# Übungsblatt 4

Musterlösung

## Aufgabe 1 Müh(l)en des Programmierens

### Aufgabe 1.1 Dokumentation

```java
import static java.lang.Math.abs; // Absolutwert
import static java.lang.Math.max; // Maximum zweier Zahlen

/**
 * Diese Klasse implementiert eine Überprüfung gültiger Züge auf einem Mühle-artigen
 * Feld. Das Feld besteht dabei aus zwei ineinander geschachtelten Quadraten. Eines
 * hat die Seitenlänge 3, das andere die Seitenlänge 5. Auf den beiden Quadraten darf
 * sich die Figur frei bewegen. Zwischen den beiden Quadraten darf aber nur in der
 * Mitte der Quadratkanten gewechselt werden.
 *
 * @author Thomas Röfer
 */
class Rules
{
    /** Die x-Koordinate der Mitte beider Quadrate, also des Spielfelds. */
    private final int centerX;

    /** Die y-Koordinate der Mitte beider Quadrate, also des Spielfelds. */
    private final int centerY;

    /**
     * Erzeugt ein neues Regel-Objekt.
     * @param x Die x-Koordinate der Mitte beider Quadrate, also des Spielfelds.
     * @param y Die y-Koordinate der Mitte beider Quadrate, also des Spielfelds.
     */
    Rules(final int x, final int y)
    {
        centerX = x;
        centerY = y;
    }

    /**
     * Überprüft, ob ein Zug legal ist, also den erlaubten Bewegungen auf dem
     * oben beschriebenen Spielfeld entspricht. Ein Zug, bei dem sich die Figur
     * nicht bewegt, ist nicht erlaubt.
     * @param object Das Spielobjekt, das den Zug ausführen soll.
     * @param dirX Die Schritte in x-Richtung. Dürfen nur -1, 0 oder 1 sein.
     * @param dirY Die Schritte in y-Richtung. Dürfen nur -1, 0 oder 1 sein.
     * @return Darf diese Bewegung ausgeführt werden?
     */
    boolean isLegal(final GameObject object, final int dirX, final int dirY)
    {
        // Es ist nicht erlaubt, sich nicht zu bewegen.
        if (dirX == 0 && dirY == 0) {
            return false;
        }

        // Es darf nur entweder in x- oder in y-Richtung gelaufen werden.
        if (dirX != 0 && dirY != 0) {
            return false;
        }

        // Schritte dürfen nicht größer als 1 sein.
        if (abs(dirX) > 1 || abs(dirY) > 1) {
            return false;
        }

        // Es ist erlaubt, auf demselben Quadrat (Level) zu bleiben.
```

```
59          final int fromLevel = maxAbs(object.getX(), object.getY());
60          final int toLevel = maxAbs(object.getX() + dirX, object.getY() + dirY);
61          if (fromLevel == toLevel) {
62              return true;
63          }
64
65          // Ab hier geht es um den Wechsel zwischen Quadraten (Leveln).
66          // Ein Wechsel ist nur in der Mitte einer Kante erlaubt.
67          if (object.getX() != centerX && object.getY() != centerY) {
68              return false;
69          }
70
71          // Das äußere Quadrat muss die Nummer 2 haben, ansonsten ist der Zug
72          // illegal. Da die Quadrate benachbart sind, hat das andere dann den
73          // Level 1.
74          final int outerLevel = max(fromLevel, toLevel);
75          return outerLevel == 2;
76      }
77
78      /**
79       * Die Methode bestimmt, auf welchem Quadrat sich eine Position befindet.
80       * @param x Die x-Koordinate der Position.
81       * @param y Die y-Koordinate der Position.
82       * @return Die Nummer des Quadrats. Dies ist der größere der x- und
83       *         y-Abstände von der Mitte.
84       */
85      private int maxAbs(final int x, final int y)
86      {
87          return max(abs(x - centerX), abs(y - centerY));
88      }
89  }
```

## Aufgabe 1.2 Fehlersuche

Beim Test des Programms fallen zwei Fehler auf. Zum einen ist es unmöglich, das innere Quadrat an den dafür vorgesehenen Orten nach außen hin zu verlassen. Außerdem kann die Figur von oben und unten in die Mitte des Spielfelds bewegt werden, also das Gitter verlassen.

Ein geeigneter Ansatz zur Fehlersuche ist es, die Figur an eine der Fehlerstellen zu bewegen, einen Haltepunkt an den Anfang von *isLegal* zu setzen und dann die Pfeiltaste in die entsprechende Richtung zu drücken. Danach kann *isLegal* Schritt für Schritt ausgeführt werden. Durch die Anzeige der lokalen Variablen und der Erkenntnis, mit welchem *return* die Methode verlassen wird, ist nachvollziehbar, was passiert.

Die Figur wird oben in der Mitte platziert und die Pfeiltaste nach oben gedrückt. In der Methode sind dann *dirX* 0 und *dirY* -1. Der Code kann nun bis in die letzte Zeile ohne besondere Vorkommnisse mit *Schritt über* durchlaufen werden. In der letzten Zeile fällt allerdings auf, dass *outerLevel* eigentlich 2 sein müsste, wenn es die größere Zahl der beiden Konstanten *fromLevel* und *toLevel* sein sollte. Statt dessen ist die Konstante aber 1, wodurch der abschließende Vergleich *false* ergibt, obwohl an dieser Stelle *true* zurückgegeben werden müsste, denn eigentlich sollte der Zug ja legal sein.

Die Methode *maxAbs* berechnet ihrem Namen nach das Maximum zweier Absolutwerte. Damit sollte sie eigentlich zum Bestimmen der größeren zweier positiver Zahlen verwendet werden können. Ein Blick in die Implementierung zeigt aber, dass zusätzlich noch die Koordinaten der Spielfeldmitte von beiden Zahlen abgezogen werden. Dadurch wird aus $max(abs(fromLevel - centerX), abs(toLevel - centerY))$ nun $max(abs(1-2), abs(2-2)) = max(abs(-1), abs(0)) = max(1,0) = 1$. Gebraucht würde hier aber einfach $max(fromLevel, toLevel)$, was tatsächlich 2 wäre. Dies wurde im Listing in Aufgabe 1.1 in Zeile 74 bereits korrigiert.

Ein weiterer Test zeigt nun, dass ein Wechsel zwischen den Quadraten nun oben und unten möglich ist und die Figur auch nicht mehr in die Mitte laufen kann. Allerdings ist ein Wechsel links und rechts immer noch nicht möglich. Also wird das oben beschriebene Vorgehen noch

einmal angewendet, diesmal mit der Figur an der Abzweigung mittig links und dem Versuch, vom inneren in das äußere Quadrat zu wechseln. Hierbei stellt sich heraus, dass das vorletzte *return* erreicht wird. Eine nähere Prüfung der Bedingung darüber zeigt, dass im rechten Teil die x-Koordinate des Spielobjekts mit der y-Koordinate der Mitte verglichen wird, was sicher ein Fehler ist und dazu geführt hat, dass der Gesamtausdruck wahr wurde. Würde hier mit *object.getY()* verglichen, wäre dieses *return* nicht erreicht worden. Dieser Fehler wurde im Listing in Aufgabe 1.1 in Zeile 67 ebenfalls korrigiert.

Ein dritter Test ergibt nun, dass sich die Figur wie erwartet bewegen lässt.

Ein Kritikpunkt an der Implementierung ist, dass der Methodenname *maxAbs* irreführend ist, da eben nicht nur das Maximum zweier Absolutwerte bestimmt wird. Ein besserer Name wäre z.B. *calcLevel* gewesen.

## Aufgabe 2 Quiz

a) *Wie kann* a == false *maximal kurz dargestellt werden?*

Entsprechend der Wahrheitstabelle aus Vorlesung 6, Folie 11: *!a*.

b) *Warum ist in booleschen Ausdrücken der Vergleich* == true *überflüssig?*

Wie der Wahrheitstabelle für == aus Vorlesung 6, Folie 11 entnommen werden kann, ist das Ergebnis eines Vergleichs mit *true* identisch zu dem verglichenen Wert selbst. Deshalb ist ein solcher Vergleich überflüssig.

c) *Wie kann* !(a > b) *maximal kurz dargestellt werden?*

Das Gegenteil von $a > b$ ist $a \leq b$. Also ist die kurze Darstellung in Java: $a <= b$.

d) *Wie kann*

```
1 if (a) {
2 }
3 else {
4     b();
5 }
```

*maximal kurz dargestellt werden?*

*b()* wird immer genau dann ausgeführt, wenn *a* falsch ist. Das kann auch direkt hingeschrieben werden:

```
1 if (!a) {
2     b();
3 }
```

e) *Was liefert nach Ausführung des folgenden Codes* a.getRotation() *und warum?*

```
1 final GameObject a = new GameObject(0, 0, 0, "player");
2 a.setRotation(1);
3 final GameObject b = a;
4 b.setRotation(2);
```

*a.getRotation()* liefert 2 zurück, da *a* und *b* auf dasselbe Objekt zeigen und *b.setRotation(2)* die Rotation für dieses Objekts auf 2 gesetzt hat.

f) *Was machen die Zeilen 3–5 (als Ganzes)?*

```
1 int a = 17;
2 int b = 4;
3 a = a + b;
4 b = a - b;
5 a = a - b;
```

Die drei Zeilen vertauschen $a$ und $b$. $a$ wird in Zeile 3 auf die Summe beider Zahlen gesetzt. Da $b$ zuerst noch unverändert bleibt, lässt sich der ursprüngliche Wert von $a$ immer noch als $a - b$ berechnen. Dies wird in Zeile 4 auch gemacht und $b$ zugewiesen, d.h. $b$ hat dann den ursprünglichen Wert von $a$. Dadurch lässt sich nun der ursprüngliche Wert von $b$ als $a - b$ bestimmen, was in Zeile 5 auch gemacht und an $a$ zugewiesen wird. Somit hat $a$ nun den ursprünglichen Wert von $b$ und $b$ den ursprünglichen Wert von $a$, d.h. sie wurden vertauscht.