PROJECT 5 REPORT

In this project, the existing architecture of “ClickCounter” and “Dots” was used to implement a game wherein the user can click on vulnerable monsters to increase their score. Vulnerable monsters are represented by yellow dots while invulnerable monsters (i.e. monsters that will not increase the score, and will not be removed from the board) are green. This project makes use of the MVA (Model-View-Adapter) architectural pattern. MVA works better for the design than MVC (Model-View-Controller), as there was no need for the model and view to interact. Other design decisions from ClickCounter were implemented, such as a listener. This was done so that the monsters could move with each clock tick (one move every second).

SOLID design principles were applied to the project. Each class was given a single responsibility (SRP). The game was also implemented to be open for extension but not modification. There can be other features implemented without having to modify any of the source code. The liskov substitution principle was used as well. Concurrency was implemented so that the user could interact with the game while the game continued to move the monsters. Using concurrency allowed the user to click on the monsters to increase the score without disrupting the process of the monsters moving and changing vulnerability states. Another concurrency issue was separating the UI thread from the timer and monsters. If they were on the same thread, it would slow the entire application down. This allowed complete separation of the two and resulted in a fast and efficient game.

The primary testing method was user testing of game protocol due to the program’s random nature, with automated testing for basic commands. Before randomization was implemented, testing was done on a static playing board to assure that the Monsters could be added and removed. Once the randomization occurred, the testing (solution) had to find a Monster’s program location, compute its physical location, then simulate a user’s touch in that location. Testing for movement was easier as two mock boards could be generated and compared after a tick event. Clock testing was implemented from a previous timer project.