Prof. Dr.-Ing. R. Roosmann, H. Plitzner

Aufgabenblatt 04

Ziel dieses Aufgabenblatts ist es, Sie initial mit der ereignisorientierten Programmierung in Java bekannt zu machen.

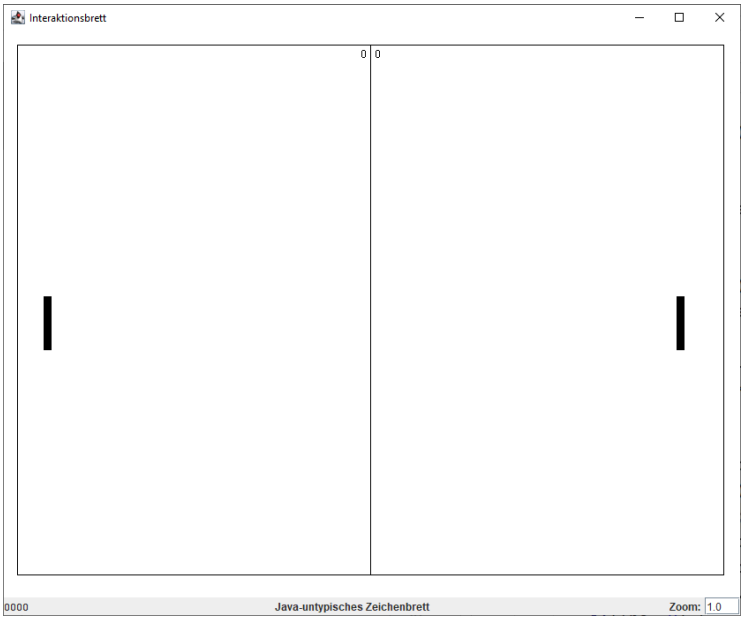
Abgabe: Gr. 3: 24.11.2021, Gr. 1+2: 29.11.2021; Max. Punktzahl: 20; Min. Punktzahl: 12 Punkte

Aufgabe 4.1: Pong (6 Punkte)

Der 1972 von Alan Alcorn für Atari entwickelte Videospiel-Klassiker Pong wurde zum ersten kommerziell erfolgreichen Videospiel überhaupt und gilt als Urvater aller Videospiele. Wurde das Spiel in den ersten Jahren nur in Spielhallen auf speziellen Arcade-Maschinen angeboten, eroberte es ab 1975 die heimischen Wohnzimmer. Informationen über das Spiel finden sie u.a. unter https://de.wikipedia.org/wiki/Pong.

Das Spielprinzip ist sehr einfach gehalten und ähnelt dem Tennisspiel. Es gibt zwei Spieler, bzw. zu sehen sind lediglich die Schläger, sowie ein Ball. Sobald das Spiel gestartet wird, bewegt sich der Ball auf dem Spielfeld zwischen den Schlägern hin und her. Wie beim historischen Tennisspiel des Mittelalters prallt der Ball beim Auftreffen an der oberen und der unteren Begrenzung des Spielfeldes ab und wird zurück ins Spielfeld gebracht (der Einfachheit halber gilt: Einfallswinkel gleich Ausfallswinkel). Trifft der Ball auf den Schläger eines Spielers, so wird der Ball zurück in Richtung des Gegners gespielt. Bringt ein Spieler den Schläger nicht rechtzeitig hinter den Ball, so erhält der Gegner einen Punkt. Gespielt wird, bis der erste Spieler 15 Punkte erreicht hat. Das Aufschlagrecht wechselt nach fünf Aufschlägen. Bei einem Aufschlag wird der Ball zufällig im Feld des aufschlagenden Spielers positioniert und bewegt sich dann in Richtung des gegnerischen Feldes, wobei die Richtung zufällig variiert werden sollte.

Nach der Ausführung des Programms Pong wird das Spielfeld als ungefülltes Rechteck mit einer Mittellinie (dem Netz), den beiden Schlägern als gefüllte Rechtecke und der Spielstand dargestellt.



Nach dem Start des Programms wird das Spiel gestartet. Durch Drücken der Taste Taste „e“ (für Ende; bspw. mit `System.exit(1)`) kann das Spiel beendet werden. Der erste Aufschlag geht immer vom linken Spieler aus. Danach erhält der Spieler den Aufschlag, der den Punkt gewonnen hat. Die Spieler bewegen die Schläger über die Tastatur. Der linke Spieler kann den Schläger mit der Taste „a“ aufwärts und der Taste „y“ abwärts bewegen. Der rechte Spieler verwendet die Tasten „Oben“ (Pfeil) und „Unten“ (Pfeil), um den Schläger vertikal zu bewegen.

Optional: über zwei zu definierende Tasten kann die Geschwindigkeit des Balls erhöht, bzw. reduziert werden.

Zur Programmierung sollen die Klassen Interaktionsbrett.java und EinUndAusgabe.java verwendet werden. Beide Klassen sollen dem Package `de.hsos.prog3.ab4.pong.util` hinzugefügt werden. Öffnen Sie Interaktionsbrett.java und ändern die Konstante `DIM` so, dass eine für Sie sinnvolle Spielfeldgröße dargestellt wird, bspw. in 800*700.

Sie können das Programm vollständig eigenständig realisieren. Das folgende Vorgehen soll Sie lediglich unterstützen, ist optional und durch Sie nicht zwingend umzusetzen!!

Mögliches Vorgehen zur Realisierung des Spiels Pong:

Zu erstellen sind die Klassen `Spielfeld`, `Spieler`, `Ball`, `KollisionsDetektion` und `PongSpiel`. Die Klasse `Rechteck` (Arbeitsblatt 2) können Sie zur Darstellung des Spielfeldes, der Spieler und des Balls wiederverwenden. Gehen Sie bei der Programmierung des Spiels iterativ vor:

1. Anpassen und erweitern der Klasse Rechteck um verschiedene Helfermethoden, wie:
   - `int oben()`: y-Wert (=> y-Wert der oberen Ecken)
   - `int unten()`: y-Wert + hoehe (=> y-Wert der unteren Ecken)
   - `int links()`: x-Wert (=> x-Wert der linken Ecken)
   - `int rechts()`: x-Wert + breite (=> x-Wert der rechten Ecken)
   - `int breite()`: Breite des Rechtecks
   - `int hoehe()`: Hoehe des Rechtecks
   - `int mitteInY()`: Y-Wert der Mittellinie durch das Rechteck
   - `int mitteInX()`: X-Wert der Mittellinie durch das Rechteck
   - `void verschiebe(int dx, int dy)`: Verschieben des Rechtecks um dx und dy
   - `void verschiebeNach(int x, int y)`: Verschieben des Rechtecks nach X/Y
   - `boolean ueberschneidet(Rechteck o)`: liefert `true`, bei Überschneidung, ansonsten `false` (Ansatz siehe bspw. https://www.baeldung.com/java-check-if-two-rectangles-overlap; Achtung: Koordinatenachsen entsprechen nicht dem Bildkoordinatensystem des Interaktionsbretts <= anpassen des Source-Codes)

2. Programmieren Sie die Klasse `Spielfeld`. Diese Klasse kennt die Spielfeldgröße (dieselben Werte, die beim Interaktionsbrett für die Konstante DIM angegeben worden sind), erzeugt über den Default-Konstruktor ein Objekt vom Typ `Rechteck` und weist dieses der Instanzvariablen `spielflaeche` zu.
   Die Klasse bietet eine öffentliche `darstellen`-Methode der ein Objekt vom Typ `Interaktionsbrett` übergeben wird. Dargestellt wird das Rechteck als

begrenzende Linie des Spielfeldes, sowie die Mittellinie (das Netz). Rund um das Spielfeld sollen Abstände (Margins) bei der Darstellung berücksichtigt werden.
Erzeugen Sie eine Klasse `App` mit einer `main`-Methode, erzeugen dort ein Objekt vom Typ `Spielfeld` und testen die Funktionalität aus.

3. Programmieren Sie die Klasse `Spieler`. Diese Klasse hat eine Instanzvariable `spielfeld` vom Typ `Spielfeld`, eine Instanzvariable `schlaeger` vom Typ `Rechteck` und eine Instanzvariable `punkte` vom Typ `int`. Das Spielfeld-Objekt wird ebenso an den Konstruktor übergeben, wie eine x- und eine y-Position für die linke obere Ecke des Schlägers. Die Breite und Höhe des Schlägers lässt sich aus der Breite und Höhe des Spielfelds berechnen. Bspw. ergibt sich die Breite aus Spielfeldbreite/100 und die Höhe aus Spielfeldhoehe/10 <= probieren Sie sinnvolle Werte aus. Das Rechteck des Schlägers soll gefüllt dargestellt werden.
   Bieten Sie nur dann Getter- und Setter-Methoden an, wenn diese benötigt werden. Die Spieler sollen sich anhand von Tastatur-Ereignissen vertikal über das Spielfeld bewegen. Dabei fungiert diese Klasse nicht selber als *Listener*, stellt jedoch die öffentlichen Methoden `void aufwaerts()` und `void abwaerts()` bereit, die vom *Listener* aufgerufen werden können. Diese Methoden verändern die Position des Schlägers nach oben oder unten, unter Berücksichtigung der Spielfeldgrenzen. Ein Überschreiten der Spielfeldgrenzen ist zu vermeiden.
   Über die öffentlichen Methoden `erhoehePunkte()` und `setzePunkteZurueck()` werden die Punkte des Spielers im `PongSpiel` verändert.
   Erweitern Sie die main-Methode der Klasse `App`, so dass neben dem Spielfeld zwei Schläger dargestellt werden.

4. Programmieren Sie die Klasse `PongSpiel` zur Steuerung des Spiels (Controller). Diese Klasse soll später erweitert werden. Jetzt geht es lediglich darum, das Spielfeld und die Schläger darzustellen, sowie auf Tasten-Ereignisse für den linken und rechten Schläger zu reagieren.
   Das `PongSpiel` besitzt die Instanzvariablen `spielfeld` vom Typ `Spielfeld`, `spielerLinks` und `spielerRechts` jeweils vom Typ `Spieler`, sowie `ib` vom Typ `Interaktionsbrett`.
   Objekte der Klasse `PongSpiel` sollen als *Listener* vom Interaktionsbrett über Tatstatur-Ereignisse benachrichtigt werden.
   Benötigt wird ein Default-Konstruktor über den ein Objekt der Klasse `Interaktionsbrett` erzeugt und der Instanzvariablen `ib` zugewiesen wird. Anschließend wird im Konstruktor die `willTastenInfo`- Methode des Interaktionsbretts aufgerufen und `this` übergeben. Damit meldet sich das `PongSpiel` als Listener für Tastatur-Ereignisse beim Interaktionsbrett an. Der Konstruktor ruft dann die private Hilfsmethode `void startAufstellung()` auf, in der die Instanzvariablen `spielfeld`, `spielerLinks` und `spielerRechts` initialisiert werden.
   In der öffentlichen Methode `void spielen()` werden in einer unendlichen `while`-Schleife (`while(true) {…}`) folgende Bearbeitungsschritte durchgeführt:
   a) abwischen des Interaktionsbrettes
   b) zeichnen des Spielfeldes
   c) zeichnen des linken und rechten Spielers
   d) darstellen des aktuellen Spielstands

   Da sich die Spieler auf dem Spielfeld bewegen, soll die Darstellung mit 60 Frames per Second (FPS) erfolgen. Warum 60 FPS? Damit eine „ruckelfreie“ Darstellung des Spiels sichergestellt ist.

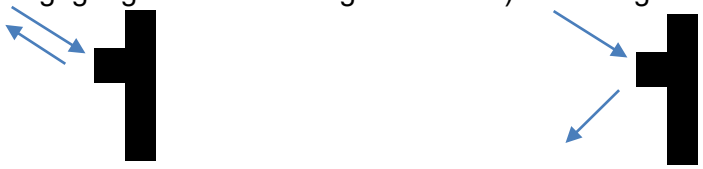
Zur Information: 60 FPS bedeutet, dass ca. alle 17 Millisekunden einmal der aktuelle Zustand des Spiels gezeichnet werden soll. Erzeugen Sie hierfür die ganzzahlige Konstante FPMS und weisen dieser den Wert 17 zu (ein Frame pro 17 Millisekunden ergibt 60 Frames pro Sekunde). Wie lange ein Durchlauf der Schleife tatsächlich dauert, lässt sich mithilfe der statischen Methode `System.currentTimeMillis()` berechnen. Ist der tatsächliche Schleifendurchlauf kürzer als der in FPMS definierte Wert, so kann mit `Thread.sleep(FPMS-differenz)` sichergestellt werden, dass nicht zu häufig gezeichnet wird (was zu Problemen führen kann).
Mit der ereignisorientierten Programmierung ist sichergestellt, dass der Zustand des Spiels im Hintergrund verändert wird. Das Ergebnis wird visualisiert, sobald es vorliegt und ein neuer Frame dargestellt wird.
Als Callback-Methode, die vom Interaktionsbrett beim Auftreten von Tatstatur-Ereignissen aufgerufen wird, ist die Methode `public void tasteGedrueckt(String s)` zu implementieren, so dass bei Übergabe von „a“ der Schläger des linken Spielers aufwärts, bei „y“ der Schläger des rechten Spielers abwärts, bei „Oben“ der Schläger des rechten Spielers aufwärts und bei „Unten“ der Schläger des linken Spielers abwärts bewegt werden kann. Rufen Sie dazu die entsprechenden Methoden der Instanzvariablen `spielerLinks` und `spielerRechts` auf. Des Weiteren soll mit „s“ das Spiel gestartet und mit „e“ beendet werden.

Passen Sie Ihre `main`-Methode in der Klasse `App` an, so dass die Klasse `PongSpiel` getestet wird.

5. Programmieren Sie die Klasse `Ball` mit den Instanzvariablen `form` vom Typ `Rechteck`, sowie `bewegungInXProFrame` und `bewegungInYProFrame`, jeweils vom Typ `int`. Die Instanzvariablen sollen über einen Konstruktor initialisiert werden. Das Rechteck soll gefüllt dargestellt werden.
   Da das Spiel in einem Raster-Koordinatensystem abläuft, bietet es sich an, die Entfernung, die der Ball zwischen zwei Frames zurücklegt über ganzzahlige Pixel-Werte für die X- und Y-Achsen des Bildkoordinatensystems zu definieren. In diesem Beispiel soll der Ball in ca. 4 Sekunden einmal über das Spielfeld mit einer Breite von 850 Pixel bewegt werden. Bei 60 FPS bedeutet dies, dass alle 17 Millisekunden ein Frame dargestellt wird und der Ball somit zwischen zwei Frames etwa 4 Pixel zurücklegt. Entsprechend können Sie den Instanzvariablen `bewegungInXProFrame` den Wert `4` und `bewegungInYProFrame` den Wert `1` zuweisen. Soll das Spiel einfacher gemacht werden, indem die Geschwindigkeit des Balls reduziert wird, können niedrige Werte gesetzt werden. Soll das Spiel schneller gemacht werden, können die Werte hochgesetzt werden. Dies kann relativ einfach über Tastatur-Ereignisse manipuliert werden, was hier jedoch nicht zwingend gefordert ist.
   In der öffentlichen Methode `void bewegen(int anzahlFrames)` wird die neue Position des Balls berechnet. Der Übergabeparameter gibt an, über wie viele Frames sich der Ball bewegen soll. Im Idealfall ist dieser Wert 1. Da später noch eine Kollisionsermittlung durchgeführt werden muss, kann die Berechnung ggf. länger dauern. Somit kann der Ball nicht alle 17 Millisekunden gezeichnet werden (entspricht dem Zeichnen eines Frames bei 60 FPS). Damit der Ball sich auch dann für den Betrachter kontinuierlich über den Bildschirm bewegt, kann der Übergabeparameter entsprechend angepasst werden.
   Trifft der Ball bei der Kollisionsermittlung auf die obere, bzw. untere Grenze des Spielfeldes oder auf den Schläger eines Spielers, so soll sich die Bewegungsrichtung ändern. Trifft der Ball auf die obere / untere Spielfeld-Grenze, so ist lediglich das Vorzeichen des Wertes `bewegungInYProFrame` zu ändern. Trifft der Ball auf den

Schläger des linken oder rechten Spielers, so ändert sich das Vorzeichen des Wertes `bewegungInXProFrame`. Diese Änderungen erfolgen in den Methoden `umkehrenDerBewegungInX` und `umkehrenDerBewegungInY`.
Diese Klasse bietet eine öffentliche Methode `void darstellen( Interaktionsbrett ib)`, über die der Ball als gefülltes Rechteck auf dem Interaktionsbrett dargestellt werden kann.
Getter- und Setter-Methoden für die Instanzvariblen sind nur dann anzubieten, wenn diese benötigt werden.

6. Erweiterung der Klasse `PongSpiel` um eine Instanzvariable `ball` vom Typ `Ball`. Anschließend ist die private Hilfsmethode `initialeAufstellung` zu erweitern, so dass in dieser Methode die Instanzvariable `ball` mit einem `Ball`-Objekt initialisiert wird. Der Einfachheit halber wird das initiale `Ball`-Objekt immer in der Nähe des linken Spielers positioniert. Als nächstes wird die Methode `spielen` erweitert, so dass in der `while`-Schleife jetzt zusätzlich der Ball auf dem Interaktionsbrett dargestellt wird. Zudem ist die neue Position des Balls über die Methode `bewegen` der Klasse `Ball` zuzuweisen. Diese Methode erwartet als Übergabeparameter die Anzahl an Frames, über die sich der Ball bewegen soll. Diese lassen sich, wie schon beschrieben, über `System.currentTimeInMillis()` berechnen (vergangene Zeit für die Darstellung des Frames geteilt durch die Konstante `FPMS` <= siehe hierzu Punkt 4.).
   Passen Sie die `main`-Methode der Klasse `App` an, so dass ein Objekt vom Typ `PongSpiel` erzeugt wird und dort die Methode `spielen` aufgerufen wird. Die Schläger der Spieler sollten bewegt werden können und der Ball sollte sich von links nach rechts bewegen.
7. Der Ball prallt in der aktuellen Lösung noch nicht an den Spielfeldgrenzen ab (Einfalls- gleich Ausfallswinkel). Auch haben die Schläger der Spieler keinerlei Auswirkungen. Aus diesem Grund soll die Klasse `KollisionsDetektor` programmiert werden. Die Klasse definiert die Instanzvariablen `spielfeld` vom Typ `Spielfeld`, sowie `spielerLinks` und `spielerRechts`, jeweils vom Typ `Spieler`.
   Die Instanzvariablen sind über den Konstruktor zu initialisieren und sollen niemals `Null` sein.
   Die Klasse `KollisionsDetektor` bietet die öffentliche Methode `checkBeruehrungBallSpielfeldGrenzen(Ball ball)`. Berührt der Ball die obere oder untere Spielgrenze, so ist `ball.umkehrenBewegungInY()` aufzurufen.
   Die Berührung des Balls durch die Schläger wird in der Methode `checkBeruehrungBallMitSchlaeger(Ball ball)` geprüft und die Bewegungsrichtung des Balls ggf. verändert. Auch wenn es sich um eine recht einfache Lösung handelt, sollte doch etwas Abwechslung geboten werden. Entsprechend soll bei der Berührung des Balls mit einem Schläger zufällig ausgewählt werden, wie der Ball abprallt:
   a) entgegengesetzte Richtung b) Einfalls- gleich Ausfallswinkel



   Beide Varianten lassen sich über die Methoden `umkehrenDerBewegungInX()` und / oder `umkehrenDerBewegungInY()` der Klasse `Ball` realisieren.
   Um festzustellen, ob sich der Ball außerhalb des Spielfeldes befindet, soll die öffentliche Methode `BallPosition checkAusserhalbDesSpielfeldes(Ball ball)` definiert werden. Diese Methode prüft, ob sich der Ball links oder rechts

außerhalb oder innerhalb des Spielfeldes befindet. Als Ergebnis wird ein Parameter des `Enums BallPosition` zurückgegeben:

```
public enum BallPosition {DRINNEN, DRAUSSEN_LINKS, DRAUSSEN_RECHTS}
```

8. Ein letztes Mal ist nun die Klasse `PongSpiel` anzupassen. Diese erhält die zusätzlich Instanzvariable `detektor` vom Typ `KollisionsDetektor`. Die Instanzvariable wird über die private Hilfsmethode `initialeAufstellung()` initialisiert. Zudem muss die Methode `spielen()` erweitert werden, so dass nachdem die neue Position des Balls über die Methode `ball.bewegen(anzahlFrames)` berechnet worden ist, geprüft wird, ob eine Kollision zwischen Ball und Spielgrenze, sowie den Schlägern der linken und rechten Spieler vorliegt. Ist dies der Fall, wird die Bewegungsrichtung des Balls verändert.
   Dies übernehmen die Methoden `checkBeruehrungBallSpielfeldGrenzen( Ball ball)` und `checkBeruehrungBallMitSchlaeger(Ball ball)` der Klasse `KollisionsDetektor`.
   Über die Methode `BallPosition checkAusserhalbDesSpielfeldes(Ball ball)` wird geprüft, ob der Ball die Spielfeldgrenze verlassen hat und auf welcher Seite. Entsprechend sind die Punkte eines Spielers zu erhöhen. Natürlich muss, wenn kein Spieler 15 Punkte erreicht hat, der Aufschlag des Balls von der Seite erfolgen, dessen Spieler den Punkt gemacht hat. Achtung: zeichnen Sie den Ball nicht zu nah an den Schläger, da ansonsten die `KollisionsDetektion` zuschlägt und den Ball nicht loslässt;-)
   Hat ein Spieler 15 Punkte erreicht, wird das Spiel beendet (`System.exit(1)`).

Puh, sehr viel Text!! Sollte etwas unklar beschrieben sein oder Sie entdecken einen unvermeidlichen Fehler, so informieren Sie die Betreuer gerne.

Viel Erfolg!!