Crash

My program does not solve any world problems, but it may help with understanding collisions from Physics.

1. The primary user of the program will be students and definitely little kids.
2. My program allows a user to input a velocity for an object. This is a stimulation for collisions in Physics. It will bounce the puck around for 20 seconds. It also has the ability to output all the positions of all the pucks. It can also redraw the lines from the pucks.

Data: This program can read from a file given an input and a series of commands.

Example file input:

10, 0,0
(initial velocity, starting x-position, starting y-position)
Go
(starts the game)
Pause
(pauses the game)
Stop
(stops the game)
Replay
(replays the puck's movements)
Restart
(restarts the game)

User interface:
The program starts when the user clicks on the start game button in the middle of the screen or types in Go. The user will get to choose a color scheme for the colors of the puck. After the user clicks on a color scheme, the board will appear with one puck in the middle of the screen.

After the user clicks on the puck, all the other pucks will appear with their colors on them. The user will be allowed to input a velocity for the Puck, and the other pucks will stay stationary until the Puck hits it. The Puck will then move with that velocity and bounce off the walls and bounce other pucks. The game will automatically stop after 20 seconds, and all the pucks will disappear to show the lines drawn by the pucks bouncing around.

There will be various commands that allow the user to pause, start, restart, replay, or quit the puck game. The user can then choose (on the bottom of the screen) to replay, restart, or stop. If the user chooses to stop, the board will disappear and a goodbye message will be shown.
If the user chooses to play again, the program will restart. If the user chooses to replay the game, the board will clear, and the lines will redraw. After each simulation, every single position of the puck, when colliding with either another puck or a wall, will be saved to a file.

There will be a class that can store the position of the puck when it updates. The positions will be stored in a ListADT. After the program terminates, the positions will be written to a file for the user.

Classes:

- IPuckListener interface
    - Behavior:
        - Gets update from Puck when its position changes

- Puck implements IPuckListener
    - Behavior:
        - How the puck moves
        - Bounce other pucks
        - Update the Listeners
    - Data:
        - Position
        - Velocity
        - Listeners

- Wall implements IPuckListener
    - Behavior:
        - Bounces the puck off the wall
    - Data:
        - Boundaries

- Velocity class
    - Behavior: none
    - Data:
        - Position
        - Speed

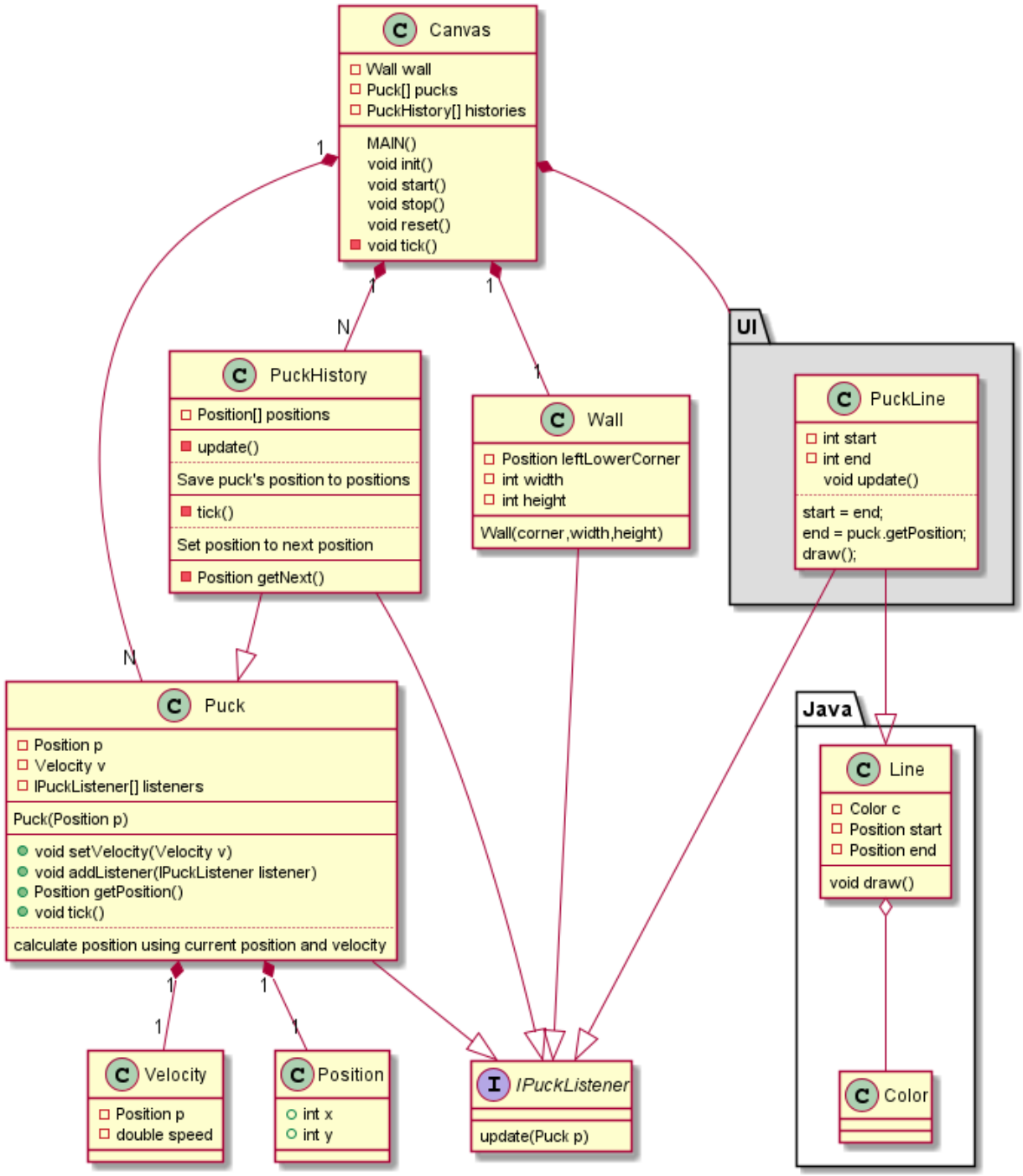
- Position class
    - Behavior: none

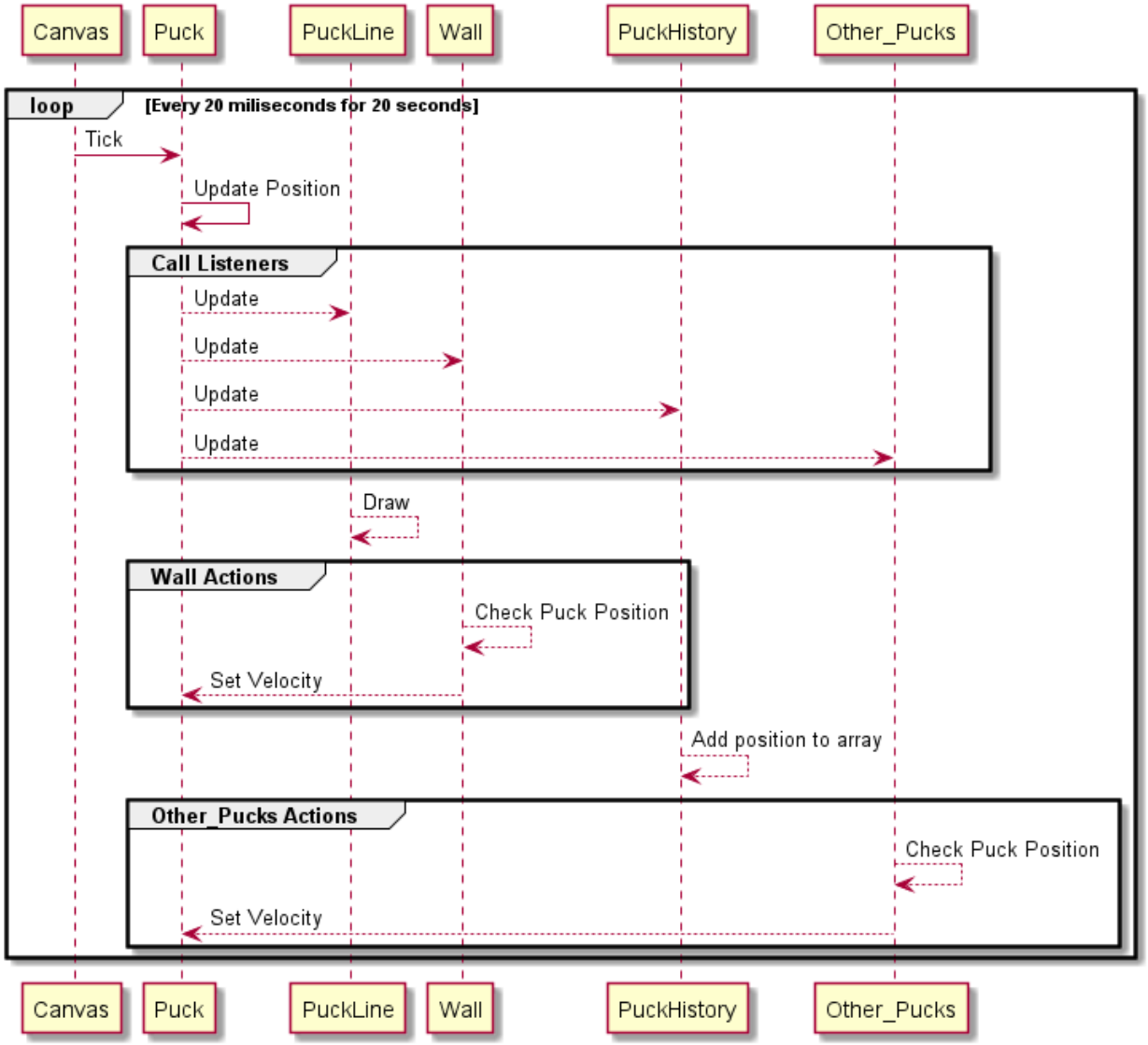
    - Data:
        - X
        - Y

- Canvas class
    - Behavior:
        - Creates all objects
        - Adds listeners
        - Timer - tick the pucks
        - Manage UI
    - Data:
        - Pucks
        - Listeners
        - GUI
        -

- PuckLine implements IPuckListener
    - Behavior:
        - Draw a line between 2 positions
    - Data:
        - 2 positions
        - Color
        - UI to draw on

- PuckHistory extends Puck implements IPuckListener
    - Behavior:
        - Stores the positions of a puck
        - Mimic a puck for UI
    - Data:
        - Position[]


Canvas
Wall wall
Puck[] pucks
PuckHistory[] histories
MAIN()
void init()
void start()
void stop()
void reset()
void tick()
1
1
1
N
1
N
PuckHistory
Position[] positions
update()
Save puck's position to positions
tick()
Set position to next position
Position getNext()
Wall
Position leftLowerCorner
int width
int height
Wall(corner,width,height)
UI
PuckLine
int start
int end
void update()
start = end;
end = puck.getPosition;
draw();
Puck
Position p
Velocity v
IPuckListener[] listeners
Puck(Position p)
void setVelocity(Velocity v)
void addListener(IPuckListener listener)
Position getPosition()
void tick()
calculate position using current position and velocity
1
1
1
1
Java
Line
Color c
Position start
Position end
void draw()
Color
Velocity
Position p
double speed
Position
int x
int y
IPuckListener
update(Puck p)


Canvas
Puck
PuckLine
Wall
PuckHistory
Other_Pucks
loop
[Every 20 miliseconds for 20 seconds]
Tick
Update Position
Call Listeners
Update
Update
Update
Update
Draw
Wall Actions
Check Puck Position
Set Velocity
Add position to array
Other_Pucks Actions
Check Puck Position
Set Velocity
Canvas
Puck
PuckLine
Wall
PuckHistory
Other_Pucks