Das Projekt – Squash

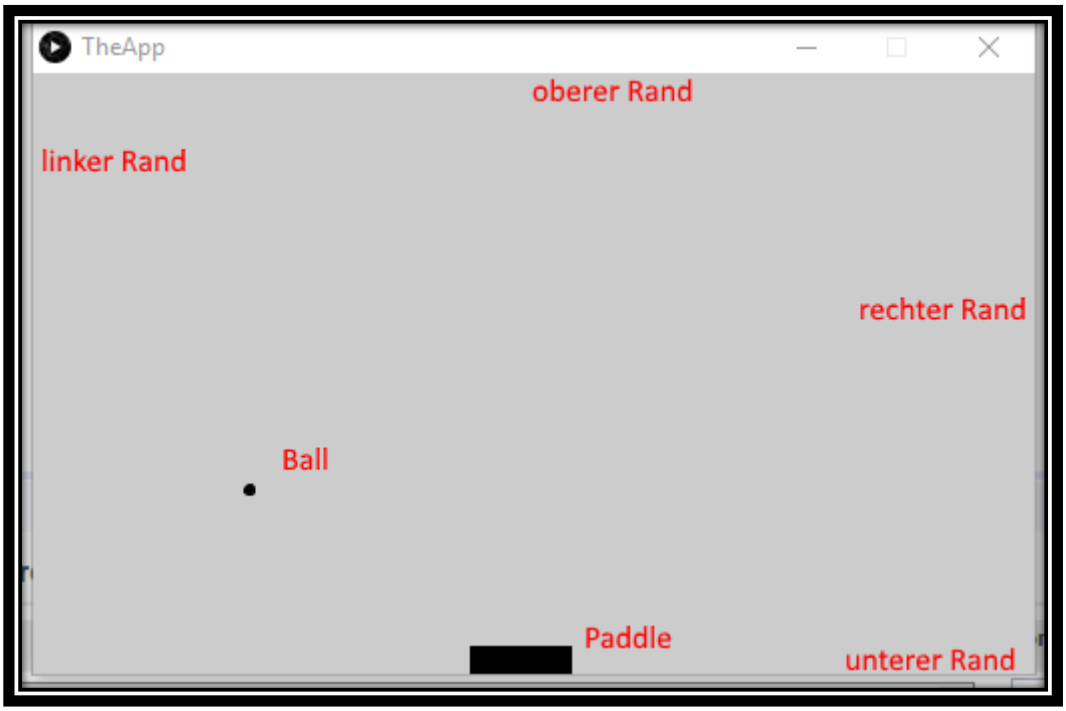
Ich habe als Projekt Squash umgesetzt. Dazu gibt es einen Ball, der sich in einem festgelegten Fenster bewegt und an der linken, oberen und rechten Seite im 90°-Winkel abprallt.

Verlässt er den unteren Bildrand, dann wird „`Ball went out of display`“ ausgegeben und es kommt ein neuer Ball.

Das Verlassen des Spielfelds am unteren Rand kann verhindert werden, indem man ein Paddle hin und her bewegt, an dem der Ball auch abprallt. Die Steuerung erfolgt mittels Maus (geht nur, wenn die Maus im Fenster ist) oder mit der Tastatur (Pfeiltasten links und rechts). Damit man mit der Tastatur steuern kann muss man einmalig in das Fenster klicken.

Spielansicht

Hier ist das Aussehen des fertigen Spiels dargestellt.



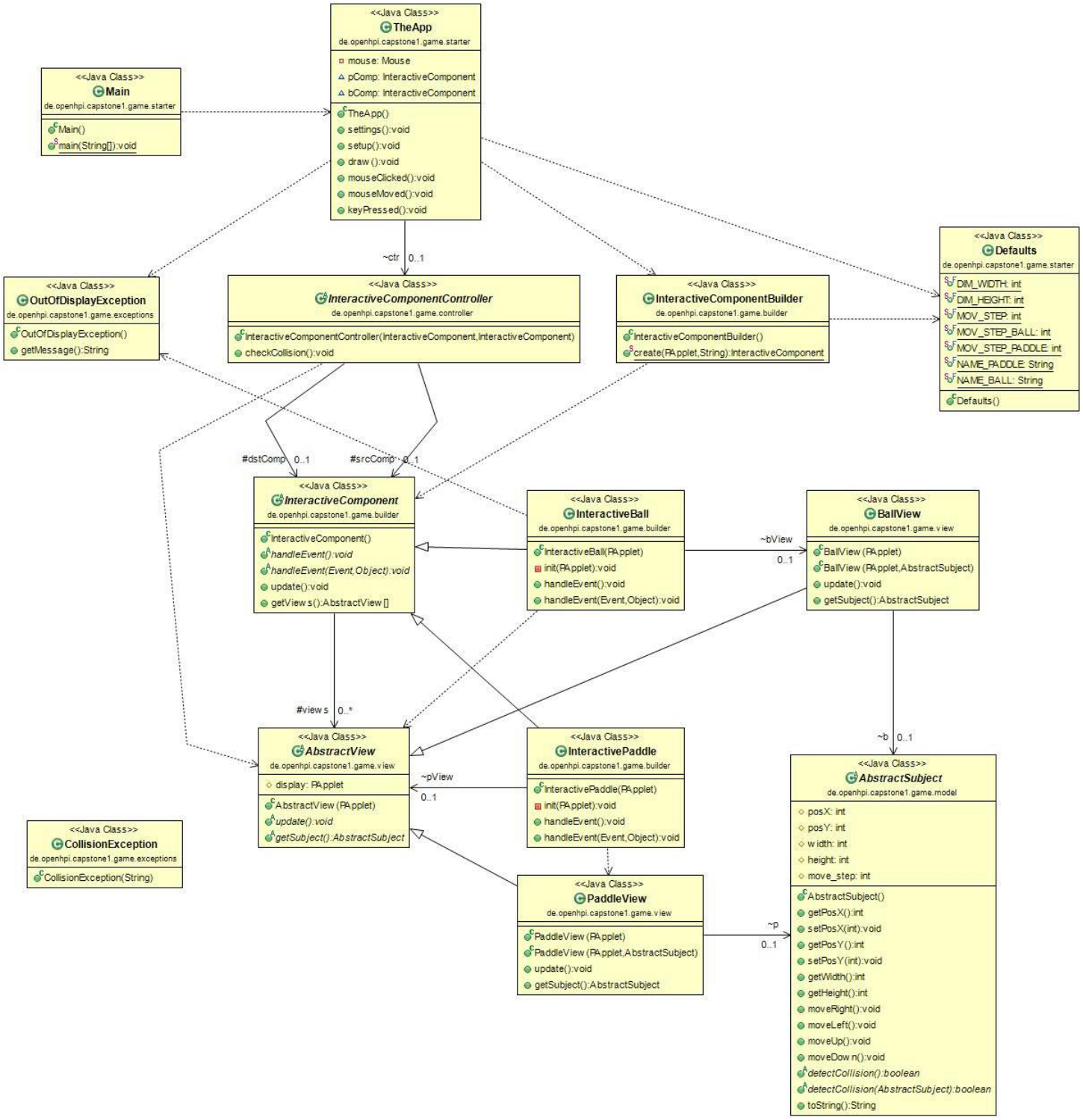
Konkrete Anforderungen an das Spiel

- Elements: a Ball and a Paddle
- Ball: once inserted to the playground, it moves (on its own/animation) to a certain set of rules.
    - Rule1: Ball bounces (changes its direction) when it hits the left, right or upper bound of the playground.
    - Rule2: Ball bounces when it hits the paddle.
    - Rule3: Ball leaves the playground when it "hits" lower bound of the playground.
    - Rule4: A new Ball is inserted when the previous Ball has left the playground (your choice if that happens automatically or on some sort of user interaction).
- Paddle: is moved by user interaction. Keyboard or Mouse

Diese Anforderungen wurden umgesetzt.

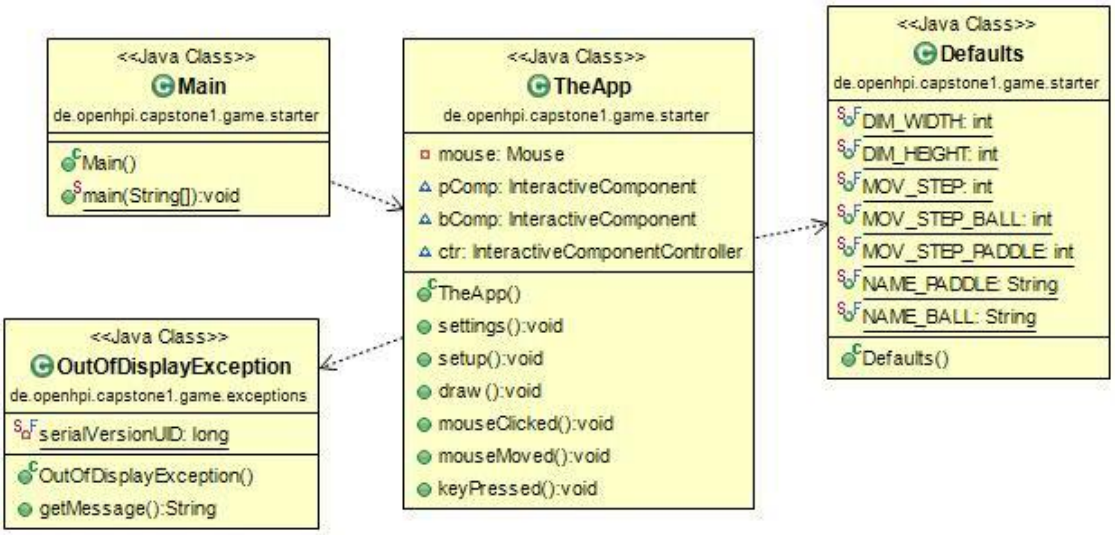
Modell

Nachfolgende Abbildung enthält den Gesamtplan der Anwendung. Dieser wird später in die einzelnen Teile zerlegt.



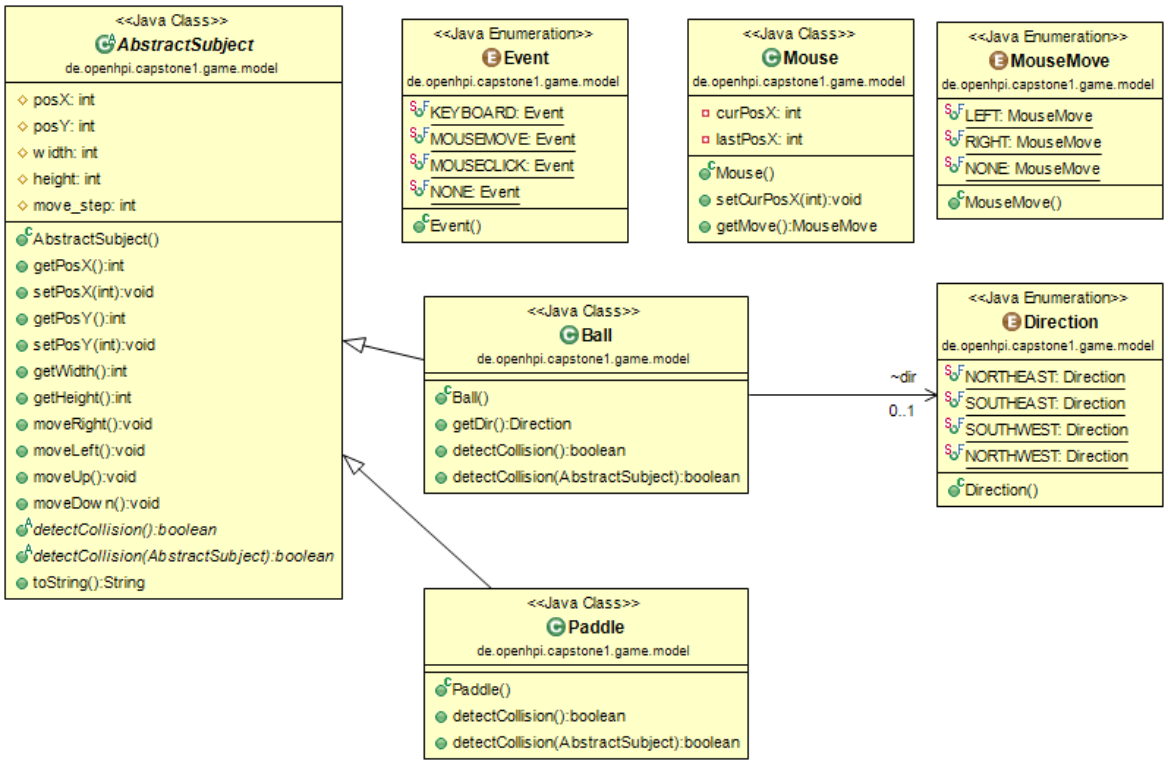
Die einzelnen Komponenten

Starter - de.openhpi.capstone1.game.starter;



Die Klasse Main dient als Startpunkt und ruft die Klasse PApplet mit der main-Methode aus der Processing-Bibliothek mit dem Argument des Namens der Klasse „TheApp“. Verschiedene Konstanten sind in der Klasse Defaults zusammengefasst und können von allen anderen Klassen im Projekt genutzt werden. Hier stehen z.B. die Breite (DIM_WIDTH) und Höhe (DIM_HEIGHT) des Fensters, sowie die Schrittweite des Paddle. Die OutOfDisplayException wird von der Klasse InteractiveBall geworfen und in der draw()-Methode von TheApp abgefangen, um einen neuen Ball zu initialisieren.

Datenmodell - de.openhpi.capstone1.game.model



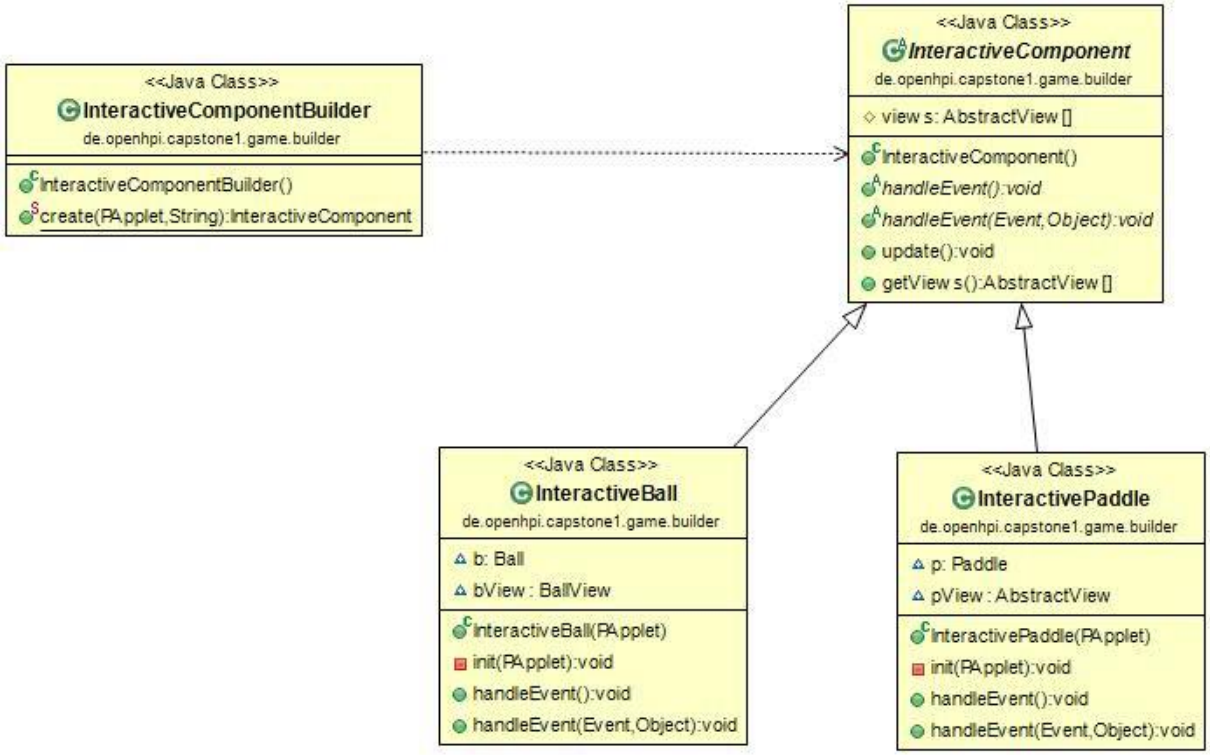
Das Datenmodell besteht aus der Maus, den Events aus Processing (KEYBOARD, MOUSEMOVE und MOUSECLICK) sowie dem Ball und dem Paddle, die sich aus dem AbstractSubject ableiten.

Die Maus verfügt über eine aktuelle und eine letzte Position, die benutzt werden, um die Bewegung herauszufinden (getMove). Die Bewegung ist eine Enumeration, die Links, Rechts oder Keine sein kann. Darauf basierend wird das Paddle nach links oder rechts bewegt.

Das AbstractSubject besitzt eine Position, eine Breite und eine Höhe. Die Position kann gelesen und gesetzt werden. Breite und Höhe können nur gelesen werden. Das Prüfen auf eine Kollision mit einer Wand (ohne Parameter) oder einem anderen AbstractSubject kann über detectCollision geprüft werden.

Der Ball besitzt zusätzlich eine Richtung (Direction), in die er sich bewegt. Diese wird genutzt, um beim Treffen auf eine Wand die 90°-gedrehte Richtung zu ermitteln.

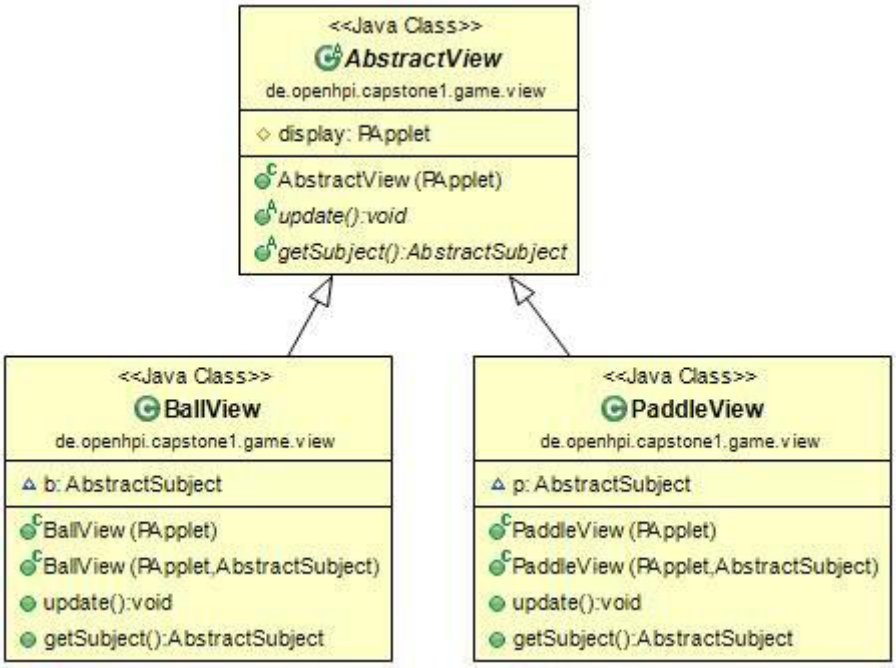
Builder und Builder Pattern- de.openhpi.capstone1.game.builder



Der InteractiveComponentBuilder erzeugt InteractiveComponents, die aktuell entweder ein interaktiver Ball (InteractiveBall) oder das Paddle (InteractivePaddle) sein können. Jede InteractiveComponent kann dabei mehrere AbstractViews enthalten (BallView oder PaddleView), die dann innerhalb des Fensters angezeigt werden. Zum erzeugen neuer InteractiveComponents wird die statische Methode create(applet, type) genutzt.

Views - de.openhpi.capstone1.game.view

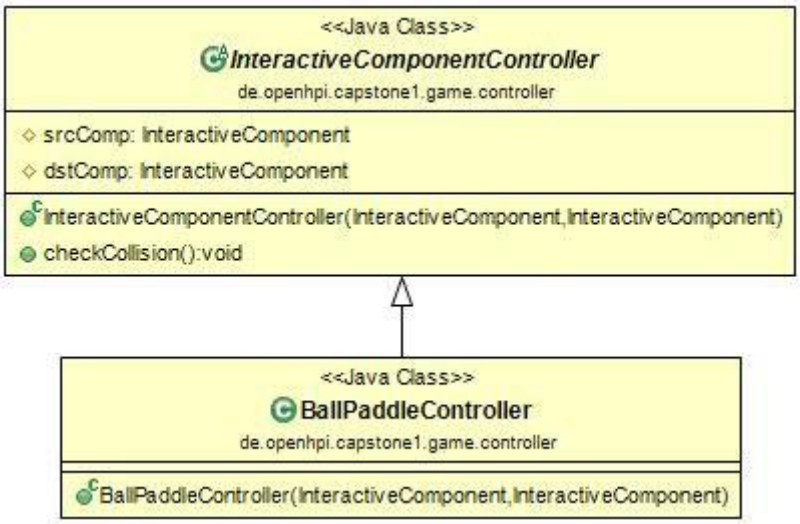
Für die Views wird eine AbstractView als Grundlage genutzt.



Von dieser werden jeweils die Views BallView und PaddleView abgeleitet. Jede View enthält den Verweis auf das diplay und ein entsprechendes Element vom Typ AbstractSubject, aus dem Modell. Bei der BallView ist dies der Ball, bei der PaddleView das Paddle.

Um zwischen Elementen Interaktionen ausführen zu können, gibt jede AbstractView mittels getSubject() ihr internes Subject nach außen. Dies kann dann von einem Controller (InteractiveComponentController) zur Interaktion genutzt werden.

Controller - de.openhpi.capstone1.game.controller

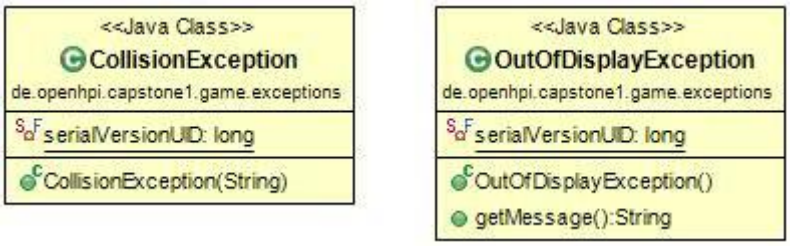


Der InteractiveComponentController ermöglicht die Kommunikation zwischen zwei InteractiveComponents. Dabei ist ein Element die Quelle der Interaktion und eins das Ziel der Interaktion.

Bei Squash ist ein BallPaddleController implementiert, der in der TheApp-Klasse zusammengebaut wird. Er dient dazu, für jeden Ball aus der InteractiveBallView die Kollision mit jedem Paddle aus der InteractivePaddleView zu ermitteln und abzuprallen.

Exceptions

Exceptions dienen in meinem Projekt dem Rückmelden von Ereignissen, z.B. dem Verlassen des Spielfelds eines Balls.



Die **OutOfDisplayException** wird genutzt, um der App im draw() zu signalisieren, dass der Ball das Spielfeld verlassen hat (kommt aus Methode handleEvent von InteractiveComponent. In diesem Fall wird ein neuer Ball erzeugt und der Controller mit diesem überschrieben.

Die **CollsionException** ist nur pro forma da, um eine falsche Richtungsänderung bei Kollision zu entdecken.