Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Studienprojekt I

Schatzinsel

Entwickeln eines Spiels durch objektorientiertes Programmieren

---

fertiggestellt am TBD

•

Fachbereich Duales Studium Wirtschaft / Technik
Hochschule für Wirtschaft und Recht Berlin

Name:	Jahn, Marko,
Name:	Schenkewitz, Mario, 685593
Name:	Zilius, Sven,
Fachrichtung:	Duales Studium Wirtschaft • Technik
Studiengang:	Informatik
Studienjahrgang:	2020
Betreuer HS:	Zimmermann, Arthur
Anzahl der Wörter:	TDB

Abstract

Kurzfassung

Inhaltsverzeichnis

Abbildungsverzeichnis

Akronyme

OOP Objektorientierte Programmierung

OOPL Objektorientierte Programmiersprachen

1 Einleitung

Die Objektorientierte Programmierung (OOP) ist ein heutzutage weit verbreitetes Computerprogrammiermodell oder Programmierparadigma. In der OOP werden Anwendungen um so genannte Objekte und deren Daten herum strukturiert und nicht um Funktionen oder Logik. So fokussiert sich die OOP auf Objekte mit denen eine Anwendung interagiert, diese Objekte können als Datenfeld mit Attributen und eigenem Verhalten angesehen werden.

Einen tieferen Einblick in das Thema der OOP soll der Abschnitt 2 bieten, in dem die Basis für die OOP näher erörtert wird.

Thema dieses Studienprojektes ist es sich dieses Programmierparadigma zu nutze zu machen um ein Spiel zu entwickeln.
Der Titel des Spiels ist „Schatzinsel“, es handelt von drei Piraten die sich auf einer Insel befinden, auf der ebenfalls ein Schatz versteckt ist. Es ist ihnen unbekannt wo sich dieser Schatz befinden. Durch zufällige Bewegung über die Insel wollen sie versuchen diesen Schatz zu finden. Jeder der Piraten hat verschiedene Eigenschaften, wie zum Beispiel Geschwindigkeit.
Sobald einer der Piraten den Schatz entdeckt, ruft er die anderen zu sich. Am Ende versammeln sie sich alle um den Schatz.

Dieser Umstand soll durch das Spiel modelliert und gegebenenfalls erweitert werden.

2 Grundlagen

2.1 Objektorientierte Programmierung

Objektorientierte Programmiersprachen (OOPL) eignen sich durch die Konzentration auf die Objekte mit denen das Programm interagieren soll dazu, dass Software Wartbar und lesbar bleibt. Objekte und ihre Methoden können getrennt von einander erstellt und Programmiert werden. Diese Unterteilung in Einzelteile ermöglicht des weiteren die Wiederverwendbarkeit von Code, also dem Quelltext, entweder in dem die Klasse selbst wiederverwendet wird, oder eine andere Klasse von ihr erbt. Dazu mehr im Abschnitt 2.3

OOP ist ein Programmierstil, der mittlerweile in Unternehmen, Industrie und akademischen Kreisen seinen Platz gefunden hat. Es wird ein natürlicheres und dadurch ein mächtigeres Entwurfsparadigma insofern erwartet, dass ein Programmierer produktiver ist, wenn das Anwendungsmodell näher am Problem modelliert werden kann, dass gelöst wird.
Anders gesagt ist eine OOPL mächtiger, weil es einem Programmierer erlaubt ein Problem zu lösen, dass die Struktur einer Welt nachbildet in welcher das Problem entstand. Vertrautheit mit solch einer Sprache eröffnet Wege zu einer klaren und natürlichen Lösung [Yae14].

Es werden also im betrachteten Fall der Schatzinsel die Objekte, die aus der echten Welt bekannt sind, so modelliert, dass die Interaktionen und Verhaltensweisen natürlich in einer Anwendung modelliert werden können.

2.2 Datenabstraktion und Verkapselung

Einer der ersten Schritte bei der OOP ist es alle Objekte zu sammeln, die manipuliert werden müssen. Im Fall des Spiels „Schatzinsel“ gibt es zum Beispiel Piraten, eine Insel, ein Schatz, aber auch eine graphische Benutzeroberfläche die alle als Objekt betrachtet werden können. Sie müssen untereinander aber auch mit dem Benutzer interagieren können.

Einer der großen Durchbrüche bei der Gestaltung von Programmiersprachen geschah, als es ermöglicht wurde, dass Programmierer eigene Datentypen definieren

konnten. C.A.R. Hoare's record class in Simula67 gilt als die Quelle der Idee. [Yae14] Verbreitung des, durch Programmierer definierten, Datentyps fand durch die Sprache Pascal statt. [Wir71]

So kann in Pascal ein Verbund definiert werden, der selbst aus Daten besteht, siehe Listing 2.1.

```
1 TYPE Auto = RECORD
2 KFZ: String[12];
3 Gewicht: Real;
4 AnzPassagiere: Integer;
5 END;
```

Listing 2.1: Verbund Beispiel Definition [Zim20]

Dieses Beispiel (Listing 2.2) definiert einen neuen Datentyp *Auto*, in dem jeweils die Art des KFZ, das Gewischt und die Anzahl der Passagiere angegeben wird. Nun kann man den Typen *Auto* nutzen um beliebig viele Daten-Entitäten zu erstellen.

```
1 VAR a : Auto;
2 BEGIN
3 a.KFZ := 'B XY 123456';
4 a.Gewicht := 555.77;
5 a.AnzPassagiere := 5;
6 END;
```

Listing 2.2: Verbund Beispiel Instantiierung [Zim20]

Manipulationen der Daten innerhalb des Verbunds werden extern vom Datentyp selbst betrachtet, die Syntax der Sprache indiziert somit nur eine Gruppierung von Daten.

Verbunde in Pascal werden homogen genannt, wenn alle Daten innerhalb des Verbunds den selben Typ haben. Sie sind heterogen, wenn es verschiedene Typen, wie im Beispiel, gibt.

Pascal erlaubt zwar die Erstellung von heterogenen Verbunden, jedoch ist es nicht möglich in diesen Verbunden Operationen einzubauen. Anders als bei Pascal ist die bei Java zum Beispiel möglich.

```
1 class Point {
2 private int x, y;
```

```
3           public int getX() { return x; }
4           public int getY() { return y; }
5           public void setXY(int x, int y) {
6                   if (x >= 0 && y >= 0) {
7                           this.x = x;
8                           this.y = y;
9                   }
10          }
11  }
```

Listing 2.3: Operation innerhalb einer Klasse [Yae14]

Im Listing 2.3 wird zum Beispiel durch die Funktion „setXY()“ermöglicht, dass x und y gesetzt werden können. Es ist zu bemerken, dass verschiedene Sichtbarkeiten gesetzt werden. Dadurch, dass x und y auf „private“gesetzt sind, können sie also von außen nur durch die Funktion „setXY()“geändert werden. Dies veranschaulicht auch das Konzept der Verkapselung [Yae14] Seite 4.

Datenabstraktion in ihrer vollen Kapazität beschreibt also die Möglichkeit für Programmierer ihre eigenen Datentypen zu definieren, diese Typen können eine Heterogene Struktur haben und die Sicht auf diese kann eingeschränkt werden. Außerdem ist es möglich Operationen innerhalb dieser Datentypen zu haben.

In diesem Zusammenhang kann gesagt werden, dass Objekte also aus Attributen und Operationen, auch Funktionen oder Methoden genannt, bestehen und Instanzen von diesen Datentypen sind. Diese Datentypen werden im allgemeinen auch als Klassen bezeichnet.

2.3 Vererbung

2.4 Polymorphismus

2.5 Verwandte Konzepte

3 Realisierung

3.1 Betrachtete Umgebungen

3.1.1 Java Swing

3.1.2 PyGame

3.1.3 Unreal Enginge

3.1.4 Unity Enginge

3.1.5 Auswahl der Umgebung

3.2 Verwendete Werkzeuge

3.2.1 Git und Github

3.2.2 Eclipse

3.2.3 Unreal Engine Editor

3.2.4 Unity Engine Editor

3.3 Zusammenfassung der Anforderungen

3.4 Anforderung 1

3.4.1 Ansatz

3.4.2 Umsetzung

3.5 Anforderung 2

3.5.1 Ansatz

4 Ergebnisse

5 Fazit

6 Ausblick

Literaturverzeichnis

[Wir71] Wirth, N. (1971). *The Programming Language Pascal.* Acta Informatica 1. pp. 35–63.

[Yae14] Yaeger, D. P. (2014). *Object-Oriented Programming Languages And Evend-Driven Programming.* Mercury Learning and Information, Dulles, Virginia; Boston, Massachusetts; New Delhi.

[Zim20] Zimmermann, A. (2020). Einführung in die Programmierung, Aggregierte Datentypen Übung. http://azdom.de/hwr/prog/.

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich:

1. dass ich meine Bachelor-Thesis selbstständig verfasst habe,

2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,

3. dass ich meine Bachelor-Thesis bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Ort, Datum	Jahn, Marko
Ort, Datum	Schenkewitz, Mario
Ort, Datum	Zilius, Sven