# Das Projekt – Squash

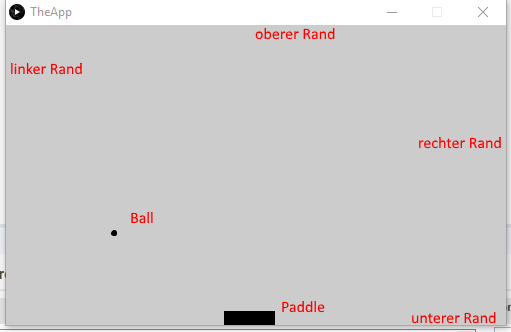
Ich habe als Projekt erstmal nur Squash umgesetzt. Dazu gibt es einen Ball, der sich in einem festgelegten Fenster bewegt und an der linken, oberen und rechten Seite im 90°-Winkel abprallt.

Verlässt er den unteren Bildrand, dann folgt, dann 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.

## 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

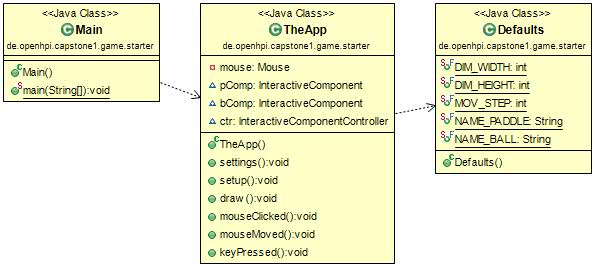
Diese Anforderungen wurden umgesetzt.

## Modell

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

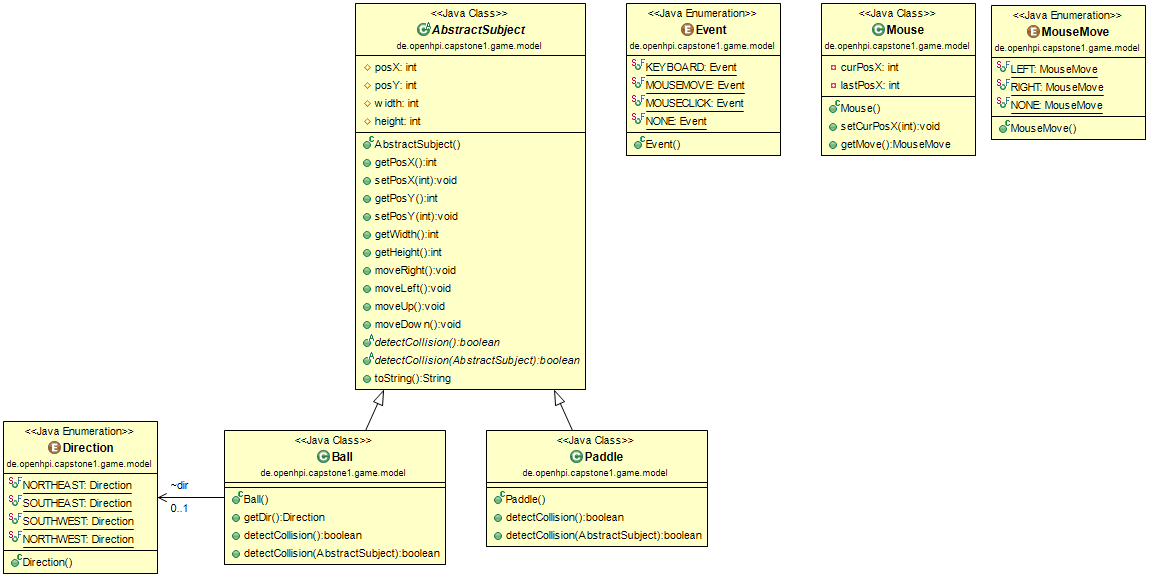
## Die einzelnen Komponenten

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



….. TODO …..

Datenmodell - de.openhpi.capstone1.game.model



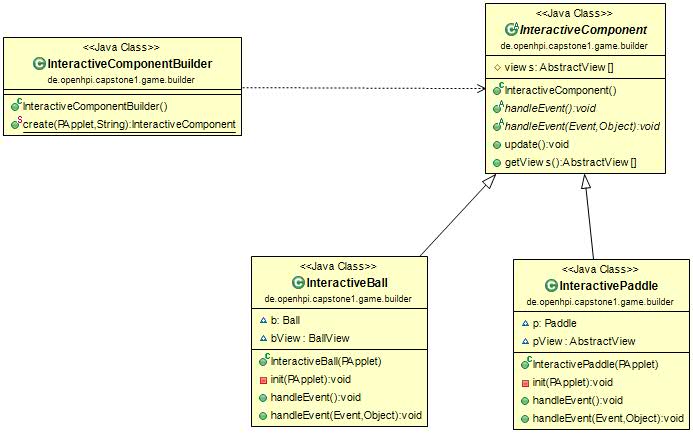
Das Datenmodell besteht aus der Maus, den Events aus Processing (KEYBOARD, MOUSEMOVE und MOUSECLICK) sowie dem Ball und dem Paddle, die sich aus dem AbstractSubject ableiten.

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 AbstrctSubject kann über detecCollision geprüft werden.

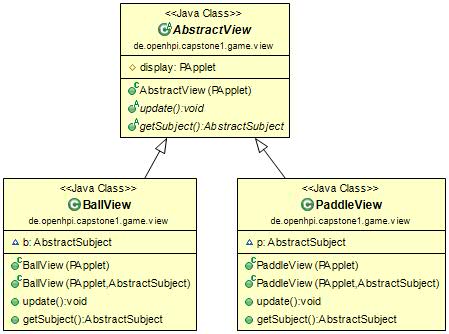
Der Ball besitzt zusätzlich eine Richtung (Direction), in die er sich bewegt..

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



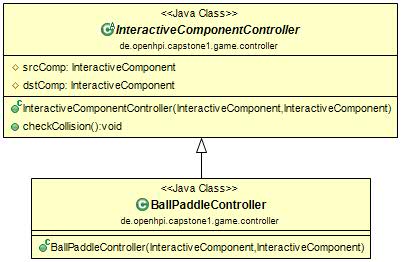
Der InteractiveComponentBuilder erzeugt InterActiveComponents, die aktuell entweder ein interaktiver Ball (InteractiveBall) oder das Paddle (InteractivePaddle) sein können. Jede InteractiveComponent kann dabei mehrere AbstractViews enthalten (BallView oder PaddleView). Zum erzeugen neuer InterActiveComponents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

Views - de.openhpi.capstone1.game.view



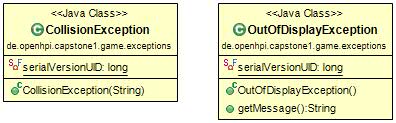
….

Controller - de.openhpi.capstone1.game.controller



Wozu gut, wie sieht er aus

Exceptions



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 Kollsion zu entdecken.

## Erweiterung

Hier soll kurz angedeutet werden, wie man das aktuelle Projekt auf weitere Spiele erweitern kann.