Environment class: Environment.prompt() method only takes a string as a parameter and returns the game board according to the parameter("You won!" or "Game Over!") .show() is the method Main class calls to start the game.It contains main animation loop and every update of positions(happens in separate classes) and active balls(balls that are in the current ballArrayList) happens there. In the beginning of the show() method and outside the main while loop which animates the game, we create 3 ball objects (one for each level). The initial ballArrayList that will be iterated in the for each loop in the main animation loop will be initialized as a list containing 3 balls of each type.Main animation loop simply does following in order: setting the background picture, drawing the bar making isactive() function of arrow to check it at every iteration if space is pressed and if true displaying the arrow, displaying the player, displaying every one of the balls(one by one with the for each loop) in the arraylist(initially three) with their changing positions and velocities(at each main animation iteration), display the current drawing to the screen(show), clear the canvas, and pausing the execution.Main animation loop is a while loop which will continue iterating if flag is true. Flag is initialized in the Bar class as true. In case of totalGameDuration has been passed and flag haven't been set to false from other conditions beforehand; it means ball-player collision haven't happened and ballArrayList is not empty yet, therefore player losed the game because there is no time left.In that case flag will be setted false with "Game Over!"prompt and iteration will stop, current game will end.In the case of a ball-player collision is occured (getting checked in the for each loop of ballarraylist which is in the main animation loop) flag is set false again with "Game Over!"prompt because current game has ended.In the case of there is no ball left in the ballArrayList flag is set false again because current game has ended, but this time with the with "You Won!"prompt.We check ball-player collisions with if statements in for each loop of ballArrayList.First, we detect which levelball we are currently dealing with by using radiuses.Then, we calculate maximum possible distance current ball can have without colliding the player and also calculate the actual distance of the ball and the player at that specific time with pythagorous' theorem.Lasty, if current distance is small or equals max distance allowed for not colliding; it means they have collide.we check ball-arrow intersetions with conditions relaying on current ball's x_position - current arrows x_position and current ball's y_position and current arrows scaledheight. Arrow's scaledheight changes with respect to passed time, therefore we calculate a passed_time variable at every iteration(placing it in the main animation loop but outside for each loop) to acquire current lenght of the arrow correctly.If current ball's current x_position's distance to the arrow's x_position is smaller than or equal to the its radius and current ball's current y_position minus arrow's current scaledHeight is less than or equal to its radius it means that arrow has hit the ball. Remove the popped ball from the list.Generate new ball objects considering the level of popped ball and add them to the ballArrayList with current x and y positions where intersection occured. Also their starting vx values should be opposite signs.If the ball that has been popped is a level0 ball only remove it form the list,do not generate anything.If a ball-arrow interaction has occured break the current loop which will continue iterating with the ballArrayList in which popped ball was belonged to, and start iterating form the beginning of the for each loop with the updated ballArrayList.Main(simal_guven) class:Flag is initially set to true therefore while loop will start operating in the beginning and everytime show() method is called (actual game method) in every situation of how the game ends, flag will be left as false.After the main game method Environment.show() a while(true) loop iterates and constantly checks for a keyboard input. It should constantly check therefore constantly iterate because we cannot give an exact time at which keyboard input is going to be taken from the user.If replay (Y) is pressed flag becomes true and with the break statement at the end of the loop computer goes back to the bigger loop with flag

equals true, therefore Environment.show() is activated again and again .If quit(N)is pressed computer closses the program.

https://drive.google.com/file/d/1QJdLw3DElNTJYTHyV6vNTKG6XHfcztO1/view?usp=drivesdk