

Snake Spiel

Mit Java Eclipse



25. Januar 2022

Almir Abdi

Modul 226B IFZ-2026-05A

Inhaltsverzeichnis

[1.Vorwort 2](#_Toc93965818)

[1.1 Vorgehensweise 2](#_Toc93965819)

[2. UML Klassendiagramm 3](#_Toc93965820)

[3.Code 4](#_Toc93965821)

[4.Links 7](#_Toc93965822)

[5.Anhang 7](#_Toc93965823)

[5.1 Selbstständigkeitserklärung 7](#_Toc93965824)

# 1.Vorwort

Als wir im ersten Block im Modul226B uns mal Gedanken machen sollten, was für ein Projekt wir mit Java programmieren sollte, habe ich lange überlegt, bis ich zu meiner Entscheidung gekommen bin.

Bevor wir uns entscheiden mussten, hatten wir in der Lernwerkstatt einen Crashkurs zur Git und GitHub. Es war schon länger her, dass ich mit GitHub gearbeitet habe und mit Git hatte ich bis dahin auch nur wenig zu tun gehabt. Nach paar Probe Test auf GitHub hochzuladen habe ich mich vertrauter mit GitHub und Git gemacht.

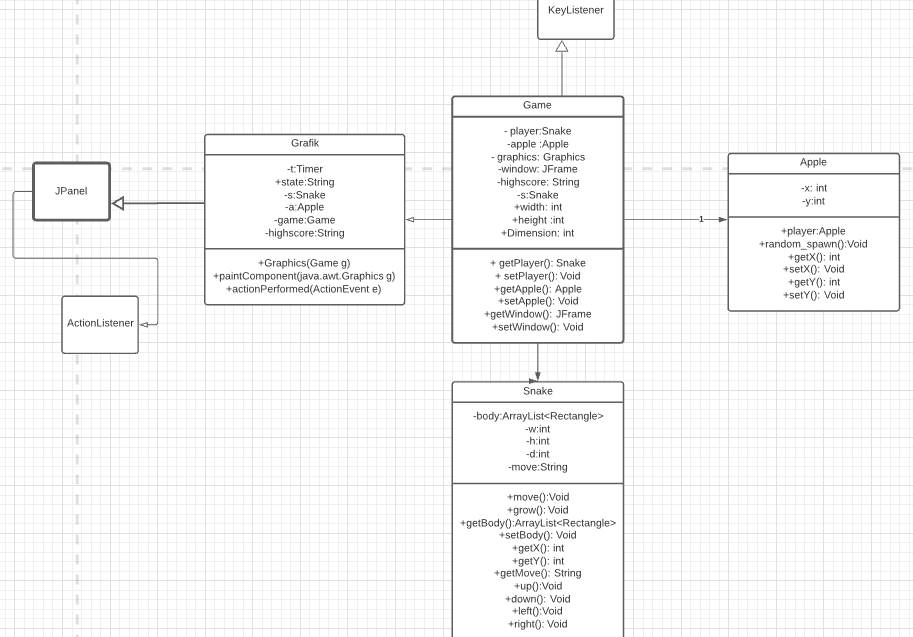
Es gab viele Möglichkeiten was man Programmieren konnte. Ich hatte erstmal eine Idee was mit Benutzerflächen zu machen mit JavaFx aber habe mich doch umentschieden ein Spiel zu programmieren. Es gab viele Spiele, die man mit Java programmieren konnte. Bevor ich mich entschieden habe welches Spiel ich programmiere, habe ich erstmal darüber nachgedacht, ob ich mit Processing oder mit Eclipse arbeiten werde und ich habe mich für Eclipse entschieden. Ich hatte drei Spiele als Auswahl Möglichkeiten Pong, Snake, Jump and Run Pong habe ich ausgeschlossen, weil ich es schonmal mit Python programmiert habe, Jump and Ran wollte ich eher nicht machen, daher habe ich mich für Snake entschieden. Ich fand das Snake eine gute Entscheidung ist, weil wir schon in anderen Modulen ähnliche Spiele, wie Snake programmiert haben, was gut ist, denn wenn es Unklarheiten gibt, kann ich dort nachschauen.

## Vorgehensweise

Ich habe mit dem UML Klassendiagramm angefangen und notiert, was ich alles einbauen will, welche Klassen, Vererbungen, Beziehungen. Ich habe auch das Bewertungsraster angeschaut, um zu schauen, was ich alles haben sollte, dabei habe ich auch noch Veränderungen beim UML Klassendiagramm vorgenommen.

Ich habe in Eclipse ein neues Projekt erstellt und Modell-View-Controller (MVC) verwendet und habe angefangen die Klassen zu erstellen.

# 2. UML Klassendiagramm



# Ein Bild, das Text enthält. Automatisch generierte Beschreibung3.Code

In der Klasse Start wird die graphics.state auf Running gesetzt.

In der Klasse Update wird geschaut ob graphics.state gleich Running ist, wenn ja soll der player und apple angezeigt werden und das Spiel ist am Laufen.

Wenn check\_wall-oder check\_self\_collision auftretet soll die State auf End wechseln und der Highscore wird angezeigt

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungBeim check\_wall\_collision wird geschaut, ob der Player die Koordinaten des Fensterende Übertritten hat.

Beim chack\_food\_collision wir geschaut, ob der Player im Radius vom Apple ist.

Beim check\_self\_collision wird geschaut, ob der Player sich selbst berührt.

Wenn State Start ist, sollte ein weisser Text «Drücke irgendeine Taste» abgebildet sein.

Wenn State auf Running ist, sollte der Apple rot aufgefüllt werden und der eigene Score und der Highscore sollten angezeigt werden, wenn nicht dann sollte nur das erreichte Score angezeigt werden.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

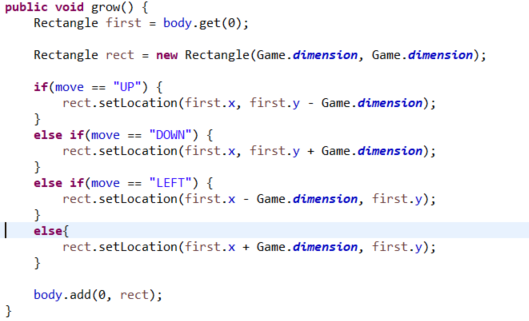
Ein Bild, das Text enthält.

Automatisch generierte Beschreibungx und y werden auf random gesetzt und Rectangle r die gleichen Koordinaten hat wie x und y dann wird onSnake auf true gesetzt und x und y bekommen wieder random Koordinaten.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Snake wird herstellt mit drei Rect’s lang und move ist auf Nothing was heisst das Snake bewegt sich noch nicht.



Bei der Klasse update wird player.grow(); abgerufen wenn es den Apple isst und in der Klasse grow wird der neue Rect an der der Snake hinten angefügt.

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungDer Player (Snake) wird mit den Pfeiltasten gelenkt, wenn State auf Running ist wenn nicht wird die Klasse Start aufgerufen.

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungIn der checkScore wird der Highscore überprüft.

Wenn der neue Score Gösser ist als der Highscore muss man seinem Namen angeben und der neue Highscore wird auf dem File highscore.txt gespeichert.

Der Alte Highscore wurde vom neuem überschrieben.

# 4.Links

1. <https://www.markdownguide.org/cheat-sheet/>
2. <https://stackoverflow.com/questions/15544549/how-does-paintcomponent-work>
3. <https://www.learnjavacoding.com/definitions/snake/>
4. <https://www.youtube.com/watch?v=91a7ceECNTc>
5. <https://www.youtube.com/watch?v=8gMd0ftWp_Y>

# 5.Anhang

## 5.1 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne

fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet

habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen

Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Datum:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Unterschrift:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bewertungsraster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name, Vorname** |  | | | |
| **Klasse** | **IFZ2026** | | | |
| **Fachliche Dokumentation** | **Hanok** | **max. Pkt.** | **erreichte Punkte** | **Begründung bei Abzug** |
| Klassendiagramm enthält mind. 2 Vererbungen | 2 | 3 |  |  |
| Klassendiagramm enthält abstrakte Basisklasse | 2 | 3 |  |  |
| Klassendiagramm enthält mindestens ein Interface | 2 | 3 |  |  |
| Vererbung anhand Code-Snippet aufgezeigt | 3 | 3 |  |  |
| Abstrakte Klasse anhand Code-Snippet aufgezeigt | 3 | 3 |  |  |
| Interface anhand Code-Snippet aufgezeigt | 3 | 3 |  |  |
| Vererbung und Interfaces plausibel und sinnvoll eingesetzt | 1 | 6 |  |  |
| mind. sinnvolle 3 Unit-Tests für unterschiedliche Klassen vorhanden und dokumentiert | 4 | 6 |  |  |
| Mockup-Test vorhanden und beschrieben | 4 | 3 |  |  |
| Hilfestellungen aufgeführt |  | 3 |  |  |
| Coding Style: Javadoc |  | 2 |  |  |
| Coding Style: Namenskonventionen |  | 2 |  |  |
| Coding Style: Einrückungen |  | 2 |  |  |
| Assoziationen im Klassendiagramm korrekt |  | 5 |  |  |
| **Dokumentation formal** |  |  |  |  |
| Vollständig, inkl. Inhaltsverzeichnis |  | 1 |  |  |
| Allg. Beschreibung des Projekts |  | 2 |  |  |
| Fachlich richtig, inkl. Fachsprache |  | 3 |  |  |
| Sauberkeit, Layout |  | 3 |  |  |
| Profimässiger Tools-Einsatz (z.B. sinnvolle  Formatierung Code-Snippets (kein  DarkMode!), Abbildungen mit  Bildunterschriften und Referenzen) |  | 5 |  |  |
| Nachvollziehbarkeit |  | 3 |  |  |
| Sprache (Stil/Orthographie) |  | 3 |  |  |
| **Codeversionierung** |  |  |  |  |
| Codeversionierung genutzt |  | 3 |  |  |
| sinnvolle Commits (Anzahl, Commit-  Messages, nur sinnvolle Dateitypen) |  | 5 |  |  |
|  |  |  |  |  |
| **Arbeitsjournal (täglich führen!)** |  |  |  |  |
| Vollständig (mind. Alle Lehrwerkstattblöcke vorhanden) |  | 5 |  |  |
| täglicher Ablauf nachvollziehbar (Ziele und Herausforderungen ersichtlich) |  | 5 |  |  |
|  |  |  |  |  |
| **Fazit** |  |  |  |  |
| Reflexion: was habe ich gelernt, wo gab es  Schwierigkeiten (analog reguläres WISS Moduljournal) |  | 3 |  |  |
| **Excellenz (max 5 Punkte)** |  | 5 |  | z.B. Serialisierung eingebunden und genutzt, Zustandsautomat Spiel sauber beschrieben oder  DB-Anbindung erstellt |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Gesamt** |  | **93** |  |  |
| **Note** |  |  |  |  |