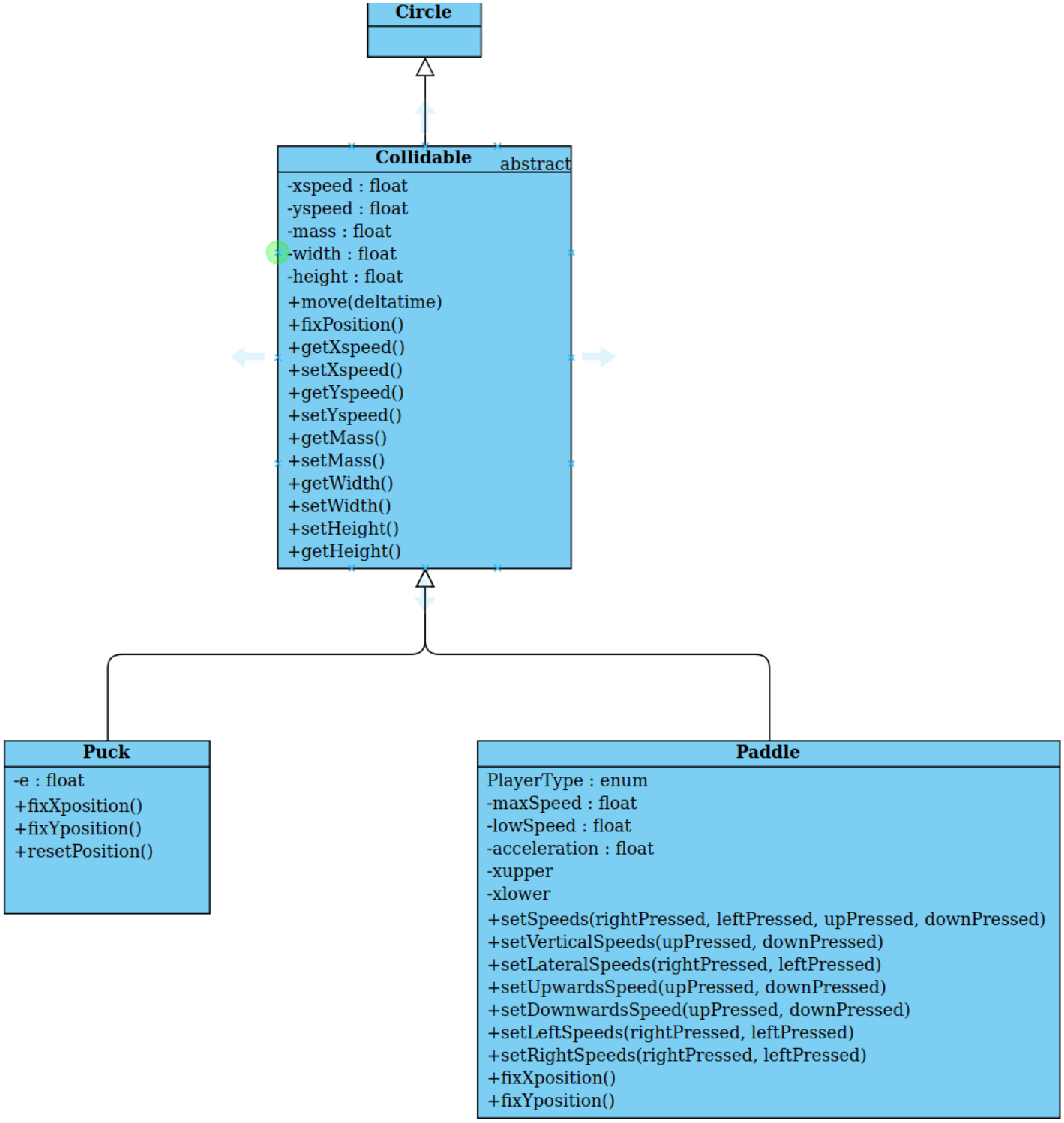
Why:
The puck and paddle have similar physical properties and behaviour in the real world, and hence should have similar properties and behaviour when implemented in our game. The puck and paddle are closely connected; they will collide with each other, and they will both be restricted by the boundaries of the board (where the game is played). But they also will differ in some regards (the puck can move around the whole board, whereas the paddles should be restricted to their own sides of the board). This led me to choose the template design pattern, as I could encapsulate shared behaviour, but also let the subclasses redefine parts of their behaviour that are unique.
I used this design pattern to implement invariant behaviour between the Puck and Paddle, and to factor out this common behaviour so that code is not duplicated (move method, fixPosition method and getters and setters of various common attributes).
I further used this pattern to define where the concrete subclasses had to do certain common tasks (such as fixing the X and Y co-ordinate positions), but the implementation would be different for each class. In this way these subclasses redefined certain parts of the behaviour that were necessary.

How:
The Collidable class is the Abstract SuperClass. The concrete classes are the Puck and Paddle classes. They are subclasses of the Collidable class. They inherit the attributes xspeed, yspeed, mass, width and height. (The super class Collidable itself inherits from libgdx' Circle class, hence inheriting the Circle class' behaviour and attributes).
The puck and paddle classes inherit the move, fixPosition, and various getter and setter methods for the shared attributes.
I used abstract methods to control subclass extensions. The abstract Collidable class defines two abstract methods fixXposition and fixYposition. These are methods that are implemented independently in the Puck and Paddle classes as their behaviour is slightly different in these respective classes. The paddle has further methods to set its speed, as this is the object that the user can directly control.


Circle
Collidable
abstract
-xspeed : float
-yspeed : float
-mass : float
-width : float
-height : float
+move(deltatime)
+fixPosition()
+getXspeed()
+setXspeed()
+getYspeed()
+setYspeed()
+getMass()
+setMass()
+getWidth()
+setWidth()
+setHeight()
+getHeight()
Puck
-e : float
+fixXposition()
+fixYposition()
+resetPosition()
Paddle
PlayerType : enum
-maxSpeed : float
-lowSpeed : float
-acceleration : float
-xupper
-xlower
+setSpeeds(rightPressed, leftPressed, upPressed, downPressed)
+setVerticalSpeeds(upPressed, downPressed)
+setLateralSpeeds(rightPressed, leftPressed)
+setUpwardsSpeed(upPressed, downPressed)
+setDownwardsSpeed(upPressed, downPressed)
+setLeftSpeeds(rightPressed, leftPressed)
+setRightSpeeds(rightPressed, leftPressed)
+fixXposition()
+fixYposition()