Patterns & Frameworks

SnakeCore, SnakeFX, SnakeServer, SnakeTest

Ostfalia Fachhochschule für angewandte Wissenschaften

Benjamin Wulfert (b.wulfert@ostfalia.de) | Mat.-Nr.: 70454350

Leonard Reidel (@ostfalia.de) | Mat.-Nr.: 70468602

Vorgelegt bei: Dipl.-Inform. Bettina Meiners

Semester: Wintersemester 2020

19.02.2020

Agenda

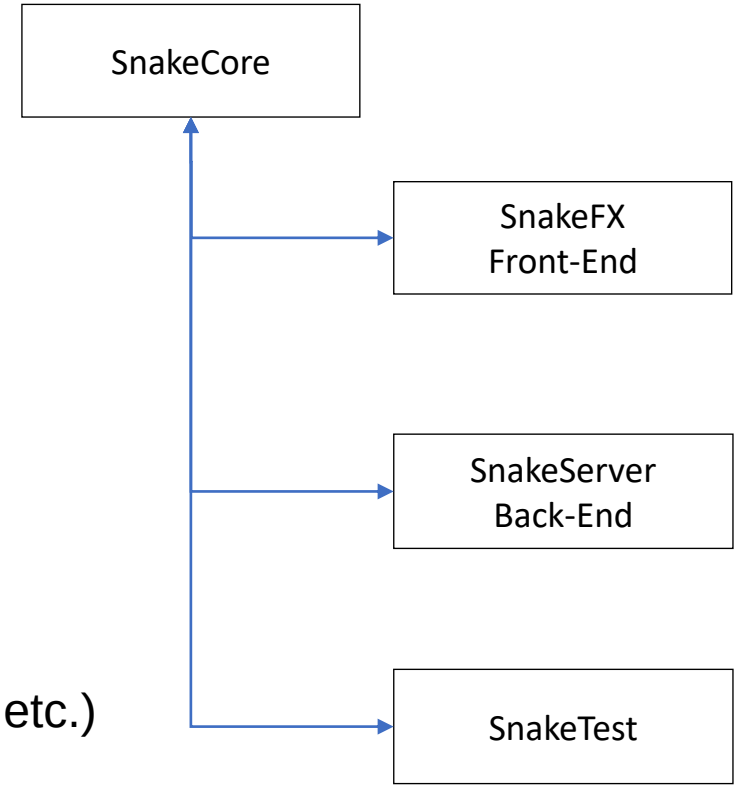
- Projekt / Architektur
- Core/Common
- Backend
- Frontend
- Hands-On

Allgemeines

- Projekt-Name „SnakeFX“
- Front-End Modul – ebenfalls „SnakeFX“ (erstes Modul)
- Build-Management-System - Apache Maven
- Multi-Module Project
- Front-End: JavaFX
- Back-End: Spring, Spring Boot, …
- Core: Plain Java

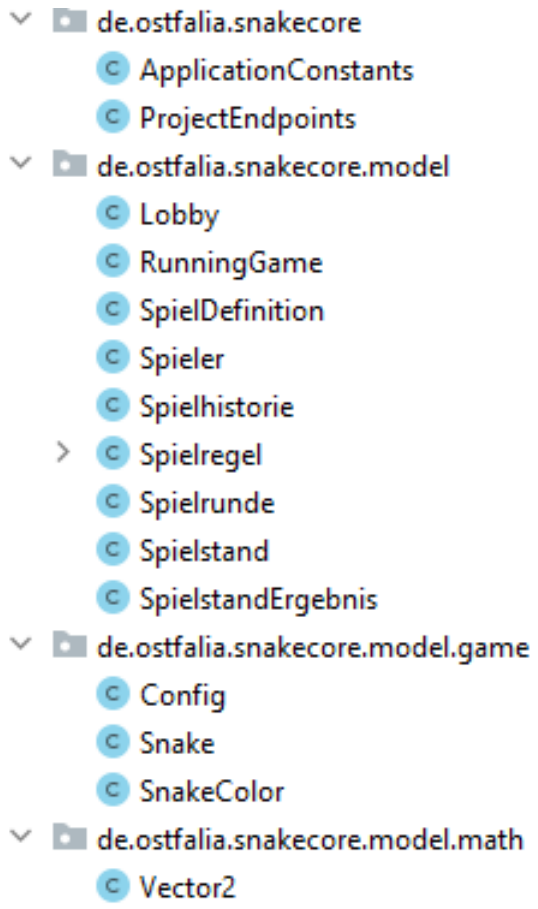
Projekt-Architektur

- SnakeCore
  Enthält grundlegende & gemeinsame Aspekte des Projekts
- SnakeFX
  Frontend-Modul - das User-Interface (GUI)
- SnakeServer
  Backend-Modul - der Application-Server
- SnakeTest
  Test-Modul - Test-Fälle für verschiedene Szenarien
  (z.B. Start des Backends & zweier Clients, auto. Login, auto. Play, etc.)



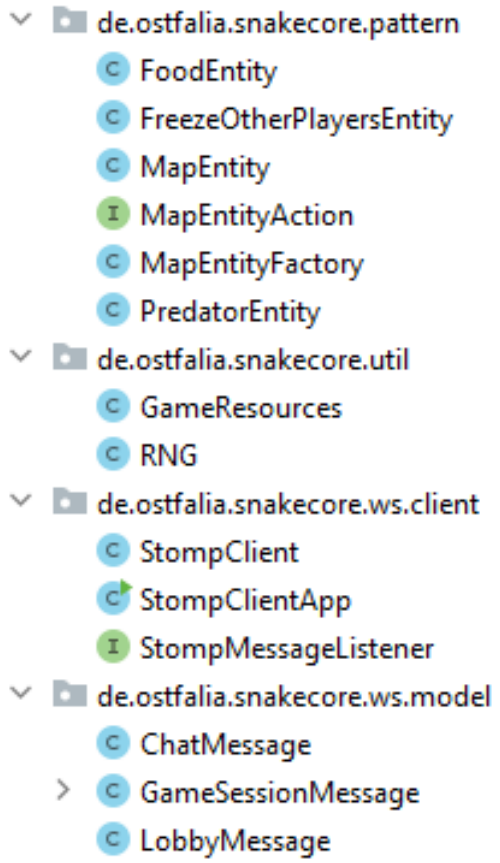
Core / Common - SnakeCore

- ApplicationConstants / ProjectEndpoints
  - String-Konstanten
- model
  - Entitäten des Projekts
- game
  - Entitäten bzgl. der Spiel-Laufzeit
- math
  - Implementierung von gängigen (2d) Vektor-Operationen



Core / Common - SnakeCore

- Pattern
  - Implementierung der Design-Pattern
- Util
  - Hilfsklassen: GameResources verwaltet Bilder & Sounds
  - RNG (Random Number Generator): Klasse zur Erzeugung von Zufallszahlen
- ws.client
  - Implementierung des Stomp-Clients, TestApp, ...
- ws.model
  - Entitäten d. Nachrichtenaustauschs



Backend - SnakeServer

- Spring Boot
- Spring Data JPA (h2, Hibernate)
- Spring Messaging (STOMP, Websockets)
- Spring Web (HTTP, RESTful Webservices, …)

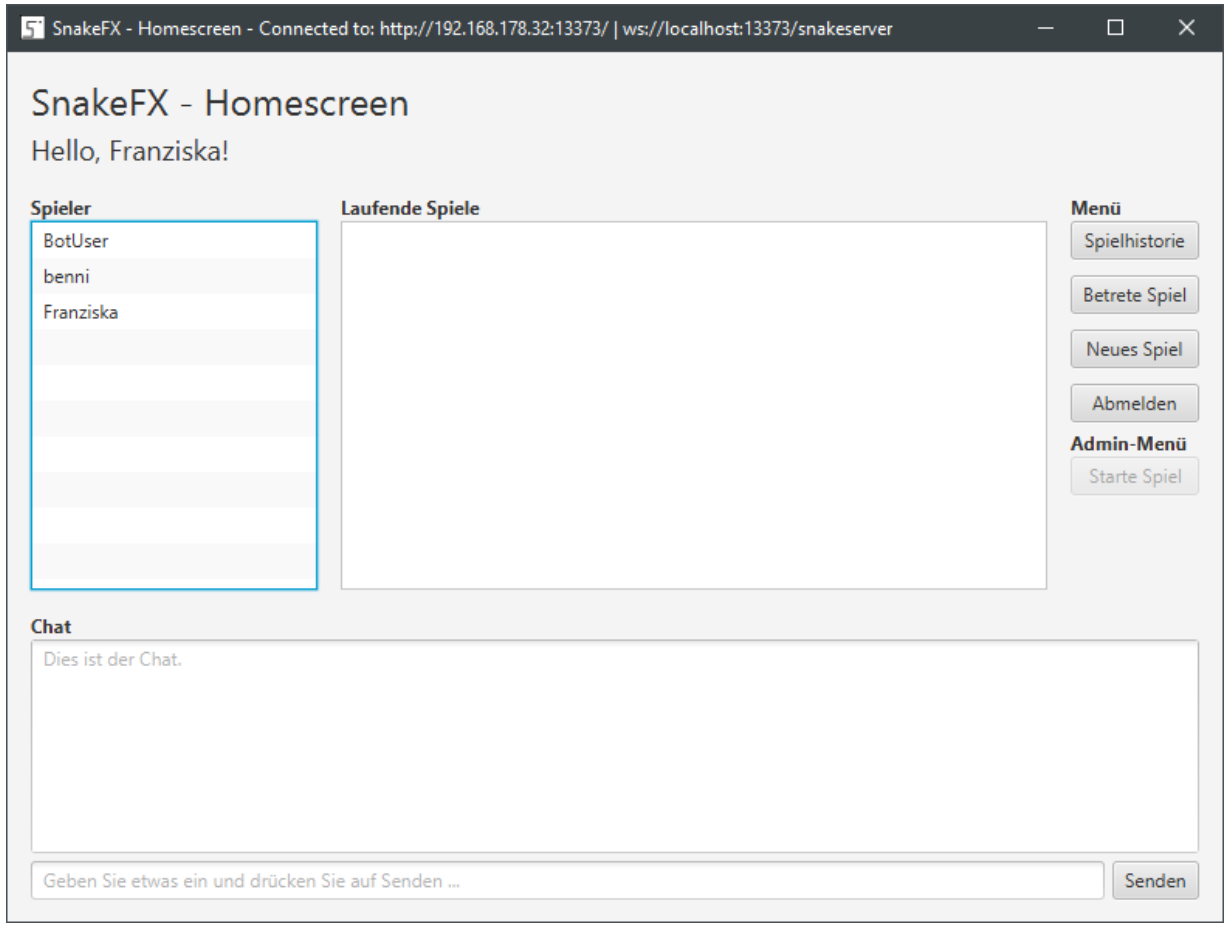
Kommunikation

- HTTP – RESTful Webservices
  - Spring Web
  - Synchron
- WebSockets – STOMP
  - Spring Messaging
  - Asynchron

Kommunikation

- HTTP – RESTful Webservices - Synchron
  - Login
  - Register
  - Initialer Bezug von Daten (Spielhistorie, Lobby, etc.)
- WebSockets – STOMP – Asynchron, Updates
  - Lobby-Daten
  - Spielhistorie
  - Spieleingaben

Kommunikation - Lobby



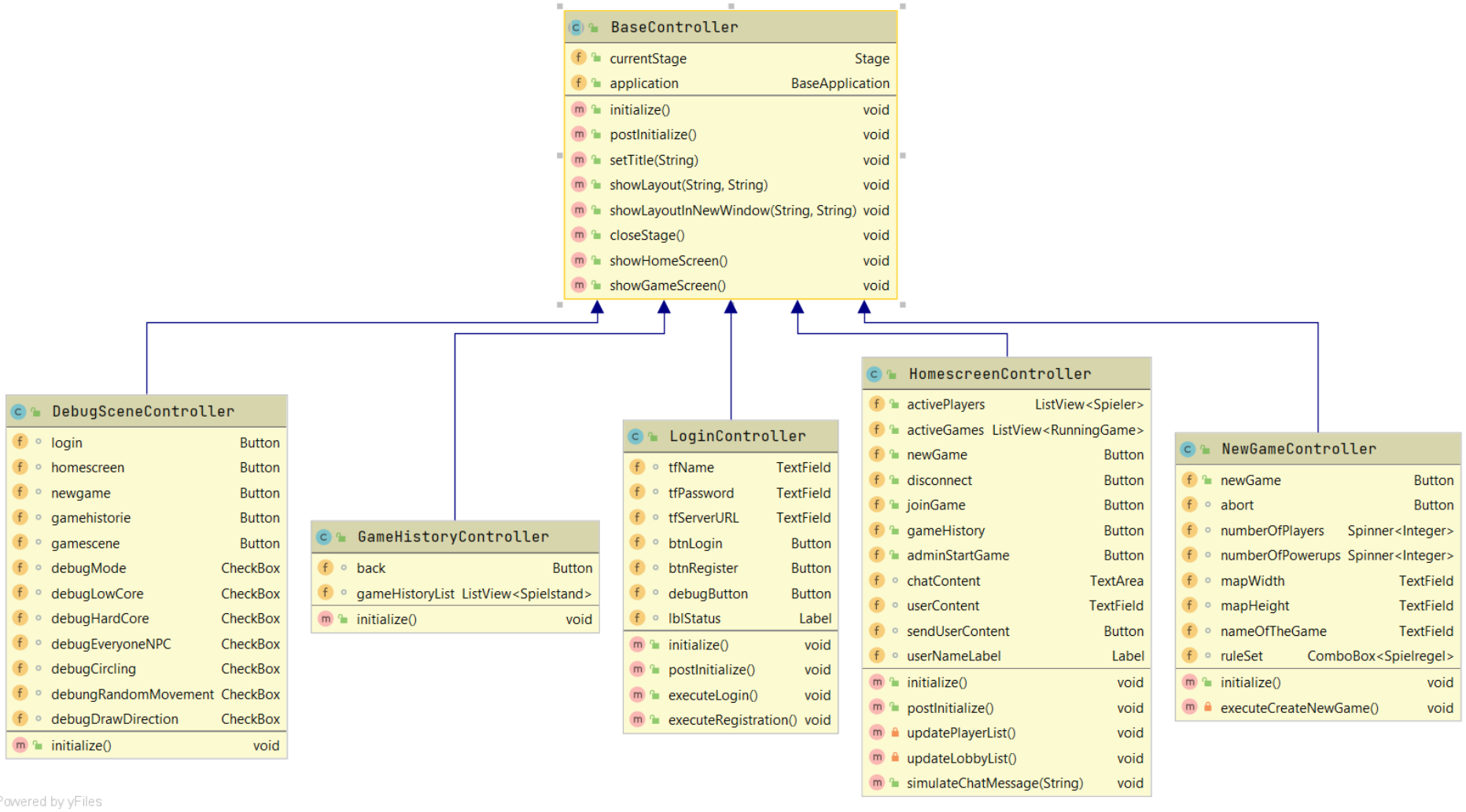
Kommunikation - Lobby

```java
@MessageMapping("/games") // "/app/games/
@SendTo("/topic/games")
public LobbyMessage broadcastGames(LobbyMessage lobbyMessage) {

    if (lobbyMessage.logout && lobbyMessage.logoutSpieler != null) {
        lobbyController.getCurrentPlayers().remove(lobbyMessage.logoutSpieler);
        System.out.println("Removing player: " + lobbyMessage.logoutSpieler + " from the active players of the lobby ...");

        lobbyMessage.activeClients = lobbyController.getCurrentPlayers();
    }

    if (lobbyMessage.spielDefinition != null) {
        System.out.println("New game definition recieved: " + lobbyMessage.spielDefinition.getNameOfTheGame());
        System.out.println("Adding it to the lobby ... ");

        // create a new runningGame based on the SpielDefinition
        RunningGame newRunningGame = new RunningGame(
                stompPath: "/topic/games/1",
                lobbyMessage.admin,
                new LinkedList<>(Arrays.asList(lobbyMessage.admin)),
                lobbyMessage.spielDefinition
        );

        // add it to the lobby
        lobbyController.add(newRunningGame);
    }

    // re-reference the current running games from the lobby to this message
    lobbyMessage.runningGames = lobbyController.getRunningGames();

    // broadcast it to every connected client
    return lobbyMessage;
}
```

Frontend - SnakeFX

- In JavaFX implementiert
- Vererbungshierarchien zwischen Controller
- Nutzt Dependency Injection (DI) für Views in Controllern
- Besitzt Zugriff auf STOMP-Client Impl.
  - Dadurch Subscribe auf Endpoints möglich, sowie Listener um auch Nachrichten zu reagieren
  - Message-Transport um Daten an das Backend zu senden
- Besitzt Zugriff auf HTTP-Client Impl.
  - Dadurch können Daten an das Backend übertragen werden,
    wie z.B. bei Login, Register Operationen oder Polling für HTTP

Frontend - SnakeFX



Snake Implementierung

- Feldbeschaffenheit
- Schlangenbeschaffenheit
- Power Ups
- Aktionen (ohne Power Ups)
  - Schlange trifft Wand
  - Schlange trifft andere Schlange
  - Schlange trifft sich selbst

Feldbeschaffenheit

Ein Koordinaten System, gezeichnet auf einer Canvas mit der Größe übergeben von der Spielerstellungsmaske

```
config.height = (int) gameCanvas.getHeight();
config.width = (int) gameCanvas.getWidth();
config.columns = runningGame.spielDefinition.getMapWidth();
config.rows = runningGame.spielDefinition.getMapHeight();
```

Schlangenbeschaffenheit

Schlange: Liste mit dem Kopf als erstes Listenelement
snake.head = snake.body(0)

Initialisierung:

```
public Snake(Vector2 spawn, SnakeColor color) {
    for (int i = 0; i < initialLength; i++) {
        Vector2 bodyPart = new Vector2(-1, -1);
        body.add(bodyPart);
    }
    head = body.get(0);

    head.x = spawn.x;
    head.y = spawn.y;

    this.color = color;

    isPredator = false;
}
```

Schlangenbeschaffenheit

Schlangenbewegung

- Spieler Eingabe wird in Richtungsvektor gewandelt

```
// initialize the map with keycode to direction-vector entries
inputDirectionMap.put(KeyCode.SPACE, Vector2.ZERO);
inputDirectionMap.put(KeyCode.UP, Vector2.UP);
inputDirectionMap.put(KeyCode.DOWN, Vector2.DOWN);
inputDirectionMap.put(KeyCode.RIGHT, Vector2.RIGHT);
inputDirectionMap.put(KeyCode.LEFT, Vector2.LEFT);
```

- Richtungsvektor und Koordinaten von snake.head ergeben Position des neuen Schlangenkopfes
- Jedes Listenelement nimmt den Wert des vorangehenden Listenelements an

Power Ups

Drei verschiedene PowerUp typen durch Factory Patterns als MAP_ENTITIES umgesetzt Umgesetzt

```
public static final int MAP_ENTITY_KIND_FOOD = 1;
public static final int MAP_ENTITY_KIND_PREDATOR_ENTITY = 2;
public static final int MAP_ENTITY_KIND_FREEZE_OTHER_PLAYERS_ENTITY = 3;
```

Koordinatenabgleich der Schlangenköpfe und der Foodpositionen

```
// the coordinates of the snakes head
int sxcord = snake.head.getX();
int sycord = snake.head.getY();

// iterate over every mapEntity instance
for (MapEntity mapEntity : mapEntityList) {

    // if the head matches the position of a mapEntity
    if (sxcord == mapEntity.getPosition().x && sycord == mapEntity.getPosition().y) {

        // mark the corresponding mapEntity to be removed from the game board
        isFrameRemoval = true;
        toRemove = mapEntity;
```

Aktionen (ohne Power Ups)

Schlange trifft andere Schlange

- Abfrage ob die Schlange durch ein vorheriges PowerUp Predator ist → GameOver falls nicht
- Falls schon, anhängen der anderen an Eigene

```
//check that predator snake doesn't eat itself
/*
for (Snake snake : snakeList) {
    for (Snake otherSnake : snakeList) {

        if (otherSnake != snake) {

            for (Vector2 part : otherSnake.body) {
                if (snake.head.equals(part)) {
                    if (!snake.isPredator) {
                        checkGameOver();
                    } else {
                        int totalLenght = otherSnake.body.size();
                        int splittingPoint = otherSnake.body.indexOf(part);
                        int growth = totalLenght - splittingPoint;

                        for (int i = splittingPoint; i < totalLenght; i++) {
                            //cut bitten Snakes body
                            otherSnake.body.remove(i);

                            //add to biting Snake
                            Vector2 newPart = new Vector2(-1, -1);
                            snake.body.add(newPart);
                        }
                    }
                }
            }
```

Aktionen

Schlange trifft Wand

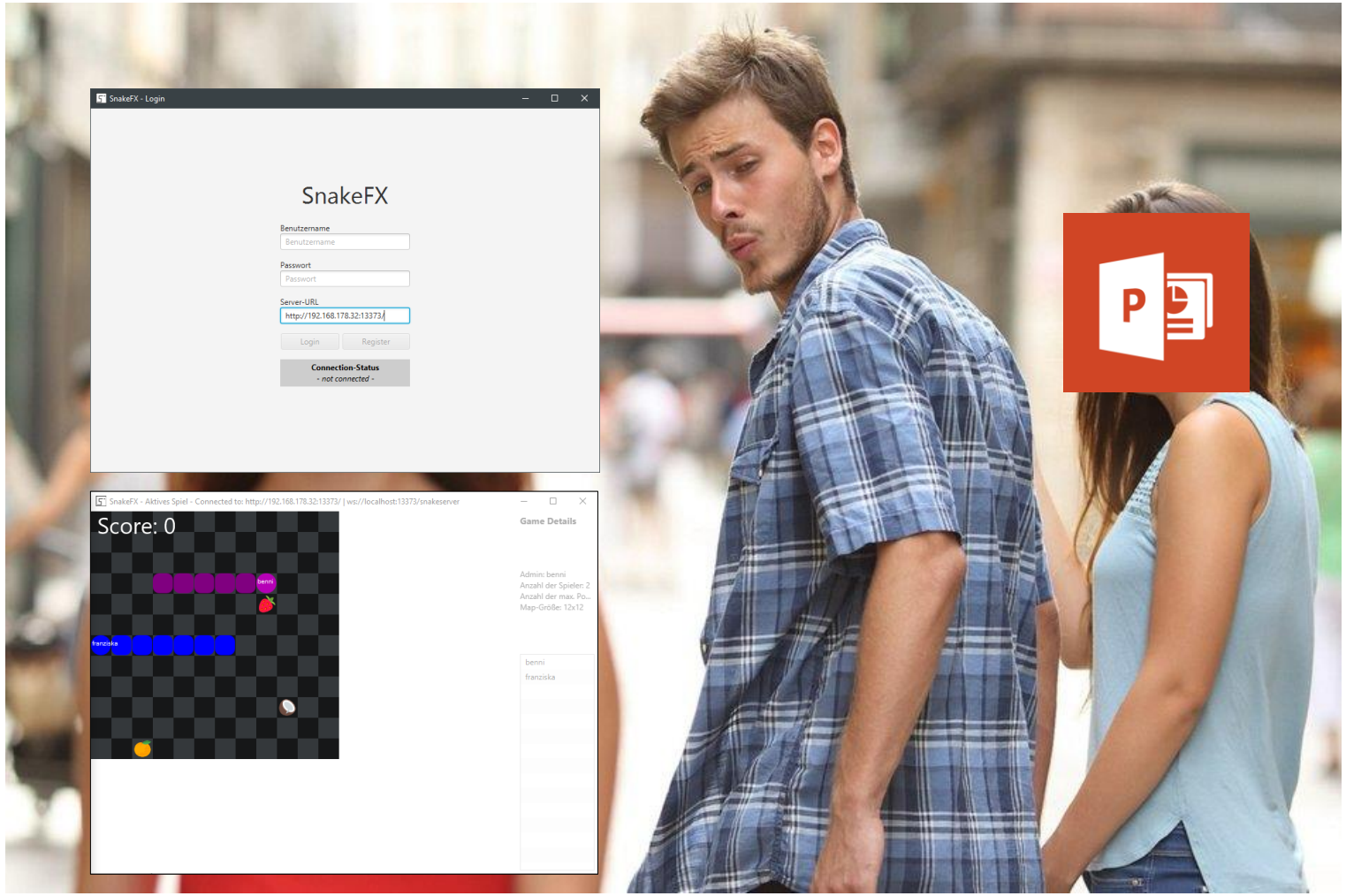
```
// check for player & wall collisions
for (Snake snake : playerSnakeMap.values()) {

    // hitting a wall teleports a player to the other side
    if (snake.head.x < 0) {
        snake.head.x = config.rows - 1;
    }
    if (snake.head.y < 0) {
        snake.head.y = config.columns - 1;
    }
    if (snake.head.x > config.rows - 1) {
        snake.head.x = 0;
    }
    if (snake.head.y > config.columns - 1) {
        snake.head.y = 0;
    }
```

Aktionen

Schlange trifft sich selbst

```
// if a snake hits itself, the related player of the snake has lost the game
for (int i = 1; i < snake.body.size(); i++) {
    if (snake.head.x == snake.body.get(i).getX() && snake.head.getY() == snake.body.get(i).getY()) {

        // update the game-over state for this player
        Spieler currentPlayerForSnake = snakePlayerMap.get(snake);
        spielerGameOverMap.put(currentPlayerForSnake, true);

        break;
    }
}
```

Hands On


SnakeFX - Login
SnakeFX
Benutzername
Benutzername
Passwort
Passwort
Server-URL
http://192.168.178.32:13373/
Login
Register
Connection-Status
- not connected -
SnakeFX - Aktives Spiel - Connected to: http://192.168.178.32:13373/ | ws://localhost:13373/snakeserver
Score: 0
Game Details
Admin: benni
Anzahl der Spieler: 2
Anzahl der max. Po...
Map-Größe: 12x12
benni
franziska
benni
franziska