OOP Projekt

# Broken Dreams

Henrik Horstmann

27. April 2015

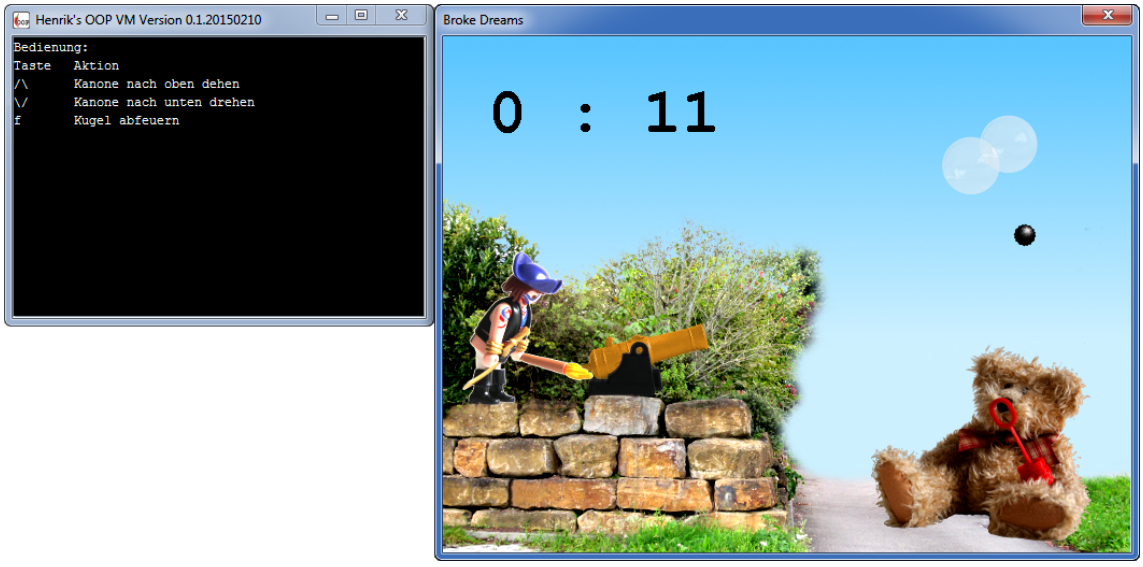
# Inhaltsverzeichnis

# 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. |

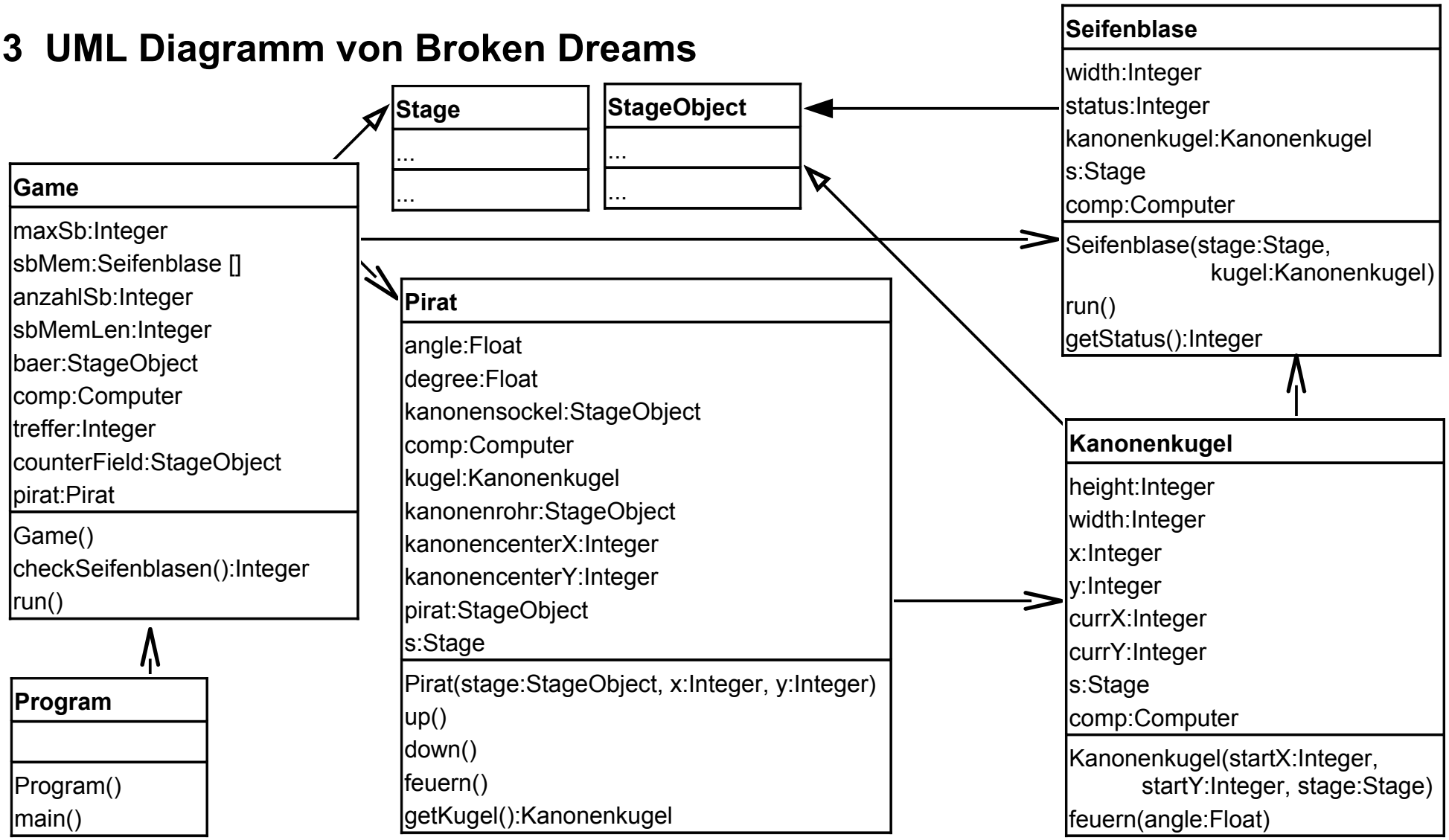
# 2 Das Spiel

Der Bär bläst Seifenblasen in die Luft. Der Kanonier (Spieler) versucht die Seifenblasen abzuschießen.




Das fertige Programm kann als *Broken_Dreams.demo.oop* in die Entwicklungsumgebung geladen und ausprobiert werden.

# 3 UML Diagramm von Broken Dreams



# 4 Klasse Kanonenkugel

*Kanonenkugel* beschreibt die Kanonenkugeln zum abschießen der Seifenblasen.

| *Attribute* | |
|---|---|
| height | Höhe der Stage. |
| width | Breite der Stage |
| x | x-Koordinate des Punktes, von dem die Kanonenkugel abgefeuert wird. |
| y | y-Koordinate des Punktes, von dem die Kanonenkugel abgefeuert wird. |
| currX | x-Koordinate der Postion, an der sich die Kanonenkugel zur Zeit befindet. |
| currY | y-Koordinate der Postion, an der sich die Kanonenkugel zur Zeit befindet. |
| s | Stage, auf der die Kanonenkugel platziert wird. |
| comp | Wird für das Generieren von Zufallszahlen benötigt. |

| Kanonenkugel(startX:Integer, startY:Integer, stage:Stage) | |
|---|---|
| startX | x-Koordinate des Punktes, von dem die Kanonenkugel abgefeuert wird. |
| startY | y-Koordinate des Punktes, von dem die Kanonenkugel abgefeuert wird. |
| stage | Stage, auf der die Kanonenkugel platziert wird. |
| Mit Hilfe des Konstruktors der Superklasse wird die Kanonenkugel an die Position (*startX, startY*) gesetzt. Die Kanonenkugel wird durch das Bild *Kanonenkugel.png* dargestellt. Die Skalierung beträgt 100%.<br>Dem Attribute *comp* wird ein neues Objekt zugewiesen.<br>Die Attribute *x*, *y* und *s* erhalten die Werte der Parameter *startX*, *startY* und *stage*. Anschließend wird die Kanonenkugel der Stage hinmzugefügt.<br>Die Attribute *width* und *height* werden auf die Breite und Höhe der Stage gesetzt. | |

Breite und Höhe dzr Stage liefern die Methoden:

```
Integer Stage.getWidth()
Integer Stage.getHeight()
```

| feuern(angle:Float) | |
|---|---|
| angle | Winkel in Grad in dem die Kugel abgefeuert werden soll. |

Als erstes wird die Kanonenkugel an die Position (*x*, *y*) (siehe Attribute) positioniert. Danach wird sie um den Winkel *angle* nach links gedreht.
Die Attribute *currX* und *currY* erhalten die Koordinaten der aktuelle Position der Kanonenkugel.
Die Kanonenkugel wird sichtbar.
Solange wie sich die Koordinate der Kanonenkugel (*currX*, *currY*) innerhalb der Bühne befindet, führe folgende Anweisungen durch:

- Verschiebe die Position der Kanonenkugel um 10 Einheiten vorwärts.
- Warte 10 ms.
- Die Attribute *currX* und *currY* erhalten die Koordinaten der aktuelle Position der Kanonenkugel.

Ist die Kanonenkugel außerhalb der Bühne, wird sie unsichtbar.
Danach wird sie um den Winkel *angle* nach rechts gedreht.

Schleifen bei denen nicht von vornherein feststeht, wie oft sie durchlaufen werden können mit einer while-Schleife umgesetzt werden:

```
while (Bedingung)
{
      // Schleifenrumpf
}
```

Hat der Schleifenrumpf keinen Einfluss auf die Werte in der Bedingung, so wird sie endlos ausgeführt und das Programm „hängt“. Beispiel:

```
i = new Integer (5)
while (i.lowerThan(10)) {
}
```

In der Schleife wird der Wert von i nicht verändert. Somit wird die Bedingung immer den Wert *TRUE* zurückliefern und die Schleife endlos ausgeführt.

```
i = new Integer (5)
while (i.lowerThan(10)) {
      i = i.add(1);
}
```

Die Schleife wird 5 mal ausgeführt und dann beendet.

# 5 Klasse Seifenblase

*Seifenblase* beschreibt die Seifenblasen, die der Bär in die Luft pustet.

| *Attribute* | |
|---|---|
| width | Breite der Stage |
| status | Zustand der Seifenblase:<br>1: Seifenblase ist schwebt nach oben.<br>0: Seifenblase ist nach oben aus der Stage geschwebt.<br>-1: Seifenblase ist mit der Kanonenkugel kollidiert. |
| kanonenkugel | Die eine Kanonenkugel, die der Kanonier abfeuern kann. |
| s | Stage, auf der die Seifenblase platziert wird. |
| comp | Wird für das Generieren von Zufallszahlen benötigt. |

| Seifenblase(stage:Stage, kugel:Kanonenkugel) | |
|---|---|
| kugel | Die eine Kanonenkugel, die der Kanonier abfeuern kann. |
| Die Seifenblase wird an Position (520,350) mit einer Skalierung von 100% gesetzt.<br>Es werden nacheinander die beiden Bilder *Seifen_Blase.png* und *Platzende_Seifen_Blase.png* geladen.<br>Das Attribut *width* wird auf die Breite der Stage gesetzt.<br>Das Attribute *comp* bekommt ein neues Objekt.<br>Dem Attribut *kanonenkugel* wird der Parameter *kugel* zugewiesen.<br>Das Attribut *status* wird der Wert 1 zugewiesen. | |

| run() |
|---|
| Eine lokale Variable *collision* wird auf *FALSE* gesetzt.<br>Zwei lokale Variablen *x* und *y* erhalten die Koordinaten der aktuellen Position der Seifenblase.<br>Solange *x* im Bereich von 0 bis Breite der Stage und *y* kleiner ist als 0 und collison=FALSE ist führe folgenden Code aus:<br>• Wenn die Seifenblase mit der Kanonenkugel kollidiert, dann<br>  • Zeige die geplatzte Seifenblase an.<br>  • Warte 100ms.<br>  • Mache die Seifenblase unsichtbar.<br>  • Setze die Variable *collision* auf *TRUE*.<br>• sonst<br>  • Subtrahiere von *y* den Wert 5.<br>  • Addiere oder subtrahiere nach dem Zufallsprinzip zu/von *x* den Wert 5.<br>  • Verschiebe die Seifenblase an die neue Position (*x*, *y*).<br>  • Warte 50ms.<br>Mache die Seifenblase unsichtbar und setze den Status *status* auf -1, wenn *collision=TRUE* ist, sonst setze *status* auf 0. |

| getStatus():Integer |
|---|
| Die Methode liefert den Wert des Attributs *status* zurück. |

# 6 Klasse Pirat

*Pirat* beschreibt den Kanonier, der auf die Seifenblasen schießt.

| *Attribute* | |
|---|---|
| angle | Winkel des Kanonenohrs. |
| degree | Grad um den das Kanonenrohr bewegt wird. |
| kanonensockel | Sockel der Kanonen. |
| kanonenrohr | Rohr der Kanone. |
| kanonencenterX | X-Koordoinate der Kanone. |
| kanonencenterY | Y-Koordoinate der Kanone. |
| kugel | Kanonenkugel, die abgefeuert werden kann. |
| pirat | Der Pirat (Kanonier), der eine Kugel abfeuert. |
| s | Stage, auf der der Pirat (Kanonier) platziert wird. |
| comp | Wird für das Generieren von Zufallszahlen benötigt. |

| Pirat(stage:Stage, x:Integer, y:Integer) | |
|---|---|
| stage | Stage, auf der der Pirat (Kanonier) platziert wird. |
| x | X-Koordinate der Position von Pirat mit Kanone. |
| y | Y-Koordinate der Position von Pirat mit Kanone. |
| Das Attribut *s* wird der Parameter *stage* zugewiesen.<br>Die Attribute *kanonencenterX* und *kanonencenterY* werden die Werte der Koordinaten (*x*+103, *y*+25) zugewiesen.<br>Das Attribut *kanonenrohr* erhält ein neues Objekt mit dem Bild *Kanonenrohr.png* (Skalierung 100%) an der Position (*kanonencenterX*, *kanonencenterY*) und wird auf die Stage platziert.<br>Das Attribut *kanonensockel* erhält ebenfalls ein neues Objekt mit dem Bild *Kanonensocel.png* (Skalierung 100%) an der Position (kanonencenterX+90, kanonencenterY+40) und wird auf die Stage platziert.<br>Das Attribut *pirat* erhält ein neues *StageObject* an der Position (*x*, *y*) (Skalierung 100%). Der *pirat* bekommt die Bilder *Pirat01.png* und *Pirat02.png*. Außerdem wird der *pirat* der Stage hinzugefügt.<br>Das Attribut *angle* erhält den Wert 0.0.<br>Das Attribut *kugel* erhält ein neues Objekt mit den Koordinaten (*kanonencenterX*, *kanonencenterY*).<br>Das Attribute *comp* bekommt ein neues Objekt.<br>Das Attribut *degree* wird der Wert 1.0 zugewiesen. | |

| up() |
|---|
| Wenn der Wert in *angle* kleiner als 45 ist, dann<br>• Erhöhe den Wert in *angle* um den Wert in *degree*.<br>• Rotiere das Kanonenrohr um den Wert -*angle* (`StageObject.setRotation(x:Float)`). |

| down() |
|---|
| Wenn der Wert in *angle* größer als -15 ist, dann<br>• Verminder den Wert in *angle* um den Wert in *degree*.<br>• Rotiere das Kanonenrohr um den Wert -*angle* (`StageObject.setRotation(x:Float)`). |

| feuern() |
|---|
| Der *pirat* beugt sich (*Pirat02.png*).<br>Die *kugel* wird im Winkel *angle* abgefeuert.<br>Der *pirat* richtet sich wieder auf (*Pirat01.png*). |

| getKugel():Kanonenkugel |
|---|
| Die Methode liefert die Kanonenkugel *kugel* zurück. |

# 7 Klasse Game

*Game* beschreibt das Spiel.

| *Attribute* | |
|---|---|
| maxSb | Maximale Anzahl von Seifenblasen, die gleichzeitig in der Luft schweben können. |
| sbMem | Feld von Seifenblasen. |
| anzahlSb | Anzahl der vom Bär in die Luft gepusteten Seifenblasen. |
| sbMemLen | Länge das Feldes sbMem |
| baer | Bär, der die Seifenblasen in die Luft pustet. |
| comp | Wird zur Steuerung des Spiels benötigt. |
| treffer | Zählt die getroffenen Seifenblasen. |
| counterField | StageObject zum Anzigen des Punktestands |
| pirat | Pirat (Kanonier) mit Kanone. |

| Game() |
|---|
| Die Stage soll eine Größe von 640×480 haben.<br>Das Hintergrundbild ist *broken_dreams_bg.png.*<br>Das Attribut *comp* erhält ein neues Objekt.<br>Das Attribut *baer* erhält ein neues Objekt. *baer* bekommt das Bild *Baer.png* (Skalierung 100%) und wird an der Position (500, 380) auf die Stage platziert.<br>Der *pirat* erhält ebenfalls ein neues Objekt und wird auf der Stage an der Position (80, 270) platziert.<br>Das Attribut *counterField* bekommt ein neues StageObject zugewiesen und wird auf der Stage an der Position (150, 50) platziert. Das *counterField* soll den Text „0:0“ in der Schriftgröße 60 anzeigen.<br>Das Attribut *sbMem* erhält ein neues Feld der Größe 3.<br>Das Attribut *sbMemLen* wird die Länge des Feldes *sbMem* zugewiesen.<br>Den Attribute *treffer* und *anzahlSb* werden jeweils der Wert 0 zugewiesen.<br>Dem Attribut *maxSb* wird dr Wrt 24 zugewiesen. |

checkSeifenblasen():Integer

Für jedes Element in dem Feld *sbMem* führe folgende Anweisungen aus:

- Wenn das Element ein Objekt enthält, dann
  - Wenn das Element (die Seifenblase) den Status -1 hat (getroffen), dann
    - Erhöhe den Wert in *treffer* um 1.
    - Setze das Element in dem Feld gleich *null*.
  - Wenn das Element (die Seifenblase) den Status 0 hat (aus der Stage geschwebt), dann
    - Setze das Element in dem Feld gleich *null*.

Gebe die Anzahl der Felder in *sbMem* zurück, die kein Objekt (=*null*) enthalten.

Mit der Methode:

```
Boolean Compter.isNull(<Variable>)
```

kann geprüft werden, ob ein Attribut, ein Parameter oder eine Variable ein Objekt (=*TRUE*) oder kein Objekt (*null*) (=*FALSE*) enthält.

run()

Wenn *anzahlSB* kleiner als *maxSB* ist oder das Feld *sbMem* leere (=null) Felder hat (dies soll mit Hilfe der Methode *checkSeifenblasen*() geschehen), dann

- Wenn die Taste <UP> gedrückt wurde, den bewege das Kanonenrohr nach oben.
  - Sonst, wenn die Taste <DOWN> gedrückt wurde, dann bewege das Kanonenrohr nach unten.
  - Sonst, wenn die Taste <f> gedrückt wurde, dann feuer eine Kanonenkugel ab.
- Erzeuge eine Zufallszahl im Bereich von 1 bis 100. Ist die Zufallszahl im Bereich von 1 bis 3 und gibt es leere Felder (=null) im Feld *sbMem* und *anzahlSB* kleiner als *maxSB*, dann
  - Erzeuge eine neu Seifenblase, übergebe ihr die Kanonenkugel des Piraten und platziere sie auf der Stage.
  - Erhöhe den Zähler *anzahlSB* um 1.
  - Suche ein leeres Feld (=*null*) in *sbMem* und weise dem Feld die soeben erzeugte Seifenblase zu.
- Aktualisiere die Punkteanzeige.

Sonst sende ein Exit Signal.

# 8 Klasse Program

*Program* enthält die Methode *main*(), mit der das Programm gestartet wird.

| main() |
| --- |
| Gebe auf der Konsole eine Anleitung für das Spiel aus (siehe Screenshot auf Seite 2).<br>Erzeuge ein Objekt vom Typ *Game* und rufe di Methode *startSingle*(10) auf.<br>Warte auf das Exit Signal. |