Team 07
Project 03
EECS 448
Dr. Johnson
10/25/20

Design Patterns

1. Memento
    a. The initial states of the *Brickset*, *Paddle*, *Ball*, and *PlayerStatus* were externalized in their corresponding *reset()* method. Thus, when a game is restarted, the states of these objects are restored to their initial states by calling the *reset()* methods of each.
2. Chain of responsibility
    a. To update and draw all game objects for each animation frame, we placed them in an array of *gameObjects,* and iterated through each to call their *update()* methods. The *Ball* and *PlayerStatus* objects each encapsulated other objects such as the *Aim* object for the *Ball* and the *Lives* and *Score* objects for the *PlayerStatus*. Thus, when the *update()* methods for *Ball* and *PlayerStatus* are called, then they in turn call the *update() methods* for the *Aim* and *Lives* and *Score* objects. Furthermore, the *Lives* object encapsulates an array of *Life* objects, which it updates when its *update()* method is called. Thus, an *update()* call to the *gameObjects* initiates two chains of *update()* calls - one starting with the *Ball* and other with *PlayerStatus*.
3. Observer or Publish/Subscribe
    a. We used the *window* object to create observers that observe whether a specific key was pressed, the mouse was moved, or the window was resized. Each of these observers then updated data members of other objects. An example is the window resizing observer, where a window resize updates the size of the canvas. Many of the *gameObjects* depend on the canvas size, and so each are updated with the new size when their *update()* methods are called. Thus, the *Ball*, *Paddle*, *PlayerStatus*, *Aim*, and *Bricks*et objects all resize when the *window* is resized.