OOP

for einige Male

Henrik Horstmann

23. September 2014

Inhaltsverzeichnis

1 Bedeutung der Symbole....1
2 Sich wiederholende Abläufe....2
3 Die for-Schleife....3
4 Optimieren des Spiels Mäuse fangen....7
5 Das Entenspiel....7
5.1 Klassenstruktur des Programms....8
5.2 Die Klasse Reagenzglas....8
5.3 Die Klasse Game....10
5.4 Die Klasse Program....13
6 Übungen....14
6.1 Luftballons picken....14
7 Lösungen zu den Aufgaben....20
7.1 Lösung zu Aufgabe 1....20
7.2 Lösung zu Aufgabe 2 (a)....21
7.3 Lösung zu Aufgabe 2 (b)....22
7.4 Lösung zu Aufgabe 2 (c)....23
7.5 Lösung zu Aufgabe 3 (a)....24
7.6 Lösung zu Aufgabe 3 (b)....25
7.7 Lösung zu Aufgabe 4 (a)....26
7.8 Lösung zu Aufgabe 4 (b)....27
7.9 Lösung zu Aufgabe 4 (c)....28
7.10 Lösung zu Aufgabe 5....29
7.11 Lösung zu Aufgabe 6....30
7.12 Lösung zu Aufgabe 7 (a)....31
7.13 Lösung zu Aufgabe 7 (b)....32
7.14 Lösung zu Aufgabe 8 (a)....33
7.15 Lösung zu Aufgabe 8 (b)....34
7.16 Lösung zu Aufgabe 8 (c)....35
7.17 Lösung zu Aufgabe 8 (d)....36
7.18 Lösung zu Aufgabe 8 (e)....37

7.19 Lösung zu Aufgabe 8 (f)....38
7.20 Lösung zu Aufgabe 8 (g)....39
7.21 Lösung zu Aufgabe 9 (a)....40
7.22 Lösung zu Aufgabe 9 (b)....41
7.23 Lösung zu Aufgabe 9 (c)....43
7.24 Lösung zu Aufgabe 9 (d)....45
7.25 Lösung zu Aufgabe 9 (e)....47
7.26 Lösung zu Aufgabe 10 (a)....49
7.27 Lösung zu Aufgabe 10 (b)....50
7.28 Lösung zu Aufgabe 11 (a)....51
7.29 Lösung zu Aufgabe 11 (b)....53
7.30 Lösung zu Aufgabe 11 (c)....55
7.31 Lösung zu Aufgabe 12....57
7.32 Lösung zu Aufgabe 13....59
7.33 Lösung zu Aufgabe 14....62

1 Bedeutung der Symbole

Symbol	Beschreibung
	Dieses Symbol kennzeichnet einen Tipp oder Hinweis.
	Aufgepasst, hier passieren leicht Fehler oder es ist mit Schwierigkeiten zu rechnen.
	Hier erfahren Sie, wie es gemacht wird.
	Es folgt eine Aufgabe die zu lösen ist.

© 2014 Henrik Horstmann

2 Sich wiederholende Abläufe

In Computerprogrammen gibt es viele Abläufe, die wiederholt ausgeführt werden müssen. Das Computerspiel *MaueseFangen.oop* aus dem Tutorial *Felder von Objekten* macht dies deutlich. In der Klasse *Game* enthält der Konstruktor folgenden Code:

```
public Game ()
{
   // Anfang Variablen
   // Ende Variablen

   super("Mäuse fangen",661,472);
   super.loadBackground("Hintergrund_gelb.png");
   punkteAnzeige = new StageObject(330,25,100);
   super.addStageObject(punkteAnzeige);
   mls = new Mauseloch[5];
   mls[0] = new Mauseloch(100,380);
   super.addStageObject(mls[0]);
   mls[1] = new Mauseloch(218,380);
   super.addStageObject(mls[1]);
   mls[2] = new Mauseloch(336,380);
   super.addStageObject(mls[2]);
   mls[3] = new Mauseloch(454,380);
   super.addStageObject(mls[3]);
   mls[4] = new Mauseloch(572,380);
   super.addStageObject(mls[4]);
   katze = new Katze(330,236);
   super.addStageObject(katze);
   maus = new Integer();
   zeit = new Integer(15);
   comp = new Computer();
   anzahlMaeuse = new Integer();
   anzahlGefangen = new Integer();
}
```

Die grau hinterlegten Zeilen beinhalten den gleichen Code (bis auf der Index für das Feld und die x-Position des Mauselochs). Wie leicht zu sehen ist muss der Code 5 Mal ausgeführt werden.

Im obigen Beispiel wird dazu der Code fünf Mal eingegeben. Dies hat gravierende Nachteile:

- Ist der Code fehlerhaft oder muss er verändert werden, so ist die Korrektur an 5 Stellen vorzunehmen. Die macht nicht nur viel Arbeit, sondern kann schnell zu erneut fehlerhaftem Code führen, wenn an einer Stelle die Korrektur vergessen wird.
- Soll das Programm dahin erweitert werden, dass 10 oder mehr Mäuselöcher benötigt werden, dann ist für jedes neue Mauseloch dr Code zu duplizieren.

Im folgenden wird gezeigt, wie sich solche Code-Redundanzen vermeiden lassen.

3 Die for-Schleife

Ist bekannt, wie oft ein Code ausgeführt werden soll, so eignet sich dazu eine sogenannte for-Schleife.

Im obigen Beispiel wird der Code

```
mls[0] = new Mauseloch(100,380);
super.addStageObject(mls[0]);
```

5 Mal ausgeführt. Bei jeder Ausführung muss der Index und die x-Position des Mauslochs angepasst werden.

Für den Indexzähler kann eine Variable vom Typ *Integer* verwendet werden, die nach jedem Durchlauf um 1 erhöht wird:

```
Integer i;
...
i = new Integer();
mls[i] = new Mauseloch(100,380);
super.addStageObject(mls[i]);
i = i.add(1);
mls[i] = new Mauseloch(218,380);
super.addStageObject(mls[i]);
i = i.add(1);
mls[i] = new Mauseloch(336,380);
super.addStageObject(mls[i]);
...
```

Aufgabe 1:

Bestimme Sie die Gesetzmäßigkeit der x-Positionen der Mäuselöcher.

Entsprechend dem Beispiel für den Index soll in einer Variablen *xp* die Postion der Mäuselöcher berechnet. Geben Sie eine Berechnung an.

Hier geht es zur Lösung auf Seite 20.

Nun unterscheiden ich die Codefragmente nicht mehr und es kann eine Schleife ins Spiel kommen, die diesen Code 5 Mal ausführt.

Deklaration einer for-Schleife:

```
for (Integer i = new Integer();
     i.lowerThan(5);
     i = i.add(1))
{
   // Code der wiederholt ausgeführt
   // werden soll
}
```

Dabei ist

`Integer i = new Integer();`	Initialisierung des Zählers
`i.lowerThan(5);`	Schleifenbedingung
`i = i.add(1)`	Inkrement des Zählers

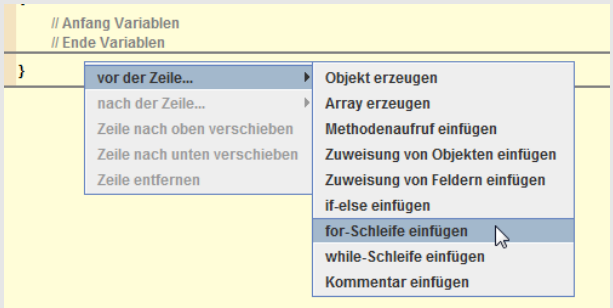
Bevor der Code im Schleifenrumpf das erste Mal ausgeführt wird, wird eine Variable als Zähler angelegt und mit einem Startwert versehen. Im obigen Beispiel heißt dieser Zähler *i* und wird mit dem Startwert 0 versehen.

Bevor der Code im Schleifenrumpf ausgeführt wird, testet die Schleife die Bedingung, im obigen Beispiel "*Ist i<5?*". Ist die Bedingung wahr, so wird der Schleifenrumpf ausgeführt, andernfalls setzt das Programm direkt nach der for-Schleife fort.

Nachdem der Schleifenrumpf ausgeführt wurde, wird das Inkrement durchgeführt. Im obigen Beispiel wird der Wert von *i* um 1 erhöht. Die Schleifenbedingung wird erneut getestet. Je nach Ergebnis wird der Schleifenrumpf erneut ausgeführt, oder mit dem Code nach der for-Schleife fortgefahren.

<u>Die for-Schleife in Henrik´s OOP Lernumgebung:</u>

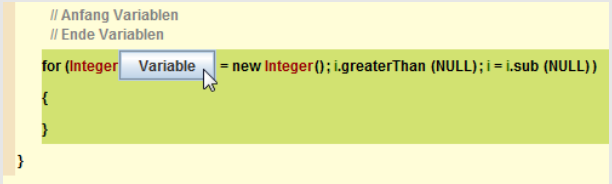
Den Mauszeiger an der Stelle positionieren, an der eine for-Schleife eingefügt werden soll. Mit der rechten Maustaste das Kontextmenü öffnen und den Eintrag *vor der Zeile ...*, bzw. *nach der Zeile ...* auswählen. In dem sich öffnenden Untermenü den Eintrag *for-Schleife einfügen* auswählen.



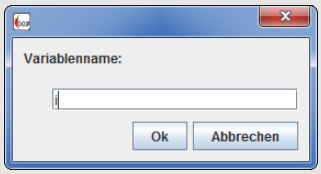
In dem Schleifenkopf kann anschließend die Schleifensteuerung programmiert werden:

1. Auswahl des Namens des für den Zähler:

   Den Mauszeiger auf die Deklaration des Zählers positionieren. Ein Button mit de Aufschrift *Variable* erscheint:

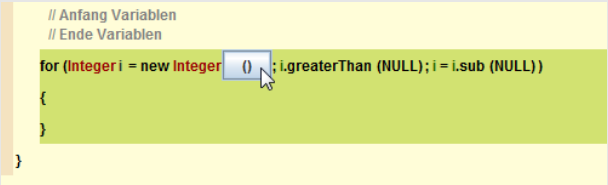


Ein Mausklick öffnet ein Dialogfenster, in dem der Name des Zählers verändert werden kann.

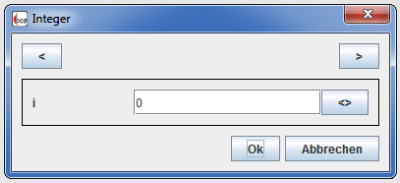


2. Festlegen des Startwerts für den Zähler:

   Den Mauszeiger auf die Parameterliste des Konstruktors für den Zähler positionieren. Ein Button erscheint.

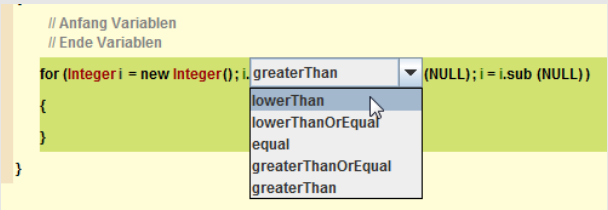


   Ein Mausklick auf den Button öffnet ein Dialogfenster mit dem der Aufruf des Konstruktor konfiguriert wird. Hier kann der Startwert für den Zähler festgelegt werden.

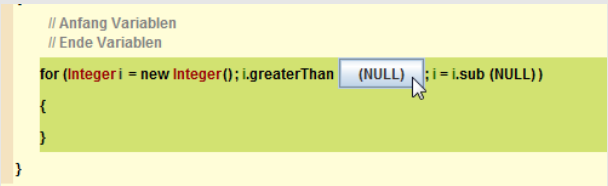


3. Schleifenbedingung festlegen:

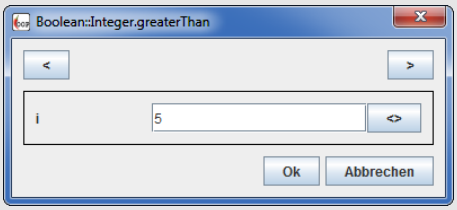
   Den Mauszeiger auf den Methodenaufruf in der Schleifenbedingung positionieren. Eine Dropdownliste mit allen zur Verfügung stehenden Methoden erscheint, die für eine Bedingung geeignet sind.



   Den Mauszeiger auf die Parameterliste des Methodenaufrufs positionieren. Ein Button erscheint.

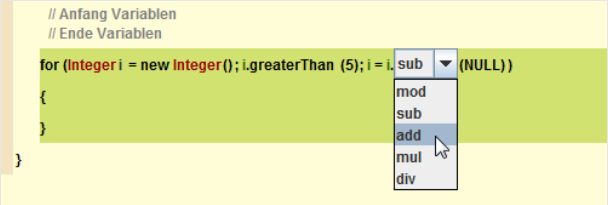


Mit einem Mausklick auf den Button erscheint ein Dialogfenster, mit dem der Methodenaufruf parametrisiert werden kann.

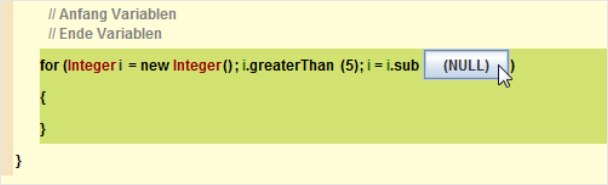


4. Inkrement festlegen:

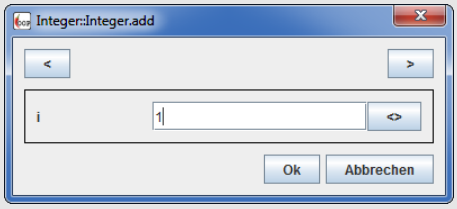
   Den Mauszeiger auf den Methodenaufruf im Schleifen-inkrement positionieren. Eine Dropdownliste mit allen zur Verfügung stehenden Methoden erscheint, die für ein Inkrement geeignet sind.



   Den Mauszeiger auf die Parameterliste des Methodenaufrufs positionieren. Ein Button erscheint.



   Mit einem Mausklick auf den Button erscheint ein Dialogfenster, mit dem der Methodenaufruf parametrisiert werden kann.



Im Anschluss kann der Schleifenrumpf nach bekannter Methode programmiert werden.

Bei der Programmierung von Schleifen ist darauf zu achten, dass sie auch beendet werden.

```
for (Integer i = new Integer(); i.equal(i); i = i.add(1))
```

Diese Schleife wird endlos ausgeführt, da die Bedingung immer wahr ist!

4 Optimieren des Spiels Mäuse fangen

Als erste Übung soll das Spiel Mäusefangen optimiert werden.

Aufgabe 2:

Öffnen Sie das Projekt *MauseFangen.oop*. In der Klasse *Game* soll im Konstruktor das Erzeugen von Mäuselöchern und platzieren auf der Stage in einer Schleife optimiert werden.

Hier geht es zur Lösung auf Seite 21.

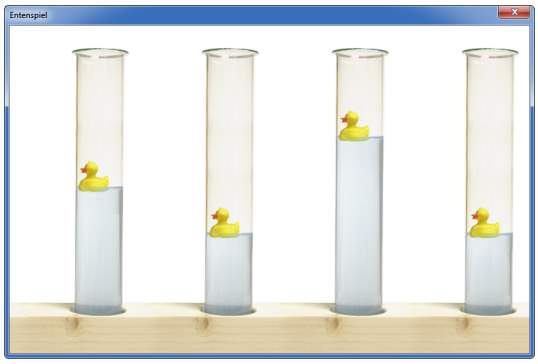
Im Konstruktor der Klasse *Katze* werden 6 Grafiken geladen, deren Namen bis auf eine Nummer alle gleich sind. Das Laden de Grafiken kann mit einer Schleife optimiert werden.
Nehmen wir an dass die Nummer der Grafik durch den Wert der Variablen *i* bestimmt wird (Katze01.0*i*.png). Notieren Sie sich auf einem Blatt ein Codefragment, das den Namen der Grafik in Abhängigkeit von *i* zusammensetzt und in einer Variablen *gName* vom Typ String abspeichert.

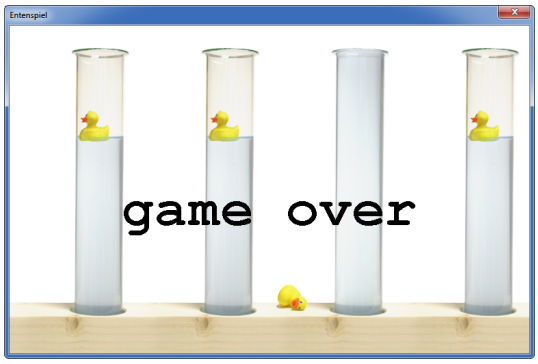
Hier geht es zur Lösung auf Seite 22.

Optimieren Sie das Laden der Grafiken mit einer for-Schleife.

Hier geht es zur Lösung auf Seite 23.

5 Das Entenspiel





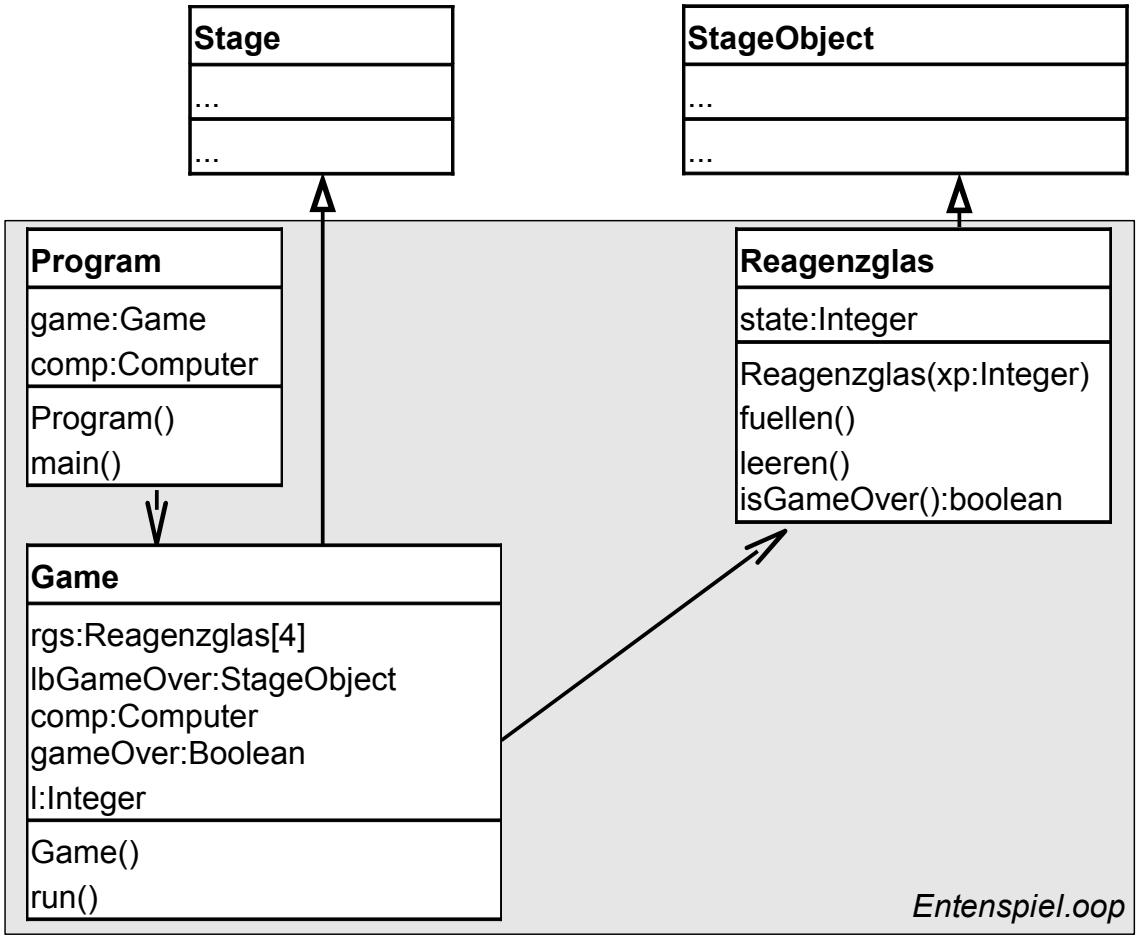
Bei diesem Spiel schwimmen in 4 Reagenzgläsern Enten. Unregelmäßig steigt der Pegel in den einzelnen Reagenzgläsern. Der Spieler muss rechtzeitig das Wasser wieder ablassen. Gelingt dies nicht, so fällt die Ente heraus und das Spiel ist beendet.

Die Projektdatei *Entenspiel.demo.oop* enthält eine Demoversion zum ausprobieren.

© 2014 Henrik Horstmann

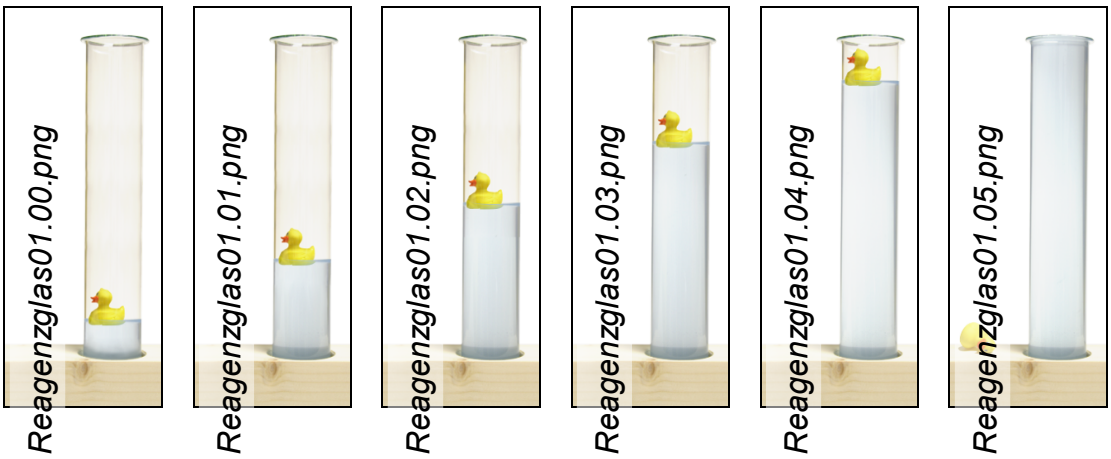
5.1 Klassenstruktur des Programms

UML-Diagram



5.2 Die Klasse Reagenzglas

Die Klasse *Reagenzglas* repräsentiert das Reagenzglas mit der Ente. Sie kann 6 verschiedene Zustände annehmen:



Aufgabe 3:

Legen Sie ein neues Projekt *Entenspiel.oop* an und erzeugen Sie eine neue Klasse *Reagenzglas*, die von der Klasse *StageObject* erbt.

Hier geht es zur Lösung auf Seite 24.

Erweitern Sie die Klasse *Reagenzglas* um alle Attribute und Methoden aus dem UML-Diagramm und passen die Parameterliste des Konstruktors an.

Hier geht es zur Lösung auf Seite 25.

Der Konstruktor

Der Parameter *xp* bestimmt die horizontale Position des Reagenzglas auf der *Stage*. Das Attribute *state* hält den Zustand fest, in dem sich das Reagenzglas befindet (*state*=5 bedeutet, dass die Ente herausgefallen ist).

Aufgabe 4:

Positionieren Sie das Reagenzglas an die Koordinaten (xp,240). Die Skalierungen der Grafiken soll 100 % betragen

Hier geht es zur Lösung auf Seite 26.

Laden Sie mit Hilfe einer for-Schleife alle Grafiken für das Reagenzglas.

Hier geht es zur Lösung auf Seite 27.

Abschließend erhält das Attribut *state* den Wert 0. Die sichtbare Grafik wird in Abhängigkeit von *state* gewählt.

Hier geht es zur Lösung auf Seite 28.

Die Methode fuellen

Mit dem Aufruf der Methode *fuellen* steigt de Pegel im Reagenzglas. Sollte das Reagenzglas voll sein, fällt die Ente heraus.

Aufgabe 5:

In der Methode *fuellen* soll der Wert des Attributs *state* um eins erhöht werden. In Abhängigkeit von dem neuen Wert im Attribut *state* wird das sichtbare Bild im Reagenzglas gewählt.

Hier geht es zur Lösung auf Seite 29.

© 2014 Henrik Horstmann

Die Methode leeren

Beim Aufruf der Methode *leeren* wird der Pegelstand zurück auf den möglichst niedrigen gesetzt.

Aufgabe 6:

Setzen Sie in der Methode *leeren* das Attribut zurück auf den Wert 0. In Abhängigkeit von dem neuen Wert im Attribut *state* wird das sichtbare Bild im Reagenzglas gewählt.

Hier geht es zur Lösung auf Seite 30.

Die Methode isGameOver

Mit der Methode *isGameOver* kann getestet werden, ob die Ente bereits herausgefallen ist.

Aufgabe 7:

Wie kann mit Hilfe des Attributs *state* festgestellt werden, ob die Ente bereits herausgefallen ist?

Hier geht es zur Lösung auf Seite 31.

Erweitern Sie die Methode *isGameOver* so, dass sie TRUE zurückliefert, wenn die Ente bereits herausgefallen ist. Andernfalls soll sie FALSE zurückgeben.

Hier geht es zur Lösung auf Seite 32.

5.3 Die Klasse Game

Mit der Klasse *Game* wird der Spielablauf gesteuert.

Der Konstruktor

Im Konstruktor werden alle Vorbereitungen getroffen. Es werden die Reagenzgläser erzeugt und auf der *Stage* platziert, sowie sämtliche Attribute mit Startwerten versorgt.

Aufgabe 8:

Legen Sie eine neue Klasse *Game* an, die von der Klasse *Stage* erbt und erweitern Sie die Klasse mit Attributen und Methoden entsprechend dem UML-Diagramm.

Hier geht es zur Lösung auf Seite 33.

© 2014 Henrik Horstmann

Die *Stage* soll den Namen *Entenspiel* tragen und eine Größe von 760×480 Pixel haben.

Hier geht es zur Lösung auf Seite 34.

Als nächstes soll ein Feld für 4 Reagenzgläser erzeugt und dem Attribut *rgs* zugewiesen werden, sowie ein Objekt für das Attribut *comp* erzeugt werden.

Hier geht es zur Lösung auf Seite 35.

Hinweis:

Die Klasse *Computer* enthält eine Methode, die Größe eines Feldes nachgeschaut werden kann.

```
Integer Computer.getArrayLength(<?>[])
```

Ein Aufruf `comp.getArrayLength(rgs)` von liefert das Ergebnis 4.

Diese Methode ist nützlich, wenn z.B. die Länge eines Feldes nicht bekannt ist.

Bestimmen Sie mit Hilfe der Methode *getArrayLength* die Größe des Feldes *rgs* und speichern diese im Attribut *l*.

Hier geht es zur Lösung auf Seite 36.

Das erste Reagenzglas soll horizontal bei 95 positioniert werden. Der Abstand zwischen den einzelnen Reagenzgläsern beträgt 190 Pixel.

Fügen Sie in den Konstruktor Code ein, der das Feld *rgs* mit Reagenzgläsern füllt und diese auf der *Stage* platziert. Verwenden Sie dazu eine for-Schleife.

Hier geht es zur Lösung auf Seite 37.

Das Attribut *lbGameOver* soll ein neues Objekt erhalten. *lbGame* soll den Text "game over" in Schriftgröße 80 erhalten. Der Text soll zentriert ausgerichtet sein. Ferner soll *lbGame* nicht sichtbar auf der *Stage* an der Position (380,240) platziert werden.

Hier geht es zur Lösung auf Seite 38.

Abschließend wird im Konstruktor das Attribut *gameOver* auf den Wert TRUE gesetzt.

Hier geht es zur Lösung auf Seite 39.

Die Methode run

In der Methode *run* findet die Ablaufsteuerung statt. Es wird entschieden, in welchem Reagenzglas der Pegel steigt und es wird auf Benutzereingaben reagiert.

Aufgabe 9:

Als erstes soll in der Methode *run* geprüft werden, ob der Benutzer die Tasten <s> oder <q> gedrückt hat.

Wurde die Taste <s> gedrückt, so soll das Attribut *gameOver* auf FALSE gesetzt werden (das Spiel kann beginnen).

Wurde die Taste <q> gedrückt, so soll das Attribut *gameOver* auf TRUE gesetzt (das Spiel beenden) und die Methode *sendExitSignal* der Klasse *Computer* aufgerufen werden.

Hier geht es zur Lösung auf Seite 40.

Ist das Attribut *gameOver* gleich FALSE, so sollen die Tasten <y>, <x>, <c> und <v> geprüft werden. Wurde eine der Tasten gedrückt, so soll das entsprechende Reagenzglas geleert werden (<y> ≙ Reagenzglas 0 , <x> ≙ Reagenzglas 1 , <c> ≙ Reagenzglas 2 , <v> ≙ Reagenzglas 3).

Hier geht es zur Lösung auf Seite 41.

In die gleiche if-Anweisung (*gameOver* ist gleich FALSE) soll eine for-Schleife eingefügt werden, deren Zähler bei 0 startet und so lange ausgeführt wird, wie der Zähler kleiner als der Wert im Attribut *l* ist. Bei jedem Schleifendurchlauf wird der Zähler um 1 erhöht.

Hier geht es zur Lösung auf Seite 43.

In der for-Schleife soll eine Zufallszahl im Bereich von 0 bis 5 erzeugt werden. Ist die Zufallszahl gleich 0, so steigt der Pegel (*fuellen*) im Reagenzglas, dessen Nummer gleich dem Schleifenzähler ist.

Hier geht es zur Lösung auf Seite 45.

Abschließend muss nach dem der Pegel im Reagenzglas gestiegen ist noch geprüft werden, ob die Ente heraus gefallen ist. Wenn dies der Fall ist, dann ist dass Attribut *gameOver* auf TRUE zu setzen und das Attribut *lbGameOver* soll sichtbar werden.

Hier geht es zur Lösung auf Seite 47.

5.4 Die Klasse Program

Die Klasse *Program* dient nur dazu ein Objekt der Klasse *Game* ins Leben zu rufen und die Anleitung auf der Konsole auszugeben.

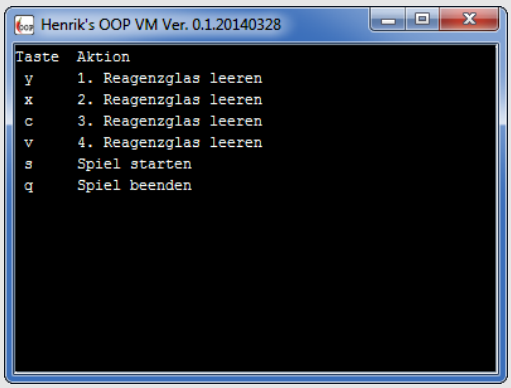
Aufgabe 10:

Erzeugen Sie in der *main* Methode der Klasse *Program* ein Objekt vom Typ *Game* und rufen Sie die Methode *startSingle(200)* auf.

Anschließend soll das Programm auf ein Exitsignal von *Game* warten.

Hier geht es zur Lösung auf Seite 49.

Geben Sie auf der Konsole eine kurze Bedienungsanleitung aus:



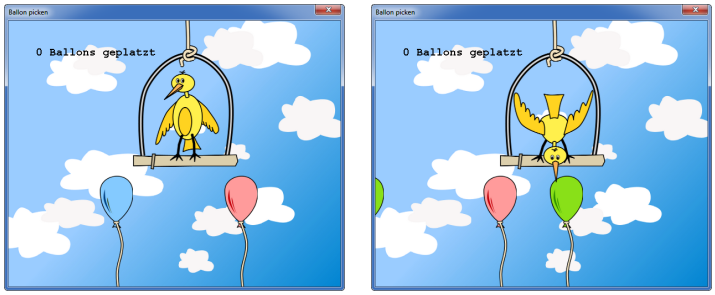
Der Code dazu soll in der Methode *main* der Klasse *Program* eingefügt werden.

Hier geht es zur Lösung auf Seite 50.

6 Übungen

6.1 Luftballons picken

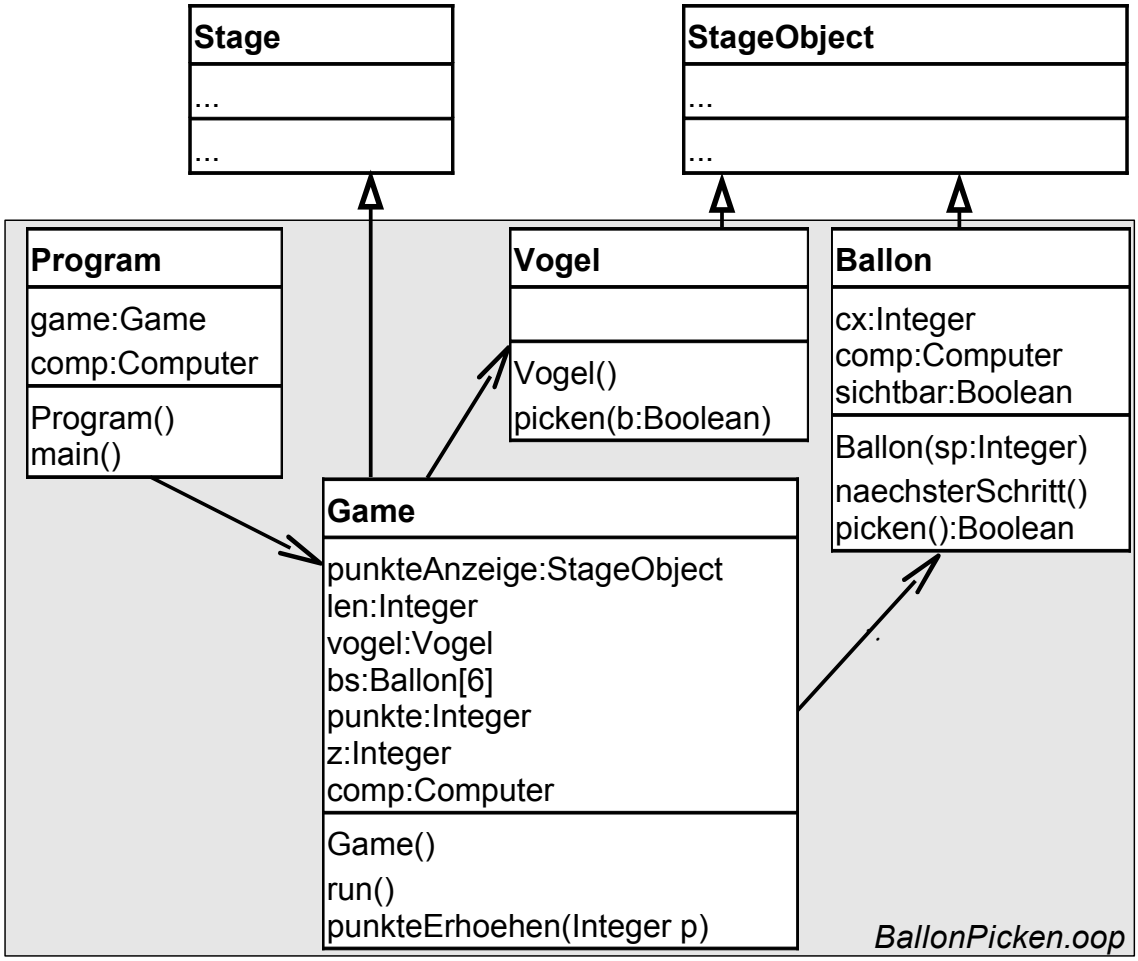
In diesem Spiel fliegen der Reihe nach Ballons an einem Vogel vorbei. Der Vogel versucht möglichst viele Ballons mit seinem Schnabel zum Platzen zu bringen. Jeder geplatzte Ballon wird gezählt.



Die Datei *BallonPicken.demo.oop* enthält das Programm zum Testen.

Entwickeln Sie ein Programm, dass das oben beschriebenes Spiel realisiert. Berücksichtigen Sie dabei folgendes UML-Diagramm und die Klassenbeschreibungen.

UML-Diagramm



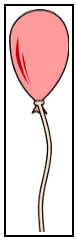
Beschreibung der Klasse Ballon

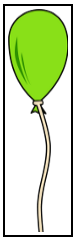
Ballon beschreibt Luftballons die von links nach rechts schweben.

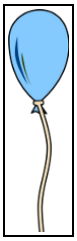
Attribute	
cx	Aktuelle horizontale Position des Ballons.
comp	Wird für das Generieren von Zufallszahlen benötigt.
sichtbar	Gibt an, ob der Ballon sichtbar ist.

Ballon(sp:Integer)	
sp	Horizontale Startposition des Ballons.

Zunächst wird das *StageObject* an der Position (sp,380) auf der Bühne platziert.
Alle Grafiken werden auf 100 % skaliert.
Es werden die folgende Grafiken in der angegebenen Reihenfolge als Sprite dem *StageObject* hinzugefügt:


Ballon.rot.png


Ballon.gruen.png


Ballon.blau.png

Das *StageObject* wird unsichtbar gemacht und das Attribut *sichtbar* auf den entsprechenden Wert (FALSE) gesetzt.

Das Attribut *cx* bekommt den Wert von *sp*.
Als letztes wird für das Attribut *comp* ein Objekt erzeugt.

naechsterSchritt()

Die aktuelle Postion in *cx* wird um 15 erhöht.
Ist der Wert in *cx* größer als 600, dann

- erhält das Attribut *cx* den Wert -30.
- Es wird eine Zufallszahl im Bereich von 0 bis 4 erzeugt.
- Ist die Zufallszahl größer als 2, dann
  - erhält das Attribut *sichtbar* den Wert FALSE
  - und wird der Ballon unsichtbar.
- Sonst
  - erhält das Attribut *sichtbar* den Wert TRUE
  - und es wird der Ballon mit der Farbe, die durch die Zufallszahl bestimmt ist angezeigt (0 ≙ rot , 1 ≙ grün , 2 ≙ blau)

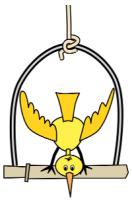
Der Ballon wird an die Position (*cx*,380) verschoben.

© 2014 Henrik Horstmann

picken():Boolean
Wenn der Ballon sichtbar ist und seine horizontale Position im Bereich von 290 bis 350 ist (Ballon vom Schnabel getroffen), dann • wird der Ballon unsichtbar • und die Methode liefert TRUE zurück. Sonst • liefert die Methode den Wert FALSE zurück.

Beschreibung der Klasse Vogel

Vogel beschreibt den Vogel, der nach den Ballons pickt.

Vogel()
Zunächst wird das *StageObject* an der Position (320,150) auf der Bühne platziert. Alle Grafiken werden auf 100 % skaliert. Es werden die folgende Grafiken in der angegebenen Reihenfolge als Sprite dem *StageObject* hinzugefügt: Vogel.wartend.png  Vogel.pickend.png

picken(b:Boolean)	
b	gibt an, ob der Vogel in Warteposition gehen oder sich nach unten bücken soll.
Wenn *b* TRUE ist, dann • zeige die Grafik *Vogel.pickend.png* an. Sonst • zeige die Grafik *Vogel.wartend.png* an.	

Beschreibung der Klasse Game

Attribute	
punkteAnzeige	*StageObject*, das den Punktestand auf der Bühne anzeigt.
bs [6]	Feld mit den Ballons.
len	Länge des Feldes *bs*.
comp	Wird für das Generieren von Zufallszahlen und zur Steuerung des Programms benötigt.
vogel	Objekt, dass den Vogel repräsentiert.
punkte	Anzahl der vom Vogel zum platzen gebrachten Ballons.
z	Zähler, der darüber wacht, dass der Vogel sich nicht zu lange bückt.

Game()
Das Spiel soll "Ballon picken" heißen. Das Spielfenster hat eine Größe von 600×480 Pixel. Das Hintergrundbild befindet sich in der Datei *Himmel01.png*. Im Attribut *bs* soll ein Feld für 6 Einträge angelegt werden. Im Attribut *len* soll die Länge des Feldes *bs* für eine weitere Verwendung gespeichert werden. Verwenden Sie für die im folgenden Absatz beschriebenen Anweisungen eine for-Schleife. Für jeden Eintrag im Feld *bs* ist ein Objekt vom Type *Ballon* zu erzeugen. Der erst Ballon soll an der horizontalen Position -30 platziert werden. Alle anderen Ballons sollen im Abstand von jeweils 120 Pixeln positioniert werden. Alle Objekte im Feld *bs* werden auf der *Stage* platziert. Das Attribut *vogel* erhält ein neues Objekt, das ebenfalls auf der *Stage* platziert wird. Letztlich erhält das Attribut *z* den Wert 0.

punkteErhoehen(p:Integer)	
p	Punkte, um die der aktuelle Punktestand erhöht werden soll.
Erhöhe den Punktezähler *punkte* um *p*. Gebe die Meldung *<?> Ballons geplatzt* (<?> ist der Wert aus *punkte*) in der Schriftgröße 20 im *StageObject punkteAnzeige* aus.	

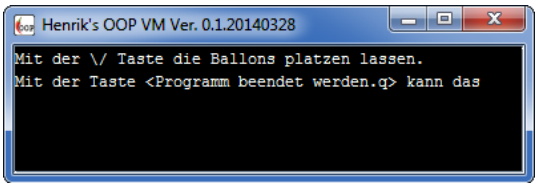
© 2014 Henrik Horstmann

run()
Zunächst einmal sollen alle Ballons einen Schritt weiter nach rechts rücken (verwenden Sie hierzu eine for-Schleife). Ist die Taste <↓> gedrückt und hat der Zaähler *z* den Wert 0, dann • soll der Vogel picken • und prüfe anschließend mit einer for-Schleife für alle Ballons, ob einer von ihnen mit dem Schnabel getroffen wurde. Ist ein Ballon getroffen, dann   • erhöhe den Punktezähler und aktuallisiere den Punktestand auf der Bühne. • Setze den Zähler *z* auf 5. Sonst prüfe, ob der Zähler *z* größer 0 ist. Wenn dem so ist, dann • nimmt der Vogel wieder seine Warteposition ein • und der Zähler *z* wird um 1 erniedrigt. Letztlich prüfe, ob die Taste <q> gedrückt wurde. In diesem Fall • wird ein Exitsignal gesendet.

Beschreibung der Klasse Program

Attribute	
game	Repräsentiert das Spiel.
comp	Wird für die Textausgabe und zur Steuerung des Programms benötigt.

Program()
-

main()
Dem Attribut *game* wird ein neues Objekt vom Type *Game* zugewiesen. Das Spiel im Attribut *game* wird mit einem Aufruf der Methode *startSingle(100)* gestartet. Auf der Konsole wird folgende Bedienungsanleitung ausgegeben:    Abschließend erfolgt ein Aufruf der Methode *waitForExitSignal* der Klasse *Computer*.

ENDE

© 2014 Henrik Horstmann

7 Lösungen zu den Aufgaben

7.1 Lösung zu Aufgabe 1

Das erste Mauseloch befindet sich an der x-Position 100.

Die Abstände der Mauselöcher sind alle gleich. Der Abstand beträgt 118 Pixel (218-100=118; 336-218=118; 454-336=118; 572-454=118).

```
Integer i;
Integer xp;
...
i = new Integer();
xp = new Integer(100);
mls[i] = new Mauseloch(xp,380);
super.addStageObject(mls[i]);
i = i.add(1);
xp = xp.add(118);
mls[i] = new Mauseloch(xp,380);
super.addStageObject(mls[i]);
i = i.add(1);
xp = xp.add(118);
mls[i] = new Mauseloch(xp,380);
super.addStageObject(mls[i]);
...
```

Zurück zur Aufgabe auf Seite 3.

7.2 Lösung zu Aufgabe 2 (a)

```
public Game ()
{
   // Anfang Variablen
   Integer xp;
   // Ende Variablen

   super("Mäuse fangen",661,472);
   super.loadBackground("Hintergrund_gelb.png");
   punkteAnzeige = new StageObject(330,25,100);
   super.addStageObject(punkteAnzeige);
   mls = new Mauseloch[5];
   xp = new Integer(100);
   for(Integer i = new Integer ); i.lowerThan(5); i = i.add(1);
   {
      mls[i] = new Mauseloch(xp,380);
      super.addStageObject(mls[i]);
      xp = xp.add(118);
   }
   katze = new Katze(330,236);
   super.addStageObject(katze);
   maus = new Integer();
   zeit = new Integer(15);
   comp = new Computer();
   anzahlMaeuse = new Integer();
   anzahlGefangen = new Integer();
}
```

Zurück zur Aufgabe auf Seite 7.

7.3 Lösung zu Aufgabe 2 (b)

```
gName = new String("Katze01.0");
gName.append(i);
gName.append(".png");
```

Zurück zur Aufgabe auf Seite 7.

7.4 Lösung zu Aufgabe 2 (c)

```
public Katze (Integer xp,Integer yp)
{
   // Anfang Variablen
   String gName;
   String nr;
   // Ende Variablen

   super(xp,yp,100);
   for(Integer i = new Integer ); i.lowerThan(6); i = i.add(1);
   {
      gName = new String("Katze01.0");
      gName.append(i);
      gName.append(".png");
      nr = new String();
      nr.append(i);
      super.addSprite(nr,gName);
   }
}
```

Zurück zur Aufgabe auf Seite 7.

7.5 Lösung zu Aufgabe 3 (a)

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   // Ende Attribute


   public Reagenzglas ()
   {
      // Anfang Variablen
      // Ende Variablen

      super(NULL,NULL,NULL);
   }
}
```

Zurück zur Aufgabe auf Seite 9.

7.6 Lösung zu Aufgabe 3 (b)

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute


   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      // Ende Variablen

      super(NULL,NULL,NULL);
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      // Ende Variablen

      return NULL;
   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 9.

7.7 Lösung zu Aufgabe 4 (a)

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute

   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      // Ende Variablen

      super(xp,240,100);
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      // Ende Variablen

      return NULL;
   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 9.

7.8 Lösung zu Aufgabe 4 (b)

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute

   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      String gName;
      String nr;
      // Ende Variablen

      super(xp,240,100);
      for(Integer i=new Integer ); i.lowerThanOrEqual(5); i=i.add(1);
      {
         gName = new String("Reagenzglas01.0");
         gName.append(i);
         gName.append(".png");
         nr = new String();
         nr.append(i);
         super.addSprite(nr,gName);
      }
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      // Ende Variablen

      return NULL;
   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 9.

7.9 Lösung zu Aufgabe 4 (c)

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute

   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      String gName;
      String nr;
      // Ende Variablen

      super(xp,240,100);
      for(Integer i=new Integer ); i.lowerThanOrEqual(5); i=i.add(1);
      {
         gName = new String("Reagenzglas01.0");
         gName.append(i);
         gName.append(".png");
         nr = new String();
         nr.append(i);
         super.addSprite(nr,gName);
      }
      state = new Integer();
      super.setSprite(state);
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      // Ende Variablen

      return NULL;
   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 9.

7.10 Lösung zu Aufgabe 5

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute

   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      String gName;
      String nr;
      // Ende Variablen

      super(xp,240,100);
      for(Integer i=new Integer ); i.lowerThanOrEqual(5); i=i.add(1);
      {
         gName = new String("Reagenzglas01.0");
         gName.append(i);
         gName.append(".png");
         nr = new String();
         nr.append(i);
         super.addSprite(nr,gName);
      }
      state = new Integer();
      super.setSprite(state);
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      // Ende Variablen

      return NULL;
   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen

      state = state.add(1);
      super.setSprite(state);
   }
}
```

Zurück zur Aufgabe auf Seite 9.

7.11 Lösung zu Aufgabe 6

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute

   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      String gName;
      String nr;
      // Ende Variablen

      super(xp,240,100);
      for(Integer i=new Integer ); i.lowerThanOrEqual(5); i=i.add(1);
      {
         gName = new String("Reagenzglas01.0");
         gName.append(i);
         gName.append(".png");
         nr = new String();
         nr.append(i);
         super.addSprite(nr,gName);
      }
      state = new Integer();
      super.setSprite(state);
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      // Ende Variablen

      return NULL;
   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen

      state = new Integer();
      super.setSprite(state);
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen

      state = state.add(1);
      super.setSprite(state);
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.12 Lösung zu Aufgabe 7 (a)

Die Ente ist bereits herausgefallen, wenn *state*>4 ist.

Zurück zur Aufgabe auf Seite 10.

© 2014 Henrik Horstmann

7.13 Lösung zu Aufgabe 7 (b)

```
public class Reagenzglas extends StageObject
{
   // Anfang Attribute
   private Integer state;
   // Ende Attribute

   public Reagenzglas (Integer xp)
   {
      // Anfang Variablen
      String gName;
      String nr;
      // Ende Variablen

      super(xp,240,100);
      for(Integer i=new Integer ); i.lowerThanOrEqual(5); i=i.add(1);
      {
         gName = new String("Reagenzglas01.0");
         gName.append(i);
         gName.append(".png");
         nr = new String();
         nr.append(i);
         super.addSprite(nr,gName);
      }
      state = new Integer();
      super.setSprite(state);
   }

   public Boolean isGameOver ()
   {
      // Anfang Variablen
      Boolean b;
      // Ende Variablen

      b = state.greaterThan(4);
      return b;   }

   public void leeren ()
   {
      // Anfang Variablen
      // Ende Variablen
      state = new Integer();
      super.setSprite(state);
   }

   public void fuellen ()
   {
      // Anfang Variablen
      // Ende Variablen

      state = state.add(1);
      super.setSprite(state);
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.14 Lösung zu Aufgabe 8 (a)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super(NULL,NULL,NULL);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.15 Lösung zu Aufgabe 8 (b)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Entenspiel",760,480);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.16 Lösung zu Aufgabe 8 (c)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.17 Lösung zu Aufgabe 8 (d)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.18 Lösung zu Aufgabe 8 (e)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.19 Lösung zu Aufgabe 8 (f)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.20 Lösung zu Aufgabe 8 (g)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
      gameOver = new Boolean(TRUE);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen
   }
}
```

Zurück zur Aufgabe auf Seite 10.

7.21 Lösung zu Aufgabe 9 (a)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
      gameOver = new Boolean(TRUE);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (super.isKeyPressed("s") && gameOver)
      {
         gameOver = new Boolean(FALSE);
      }
      else if (super.isKeyPressed("q"))
      {
         gameOver = new Boolean(TRUE);
         comp.sendExitSignal();
      }
   }
}
```

Zurück zur Aufgabe auf Seite 12.

7.22 Lösung zu Aufgabe 9 (b)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
      gameOver = new Boolean(TRUE);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (super.isKeyPressed("s") && gameOver)
      {
         gameOver = new Boolean(FALSE);
      }
      else if (super.isKeyPressed("q"))
      {
         gameOver = new Boolean(TRUE);
         comp.sendExitSignal();
      }
```

```
        if (gameOver.not())
        {
           if (super.isKeyPressed("y"))
           {
              rgs[0].leeren();
           }
           if (super.isKeyPressed("x"))
           {
              rgs[1].leeren();
           }
           if (super.isKeyPressed("c"))
           {
              rgs[2].leeren();
           }
           if (super.isKeyPressed("v"))
           {
              rgs[3].leeren();
           }
        }
    }
}
```

Zurück zur Aufgabe auf Seite 12.

7.23 Lösung zu Aufgabe 9 (c)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
      gameOver = new Boolean(TRUE);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (super.isKeyPressed("s") && gameOver)
      {
         gameOver = new Boolean(FALSE);
      }
      else if (super.isKeyPressed("q"))
      {
         gameOver = new Boolean(TRUE);
         comp.sendExitSignal();
      }
```

```
      if (gameOver.not())
      {
         if (super.isKeyPressed("y"))
         {
            rgs[0].leeren();
         }
         if (super.isKeyPressed("x"))
         {
            rgs[1].leeren();
         }
         if (super.isKeyPressed("c"))
         {
            rgs[2].leeren();
         }
         if (super.isKeyPressed("v"))
         {
            rgs[3].leeren();
         }
         for(Integer i=new Integer(); i.lowerThan(l); i=i.add(1))
         {
         }
      }
   }
}
```

Zurück zur Aufgabe auf Seite 12.

7.24 Lösung zu Aufgabe 9 (d)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
      gameOver = new Boolean(TRUE);
   }

   public void run ()
   {
      // Anfang Variablen
      Integer r;
      // Ende Variablen

      if (super.isKeyPressed("s") && gameOver)
      {
         gameOver = new Boolean(FALSE);
      }
      else if (super.isKeyPressed("q"))
      {
         gameOver = new Boolean(TRUE);
         comp.sendExitSignal();
      }
```

```
      if (gameOver.not())
      {
         if (super.isKeyPressed("y"))
         {
            rgs[0].leeren();
         }
         if (super.isKeyPressed("x"))
         {
            rgs[1].leeren();
         }
         if (super.isKeyPressed("c"))
         {
            rgs[2].leeren();
         }
         if (super.isKeyPressed("v"))
         {
            rgs[3].leeren();
         }
         for(Integer i=new Integer(); i.lowerThan(l); i=i.add(1))
         {
            r = comp.rand(0,5);
            if (r.lowerThan(1))
            {
               rgs[i].fuellen();
            }
         }
      }
   }
}
```

Zurück zur Aufgabe auf Seite 12.

7.25 Lösung zu Aufgabe 9 (e)

```
public class Game extends Stage
{
   // Anfang Attribute
   private Integer l;
   private StageObject lbGameOver;
   private Computer comp;
   private Boolean gameOver;
   private Reagenzglas[] rgs;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      Integer xp;
      // Ende Variablen

      super("Entenspiel",760,480);
      rgs = new Reagenzglas[4];
      comp = new Computer();
      l = comp.getArrayLength(rgs);
      xp = new Integer(95);
      for(Integer i=new Integer ); i.lowerThan(l); i=i.add(1);
      {
         rgs[i] = new Reagenzglas(xp);
         super.addStageObject(rgs[i]);
         xp = xp.add(190);
      }
      lbGameOver = new StageObject(380,240,0);
      lbGameOver.setTextAlignCenter();
      lbGameOver.setText("game over",80);
      lbGameOver.hide();
      super.addStageObject(lbGameOver);
      gameOver = new Boolean(TRUE);
   }

   public void run ()
   {
      // Anfang Variablen
      Boolean b;
      Integer r;
      // Ende Variablen

      if (super.isKeyPressed("s") && gameOver)
      {
         gameOver = new Boolean(FALSE);
      }
      else if (super.isKeyPressed("q"))
      {
         gameOver = new Boolean(TRUE);
         comp.sendExitSignal();
      }
```

```
      if (gameOver.not())
      {
         if (super.isKeyPressed("y"))
         {
            rgs[0].leeren();
         }
         if (super.isKeyPressed("x"))
         {
            rgs[1].leeren();
         }
         if (super.isKeyPressed("c"))
         {
            rgs[2].leeren();
         }
         if (super.isKeyPressed("v"))
         {
            rgs[3].leeren();
         }
         for(Integer i=new Integer(); i.lowerThan(l); i=i.add(1))
         {
            r = comp.rand(0,5);
            if (r.lowerThan(1))
            {
               rgs[i].fuellen();
               b = new Boolean(FALSE);
               b = rgs[i].isGameOver();
               gameOver = gameOver.or(b);
               if (gameOver)
               {
                  lbGameOver.show();
               }
            }
         }
      }
   }
}
```

Zurück zur Aufgabe auf Seite 12.

7.26 Lösung zu Aufgabe 10 (a)

```
public class Program
{
   // Anfang Attribute
   // Ende Attribute


   public Program ()
   {
      // Anfang Variablen
      // Ende Variablen

   }

   public void main ()
   {
      // Anfang Variablen
      Game game;
      Computer comp;
      // Ende Variablen

      game = new Game();
      game.startSingle(200);
      comp = new Computer();
      comp.waitForExitSignal();
   }
}
```

Zurück zur Aufgabe auf Seite 13.

7.27 Lösung zu Aufgabe 10 (b)

```
public class Program
{
   // Anfang Attribute
   // Ende Attribute


   public Program ()
   {
      // Anfang Variablen
      // Ende Variablen

   }

   public void main ()
   {
      // Anfang Variablen
      Game game;
      Computer comp;
      // Ende Variablen

      game = new Game();
      game.startSingle(200);
      comp = new Computer();
      comp.print("Taste  Aktion");
      comp.println();
      comp.print(" y     1. Reagenzglas leeren");
      comp.println();
      comp.print(" x     2. Reagenzglas leeren");
      comp.println();
      comp.print(" c     3. Reagenzglas leeren");
      comp.println();
      comp.print(" v     4. Reagenzglas leeren");
      comp.println();
      comp.print(" s     Spiel starten");
      comp.println();
      comp.print(" q     Spiel beenden");
      comp.waitForExitSignal();
   }
}
```

Zurück zur Aufgabe auf Seite 13.

7.28 Lösung zu Aufgabe 11 (a)

```
public class Game extends Stage
{
   // Anfang Attribute
   private StageObject ball;
   private Integer cy;
   private Integer cx;
   private Computer comp;
   private Integer sy;
   private StageObject spieler;
   private Integer dy;
   private Integer punkte;
   private Integer dx;
   private StageObject punkteAnzeige;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Fußball",640,480);
      this.loadBackground("Fussballfeld.png");
      this.resetBall();
      ball = new StageObject("Ball",cx,cy,"Fussball.png",100);
      super.addStageObject(ball);
      comp = new Computer();
      punkteAnzeige = new StageObject(320,100,100);
      punkteAnzeige.setTextAlignCenter();
      super.addStageObject(punkteAnzeige);
      punkte = new Integer();
      this.setzePunkte(0);
      sy = new Integer(240);
      spieler = new StageObject("Spieler",150,sy,
                                "Fussballer.png",100);
      super.addStageObject(spieler);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (cx.lowerThanOrEqual(45))
      {
         dx = dx.mul(-1);
      }
      else if (cx.greaterThanOrEqual(565))
      {
         if (cy.greaterThanOrEqual(160) && cy.lowerThanOrEqual(320))
         {
            this.resetBall();
            this.setzePunkte(1);
         }
         else
```

```
            {
               dx = dx.mul(-1);
            }
         }
         else if (ball.checkCollision(NULL))
         {
         }

         if (cy.lowerThanOrEqual(45) || cy.greaterThanOrEqual(435))
         {
            dy = dy.mul(-1);
         }
         if (super.isKeyPressed("s"))
         {
            dx = comp.rand(-10,-5);
            dy = comp.rand(5,10);
         }
         else if (super.isKeyPressed("UP"))
         {
            sy = sy.sub(10);
            spieler.moveToPosition(150,sy);
         }
         else if (super.isKeyPressed("DOWN"))
         {
            sy = sy.add(10);
            spieler.moveToPosition(150,sy);
         }
         cx = cx.add(dx);
         cy = cy.add(dy);
         ball.moveToPosition(cx,cy);
      }

      public void setzePunkte (Integer aenderung)
      {
         // Anfang Variablen
         String msg;
         // Ende Variablen

         punkte = punkte.add(aenderung);
         msg = new String("Punkte: ");
         msg.append(punkte);
         punkteAnzeige.setText(msg,24);
      }

      public void resetBall ()
      {
         // Anfang Variablen
         // Ende Variablen

         cx = new Integer(320);
         cy = new Integer(240);
         dx = new Integer();
         dy = new Integer();
      }
   }
```

Zurück zur Aufgabe auf Seite Fehler: Referenz nicht gefunden.

7.29 Lösung zu Aufgabe 11 (b)

```
public class Game extends Stage
{
   // Anfang Attribute
   private StageObject ball;
   private Integer cy;
   private Integer cx;
   private Computer comp;
   private Integer sy;
   private StageObject spieler;
   private Integer dy;
   private Integer punkte;
   private Integer dx;
   private StageObject punkteAnzeige;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Fußball",640,480);
      this.loadBackground("Fussballfeld.png");
      this.resetBall();
      ball = new StageObject("Ball",cx,cy,"Fussball.png",100);
      super.addStageObject(ball);
      comp = new Computer();
      punkteAnzeige = new StageObject(320,100,100);
      punkteAnzeige.setTextAlignCenter();
      super.addStageObject(punkteAnzeige);
      punkte = new Integer();
      this.setzePunkte(0);
      sy = new Integer(240);
      spieler = new StageObject("Spieler",150,sy,
                                "Fussballer.png",100);
      super.addStageObject(spieler);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (cx.lowerThanOrEqual(45))
      {
         dx = dx.mul(-1);
      }
      else if (cx.greaterThanOrEqual(565))
      {
         if (cy.greaterThanOrEqual(160) && cy.lowerThanOrEqual(320))
         {
            this.resetBall();
            this.setzePunkte(1);
         }
         else
```

```
            {
               dx = dx.mul(-1);
            }
         }
         else if (spieler.checkCollision(ball))
         {
         }

         if (cy.lowerThanOrEqual(45) || cy.greaterThanOrEqual(435))
         {
            dy = dy.mul(-1);
         }
         if (super.isKeyPressed("s"))
         {
            dx = comp.rand(-10,-5);
            dy = comp.rand(5,10);
         }
         else if (super.isKeyPressed("UP"))
         {
            sy = sy.sub(10);
            spieler.moveToPosition(150,sy);
         }
         else if (super.isKeyPressed("DOWN"))
         {
            sy = sy.add(10);
            spieler.moveToPosition(150,sy);
         }
         cx = cx.add(dx);
         cy = cy.add(dy);
         ball.moveToPosition(cx,cy);
      }

      public void setzePunkte (Integer aenderung)
      {
         // Anfang Variablen
         String msg;
         // Ende Variablen

         punkte = punkte.add(aenderung);
         msg = new String("Punkte: ");
         msg.append(punkte);
         punkteAnzeige.setText(msg,24);
      }

      public void resetBall ()
      {
         // Anfang Variablen
         // Ende Variablen

         cx = new Integer(320);
         cy = new Integer(240);
         dx = new Integer();
         dy = new Integer();
      }
   }
```

Zurück zur Aufgabe auf Seite Fehler: Referenz nicht gefunden.

7.30 Lösung zu Aufgabe 11 (c)

```
public class Game extends Stage
{
   // Anfang Attribute
   private StageObject ball;
   private Integer cy;
   private Integer cx;
   private Computer comp;
   private Integer sy;
   private StageObject spieler;
   private Integer dy;
   private Integer punkte;
   private Integer dx;
   private StageObject punkteAnzeige;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Fußball",640,480);
      this.loadBackground("Fussballfeld.png");
      this.resetBall();
      ball = new StageObject("Ball",cx,cy,"Fussball.png",100);
      super.addStageObject(ball);
      comp = new Computer();
      punkteAnzeige = new StageObject(320,100,100);
      punkteAnzeige.setTextAlignCenter();
      super.addStageObject(punkteAnzeige);
      punkte = new Integer();
      this.setzePunkte(0);
      sy = new Integer(240);
      spieler = new StageObject("Spieler",150,sy,
                                "Fussballer.png",100);
      super.addStageObject(spieler);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (cx.lowerThanOrEqual(45))
      {
         dx = dx.mul(-1);
      }
      else if (cx.greaterThanOrEqual(565))
      {
         if (cy.greaterThanOrEqual(160) && cy.lowerThanOrEqual(320))
         {
            this.resetBall();
            this.setzePunkte(1);
         }
         else
```

```
            {
               dx = dx.mul(-1);
            }
         }
         else if (spieler.checkCollision(ball))
         {
            dx = dx.mul(-1);
         }

         if (cy.lowerThanOrEqual(45) || cy.greaterThanOrEqual(435))
         {
            dy = dy.mul(-1);
         }
         if (super.isKeyPressed("s"))
         {
            dx = comp.rand(-10,-5);
            dy = comp.rand(5,10);
         }
         else if (super.isKeyPressed("UP"))
         {
            sy = sy.sub(10);
            spieler.moveToPosition(150,sy);
         }
         else if (super.isKeyPressed("DOWN"))
         {
            sy = sy.add(10);
            spieler.moveToPosition(150,sy);
         }
         cx = cx.add(dx);
         cy = cy.add(dy);
         ball.moveToPosition(cx,cy);
      }

      public void setzePunkte (Integer aenderung)
      {
         // Anfang Variablen
         String msg;
         // Ende Variablen

         punkte = punkte.add(aenderung);
         msg = new String("Punkte: ");
         msg.append(punkte);
         punkteAnzeige.setText(msg,24);
      }

      public void resetBall ()
      {
         // Anfang Variablen
         // Ende Variablen

         cx = new Integer(320);
         cy = new Integer(240);
         dx = new Integer();
         dy = new Integer();
      }
   }
```

Zurück zur Aufgabe auf Seite Fehler: Referenz nicht gefunden.

7.31 Lösung zu Aufgabe 12

```
public class Game extends Stage
{
   // Anfang Attribute
   private StageObject ball;
   private Integer cy;
   private Integer cx;
   private Computer comp;
   private Integer sy;
   private StageObject spieler;
   private Integer dy;
   private Integer punkte;
   private Integer dx;
   private StageObject punkteAnzeige;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Fußball",640,480);
      this.loadBackground("Fussballfeld.png");
      this.resetBall();
      ball = new StageObject("Ball",cx,cy,"Fussball.png",100);
      super.addStageObject(ball);
      comp = new Computer();
      punkteAnzeige = new StageObject(320,100,100);
      punkteAnzeige.setTextAlignCenter();
      super.addStageObject(punkteAnzeige);
      punkte = new Integer();
      this.setzePunkte(0);
      sy = new Integer(240);
      spieler = new StageObject("Spieler",150,sy,
                                "Fussballer.png",100);
      super.addStageObject(spieler);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (cx.lowerThanOrEqual(45))
      {
         this.setzePunkte(-1);
         this.resetBall();
      }
      else if (cx.greaterThanOrEqual(565))
      {
         if (cy.greaterThanOrEqual(160) && cy.lowerThanOrEqual(320))
         {
            this.resetBall();
            this.setzePunkte(1);
         }
```

```
         else
         {
            dx = dx.mul(-1);
         }
      }
      else if (spieler.checkCollision(ball))
      {
         dx = dx.mul(-1);
      }
      if (cy.lowerThanOrEqual(45) || cy.greaterThanOrEqual(435))
      {
         dy = dy.mul(-1);
      }
      if (super.isKeyPressed("s"))
      {
         dx = comp.rand(-10,-5);
         dy = comp.rand(5,10);
      }
      else if (super.isKeyPressed("UP"))
      {
         sy = sy.sub(10);
         spieler.moveToPosition(150,sy);
      }
      else if (super.isKeyPressed("DOWN"))
      {
         sy = sy.add(10);
         spieler.moveToPosition(150,sy);
      }
      cx = cx.add(dx);
      cy = cy.add(dy);
      ball.moveToPosition(cx,cy);
   }

   public void setzePunkte (Integer aenderung)
   {
      // Anfang Variablen
      String msg;
      // Ende Variablen

      punkte = punkte.add(aenderung);
      msg = new String("Punkte: ");
      msg.append(punkte);
      punkteAnzeige.setText(msg,24);
   }

   public void resetBall ()
   {
      // Anfang Variablen
      // Ende Variablen

      cx = new Integer(320);
      cy = new Integer(240);
      dx = new Integer();
      dy = new Integer();
   }
}
```

Zurück zur Aufgabe auf Seite Fehler: Referenz nicht gefunden.

7.32 Lösung zu Aufgabe 13

```
public class Game extends Stage
{
   // Anfang Attribute
   private StageObject ball;
   private Integer cy;
   private Integer cx;
   private Computer comp;
   private Integer sy;
   private StageObject spieler;
   private Integer dy;
   private Integer punkte;
   private Integer dx;
   private StageObject punkteAnzeige;
   // Ende Attribute


   public Game ()
   {
      // Anfang Variablen
      // Ende Variablen

      super("Fußball",640,480);
      this.loadBackground("Fussballfeld.png");
      this.resetBall();
      ball = new StageObject("Ball",cx,cy,"Fussball.png",100);
      super.addStageObject(ball);
      comp = new Computer();
      punkteAnzeige = new StageObject(320,100,100);
      punkteAnzeige.setTextAlignCenter();
      super.addStageObject(punkteAnzeige);
      punkte = new Integer();
      this.setzePunkte(0);
      sy = new Integer(240);
      spieler = new StageObject("Spieler",150,sy,
                                "Fussballer.png",100);
      super.addStageObject(spieler);
   }

   public void run ()
   {
      // Anfang Variablen
      // Ende Variablen

      if (cx.lowerThanOrEqual(45))
      {
         this.setzePunkte(-1);
         this.resetBall();
      }
      else if (cx.greaterThanOrEqual(565))
      {
         if (cy.greaterThanOrEqual(160) && cy.lowerThanOrEqual(320))
         {
            this.resetBall();
            this.setzePunkte(1);
         }
```

```
         else
         {
            dx = dx.mul(-1);
         }
      }
      else if (spieler.checkCollision(ball))
      {
         dx = dx.mul(-1);
      }
      if (cy.lowerThanOrEqual(45) || cy.greaterThanOrEqual(435))
      {
         dy = dy.mul(-1);
      }

      if (super.isKeyPressed("s"))
      {
         dx = comp.rand(-10,-5);
         dy = comp.rand(5,10);
      }
      else if (super.isKeyPressed("UP"))
      {
         sy = sy.sub(10);
         spieler.moveToPosition(150,sy);
      }
      else if (super.isKeyPressed("DOWN"))
      {
         sy = sy.add(10);
         spieler.moveToPosition(150,sy);
      }
      else if (super.isKeyPressed("q"))
      {
         comp.sendExitSignal();
      }

      cx = cx.add(dx);
      cy = cy.add(dy);
      ball.moveToPosition(cx,cy);
   }

   public void setzePunkte (Integer aenderung)
   {
      // Anfang Variablen
      String msg;
      // Ende Variablen

      punkte = punkte.add(aenderung);
      msg = new String("Punkte: ");
      msg.append(punkte);
      punkteAnzeige.setText(msg,24);
   }
```

```
   public void resetBall ()
   {
      // Anfang Variablen
      // Ende Variablen

      cx = new Integer(320);
      cy = new Integer(240);
      dx = new Integer();
      dy = new Integer();
   }
}
```

Zurück zur Aufgabe auf Seite Fehler: Referenz nicht gefunden.

7.33 Lösung zu Aufgabe 14

```
public class Program
{
   // Anfang Attribute
   // Ende Attribute


   public Program ()
   {
      // Anfang Variablen
      // Ende Variablen

   }

   public void main ()
   {
      // Anfang Variablen
      Game game;
      Computer comp;
      // Ende Variablen

      game = new Game();
      game.startSingle(5);
      comp = new Computer();
      comp.print("Bedienung:");
      comp.println();
      comp.print("/\ - Spieler nach oben");
      comp.println();
      comp.print("\/ - Spieler nach unten");
      comp.println();
      comp.print(" s - Runde beginnen");
      comp.println();
      comp.print(" q - Spiel beenden");
      comp.waitForExitSignal();
   }
}
```

Zurück zur Aufgabe auf Seite Fehler: Referenz nicht gefunden.