Quark--Qian Zhou, Victor Lin, Kevin Lin
Pd 1

Programming base: Processing
Project: Pinball

Like the classical pinball game, the user will have several balls in his/her inventory. The user presses two different keyboard keys to move each to move one flipper up and down; there will also be another key to shoot the ball from the loading zone. The entire "field" will be composed of different objects which, if hit by the ball, will give the user points.

Classes/objects:

- Ball
- Obstacle(Abstract)
    - Wall
    - Bumper
- Flipper
- Game

Instance Variables:

- Ball
    - int xvel, yvel
    - int xpos, ypos
    - radius
- Flipper
    - int flipperState
    - int point//fixed point
    - float angle???
- Game
    - int numBalls
    - int points
    - Flipper left, right
    - Ball
    - Arraylist<Wall & Bumpers>
- Obstacle
    - int xpos, ypos, size(radius? length?)
    - Subclasses:
        - Wall: linear obstacles used to construct the walls, the kickers and the ramps

            - Ypoint2
            - Xpoin2
        - Bumper : circular obstacles

Methods:

- Ball
    - move() : Changes velocity of ball and moves
    - contact(Object other) : Returns true if touching other object
    - bump(Object other) : Changes velocity of ball based on object hit
- Game
    - shoot() : Releases ball into field (if no ball currently in the field)
    - isAlive() : Returns false if there are no usable balls left
    - leftFlipper() : Moves left flipper
    - rightFlipper() : Moves right flipper
    - createField(): draws the field
    - designField(): places all the obstacles in place
- Flipper
    - move() : Moves flipper
    - getState(): returns flipper state (maybe to determine if it's falling down or moving up?)
- Obstacles
    - inContact(): returns true if ball touches the obstacle, false otherwise
    - Subclasses:
        - Wall
            - inContact(): point line distance formula
            - reflect(): ball moves in reflected direction
        - Bumper
            - bounce(): ball moves in reflected direction

TODO:

1) Create objects with basic physical properties
2) Places objects onto the gameboard and maintain functionality
3) Implement user experience
4) Finalize and optimize
5) Any add-ons if time