We made changes in the overall rules of the game. In our initial program specification, the game was lost every time the ball hit the floor. This made the game very short-lived and unexciting so we implemented lives instead to give the user relatively longer playtime.

Initially, we only had one level so the game was won when the first set of bricks was all broken. To add to the complexity of the game, we decided to add a second level because of which the implementation of winning the game changed. After level 1, the user gets to choose to quit or continue and then level 2 decides if the user wins or doesn't.

We first planned on using solely drawstring to communicate with the user e.g. for game won, game lost, game over etc. The game did not look good aesthetically that way so we decided to import images for these purposes instead. Now there are buffered images that crop up in all these cases.

We were only using KeyBoardListener first. This was to control the saucer. But that did not allow the user to choose levels of difficulty or pick an option on screen conveniently. So, we implemented MouseListener in order to detect clicks and add to the choice-provision and layers of the game.

Initially, we had separate methods in world to check if the game is won or lost and to add the score. Later we felt that it just added to the length of the code unnecessarily so we just used variables like Boolean gamewon, gamelost, and public static int score for these purposes instead. Gamewon and gamelost were changed according to the number of lives and the count of broken bricks. This interaction between variables, especially making score static allowed us to significantly cut down on the length of our code.

We added a lot to our paintcomponent. We realized that we were trying to control and change the contents of the screen from outside the paint component which made things a lot more convoluted than necessary. So we added if statements to our paint component to deal with each scenario of the change of screen e.g. if the game is won or over or lost.

We were initially only making some bricks bonus by giving them 20 points instead of the normal 10 points. But we wanted to be more generous, so we added power ups as another feature of the bonus bricks. Each power up is different and provides different perks e.g. freezing the ball for some time, speeding up the saucer, giving more lives, giving more points etc. This is why we made a new class called power up and imported images to fall down from each bonus brick and be caught by the saucer for the power up to act.

We also added sound to the game in order to make it more fun. There is a constant background sound (essentially Game of Thrones intro song) and then instantaneous sounds like the collision of the ball with bricks and the sound of power ups.

We also needed a Timer Object to execute the power ups because the power ups should only happen for some time. We used time object to limit the length of the power up to a fixed length of time.

We also modified the front screen to have not just the play game option but also the option of choosing a difficulty level from among easy, medium and difficult. These levels vary in their speed of the ball. The harder the level, the faster the ball. We used mouseClicked to detect the click on a level and used setVelocity method of the ball to change the velocity of the ball.