**Revised Specification MW-Wed Team 4**

* create and edit squares, circles triangles, & bumpers
* create and edit flippers
* connect the actions to the flippers and bumpers to triggers
* use keyboard or mouse events to trigger actions within the game using the Gizmo-ball controller listener classes
* create physics loop identifying the sequence of events happening within each clock tick of the game
* research the Java geometry classes to implement the collision detection, and how the ball will react depending on where it collides within the game, possibly changing velocity and direction depending on collision detection
* decide on the timer within the game, it will likely be every 5 Milliseconds, and if the ball collides the update method will be called to repaint the GUI
* implement the use case scenarios, including pre and post conditions as well as alternative paths and the course of events the user takes to implement each identified use case in the system
* create class diagram: identify class relationships, and which classes belong in the model, view and controller of the system, and the purpose of each class within the system
* assign tasks to members using project management plan, and add milestones and set realistic targets
* complete Gantt chart along with the project plan to diagrammatically display tasks and milestones, displaying the key dates and deadlines of the project
* brainstorm ideas of how the GUI should look, and build a prototype using a GUI builder, this will be done using a RAD approach to make a mock-up of how the interface should look
* describe the triggering system connecting triggers to actions, i.e. user key events, flipper actions, collisions and timer of ball movement
* screen shots of editor for build and play
* Identify and implement exactly which buttons should be included in the build mode of the game, these will also be the main the listener classes of the controller package. This will also let the user control how the interface of the game-play should look in run mode
* identify if the ball itself is a Gizmo, or if it’s implemented as a circle in Java
* let the user alter the friction and gravity of the game, but decide on how this will be implemented, i.e. will it be a text box for the user to enter numerical values or will we include an adjustable slider to change the setting?
* create Zero-Radius circles placed at the end of lines included in the GUI, this will make the collisions look smoother as the ball will not travel through the corners of Gizmo’s
* for the next submission in week 6 we will have identified the final class diagram, as well as the methods and attributes within each class and any changes made to the design
* create the validation testing strategy, identifying if the system meets the initial functional requirements, i.e. the system does what it is supposed to do
* design the J Unit testing strategy testing the main test cases, and if the system meets the desired expected output
* implement the 4 required prototypes:

1) Flipper triggering using mouse events to move the flipper up and down

2) The absorber ball motion, gravity and friction of the system

3) Ball collisions with bumpers and walls

4) Loading files from previously saved state

* implement any realistic Java assertions and include method specifications