The journey of project 4 was pretty challenging and interesting. Initially when we saw the description of project 4, looks like really difficult to start. However, thanks to Professor Konstantin who posted UI demo app for initial start, it made our work easy in order to complete this project. As we learned MVA architecture in project 3, the use of MVC architecture in this project 4 was a great learning experience for us and we learned comparison between this two architecture.

In MVA architecture, the Adapter is strictly mediate between Model and View. The Adapter holds a pointer both to the Model and to the View and directly calls methods on both. At the same time, it attaches itself as a listener both to the Model and to the View in order to receive events. Moreover, the Adapter is entirely responsible for keeping the Model and the View in sync; the Model and View do both relatively know nothing about the other.

Whereas in case of MVC architecture, Model, View and Controller are directly connected and communicate with each other. Here the Model part directly manages the data, logic and rules of the application. A View can be any output representation of information, such as a chart or a diagram. The third part, the Controller, accepts input and converts it to commands for the Model or View. In this project, our Model part implements Monster movements, its states, Monster creation, death etc. The View part shows layout and graphical representation of this app such as grid layout. The Controller part illustrates touch interface of this project.

As the progress going on in this project, we had a trouble to achieve SOLID principle in order to increase our project quality. We think we were able to achieve the "open/closed principle" (adding new functionality with minimal changes to existing code).

The most challenging on implementing this game was about treating concurrency. In the final version we are using a ScheduledThreadPoolExecutor in which we add the two tasks of each monster to be executed by 2 threads: One of these tasks move the monster and the other one changes its vulnerability state. As our monsters and cells are mutable, it is important synchronize each access to these instances to avoid race condition.

Besides that, organize the project in a well defined structure, respecting the good practices and design patterns we have learnt throughout the course also took us a long time and we realized that it is not a simple task. Much more could be improved.