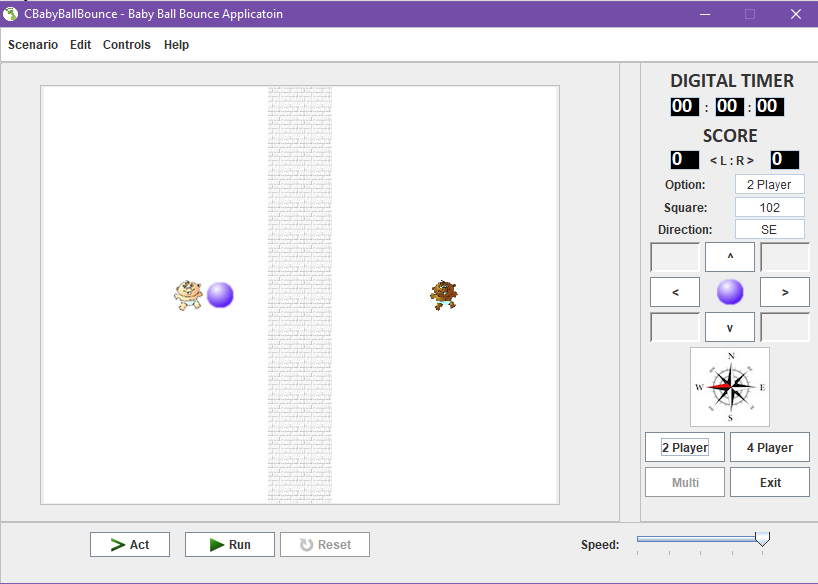
**CSY1020: Problem Solving & Programming**

**Assignment 2: Programming (Java) (50%)**

Sunil Thapa

17421492

BSc Computing



University of Northampton

Submission: **Sunday, 30th July 2017, by 23h59min.**

Contents

[University of Northampton 1](#_Toc489209663)

[1. INTRODUCTION 3](#_Toc489209664)

1. [Aim and Objective 3](#_Toc489209665)

[2. ANALYSIS 5](#_Toc489209666)

1. [Essential GUI 5](#_Toc489209667)
2. [Additional Functionality & Complexity 5](#_Toc489209668)

[3. DESIGN 6](#_Toc489209669)

1. [Graphical User Interface Design: 6](#_Toc489209670)
2. [Pseudo Code: 8](#_Toc489209671)

[4. IMPLEMENTATION: 11](#_Toc489209672)

[5. TESTING 15](#_Toc489209673)

1. [kickBall() method: 15](#_Toc489209674)
2. [Additional features: 15](#_Toc489209675)
3. [Here is Testing criteria provided. 15](#_Toc489209676)

[System Requirements: Additional (Functionality & Complexity): 15](#_Toc489209677)

[6. Conclusion 16](#_Toc489209678)

1. [Summary: 16](#_Toc489209679)
2. [Future Improvements: 16](#_Toc489209680)

# INTRODUCTION

## Aim and Objective

A Green Foot Scenario was provided as shown in figure 1.1. Here, Java GUI needs to be created where there must be feature of simulation of ball when keys are clicked (up/left/right/down arrows) around the white background and also ball must be kicked(bounce) after touching the babies on opposite direction.

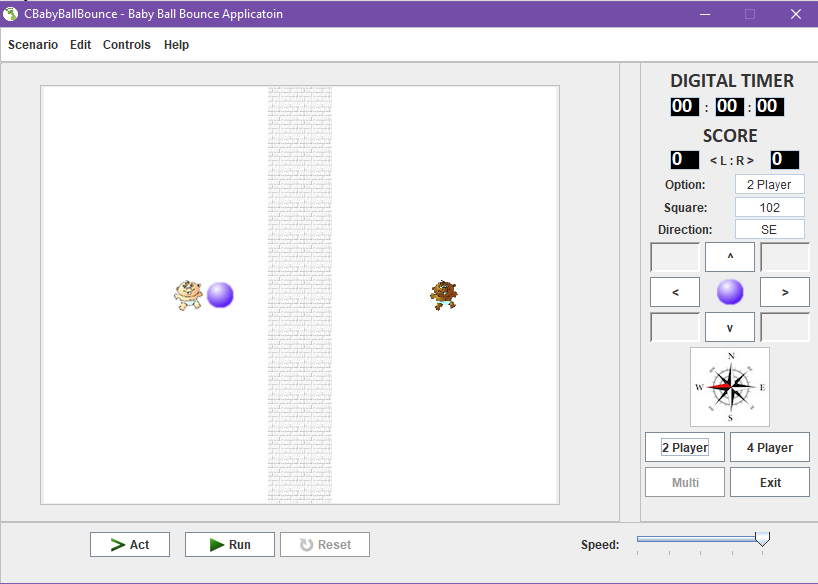
**

Figure 1.1: CBabyBallBounce.java Application – Opening State Microsoft OS.

Here are the objective provided:

**Rules (Basic)** Create a simulation of the ball moving around the pitch, where:

* The ball can move anywhere within the pitch and across the wall in the middle of the screen.
* If the ball touches a baby it is deflected/rebounds back in the opposite direction.
* The ball must move one whole ‘white/wall’ block at a time every time a movement button (via a direction button (<, > v ^)) /key is pressed (when movement is possible).
* The solution must use the scenario provided. i.e. If the babies are left in the unmodified codes starting position the ball would move between them continuously.
* The ball must stop when the perimeter of the pitch is reached.
* The basic solution must be completed using the ‘act’ button (accessing the **kickBall()** method within the **CBabyBallBounce.class**).

**Rules (Intermediate and advanced)** Create a simulation of a ball moving around the pitch, where:

* Your solution must still use the scenario provided (all the basic features above).
* Add appropriate extra features to the solutions, e.g. a) The ball can bounce off a baby or babies in random direction, b) Two new ‘player’ babies are added to each side of the line, that move vertically towards the ball ready to potentially bounce the ball back, c) Add a scoring system for each side (a, b or c for **Intermediate,** a, b & c for **Advanced**).
* For higher grades on the solution part of the assignment see the marking scheme at the front of the brief.
* You must NOT change the layout and all changes should still meet the criteria of **Rules (Basic**).

# ANALYSIS

Below are the system requirements:

## Essential GUI

* JButton's 13\*16 grid needed to be created.
* '2 Player', '4 Player', 'multi' and 'exit' buttons are game options button.
* 'Act', 'Run' and 'Reset' button
* 9 Buttons having icon '<', '>', '^', 'v' and 5 additional blank buttons are needed.
* All the 9 buttons must have potential to move the ball in respective direction.
* Compass icon of JButton should show direction of movement of ball in all 4 direction.
* 'Options', 'Square' and 'Direction' needs 3 JLabel.
* Options should show selected number of players, square should show position of ball whereas direction should show bal movement direction.
* JLabels to show Digital Timer having 3 JTextField showing timer having minute, second and hours.
* 'Score' JLabel shall contain 2 JTextField having caption '<L : R>'.
* 825\*585 size JFrame application needs to be created.
* "*CBabyBallBounce – Baby Ball Bounce Application*" is title of the JFrame application.

## Additional Functionality & Complexity

GUI Application should contain Greenfoot icon. The 'Run' button should move ball continuously left to right and right to left in between the babies from the initial position. 2 Player should be default state after launching the application. The 'Reset' button should reset all the positions including ball, babies and also should reset the score and timer as it was in default position. The 'Act' button should move ball by one grid at a time. The player selection (2 Player, 4 Player and multi) should increase obstacles configuration. JMenuBar could be added with JMenu including Scenario, Edit, Control and Help. These may also contain JMeuItem's such as Exit, Open, Save options in Scenario, Help Topic and About in Help JMenu. Additional button might be added to make improvement in the applications by either moving ball in random direction, bounce of the babies and as well as borders and so on. JFrame application created mustn't be resizable and should lie on centre of monitor itself. Number of babies can be added to indicate current position of the babies as well as its direction. Ball should not go beyond the grid. Ball must bounce after hitting the boundaries. Digital timer must start after pressing run button and pause/stop after pressing pause/reset button. kickBall() method should be implement so that it contains solution for moving ball in 4 directions(right/left/up/down). kickBall() must also contain methods such as move(left), move(up), move(down) and move(right) to move ball in those directions.

public void kickBall(){

move(left);

...........

move(right);

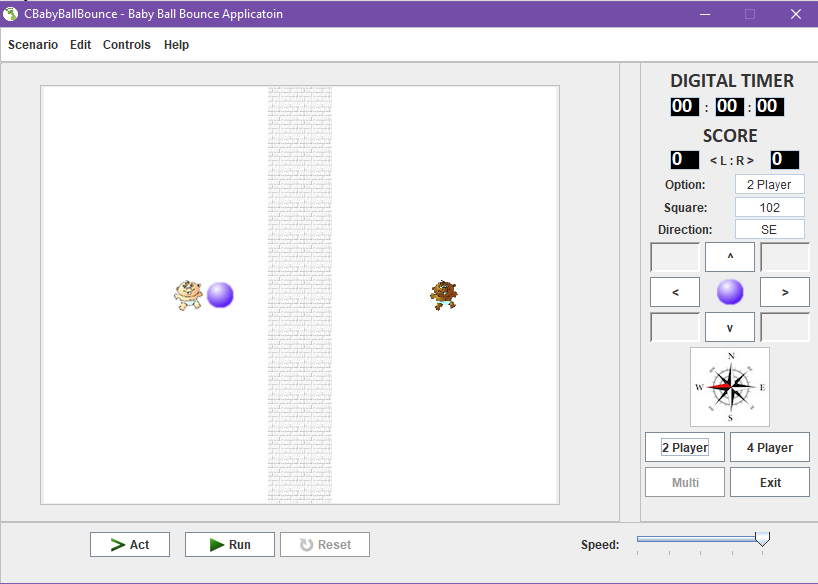
}

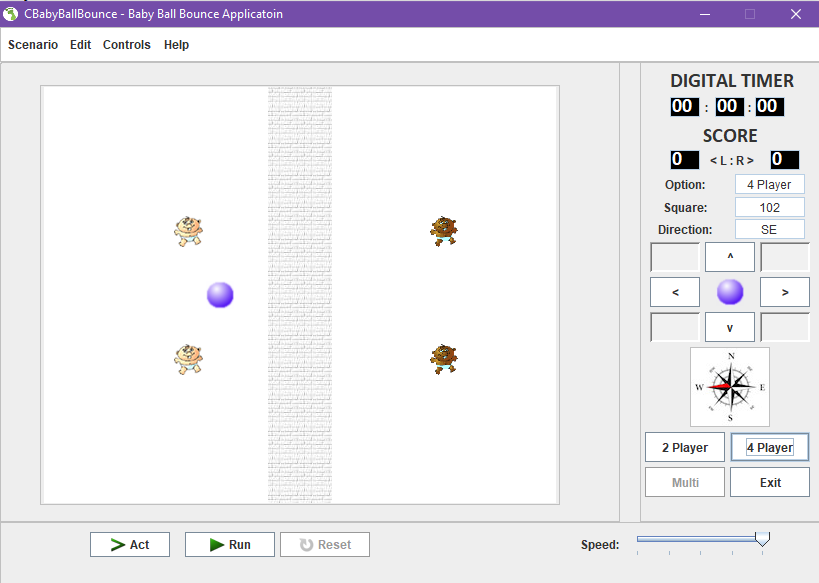
The application must be demonstrated. The name of file containing main method should have file named CBabyBallBounce.java & the compiled byte code class file should be named CBabyBallBounce.class.

# DESIGN

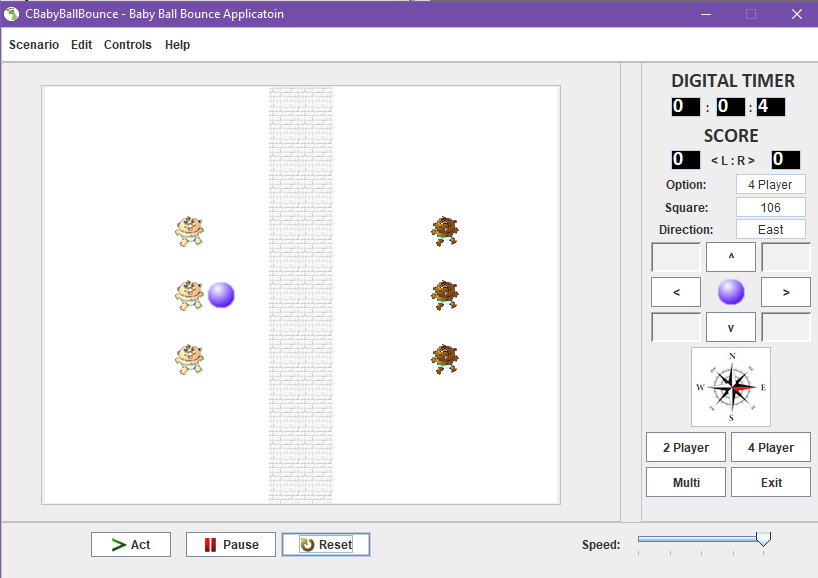
## Graphical User Interface Design:

According to scenario of green foot provided, here is GUI design created. There is presence of JMenu, white playing background where there are two/four baby or even six baby for multiplayer, keys to move the ball and text field which shows position of ball, direction of ball and option selected by the user, also there is score designed and timer to determine time running after run button is clicked.

***Figure 3.1: CBabyBallBounce.java Application – Opening State Microsoft OS.*



*Figure 3.2: CBabyBallBounce.java Application – Opening State Microsoft OS.*



*Figure 3.2: CBabyBallBounce.java Application – Opening State Microsoft OS.*

## Pseudo Code:

There is multiple levels for user to play. Here are pseudo codes for each level.

Two Player:

1. Start
2. Select 2 player option
3. Start game by pressing on Run Button
4. Move left on pressing left arrow
5. Move right on pressing right arrow
6. If ballPosition == baby2Position
7. Then
8. Move ball to left direction
9. Else if ballPosition == baby1Position
10. Then
11. Move ball to right direction
12. Increase score
13. If Score == 3
14. Then
15. Display winner
16. Stop

Four Player:

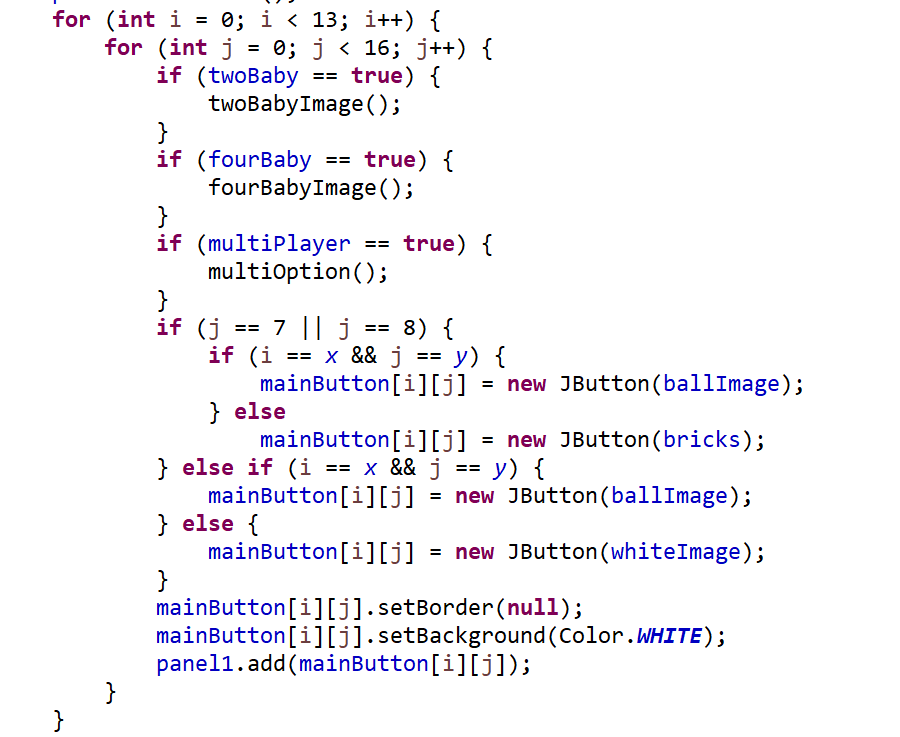
1. Start
2. Select 4 player option
3. Start game by pressing on Run Button
4. Move left on pressing left arrow
5. Move right on pressing right arrow
6. Move up on pressing up arrow
7. Move down on pressing down arrow
8. If ballPosition == baby2Position
9. Then
10. Move ball to left direction
11. Else if ballPosition == baby1Position
12. Then
13. Move ball to right direction
14. Else if ball [x] Position == 0
15. Then
16. Stop ball movement to up direction
17. Else if ball [x] Position == 12
18. Then
19. Stop ball movement to down direction
20. Else if ball [y] Position == 0
21. Then
22. Stop ball movement to left direction
23. Increase score of baby2
24. If Score == 3
25. Then
26. Display winner
27. Else is ball [y] Position == 15
28. Then
29. Stop ball movement to right direction
30. Increase score of baby1
31. If Score == 3
32. Then
33. Display winner
34. Stop

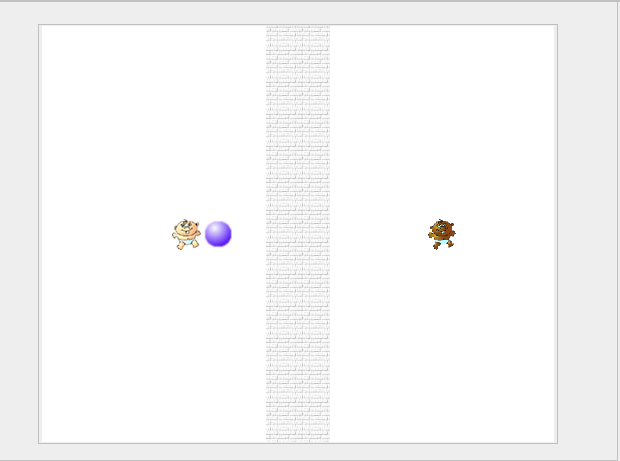
Multi Player:

1. Start
2. Select multi player option
3. Ball start moving through 1 grid at each interval
4. New random number generate at each interval
5. If ball [y] Position == (baby2-1) and random number == 1
6. Then
7. Ball move at top left position in diagonal position
8. If ball [x] Position == 0
9. Then
10. Ball move at bottom left position in diagonal position
11. Else if ball [y] Position == 0
12. Then
13. Ball move at bottom right position in diagonal position
14. Increment baby2 score by 1
15. If Score == 3
16. Then
17. Display winner
18. Else if ball [x] Position == 12
19. Then
20. Ball move at top right position in diagonal position
21. Else if ball [y] Position == 15
22. Then
23. Ball move at bottom left position in diagonal position
24. Increment baby1 score by 1
25. If Score == 3
26. Then
27. Display winner
28. If ball [y] Position == (baby2-1) and random number == 0
29. Then
30. Ball move at bottom left position in diagonal position
31. If ball [x] Position == 12
32. Then
33. Ball move at top left position in diagonal position
34. Else if ball [y] Position == 0
35. Then
36. Ball move at top right position in diagonal position
37. Increment baby2 score by 1
38. If Score == 3
39. Then
40. Display winner
41. Else if ball [x] Position == 0
42. Then
43. Ball move at bottom right position
44. Else if ball [y] Position == 15
45. Then
46. Ball move at bottom left position in diagonal position
47. Increment baby1 score by 1
48. If Score == 3
49. Then
50. Display winner
51. Stop

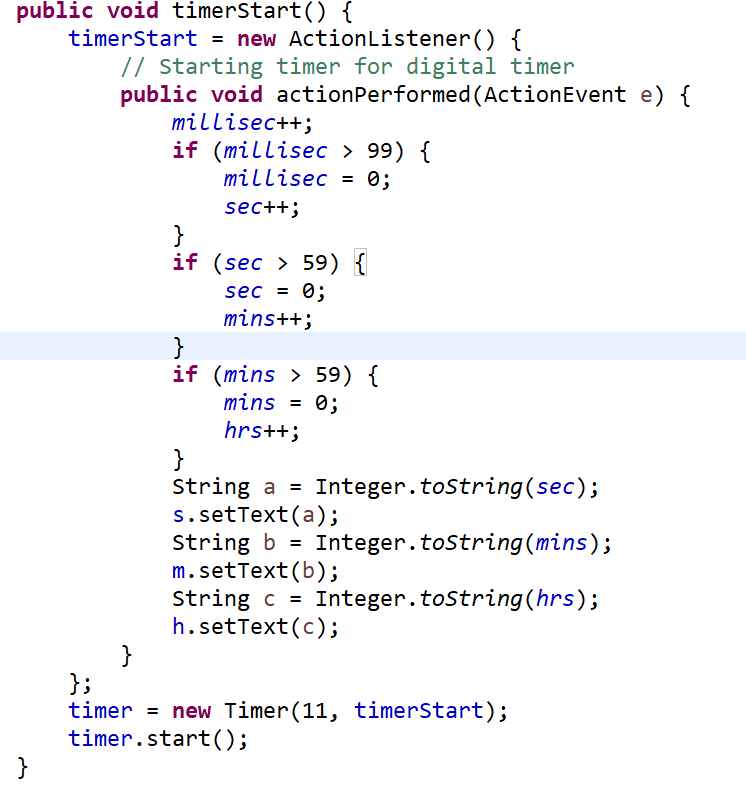
# IMPLEMENTATION:

There has been used of 5 panels. One panel is used to make parent panel which carries two panel (grid layout panel + panel to set border), another panel has been used to create right side of the panel whereas another panel is used to create bottom panel. Grid layout contains grid of 13\*16 grid. 2D array has been used to full grid of 13\*16 with button and default image for all those button is set to white image. To make walls in between the grid, 7 and 8 grid position(whole column) had been set brick image and when ball passes through the wall, ball image replaces brick image at that particular position. Here is an example of nested for loop used to create 2D array for grid layout.

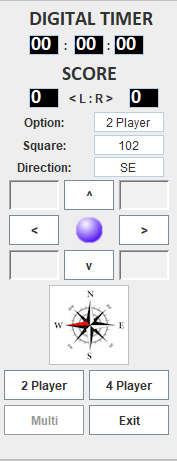




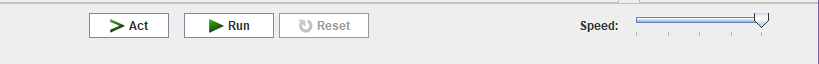
At right panel, timer had been set to acknowledge user about the start time. To make timer, an object of timer had been created which calls a global action listener at 11 millisecond. 4 variables are created to introduce timer (millisec, sec, mins, hrs). Since 99\*11==999 nearly equal to 1 sec, after millisec value is 99, sec value increase by 1 and when sec value goes beyond 59, mins value is increment by 1 and also when mins value reaches at 59, hrs value is increment by 1.



Score is also introduced after digital timer and options, square, direction are set to right-hand side. Options displays user selecting either 2 player or 4 player or multi player. Square displays number of grid on which ball is placed and direction shows movement of ball. 4 arrow keys is introduced that helps ball [x], [y] position either to increase or decrease according to the button clicked. At last of the panel, user is allowed to choose number of player to play. This acts on main white panel. If 2 player is clicked, 2 babies is set on the white screen, if 4 player is clicked, 4 babies is set on the white screen and if multi is clicked, 6 babies is set on the white screen.



At bottom panel, there is action buttons (Act, Run, Reset). Act button is used to move ball by value 1 grid. Run allows user to play the game. It enables ball to move and increase score which helps to determine winner. Reset button is used to set ball position to default position. Slider value ranges from 1-5. When slider is increased ball speed increases and if slider is decreased ball speed decreases.



Meaningful variables are used so that on future it makes code more readable at neatness.



# TESTING

## kickBall() method:

kickBall() method is created so that ball could move in left, right, up and down directions. This method contains 4 child method named move(left), move(right), move(up) and move(bottom). These 4 methods have statements to move ball in respective direction as mentioned inside parenthesis of the function name. To test those methods, each position of ball was checked on console so to make sure that written code is accurate. Try....Catch method was also used to make sure that if any error occurs, it is displayed in readable simple English format so that user doesn't get panicked.

## Additional features:

## Here is Testing criteria provided.

|  |  |
| --- | --- |
| System Requirements: Essential (Graphical User Interface): |  |
| 13 x 16 grid of **JButton**’s or Icon’s. | 🗸 |
| 4 **JButton**’s for the game options ‘2 Player, 4 Player, Multi’ and ‘*Exit*’. | 🗸 |
| 3 **JButton**’s for ‘Act’, ‘Run’ and ‘*Reset*’. | 🗸 |
| 9 **JButton**’s for ‘*Forward >*’, ‘Backwards <’, ‘Up ^’, ‘Down v’ should move the ball in the appropriate direction by one square for each press (plus 5 blank). | 🗸 |
| The compass icon (**JButton)** should illustrate the current direction for the ball. | 🗸 |
| **3 JLabel**’s for ’Option’, ‘*Square*’ and ‘*Direction*’. | 🗸 |
| **3 JTextField**’s for the current ‘Option’, Location/*’Square’* and *‘Direction’* of the ball. Use the square identification method e.g. 0 to 207 and N, E etc. | 🗸 |
| 3 **JLabel’s** for the ‘DIGITAL TIMER and the two :’, with 3 **JTextField’**s for the hours, minutes and seconds. | 🗸 |
| 2 **JLabel’s** for the ‘SCORE and ‘<L:R>’, with 2 **JTextField’**s for the scores (L & R). | 🗸 |
| Create a **JFrame** application, which opens to the set size (825 \* 585). | 🗸 |
| **JFrame** title set as "*CBabyBallBounce – Baby Ball Bounce Application*". |  |
| System Requirements: Additional (Functionality & Complexity): |  |
| Application icon for the **JFrame** used. | 🗸 |
| The ‘Run’ **JButton** should show the ball moving between the babies continuously from the initial position (2 Player – default opening state). | 🗸 |
| The ‘Reset’ **JButton** should clear/reset the application to its starting/default opening state. | 🗸 |
| The ‘Act’ **JButton** should step through the above ‘Run’ sequence one move at a time. | 🗸 |
| Discuss and implement the different options for the 3 configurations. | 🗸 |
| The ‘2 Player, 4 Player, Multi’ **JButton**’s should display different obstacle/car configurations/locations. | 🗸 |
| A **JMenuBar** could be included with **JMenu**’s for the *Scenario, Edit, Controls* and *Help*, which include **JMenuItem**’s of *Exit (Scenario)*, *Help Topic* and *About (Help)*. | 🗸 |
| Additional **JButton**’s may be used to improve the applications usability e.g. ball bounce – in random direction, deflection angle etc. | 🗸 |
| Create a **JFrame** application, which is not resizable. | 🗸 |
| Create a **JFrame** application, which centres itself on the monitor. | 🗸 |
| Use of additional baby images indicating the current position and direction of the baby. | 🗸 |
| Discuss the possibilities for incorporating intelligence/checks for whether moves are valid. | 🗸 |
| Digital Timer should start and stop when run is pressed and stopped when a baby misses the ball (with the ball continuing to the left or right boundary and stopping itself and the timer). | 🗸 |
| Implement intelligence/checks for whether moves are valid. | 🗸 |
| A **kickBall()** method should be used to solve the problem. The **kickBall()** method should include **move(left), move(right), move(up), move(down)** methods | 🗸 |
| **CBabyBallBounce.java** & **CBabyBallBounce.class** | 🗸 |
| **Predicted Grade:** | A |

Key: Blue GUI; Yellow Testing Application; Red Code.

***Table 5.1 Testing Criteria***

# Conclusion

## Summary:

Ball and baby game had been created that contains all the basic and additional feature mentioned in the document file. Ball bounces off the babies when it touches babies and also bounces off the wall in random direction. Timer is also set to acknowledge user to time elapsed. Score counter is also introduced on the application. When score is reaches to 3, a dialog box pops-up defining the winner. Arrow can be used to change ball direction as per click. Compass image that determines direction of ball moving whether by user using arrow keys or automatic movement of the ball. Square helps to define position of the ball at the current moment. To increase difficulty user can choose 2 Player, 4 Player, Multi Player options. Movement of baby is only the weakness of this application. Baby moves only if it is clicked. Ball motion can be increased or decreased according to slider action.

## Future Improvements:

For future improvements, movement of babies could be improved, movement of ball could be made more random and more unpredictable. At present scenario developed, movement of baby is not precised. And also, baby movement could be set limitation to not to go beyond the boundaries. Ball movement can be made more random so that ball can move in any direction not only diagonal but in random direction like up, down and so on. Level can be introduced so that game could be more realistic.