Firstly, from a bird's eye view to classes' functions , Main Class is to only call the function to play the game. Environment class keeps all game related constants, some static methods to check intersections between game objects and playTheGame() method which is the foundational part of the project. In playTheGame() method, canvas is set, game objects are created. Inside the method, in a while loop to make the animation, objects positions and velocities are updated and the conditions that makes the game over are checked by calling other methods. If for some reason game is over, this animation loop is broken. After the loop, there are conditions checking the reason of the finishing game, and according to those, end station of the game is drawn. Quitting and replaying the game is also handling here.

In Player class, the player's movement is controlled by the move() method in the Player class. This method checks if the left or right arrow keys are pressed and updates the player's x-coordinate accordingly. The player's movement is limited by the width of the canvas and the width of the player image. Also, there are two variables isAlive and winner. Firstly, isAlive is set as true which means player is alive. When a ball hits the player, this variable will be false and game will be over. Winner variable is set initially false and if player wins the game, it is set as true and game will be over.

In Ball class Each ball has an x and y coordinate, a radius, and a level. The level of the ball determines its size and behavior when hit by the arrow. The Ball class includes methods for updating the ball's velocity and position over time. The updateVelocity() method updates the ball's velocity based on gravity and the time that has passed since the last update. The move method updates the ball's position based on its current velocity and the time that has passed since the last update. When a ball is hit by the arrow, it can split into two smaller balls if its level is greater than 0. This behavior is controlled by the method arrowHitsBall() in the Environment class. This method checks if the arrow has hit a ball and, if so, creates two new balls with a lower level and adds them to the game.

The game also includes a time bar that shrinks over time and changes color from green to red. The time bar's size and color are updated over time using the pass and setColor() methods in this class. Animation of shrinking bar over time is provided by this idea: As time passes, slide left the time bar by its velocity*time. Actually, the length of the bar is always same, but it just slides left. The arrow is controlled by the player and can be shot upwards by pressing the space bar. The arrow moves upwards at a constant y velocity, which is controlled by the moveUp() method in the Arrow class. The arrow becomes inactive if it hits a ball or reaches the top of the canvas. The game ends if the player runs out of time, if a ball hits the player, or if all the balls are eliminated. If the game ends, a game over screen is displayed with options to replay or quit. The user can choose to replay by pressing the "Y" or quit by the "N" key.

Apart from those, one of the main idea behind the code was that. At each two successive while iteration, time difference is calculated and objects new coordinates are set by multiplying this tiny value by the velocities. Also, the velocities are updated similarly to make balls do projectile motion: add the time*gravity to the current velocity of the ball. This also implies that: as the time difference between each two successive iteration in the loop gets larger, the animation will appear to slow down over time and its quality may deteriorate. Another essential concept was the mathematical inequalities to check the intersections between the objects of the game as well as physical knowledge.

Drive link of the game video:
https://drive.google.com/file/d/12tNQBcaGd37QdcQ0Aua_qZEGAG3OM7fL/view?usp=share_link