem schickt nicht jeden Tastendruck sofort zum Programm, sondern sammelt sie erst an, bis die Eingabetaste gedrückt wird.

[24] Eine indirekte Testmöglichkeit wäre eine genaue Messung des Zeitverhaltens einzelner Methodenaufrufe.

[25] Eingabeseitig gibt es keine vergleichbare Möglichkeit.

[26] Der Aufruf `Thread.sleep(100)` blockiert für 100 Millisekunden. In dieser Zeit tut das Programm nichts, außer auf den Ablauf der Zeitspanne zu warten. Es verbraucht dabei auch keine Rechenleistung. `sleep` kann eine `InterruptedException` werfen, die hier nicht behandelt wird. Die Methode wird auf Seite 384 genauer besprochen.