if, switch, while, do-while, for

Aufgabe 1: NumberParser

Gegeben ist folgende Grammatik für die Darstellung von Gleitkommazahlen.

```
Gleitkommazahl = ["+" | "-"] Zahl "." Zahl ["E" ["+" | "-"] Zahl] "!".
```

Beachten Sie, dass nach der Grammatik jede Gleitkommazahl mit einem Rufzeichen abgeschlossen werden muss. Realisieren Sie ein Programm, das Gleitkommazahlen von der Konsole einliest und überprüft, ob es sich um eine gültige Gleitkommazahl nach dieser Grammatik handelt. Verwenden Sie dazu folgende Methoden der Klasse `In`:

- *readInt()*, um eine ganze Zahl einzulesen.
- *peek()*, um das nächste Zeichen zu lesen, ohne es vom Eingabestrom zu entfernen.
- *done()*, um zu überprüfen, ob die letzte read-Anweisung erfolgreich war.
- *read()*, um das nächste Zeichen zu lesen.

Beschreiben Sie zuerst das Programm in Prosa und realisieren Sie es anschließend in Java. Achten Sie auf eine ansprechende Formatierung des Programms. Testen Sie das Programm mit einigen interessanten Testfällen.

Lösungen zu den Aufgaben

Lösung 1: NumberParser

Prosabeschreibung

```
Schritt 1:
    PRÜFE Zeichen
    Wenn Zeichen '+' oder '-', dann LIES Zeichen
Schritt 2:
    LIES Zahl
    Wenn OK, dann weiter, sonst GIB Fehler AUS
Schritt 3:
    LIES Zeichen
    Wenn Zeichen '.', dann weiter, sonst GIB Fehler AUS
Schritt 4:
    LIES Zahl
    Wenn OK, dann weiter, sonst GIB Fehler AUS
Schritt 5:
    LIES Zeichen
    Wenn Zeichen 'E', dann weiter, sonst weiter bei Schritt 8a
Schritt 6:
    PRUEFE Zeichen
    Wenn Zeichen '+' oder '.', dann LIES Zeichen
Schritt 7:
    LIES Zahl
    Wenn OK, dann weiter, sonst GIB Fehler AUS
Schritt 8:
    LIES Zeichen
Schritt 8a:
    Wenn Zeichen '!', dann GIB gueltig AUS, sonst GIB Fehler AUS
```

Source Code

```
1  class NumberParser {
2    public static void main(String[] args) {
3      // Gleitkommazahl = ["+" | "-"] Zahl "." Zahl ["E" ["+" | "-"] Zahl] "!".
4      //                        1       2    3    4    5       6       7     8
5
6      Out.print("Bitte␣eine␣Gleitkommazahl␣eingeben:␣");
7
8      char ch = In.peek();
9      if (ch == '+' || ch == '-') {
10       In.read();
11     }
12
13     In.readInt();
14     if (!In.done()) {
15       Out.println("--2␣Fehler:␣Zahl␣erwartet");
16       return;
17     }
18
19     ch = In.read();
20     if (ch != '.') {
21       Out.println("--3␣Fehler:␣Punkt␣erwartet");
22       return;
23     }
24
25     In.readInt();
26     if (!In.done()) {
27       Out.println("--4␣Fehler:␣Zahl␣erwartet");
28       return;
29     }
30
31     ch = In.read();
32     if (ch == 'E') {
33       ch = In.peek();
34       if (ch == '+' || ch == '-') {
35         In.read();
36       }
37
38       In.readInt();
39       if (!In.done()) {
40         Out.println("--7␣Fehler:␣Zahl␣erwartet");
41         return;
42       }
43
44       ch = In.read();
45     }
46
47     if (ch != '!') {
48       Out.println("--8␣Fehler:␣Rufzeichen␣erwartet");
49       return;
50     }
51
52     Out.println("Gleitkommazahl␣gültig!");
53   }
54 }
```

Testausgaben

Tabelle 1: Testfälle

#	Beschreibung	Erwartetes Ergebnis	Status
1	+10.05!	Gleitkommazahl OK	OK
2	-10.05!	Gleitkommazahl OK	OK
3	10.05!	Gleitkommazahl OK	OK
4	0.05!	Gleitkommazahl OK	OK
5	-.05!	Keine Gleitkommazahl. Zahl nicht gefunden.	OK
6	+10!	Keine Gleitkommazahl. Dezimalpunkt nicht gefunden.	OK
7	+10.!	Keine Gleitkommazahl. Zahl nicht gefunden.	OK
8	+10.05 x	Keine Gleitkommazahl. Rufzeichen nicht gefunden.	OK
9	10.05 x	Keine Gleitkommazahl. Rufzeichen nicht gefunden.	OK
10	10.05	Keine Gleitkommazahl. Rufzeichen nicht gefunden.	OK

```
Testfall #1
+10.05!
Gültige Gleitkommazahl
Testfall #2
-10.05!
Gültige Gleitkommazahl
Testfall #3
10.05!
Gültige Gleitkommazahl
Testfall #4
0.05!
Gültige Gleitkommazahl
Testfall #5
-.05!
--2: Fehler, Zahl erwartet
Testfall #6
+10!
--3: Fehler, Punkt erwartet
Testfall #7
+10.!
--4: Fehler, Zahl erwartet
Testfall #8
+10.05
--8: Fehler, Rufzeichen erwartet
Testfall #9
10.05 x
--8: Fehler, Rufzeichen erwartet
Testfall #10
10.05
--8: Fehler, Rufzeichen erwartet
```