Die Hausaufgaben müssen von jedem Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws1920/programming-methodologies/` zu berücksichtigen.

Abgabefrist ist der 30.01.2020 - 23:59 Uhr

Abgabe

Wir benutzen für die Abgabe der Hausaufgaben Git. Jedes Repository ist nur für den Studierenden selbst, sowie für die Betreuer und Korrekteure sichtbar. Für diese und voraussichtlich alle zukünftigen Abgaben wird kein neues Repository benötigt. Nutzt weiterhin jenes, welches mit folgendem Link angelegt wurde:

`https://classroom.github.com/a/X9QUkSjx`

Nicht, oder zu spät gepushte (Teil-)Abgaben werden mit 0 Punkten bewertet.

Vorbereitung

Zur Bearbeitung der Hausaufgabe sollte eine Entwicklungsumgebung verwendet werden. Wir empfehlen aufgrund der Nachvollziehbarkeit die Verwendung von IntelliJ (siehe Aufgabenblatt 3). Die Abgabe muss als **lauffähiges** Projekt abgegeben werden.

Abgaben, die nicht lauffähig sind, werden mit 0 Punkten bewertet!

Aufgabe 1 - Houses & Shrooms (40P)

In dieser Aufgabe ist das Ziel, die Darstellungen der Houses und Shrooms mithilfe eigener View- und Controller-Dateien zu realisieren. Hierzu sollen die selbst gewählten Darstellungsformen in eine FXML-Datei verschoben und wie gewohnt mit einer Controller-Klasse verknüpft werden. Die Initialisierung der Komponenten geschieht dabei jeweils über eine setContent() genannte Methode.

Eine weitere Besonderheit stellt die ShroomController-Liste in Codesnippet 2 dar. Diese weicht von dem Grundsatz ab, dass die Controllerklassen nicht miteinander verbunden sein dürfen. Die Verbindung ist maßgeblich für die Darstellung der Shrooms notwendig, da sich diese relativ zu den Houses positionieren.

```
package de.uniks.pmws1920.controller.ingame;

public class IngameController {

    private ArrayList<House> houseList = new ArrayList<>();

    // ========================================================================
    // Controller Init
    // ========================================================================

    @Override
    public void setBuilder(ModelBuilder builder) {
        super.setBuilder(builder);
        // manage to get House list
        this.placeHouses();
    }

    // ========================================================================
    // Fake Ticks
    // ========================================================================

    @FXML
    public void nextTick() {
        // call reinforce Houses
    }

    // ========================================================================
    // View Init
    // ========================================================================

    private void placeHouses() {
        // for every house in the list
        // Load View Component
        // Calc coordinates for the house
        // Get Controller
        // Set Controller Content
    }
}
```

Abbildung 1: IngameController

```
package de.uniks.pmws1920.controller.ingame;

public class HouseController {

    @FXML
    public <YOUR_HOUSE_SHAPE> <YOUR_VARIABLE_NAME>;

    private ArrayList<ShroomController> shroomController = new ArrayList();
    private ModelBuilder builder;
    private AnchorPane pane;
    private House house;
    private Player player;

    private double xPos;
    private double yPos;
    private double shapeXOrigin;
    private double shapeYOrigin;

    // ============================================================================================
    // Init
    // ============================================================================================

    public void setContent(ModelBuilder builder, AnchorPane pane, House house, double xPos, double yPos) {
        // save parameter in fields
        // maybe additional location calculations
        this.setup();
    }

    private void setup() {
        this.bindValues();
        this.placeSelf();
        this.placeShrooms();
    }

    // ============================================================================================
    // Init View
    // ============================================================================================

    // A tip to make the future homework a lot easier.
    // Follow the instructions about shroom-object and shroom-controller comparison.
    private void bindValues() {
        // compare the referenced shrooms of the house with the controllers in the list above
        // If new shrooms are contained, create a new controller for these
        // Bind shape to pane
        // Bind property change listener (aka new shrooms target this house)
        // -> see if shroom-controller exists
        // -> manage display of shrooms
    }

    // Another tip for clean coding, extract as much reused functionality into their own
    // methods as you can. This controller creation is just one good example.
    private void createNewShroomController(Shroom shroom) {
        // Load View Component
        // Calc coordinates for the shroom
        // Get Controller
        // Set Controller Content
    }
```

```
    // ========================================================================================
    // Update View
    // ========================================================================================

    public void placeSelf()  {
        // place shape at given (or calculated) position
    }


    private void placeShrooms() {
        // for every shroom-controller
        // update shroom positions
    }

    // ========================================================================================
    // Logic
    // ========================================================================================

    @FXML
    public void onClicked() {
        // change appearance of the house on click
    }

    // ========================================================================================
    // Helping Methods
    // ========================================================================================

    // feel free to add setter/getter or other kinds of self written methods
}
```

Abbildung 2: HouseController

```
package de.uniks.pmws1920.controller.ingame;

public class ShroomController {

    @FXML
    public <YOUR_SHROOM_SHAPE> <YOUR_VARIABLE_NAME>;

    private ModelBuilder builder;
    private AnchorPane pane;
    private Shroom shroom;

    private double xPos;
    private double yPos;

    // ========================================================================
    // Init
    // ========================================================================

    public void setContent(ModelBuilder builder, AnchorPane pane, Shroom shroom, double xPos, double yPos) {
        // save parameter in fields
        // maybe additional location calculations
        this.setup();
    }

    private void setup() {
        this.bindValues();
        this.relocateSelf(<YOUR_X_POSITION>, <YOUR_Y_POSITION>);
    }

    // ========================================================================
    // Init View
    // ========================================================================

    private void bindValues() {
        // Bind shape to pane
        // Optional bind property change listener to shroom
    }

    // ========================================================================
    // Update View
    // ========================================================================

    // Use this method for the view update in the HouseController
    public void relocateSelf(double xPos, double yPos) {
        // optional calculations for shroom position
        this.placeSelf();
    }

    public void placeSelf() {
        // place shape at given (or calculated) position
    }


    // ========================================================================
    // Helping Methods
    // ========================================================================

    // feel free to add setter/getter or other kinds of self written methods
}
```

Abbildung 3: ShroomController

Aufgabe 2 - Clicklistener (10P)

Ziel dieser Aufgabe ist es, beim Klicken auf eines der Houses ein visuelles Feedack zu geben. Verbinde hierzu die OnMouseClicked-Methode der geometrischen Form des Houses mit der onClicked-Methode in Codesnippet 2.

Nach einem Klick auf eines der Houses soll dessen Darstellung so verändert werden, dass dem potenziellen Nutzer klar ist, welche Houses als ausgewählt oder aktiv gelten. Folgende Abbildung 4 gibt ein Beispiel des Highlightings vor, es dürfen aber auch eigene Darstellungsformen gewählt werden.

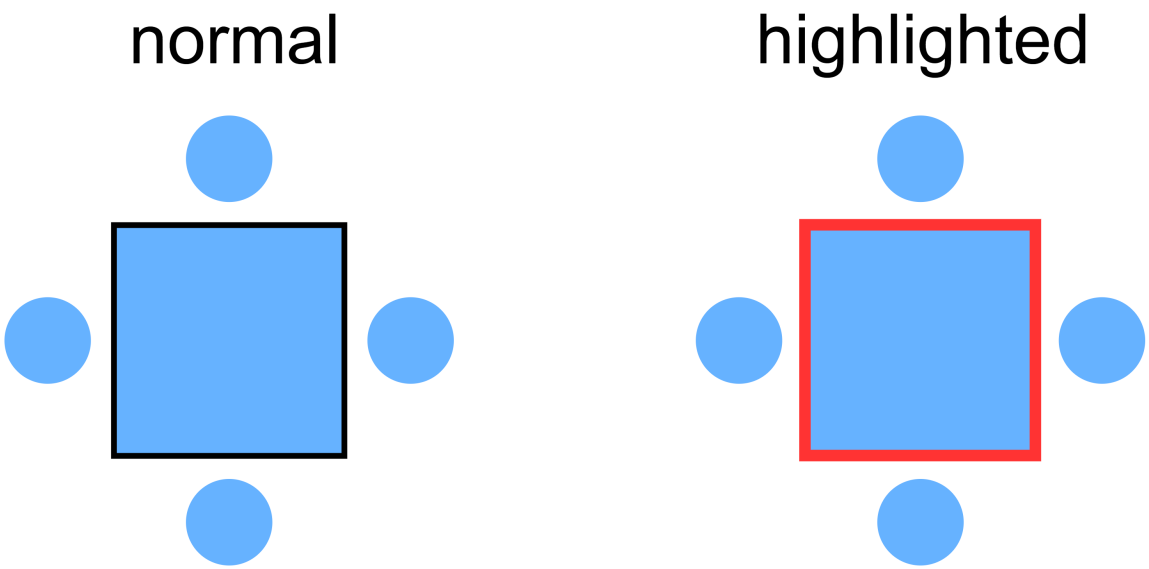

Abbildung 4: Beispiel einer inaktiven/aktiven House-Darstellung

Aufgabe 3 - Reinforce (50P)

Abschließend sollen die ersten Schritte in Richtung Spielelogik gemacht werden. Zu einem späteren Zeitpunkt dieser Veranstaltung werden wir diverse Methoden durch einen Timer ausführen lassen. Das Auftreten solcher automatischen Aktionen durch einen Timer nennen wir Ticks. Diese sollen bis zum Abschluss der Grundfunktionalität des Spiels zunächst einmal manuell ausgelöst werden.

Verbinde hierzu den Button in Abbildung 5 mit der nextTick()-Methode aus Codesnippet 1. Diese ruft die reinforce()-Methode aus Codesnippet 6 auf.

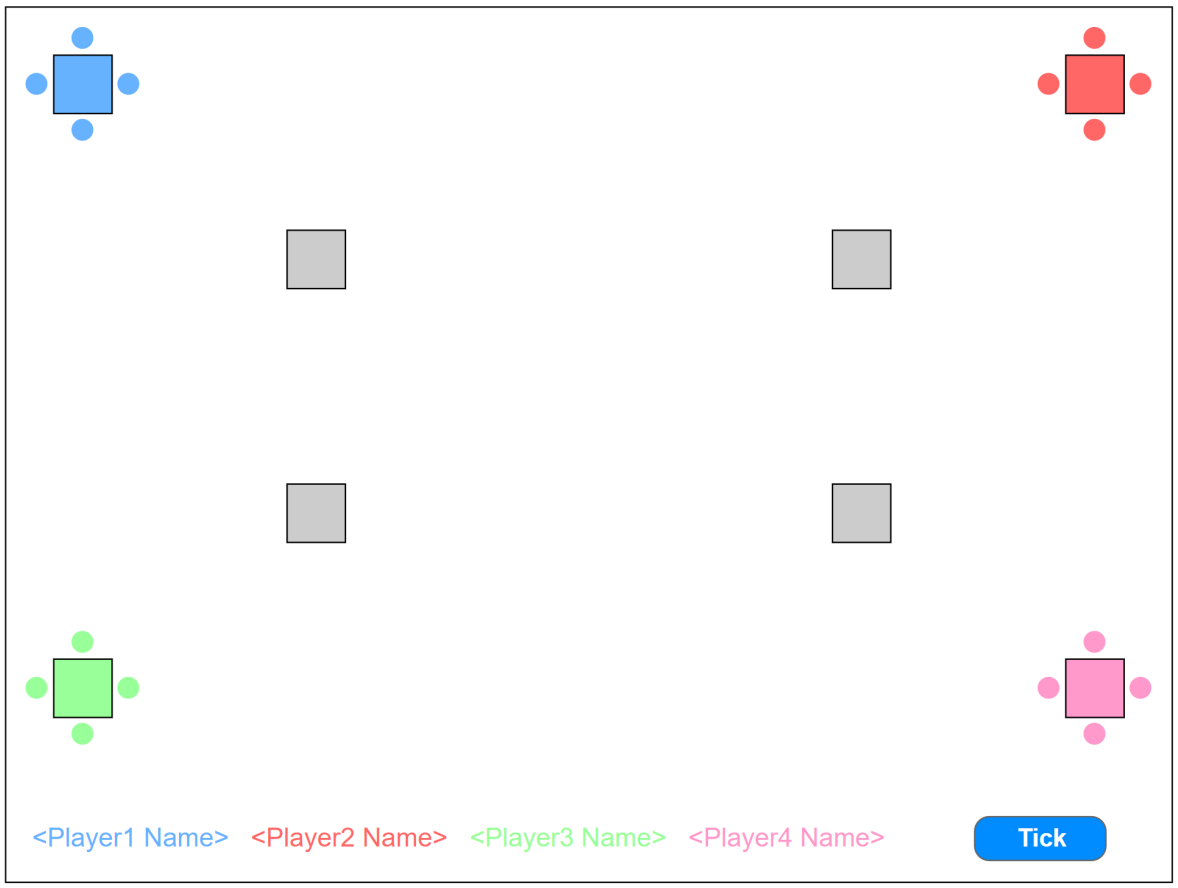

Abbildung 5: Ingame mit Tick-Button

```
package de.uniks.pmws1920.builder;

public class ModelBuilder {

    // ============================================================================
    // Game Rule Methods
    // ============================================================================

    public void reinforce() {
        // put code here
    }
}
```

Abbildung 6: ModelBuilder

Model

Innerhalb der reinforce()-Methode sollen alle Houses, die von einem Spieler eingenommen sind und deren eigenen Shroom-Anzahl an diesem House kleiner als die Capacity des Houses ist, Verstärkung erhalten. Die Anzahl der Shrooms in dieser Verstärkung kann selbst gewählt werden, sie muss jedoch mindestens 1 betragen und darf nicht einfach die Differenzen zwischen capacity und Shroomanzahl sein.

Controller

Eine Änderung der Shroomanzahl innerhalb eines Houses wird ein Property Change auslösen. Da wir für jedes House, im entsprechenden Controller (siehe Codesnippet 2), einen Listener für diese Property Changes angelegt haben, wird dieser bei der Erstellung neuer Shrooms im Datenmodell benachrichtigt.

In der Aktion des Listeners kann nun für die neu erstellten Shrooms je ein eigener Controller angelegt und in der Liste hinterlegt werden, wie dies bereits zur Initialisierung des HouseControllers in Codesnippet 2 geschehen ist.

View

Nach Abschluss der Instantiierung neuer Controller sollen die neuen und alten Shrooms (eventuell) neu positioniert werden. Die Positionierung hängt von der eigenen Darstellungsform ab. Dies bedeutet, dass je nach gewählter Darstellung eine Neupositionierung nach einem reinforce nicht benötigt wird. In Voraussicht auf die Dezimierung der Shrooms durch einen Kampf sollte eine Neupositionierung jedoch bereits jetzt umgesetzt werden.