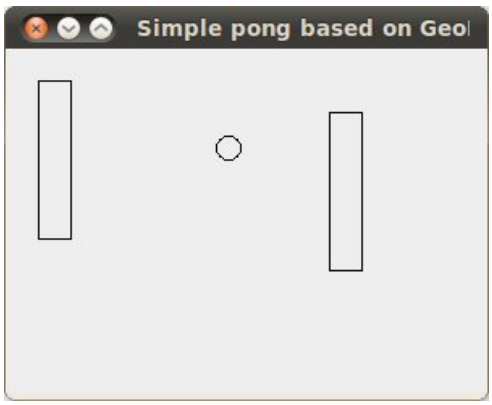
# Aufgabe 3 (*Pong*)




***Lernaspekte:***

- Modellierungsfragen/ -entscheidungen,
- Ereignisbearbeitung (Tastatur),
- Klassenbeziehungen:
  - Vererbung,
  - insbesondere kennt-Beziehung,
- Objektkommunikation,
- Spieleprogrammierung.

***Materialien (auf der CD):***

- GeoFaSC-Bibliothek,
- Vorgefertigtes BlueJ-Projekt mit vollständigen Quelltexten,
- GeoFaSC-Dokumentation (als JavaDoc)/ -Referenz (als pdf-Datei).

***Aufgabenstellung:***

`Pong` ist ein populäres Atari-Spiel aus dem Jahr 1972, das es in vielen Varianten gibt. In einer einfachen Variante spielen sich zwei in vertikaler Richtung bewegbare `Schlaeger` (Rechtecke) einen `Ball` (Kreis) zu. Kollidiert der `Ball` mit einem `Schlaeger`, der oberen oder unteren Spielfeldbegrenzung, so prallt er abhängig von seiner Bewegungsrichtung ab. Überschreitet der `Ball` die rechte oder linke Spielfeldbegrenzung, so positioniert er sich automatisch in etwa der Spielfeldmitte neu. Üblicherweise gewinnt der `Schlaeger` mit den meisten Ballberührungen (hier nicht implementiert). Aus Effizienzgründen sind Teile dieses Programms nebenläufig mit Threads programmiert, die ignoriert werden können.

a) Machen Sie sich zunächst mit der Funktionsweise des Programms durch Ausführen der Klasse `Pong` (= Hauptprogramm) vertraut. Der linke und rechte `Schlaeger` werden durch die Tasten 'F' und 'D' bzw. 'J' und 'K' hoch- bzw. runter bewegt.

b) Analysieren Sie die Quelltexte und entwickeln Sie ein Modell in Form eines Klassendiagramms. Verwenden Sie dazu die in der UML üblichen Notationselemente (siehe umseitig).

c) Vollziehen Sie die Swing-nahe Lösung der Verarbeitung von Tastaturereignissen in diesem Programm nach. Welches Objekt registriert sich bei welcher Komponente zum Zuhören und Verarbeiten der Tastaturereignisse?

d) Damit ein `Ball` bei Kollision mit dem Rand des Spielfeldes und mit den `Schlaegern` seine Bewegungsrichtung ändern kann, müssen diese Objekte untereinander kommunizieren (sog. Inter-Objekt-Kommunikation). Diskutieren Sie folgende Fragen:

   1. Wie kommunizieren die genannten Objekte in diesem Programm zur Lösung des „Kollisionsproblems“?
   2. Wie ist die kennt-Beziehung der Objekte untereinander implementiert und wie ist diese gerichtet?
   3. Wie könnte eine alternative funktionierende Inter-Objekt-Kommunikation zur Lösung des „Kollisionsproblems“ aussehen? Hätte diese ein wirklichkeitsnäheres Modell und brächte diese Vorteile für die Programmierung?

***Ihr Klassendiagramm:***

## ***Notationselemente Klassendiagramm UML:***