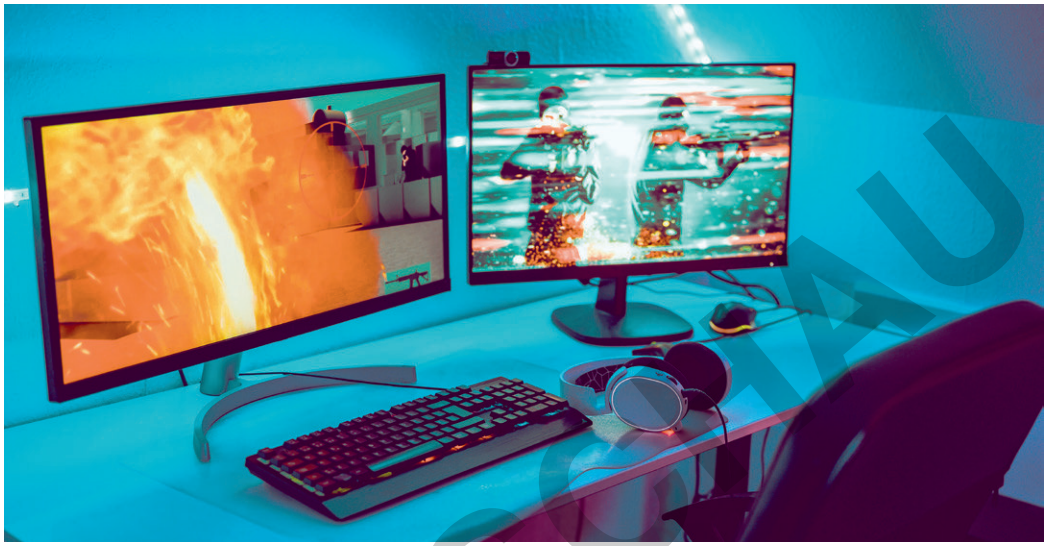
B.I.13

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 Basisklassen 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 Hilfekarten 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	**Videospiel Pong – Welche Anforderungen sollen umgesetzt werden?**
M 2	**Analyse der Basisklassen – Welche Funktionalität ist 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: Anzeige der Spielelemente Ball und Schläger**
M 6	**Hilfematerial: Bewegung des Balls implementieren**
M 7	**Hilfematerial: Tastatureingabe und Schlägerbewegung**
M 8	**Hilfematerial: Kollisionsüberprüfung und Punktezählung**

Ergebnissicherung

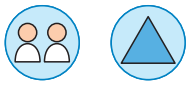
M 9	**Präsentation und Feedback**
M 10	**Bewertungsbogen**

Benötigte Dateien

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

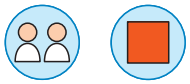
Planung – Welche Implementierungsschritte sind notwendig?

Nach der groben Planung der Programmstruktur ist die Reihenfolge der Implementierungsschritte festzulegen. Diese Schritte werden so geplant, dass nach der Umsetzung eines Schrittes die Software um neue Funktionalität erweitert wurde und auch lauffähig ist. So entsteht nach und nach ein fertiges Produkt, das auch während der Entwicklung immer getestet werden kann.



Aufgabe 1

Auf den folgenden Karten sind die notwendigen Implementierungsschritte aufgelistet. Schätzt für jeden Schritt wie lange ihr vermutlich für die Umsetzung benötigen werdet und notiert die Zeit in Minuten auf der Karte.



Aufgabe 2

Bringt die Schritte anschließend in eine geeignete Reihenfolge, sodass das Programm Schritt für Schritt aufgebaut und um neue Funktionalität erweitert wird. Wenn ihr mit der Planung der Reihenfolge fertig seid, dann nummeriert die Karten entsprechend.

Spiel um Kollisionsabfrage Ball Schläger erweitern. Zeitaufwand: _____ min Schritt-Nummer: _____	**Spiel um Tastatureingabe erweitern, so dass sich die Schläger bewegen.** Zeitaufwand: _____ min Schritt-Nummer: _____	**Spiel um Kollisionsabfrage Ball mit Spielfeldrand erweitern.** Zeitaufwand: _____ min Schritt-Nummer: _____
Klasse Ball erstellen und Ball anzeigen lassen. Zeitaufwand: _____ min Schritt-Nummer: _____	**Spiel um Punktezählung erweitern.** Zeitaufwand: _____ min Schritt-Nummer: _____	**Klasse Racket erstellen und zwei Schläger anzeigen.** Zeitaufwand: _____ min Schritt-Nummer: _____
Klasse Ball um Bewegung erweitern. Zeitaufwand: _____ min Schritt-Nummer: _____	**Klasse für das Spielfeld (*PongPanel*) anlegen und der Anzeige hinzufügen.** Zeitaufwand: _____ min Schritt-Nummer: _____	**Spiel um Überprüfung und Anzeige Spielende erweitern.** Zeitaufwand: _____ min Schritt-Nummer: _____

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*):

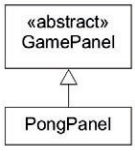
Erklärvideo:

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

Schritt 2

Damit das Spiel durchgeführt werden kann, wird auch ein Spielfeld benötigt. Dieses Spielfeld verwaltet Ball, Schläger und die 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*):

Erklärvideo:




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

Schritt 3

Nun musst du den **Konstruktor der Klasse *PongPanel* implementieren**. An dieser Stelle müssen Objekte der Klassen *Ball* und *Racket* erstellt und dem Panel hinzugefügt werden. Am besten du speicherst 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.

Hinweis:
Für jede Form, die dem Panel hinzugefügt werden soll, muss die Methode `addShape` aufgerufen werden. Nur so wird die Form auch gezeichnet.

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

M 10 Bewertungsbogen

Bewertung für:	Ja	Zum Teil	Nein	Punkte
Funktionalität				
Der Ball bewegt sich über das Spielfeld.				
Der Ball prallt vom oberen und unteren Rand ab.				
Der Ball prallt von den Schlägern ab.				
Zwei Spielende können über die Tastatur jeweils ihren Schläger bewegen.				
Verlässt der Ball nach links oder rechts das Spielfeld, bekommt ein Spieler einen Punkt				
Der Punktestand wird korrekt angezeigt.				
Das Spiel endet, wenn ein Spieler zehn Punkte erreicht hat.				
Es wurde zusätzliche Funktionalität implementiert.				
Code-Qualität				
Der Quelltext ist vollständig kommentiert.				
Es werden sprechende Namen verwendet.				
Es gibt keinen unnötigen Quelltext.				
Das Programm ist strukturiert, d. h. es werden geeignete Methoden verwendet.				
Es gibt keinen Quelltextduplikate.				
Kreativität/Gestaltung				
Das Programm wurde besonders kreativ umgesetzt und die Oberfläche wurde schön gestaltet.				
Insgesamt				

Das ist euch **besonders gut** gelungen:

An diesen Stellen könnt ihr noch **Verbesserungen** vornehmen:

Note: Datum: Unterschrift: