Die Hausaufgaben müssen von den Studierenden einzeln bearbeitet und abgegeben werden. Für die Hausaufgabe sind die aktuellen Informationen vom Blog `https://seblog.cs.uni-kassel.de/ws2223/einfuhrung-in-die-informatik/` zu berücksichtigen.

Abgabefrist ist der 02.02.2023 - 23:59 Uhr

Vorbereitung

Zur Abgabe dieser Hausaufgabe muss zunächst ein neues Respository angelegt werden. Mit dem folgenden Link kannst du das Respository für Hausaufgabe 11 anlegen bzw. einsehen.

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

Achte bei jeder Aufgabe auf das verlangte Dateiformat. Andere Formate werden nicht akzeptiert und folglich mit **0** Punkten gewertet. Handschriftliche Abgaben werden nicht akzeptiert.

Es sollen nur die bis zur jeweiligen Übung vermittelten Konzepte verwendet werden.

Nicht oder zu spät abgegebene (Teil-)Aufgaben werden mit 0 Punkten bewertet.

Code-Formatierung

Formatiere deinen Code vor dem Abgeben wie in Vorlesung und Übung gezeigt. Verstöße gegen Formatierungs- und Namenskonventionen führen zu Punktabzug.

Lauffähigkeit

Nicht ausführbare Abgaben werden mit 0 Punkten bewertet.

3-Credit Abschnitt

Diese Hausaufgabe zählt zu dem in Übung 1 erläuterten „3-Credit Abschnitt" der Veranstaltung. Wenn du dir unsicher bist, welche Prüfung du in dieser Veranstaltung ablegen musst, informiere dich bitte bei deinem Studienservice.

Aufgabe 1 - Rock Paper Scissors Simulator (40P)

Einbinden von Processing

Lege ein neues Java-Projekt an. Um zu ermöglichen, dass die Processing-Bibliothek verwendet werden kann, muss zunächst die benötigte .jar-Datei eingebunden werden.

Wenn nicht bereits durch frühere Hausaufgaben geschehen, kann Processing hier heruntergeladen werden:

`https://processing.org/download`

Die Datei `core.jar` befindet sich im Ordner `processing-4.1.2/core/library` und muss in dein Java-Projekt eingebunden werden.

In der Übung wird gezeigt, wie Processing nun verwendet wird.

Simulator

Es soll ein Programm geschrieben werden, welches visuell einen Schere-Stein-Papier-Kampf darstellt. Dabei sollen im Programmfenster einige Scheren, Steine und Papiere in zufällige Richtungen fliegen und dabei von den Wänden abprallen bis sie auf einen Gegenstand treffen. Bei einer Kollision von zwei Gegenständen gelten die bekannten Regeln:

- Schere schlägt Papier
- Papier schlägt Stein
- Stein schlägt Schere

Wird ein Gegenstand bei einer solchen Kollision geschlagen, ändert er sich ebenfalls zu diesem Gegenstand. Bei Kollisionen von Gegenständen prallen diese außerdem auch immer voneinander ab.

Das Programmfenster soll 720 x 720 Pixel groß sein. Scheren sollen als Dreiecke, Papiere als Quadrate und Steine als Kreise dargestellt werden. Bei der Kollisionserkennung reicht es zu prüfen, ob die quadratische Hitbox um die jeweilige Form eine andere berührt bzw. überlappt.

Im Verlauf der Simulation kann es dazu kommen, dass alle Gegenstände vom gleichen Typ sind, z. B. wenn nur noch Dreiecke (bzw. Scheren) auf dem Spielfeld zu sehen sind. In diesem Fall soll als Text eingeblendet werden, welcher Gegenstand der Gewinner ist.

Programmstruktur

Im Standard-Package befinden sich folgende Klassen:

- `Main`: In dieser Klasse befindet die main-Funktion. Wie in der Übung gezeigt, soll sie lediglich die Aufrufe zum Starten des Processing-Programms beinhalten.
- `RPS`: Diese Klasse erbt von `PApplet` und beinhaltet die Methoden `settings()` und `draw()`.

Im Package `tools` befinden sich folgende Klassen bzw. Interfaces:

- das Interface `Beatable`
- die abstrakte Klasse `Tool`, die das `Beatable` implementiert
- die Klasse `Rock`, die von `Tool` erbt

- die Klasse `Paper`, die von `Tool` erbt
- die Klasse `Scissors`, die von `Tool` erbt

Es folgen genauere Erklärungen zu den einzelnen Klassen.

RPS

In dieser Klasse soll die Simulation gezeichnet und aktualisiert werden. Beispielsweise müssen zu Beginn die Scheren, Steine und Papiere erzeugt werden und im weiteren Verlauf die Methoden zum Aktualisieren der Gegenstände aufgerufen werden. Diese Klasse soll sich außerdem um das Prüfen und Anzeigen des Gewinners kümmern.

Zu Beginn der Simulation sollen alle Gegenstände so spawnen, dass sich keine berühren. Außerdem soll es möglich sein, die **Anzahl** und **Größe** der Gegenstände jeweils durch die Anpassung einer Konstante einfach ändern zu können. Bei der Abgabe soll die Simulation mit insgesamt 60 Gegenständen starten (20 pro Typ) und die Gegenstände eine Größe von 30 Pixeln haben. (D. h. Quadrate haben eine Seitenlänge und Kreise einen Durchmesser von 30 Pixeln. Die Dreiecke können z. B. so umgesetzt sein, dass eine ihrer Seiten und ihre Höhe 30 Pixel lang sind. Entscheidend ist, dass alle Hitboxen gleich groß sind.)

Beatable

Das Interface `Beatable` beinhaltet folgende Methoden-Schnittstellen:

- `boolean canBeatMe(Tool tool)`
- `void display()`

Tool

Die abstrakte Klasse `Tool` soll per Schlüsselwort `implements` das Interface `Beatable` implementieren.

Folgende Attribute müssen in der Klasse vorhanden sein:

- `public PApplet sketch`
- `public float x`
- `public float y`
- `public float xSpeed`
- `public float ySpeed`
- `public float size`

Folgende Methoden müssen in der Klasse vorhanden sein:

- Konstruktor `public Tool(PApplet sketch, float x, float y, float size)`
- Methode `public void move()`, die die Koordinaten modifiziert
- Methode `public void wallBounce()`, die prüft, ob eine Kollision mit dem Rand des Programmfensters stattfindet und die Geschwindigkeit entsprechend modifiziert

- Methode `public boolean collidesWith(Tool tool)`, die zurückgibt, ob eine Kollision mit dem übergebenen tool stattgefunden hat

Rock bzw. Paper bzw. Scissors

Diese Klassen müssen jeweils per Schlüsselwort `extends` von der Klasse `Tool` erben. Aufgrund der entstandenen Hierarchie müssen hier nun die Methoden `canBeatMe` und `display` mit Leben befüllt werden. Die Methode `canBeatMe` soll jeweils zurückgeben, ob das übergebene Tool laut Schere-Stein-Papier-Regeln stärker ist. Die Methode `display` soll dazu dienen, die Form zu zeichnen. Dabei kann per `this.sketch` auf die bekannten Processing-Methoden zugegriffen werden.

Hinweise

Farben dürfen selbst gewählt werden. Achte darauf, dass alle Gegenstände erkennbar sind.

In RPS muss beim Erzeugen von Rock- / Paper- / Scissors-Objekten im Konstruktor als `sketch` das aufrufende RPS übergeben werden, also `this`.

Es **dürfen** zusätzliche Hilfsmethoden und -variablen verwendet werden. Es dürfen Enums verwendet werden (diese müssen mit abgegeben werden). Weitere Klassen sind **nicht** erlaubt. Verwende sprechende Namen.

Achte darauf, dass sich die Klassen im richtigen Package befinden.

Abgabe

Gib deine Dateien

- `Main.java`
- `RPS.java`
- `Beatable.java`
- `Tool.java`
- `Rock.java`
- `Paper.java`
- `Scissors.java`

mit der Lösung im GitHub-Repository zur aktuellen Hausaufgabe ab.