Übung03: if, switch, while, do-while, for

Abgabetermin: 29.03.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe09	Prosabeschreibung, Java Programm, Testfälle und Ausgaben	Java Programm	☐
Aufgabe10	Prosabeschreibung, Java Programm, Testdurchlauf und Ausgaben	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Verzweigungen (if, switch) (12 Punkte)

In folgender Aufgabe sollen Sie sich mit Verzweigungen, nämlich mit den if- und switch-Konstrukten in Java vertraut machen.

Aufgabe09: Time Formatter

Eine Uhrzeit kann nach folgender Grammatik eingegeben werden:

```
hh ":" mm [":" ss] [", " ("am" | "pm")] "."
 1  2   3   4   5     6        7         8
```

Das heißt, eine Uhrzeit wird wie folgt gebildet:

- Die Uhrzeit beginnt mit einer Stunde (hh) als Zahl zwischen 0 und 23.
- Nach einem Doppelpunkt (":") folgt die Minute (mm) als Zahl zwischen 0 und 59.
- Optional kann auf die Minute, durch einen Doppelpunkt eingeleitet, die Sekunde (ss) als Zahl zwischen 0 und 59 folgen.
- Auf die Minute (oder Sekunde) kann optional, eingeleitet durch einen Beistrich und Leerzeichen (", "), entweder "am" oder "pm" folgen. In diesem Fall muss die Stunde (hour) zwischen 1 und 12 liegen.
- Die Eingabe wird mit einem Punkt "." abgeschlossen.

Beispiele für gültige Eingaben:

- 16:17.
- 23:59:59.
- 6:20, pm.
- 03:06:00.
- 8:6:4, am.

Schreiben Sie ein Java Programm, welches eine Uhrzeit einliest und überprüft, ob die Eingabe der gegebenen Grammatik entspricht. Prüfen Sie schon beim Einlesen, ob die Angaben den gültigen Werten entsprechen (Stunde zwischen 0 und 23 oder 1 und 12 bei am/pm, Minuten zwischen 0 und 59, usw.).

Bei ungültigen Eingaben soll das Programm eine Fehlermeldung mit Angabe der Position des Fehlers ausgeben, wie in der Grammatik markiert. So soll z.B. für die Eingabe `10:63.` die Fehlermeldung `Fehler bei Position 3: Ungültige Minuten` ausgegeben werden.

Bei einer gültigen Eingabe soll die Uhrzeit anschließend in folgendem Format ausgegeben werden (verwenden Sie hierfür *Out.format*):

```
hh ":" mm ":" ss
```

`hh`, `mm` und `ss` bezeichnen jeweils zweistellige Zahlen für Stunden (00 bis 23), Minuten und Sekunden (00 bis 59). Sind Sekunden in der Eingabe nicht gegeben, nehmen Sie 00 an. Geben Sie das Datum stets im 24-Stunden-Format aus und konvertieren Sie gegebenenfalls von am/pm nach folgenden Regeln:

- 12:00 am ist 00:00 Uhr
- 12:59 am ist 00:59 Uhr
- 6:32 am ist 06:32 Uhr

- 12:00 pm ist 12:00 Uhr
- 12:59 pm ist 12:59 Uhr
- 8:15 pm ist 20:15 Uhr

Beispiele für Eingaben und formatierte Ausgaben:

Eingabe: `7:10.`	Ausgabe: `07:10:00`
Eingabe: `11:59:50, pm.`	Ausgabe: `23:59:50`
Eingabe: `12:00, am.`	Ausgabe: `00:00:00`

Hinweise:

Verwenden Sie folgende Leseoperationen der *In*-Klasse:

- *peek* prüft das nächste Zeichen, ohne es zu lesen, und überliest dabei Leerzeichen.
- *read* liest das nächste Zeichen und liefert auch Leerzeichen.
- *readInt* liest die nächste ganze Zahl.
- *done* zeigt, ob die letzte Leseoperation erfolgreich war (true oder false), z.B. ob *readInt* eine ganze Zahl lesen konnte.

Lesen Sie für "am" und "pm" mit *read* jeweils zwei Zeichen ein und verketten Sie beide zu einem String wie folgt:

```
char period1 = In.read();
char period2 = In.read();
String period = "" + period1 + period2;
```

Sie können Strings mit einer switch-Anweisung unterscheiden, z.B.:

```
switch (period) {
  case "am":
    Out.println("input was am");
    break;
  case "pm":
    Out.println("input was pm");
    break;
  …
}
```

Sie können aber Strings nicht mit == vergleichen, z.B.:

~~`if (period == "am") { … }`~~ `// Funktioniert nicht`

Sie müssen Strings mit equals auf Gleichheit überprüfen, z.B.:

```
if (period.equals("am")) { … }
```

Abzugeben:

- Prosabeschreibung
- Java Programm
- Testfälle und Ausgaben

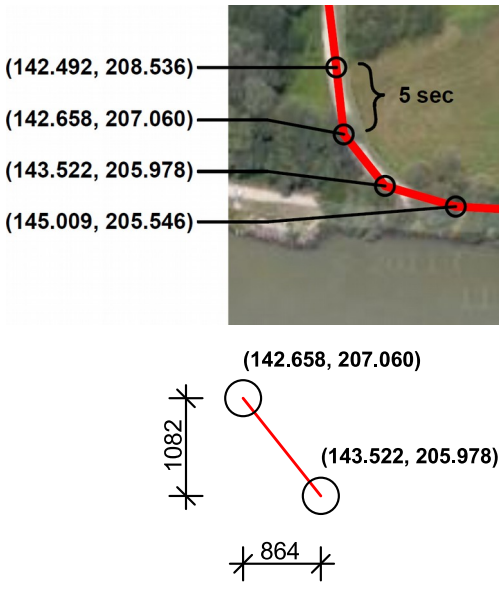
Schleifen (while, do-while, for) (12 Punkte)

In folgender Aufgabe sollen Sie sich mit Schleifen, nämlich mit den while-, do-while- und for-Konstrukten in Java vertraut machen.

Aufgabe10: Position Tracker

Ein Aufzeichnungsgerät speichert zurückgelegte Routen, indem es alle fünf Sekunden die aktuelle Position in Form von zwei Zentimeterkoordinaten in eine Datei `Tracks.txt` schreibt. Entwickeln Sie ein Programm, das die aufgezeichneten Routen aus der Datei liest und verarbeitet. Berechnen Sie für jede Route die Dauer, Länge und Durchschnittsgeschwindigkeit (siehe Abbildungen) und geben Sie diese formatiert auf der Konsole aus:

- Die Dauer der Route in Sekunden ergibt sich aus der Anzahl der Segmente (aufgezeichnete Positionen - 1) * fünf Sekunden
- Die Länge der Route in Meter wird berechnet, indem Sie jeweils die Länge zwischen zwei Positionen aufsummieren. Berechnen Sie die Länge zwischen zwei Positionen mit dem Satz des Pythagoras ($a^2 + b^2 = c^2$). Achtung: um Rundungsfehler zu vermeiden, summieren Sie zunächst alle Distanzen in Zentimeter auf und rechnen erst am Ende die Gesamtlänge von Zentimeter in Meter um.
- Die Durchschnittsgeschwindigkeit in cm/s.



Die Datei `Tracks.txt`, die die Routendaten enthält, ist wie folgt aufgebaut: Jede Zeile enthält eine Route. Die erste Zahl gibt an, wie viele Positionen für die Route aufgezeichnet wurden. Darauf folgen abwechselnd die x- und y-Koordinaten der einzelnen Positionen (in Zentimeter). Folgende Beispielroute besteht aus den vier Positionen, wie sie im Bild rechts oben dargestellt sind.

```
4 142492 208536 142658 207060 143522 205978 145009 205546
```

Beispielausgabe:

```
# java GpsTracker

=========================
   GPS Track Evaluation
=========================

Track 1:    493 m      160 sec   308 cm/sec
Track 2:   8449 m     2560 sec   330 cm/sec
Track 3:   7542 m     2695 sec   279 cm/sec
```

Hinweise:

Testen Sie Ihr Programm mit der Vorgabedatei `Tracks.txt`. Sie dürfen davon ausgehen, dass die Datei nur gültige Werte enthält. Öffnen Sie zu Beginn die Datei mit *In.open(dateiName)* und schließen Sie am Ende die Datei mit *In.close()*. Sie können mit ***In.isEOF()*** prüfen, ob Sie am Ende der Datei angelangt sind, und wenn nicht, in einer Schleife mit weiteren *read*-Befehlen durch die Datei wandern. **Verwenden Sie hierfür unbedingt die *In.java* Datei aus dem im Moodle bereitgestellten *NumberReader.zip* Archiv!**

Berechnen Sie die Wurzel einer Zahl mit der Funktion *Math.sqrt* und konvertieren Sie das Ergebnis mit einer Typumwandlung nach *int* wie folgt:

```
int x = (int) Math.sqrt(num);
```

Abzugeben:

- Prosabeschreibung
- Java Programm
- Testdurchlauf mit Beispieldatei und Ausgaben