Einführung in die Informatik: Übung 2

Alexander Waldenmaier

18. November 2020

Präsenzaufabe

- *Mathematisch*: alter <= 19
  In Java:

```
alter <= 19
```

- *Mathematisch*: $\neg$ (uhrzeit >= $t_{sonnenaufgang}$ $\wedge$ uhrzeit <= $t_{sonnenuntergang}$)
  In Java:

```
!((uhrzeit >= t_sonnenaufgang) && (uhrzeit < t_sonnenuntergang))
```

- *Mathematisch*: $\neg$istAngeschaltet
  In Java:

```
!istAngeschaltet
```

- *Mathematisch*: (zucker $\wedge$ $\neg$milch) $\vee$ ($\neg$zucker $\wedge$ milch)
  In Java:

```
zucker ^ milch
```

Aufgabe 1: Pseudocode Algorithmus

```
int ware = get_rand_int(1, 5);
int preis = get_rand_int(1, 10);
int tries = 5;
boolean success = false;
while (tries > 0) {
    println("Ware eingeben (int zwischen 1 und 5)");
    in_ware = Input();
    println("Preis eingeben (int zwischen 1 und 10)");
    in_preis = Input();

    if ((ware == in_ware) && (preis == in_preis)) {
        success = true;
        break;
    } else {
        println("Sie haben falsch geraten!");
        if (in_ware > ware) {
            println("Der Wert fuer Ware lag zu hoch!");
        } else if (in_ware < ware) {
            println("Der Wert fuer Ware lag zu niedrig!");
        }
        if (in_preis > preis) {
            println("Der Wert fuer Preis lag zu hoch!");
        } else if (in_preis < preis) {
            println("Der Wert fuer Preis lag zu niedrig!");
        }
    }
    tries--;
```

```
}
if (success) {
    println("Sie haben erfolgreich geraten!");
} else {
    println("Sie haben leider keine weiteren Versuche!");
    println("Die richtigen Werte waeren gewesen:");
    println("Ware = %d, Preis = $d", ware, preis);
}
```

Aufgabe 2: Datentypen

- Das „kommt drauf an“ - generell ist bei der Zeitmessung wohl eher der Datentyp *float*, vielleicht bei höheren Präzisionsanforderungen auch *double* geeignet. Dies ist insbesondere der Fall, wenn beispielsweise die Systemuhr in der Einheit „Sekunden“ ausgelesen wird und man beispielsweise Benchmarks zur Programmlaufzeit durchführt. Bei wissenschaftlichen Berechnungen ist die Verwendung von Kommazahlen selbstverständlich. Gibt die Uhr, mit der gemessen wurde, allerdings nur Ganzzahlen heraus (beispielsweise in der Einheit „Mikrosekunden“) muss auch auf Programmseite etwa ein *int* oder besser *long* verwendet werden, um diese Eigenschaft wiederzuspiegeln. Damit ist dann auch klar, dass Bruchteile einer Mikrosekunde schlicht nicht erfasst werden können.
- Hier verwendet man eindeutig den Datentyp *char*.
- Hausnummern sind im Normalfall Ganzzahlen mit überschaubaren Größen, also am ehesten ein *int* oder sogar *short*. Allerdings gibt es auch Zusätze, z.B. Hausnummer "12a". Aus diesem Grund ist die Wahl eines *char*-arrays (*string*) wahrscheinlich besser.
- Je nach zu erwartender Anzahl an Nutzern funktionieren *short* oder *int*.
- Da es nur zwei mögliche Ausgänge gibt (wir nehmen an, dass die Münze nicht auf ihrer Kante landen kann), reicht sogar der Datentyp *boolean*.
- Ein Bruch kann in Form zweier Zahlen vom Typ *int* oder ähnlich wiedergegeben werden. Der Dezimalwert hingegen kann nur mit *float* oder *double* dargestellt werden (abgängig von der geforderten Präzision).

Aufgabe 3: Boolsche Ausdrücke

```
  (A && !A) || !(5 != 6 ^ (1 > 42) == (23 < 23))
= false || !(true ^ (false) == (false))
= false || !(true ^ true)
= false || !(false)
= false || true
= true
```