

B.I.13

Algorithmen – Unterrichtseinheit

Projektaufgabe Pong: Objektorientierte Programmierung mit *Java*

Ein Beitrag von Johann-Georg Vogelhuber



© mikkelwilliam/E+

Die objektorientierte Programmierung ist zentraler Bestandteil des Informatikunterrichts. Mit dieser Unterrichtseinheit wendet ihre Klasse die Prinzipien moderner Softwareentwicklung anhand einer motivierenden Projektaufgabe in einem realitätsnahen Kontext an und vertieft so ihr Wissen. Sie planen das Videospiel Pong, indem sie die Anforderungen an die zu erstellende Software und die zur Verfügung stehenden Ressourcen analysieren, ein UML-Klassendiagramm erstellen und die notwendigen Implementierungsschritte planen und priorisieren. Anschließend programmieren Sie das Videospiel in Partnerarbeit in *Java*. Für die Durchführung der Implementierung stehen vier Hilfefkarten zur Verfügung, auf denen die Schritte detailliert erläutert werden. Zum Üben der Kommunikations- und Reflexionskompetenz geben Sie sich gegenseitig konstruktives Feedback.

KOMPETENZPROFIL

Klassenstufe: Sek. II

Dauer: 8–10 Unterrichtsstunden

Lernziele: Die Lernenden ... 1. modellieren und implementieren durch Planen und Programmieren in *Java* für das Videospiel Pong, 2. kommunizieren und kooperieren durch das Geben von konstruktivem Feedback zu Projektarbeiten

Kompetenzbereiche: Modellieren und Implementieren, Kommunizieren und Kooperieren

Thematische Bereiche: Algorithmen, objektorientierte Programmierung mit *Java*, Vererbung, Klassen und Objekte, UML, Softwareengineering

Auf einen Blick

Benötigt

- Tablet/Laptop/PC pro Schüler/in oder pro Schülerpaar
- Internetzugang
- BlueJ zur Programmierung, installiert auf den Schülerendgeräten



Einstieg

Thema: Objektorientierte Analyse und Design

M 1 Videospiele Pong – Welche Anforderungen sollen umgesetzt werden?

M 2 Analyse der Basisklassen – Welche Funktionalitäten sind schon vorhanden?



Benötigt: Vorlage.zip

M 3 Erster Softwareentwurf – Welche neuen Klassen müssen erstellt werden?



M 4 Planung – Welche Implementierungsschritte sind notwendig?

Erarbeitung

Thema: Objektorientierte Programmierung

M 5 Hilfematerial: Analyse der Spielelemente Ball und Schläger



M 6 Hilfematerial: Bewegung des Balls implementieren

M 7 Hilfematerial: Tastatureingabe und Schlägerbewegung

M 8 Hilfematerial: Kollisionüberprüfung und Punktezählung

Ergebnisbewertung

M 9 Präsentation und Feedback

M 10 Bewertungsbogen

Benötigte Dateien

- Vorlage.zip (M 2)
- Pong_Schritt1.zip (Zwischenlösungen als Hilfestellung zur Implementierung)
- Beispiellösung.zip



M 1

Videospiel Pong – Welche Anforderungen sollen umgesetzt werden?

Das zu Beginn der 1970er Jahre von dem Unternehmen Atari veröffentlichte Videospiel Pong gilt als Urvater der Videospiele und wurde zunächst auf Geräten in Spielhallen gespielt. Zwar war es nicht das erste Videospiel, dennoch war es das erste, das weltweit erfolgreich wurde. Schau dir das kurze Video zum Ablauf des Videospieles Pong an, um dir einen ersten Eindruck zu verschaffen:

<https://raabe.click/Videospiel-Pong>

Die Spielregeln

Das Spielprinzip von Pong ist sehr einfach gehalten und ähnelt Tischtennis: Ein Ball, dargestellt als Bildpunkt, bewegt sich geradlinig auf dem Bildschirm hin und her. Jeder der zwei Spieler hat einen „Schläger“, den er nach oben oder unten bewegen kann. Den Schläger muss man dabei so bewegen, dass der Ball dort abprallt und wieder zum Gegner zurückgespielt wird. Verpasst man den Ball und lässt ihn am Schläger vorbeigehen, so erhält der Gegner einen Punkt. Trifft der Ball auf den Schläger bzw. auf den oberen oder unteren Bildschirmrand, so prallt er von diesem Hindernis ab. Die Geschwindigkeit wird beibehalten und die Richtung verändert sich so, dass sich der Ball wieder vom Hindernis wegbewegt. Dabei soll der „Auftreffwinkel“ gleich dem „Abprallwinkel“ sein.



Commons/Azerion/CC BY-SA 4.0

Situationsbeschreibung

Das Spiel Pong soll als innovative App neu entwickelt werden. Der erste Schritt besteht aus der Erstellung der grundlegenden App-Funktionalität. Einige der notwendigen Klassen sind bereits implementiert worden. Bei der Umsetzung sollen zuerst folgende Anforderungen umgesetzt werden:

- Zwei Spieler spielen das Spiel gemeinsam an einem PC.
- Das Spiel wird über eine Tastatur gesteuert.
- Wird der Ball nicht zurückgespielt, bekommt der Gegner einen Punkt.
- Beim zehnten Punkt hat der entsprechende Spieler das Spiel gewonnen.
- Der Ball startet in der Mitte des Spielfeldes in eine zufällige Richtung.
- Der jeweilige Punktestand soll angezeigt werden.
- Die Spieldarstellung soll ähnlich zu der obigen Abbildung erfolgen.

Aufgabe 1

Erstelle einen groben Ablaufplan mit den durchzuführenden Arbeitsschritten bis hin zur fertig programmierten App. Beachte dabei, dass für die Umsetzung der App bereits ein Framework mit Basis-Klassen erstellt wurde. Auf Details des Programms sollst du an dieser Stelle noch nicht eingehen.

Aufgabe 2

- Erstelle eine Checkliste für das beschriebene Programm. Diese Checkliste soll die notwendigen Funktionalitäten und Qualitätskriterien für den Quelltext umfassen.

Aufgabe 3

Überlege welche Objekte in diesem Spiel beteiligt sind und erstelle ein entsprechendes Objektdiagramm. Welche Eigenschaften haben diese Objekte? Entwickle das Diagramm so detailliert wie möglich.

M 5

Hilfematerial: Anzeige von Ball und Schläger

Der erste Implementierungsschritt zur Anzeige von Ball und Schläger ist auf den ersten Blick relativ komplex, da dazu mehrere Klassen erstellt und zueinander in Beziehung gesetzt werden müssen. Zwischenlösungen kannst du den folgenden Hinweisen sowie den Dateien *Config.java*, *Pong.java*, *PongPanel.java*, *Ball.java* und *Racket.java* im ZIP-Ordner *Pong_Schritt1.zip* entnehmen.

Schritt 1

Damit Ball und Schläger angezeigt werden können, musst du als Erstes **entsprechende Klassen erstellen**. Die Klasse *Ball* muss dabei von *Circle* und die Klasse *Racket* von *Rectangle* erben. Zum Beispiel (<https://raabe.click/Video-VererbungJava/>):

```
public class Ball extends Circle
{
    // hier kommen die zusätzlichen Attribute hinzu
    public Ball(int startX, int startY, int radius, int color)
    {
        super(startX, startY, radius, color);
        // hier kommt die Logik für den Konstruktor hinzu
    }
    // hier wird später die weitere Logik für den Ball ergänzt
}
```

Schritt 2

Damit das Spiel durchgeführt werden kann, benötigst du ein Spielfeld. Dieses Spielfeld verwaltet Ball, Schläger und alle anderen Spielobjekte. Dazu musst du eine Klasse *PongPanel* erstellen, die von der abstrakten Klasse *GamePanel* abgeleitet ist. D. h. die abstrakten Methoden von *GamePanel* müssen in *PongPanel* implementiert werden. Die Methoden können dabei zunächst leer bleiben (<https://raabe.click/Video-AbstrakteKlassenJava/>):

```
@Override
public void update() { }
```

Schritt 3

Nun muss **der Konstruktor der Klasse *PongPanel* implementieren**. An dieser Stelle müssen Objekte der Klassen *Ball* und *Racket* erstellt und dem Panel hinzugefügt werden. Am besten du erst die Objekte zusätzlich in einem entsprechenden Attribut. Die Einstellungen für Farbe und Größe kannst du in Konstanten der Klasse *Config* anlegen und dann über den Konstruktor an die Klasse *Ball* übergeben. So sind alle Einstellungen an einer Stelle und später leicht anzupassen.

```
private Ball ball;
public PongPanel(Config gameConfig)
{
    super(gameConfig);
    this.ball = new Ball(gameConfig.WIDTH / 2 - gameConfig.BALL_RADIUS,
        gameConfig.BALL_COLOR);
    this.addShape(this.ball);
    // ...
}
```

Erklärvideo:

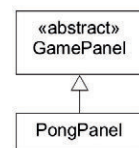


Erklärvideo:



Hinweis:

Für jede Form, die dem Panel hinzugefügt werden soll, muss die Methode `addShape()` aufgerufen werden. Nur wenn dies geschieht, wird die Form auch gezeichnet.



Sie wollen mehr für Ihr Fach?

Bekommen Sie: Ganz einfach zum Download im RAABE Webshop.



✓ **Über 5.000 Unterrichtseinheiten**
sofort zum Download verfügbar

✓ **Webinare und Videos**
für Ihre fachliche und
persönliche Weiterbildung

✓ **Attraktive Vergünstigungen**
für Referendar:innen
mit bis zu 15% Rabatt

✓ **Käuferschutz**
mit Trusted Shops



Jetzt entdecken:
www.raabe.de