Design Document

Intro:

Period: 2

Names: Justin Codner, Thomas Mackey, Ethan Lin
Emails: jcodner60@stuy.edu, tmackey@stuy.edu, elin61@stuy.edu

Group Name: The Eight-Ballers

Project Title: 8-Ball

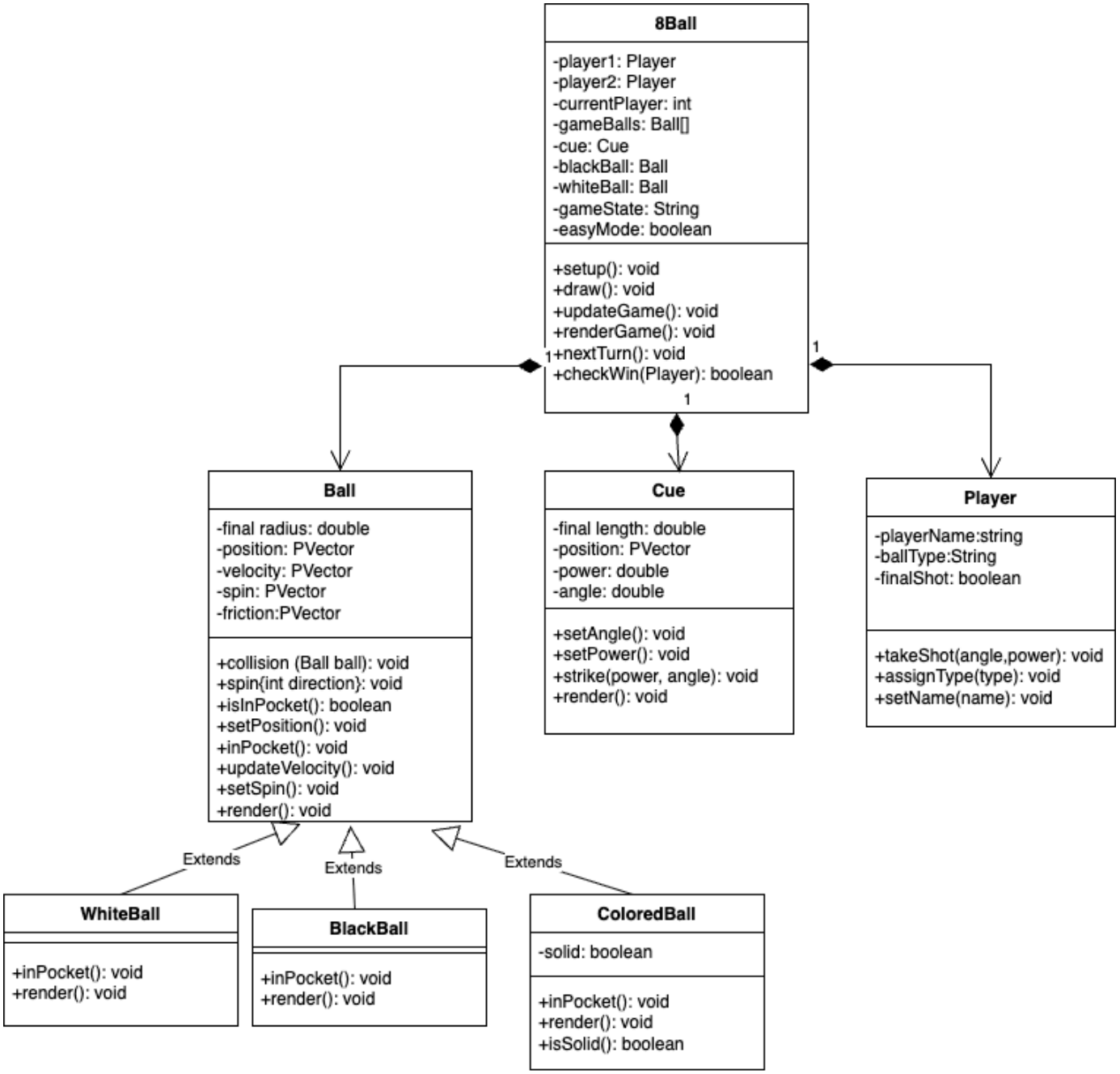
Description:

We are creating a two player java version of the classic billiard game.
Functionalities: Players will be able to control the spin and power on their shots. We will make use of velocity and acceleration vectors to make this possible. Additionally, we intend to implement elastic collisions using the conservation of momentum, and also static friction through acceleration vectors. We are also creating a scoring system to keep track of stripes vs solids using the pockets of the pool table. We are going to create a ball class, a cue class, and a class for the pool table for the scoring mechanics. The ball class will have multiple subclasses for the different ball types in the game. Additionally, we will implement two difficulties, easy and hard mode. In easy mode, you will be able to see the trajectories of the balls after an elastic collision, but in hard mode, those will not be visible.
Libraries: None as of now.

How does it work:

This is a two player game, with player 1 and 2. The game starts with player one breaking the balls in the middle of the table. If player 1 gets a ball in the pocket when he breaks, he is assigned stripes of solids based on the first ball he gets in, if he doesn't, then it becomes player two's turn, if he gets a ball in a pocket, then he is assigned stripes or solids based on the first ball he gets in. If they both don't manage to get a ball in, they will go back and forth until they get a ball in and assign stripes and solids to each player. During your turn, it is your goal to get balls of your type into the pockets, if you get a ball of your type into a pocket, then you may go again. To do this, you must manipulate the power you use to hit the ball, and also the point of impact on the ball in order to dictate its spin. If you get all of your types of balls into the pocket, then you can hit the black ball into a pocket that you choose before hitting in order to win. If at any time in the game, the black ball goes into a pocket and the player has not gotten in all of his balls or the pocket wasn't the one he chose before hitting, then that player automatically loses. In the case that the white ball falls in the pocket, the turn goes to the other player and the other player can choose where to place the ball.

UML Diagram:


8Ball
-player1: Player
-player2: Player
-currentPlayer: int
-gameBalls: Ball[]
-cue: Cue
-blackBall: Ball
-whiteBall: Ball
-gameState: String
-easyMode: boolean
+setup(): void
+draw(): void
+updateGame(): void
+renderGame(): void
+nextTurn(): void
+checkWin(Player): boolean
1
1
1
1
Ball
-final radius: double
-position: PVector
-velocity: PVector
-spin: PVector
-friction:PVector
+collision (Ball ball): void
+spin{int direction}: void
+isInPocket(): boolean
+setPosition(): void
+inPocket(): void
+updateVelocity(): void
+setSpin(): void
+render(): void
Cue
-final length: double
-position: PVector
-power: double
-angle: double
+setAngle(): void
+setPower(): void
+strike(power, angle): void
+render(): void
Player
-playerName:string
-ballType:String
-finalShot: boolean
+takeShot(angle,power): void
+assignType(type): void
+setName(name): void
Extends
Extends
Extends
WhiteBall
+inPocket(): void
+render(): void
BlackBall
+inPocket(): void
+render(): void
ColoredBall
-solid: boolean
+inPocket(): void
+render(): void
+isSolid(): boolean