# Architectural Document

This document presents the first milestone i.e. architectural design of a Tower Defense Game built in Java by adapting to MVC Architecture. The tower defense game is developed using many agent objects like creatures, towers, Firing bullets and the map which acts as an arena to represent all these objects. All these Objects are used in the logic of the game. Quick usage and communication with the player and an engine in behind to understand the application itself; since interdependencies among all of the modules causes resilient ripple effects at every time change is made anywhere. High coupling makes classes hard or difficult to reuse because of their dependency on other classes. In process of adding new data views often requires frequent alterations in implementation part and game logic code, which then needs maintenance in various places of code. Mainly, bearing in mind the game being developed, it is anticipated that there will be lots of agents that have their own attributes and operations; logic to preserve the relationship between these agents and the situation; and basically all changes in the game should be reflected to the player. Thus, an architectural pattern selection is needed to choice all these concerns.

The Model-View-Controller (MVC) design pattern solves these problems by decoupling agents, game logic, and graphical presentation and user interaction. We developed the system in a way that agent objects such as towers, creatures, firing bullets, and the map are in model package that represent the domain specific objects and also Model is responsible for storing the vital information of these objects which is needed by both Controller and Model package classes. And we implemented in a way that the responsibility of controlling, managing and updating of these agent objects was done by Controller Package. Their changes, although generates the domain knowledge, but they are not directly controlled by themselves nor the player. Game logic classes which are in controller part are responsible for that.

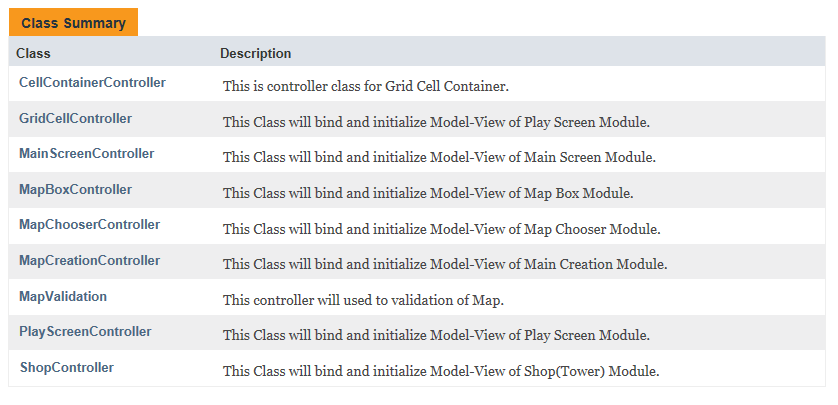
The controller package consist of the manager classes; screen manager, game logic, tower manager, creature manager. Also, inputs and outputs that influence the game logic is also part of controller classes. These classes create the game logic, with a certain hierarchy amongst themselves. The controlling issues can be concise as taking inputs either directly or from view classes, changing them to data that is directed to be managed and modified in the model and reflect these changes on the view, to the user.

The View Package has the classes that are responsible for the interactions between the game and the user.

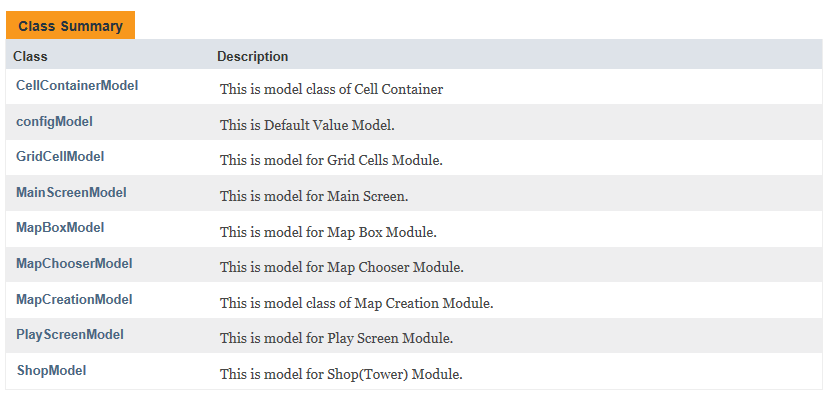
These interactions are mainly dealt in interface classes; which consist of all screen classes, panel and label components. This View package classes uses most of the information from the Model package classes, though is usually controlled and directs interactions to the controller. While implementation using MVC we separated the functions of View package (GUI) and Model package (game data) which is considered to be one of the good engineering practice.

**Class Description:**

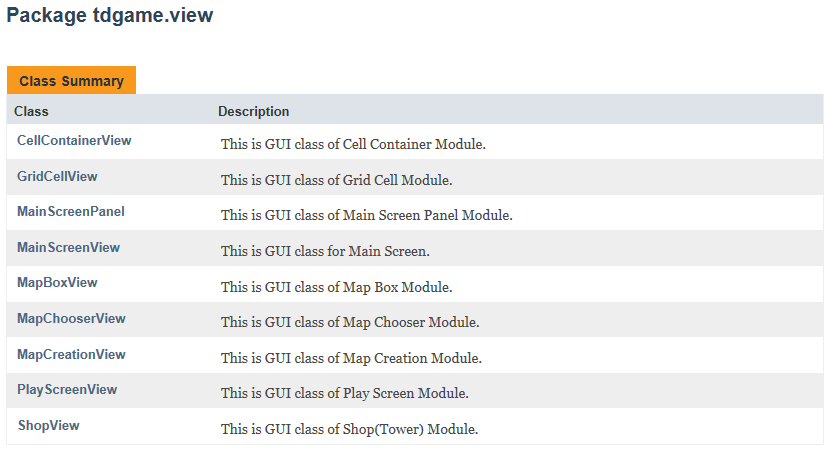
* **tdgame.controller:**



* **tdgame.model:**

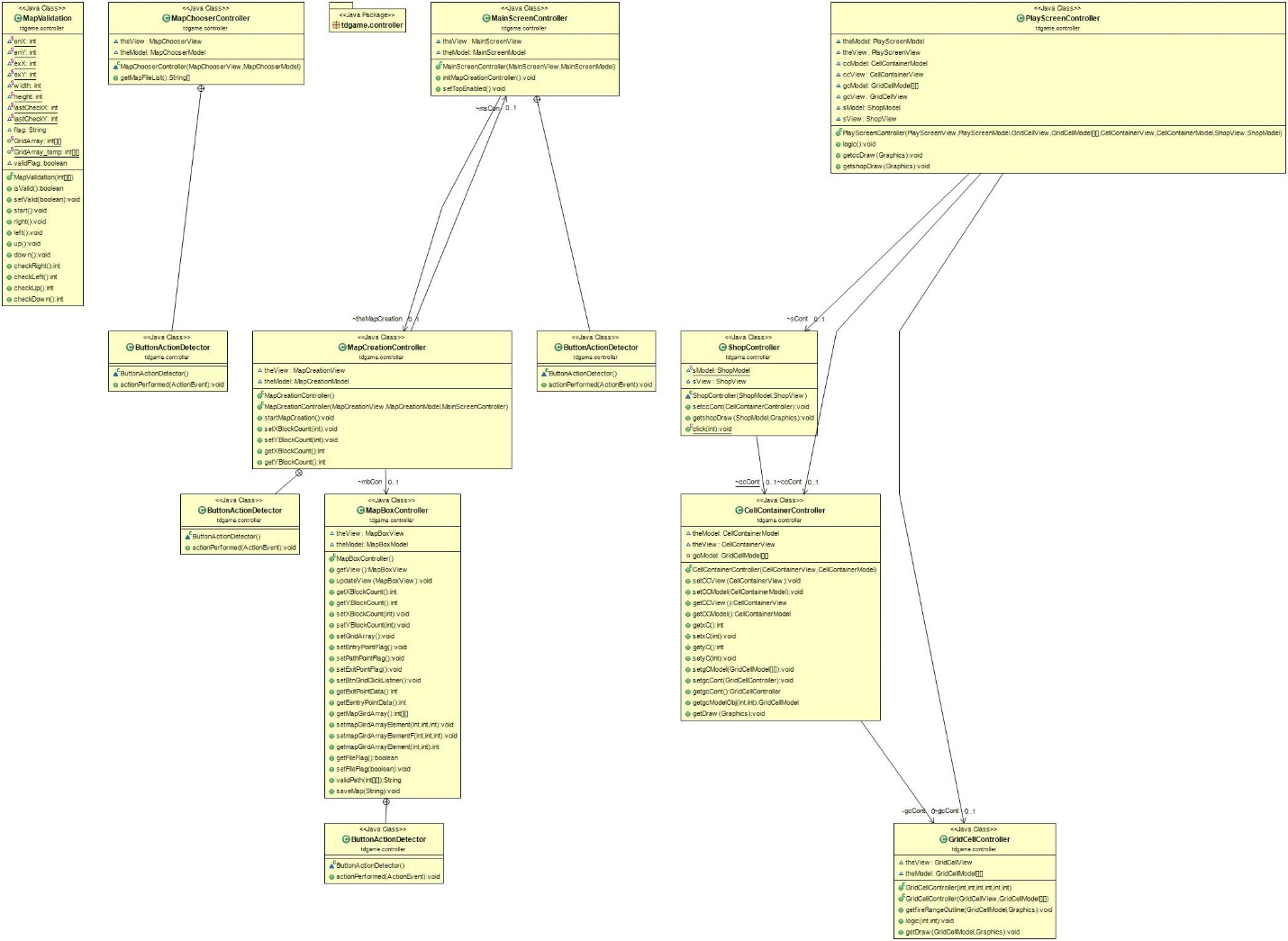


* **tdgame.view:**

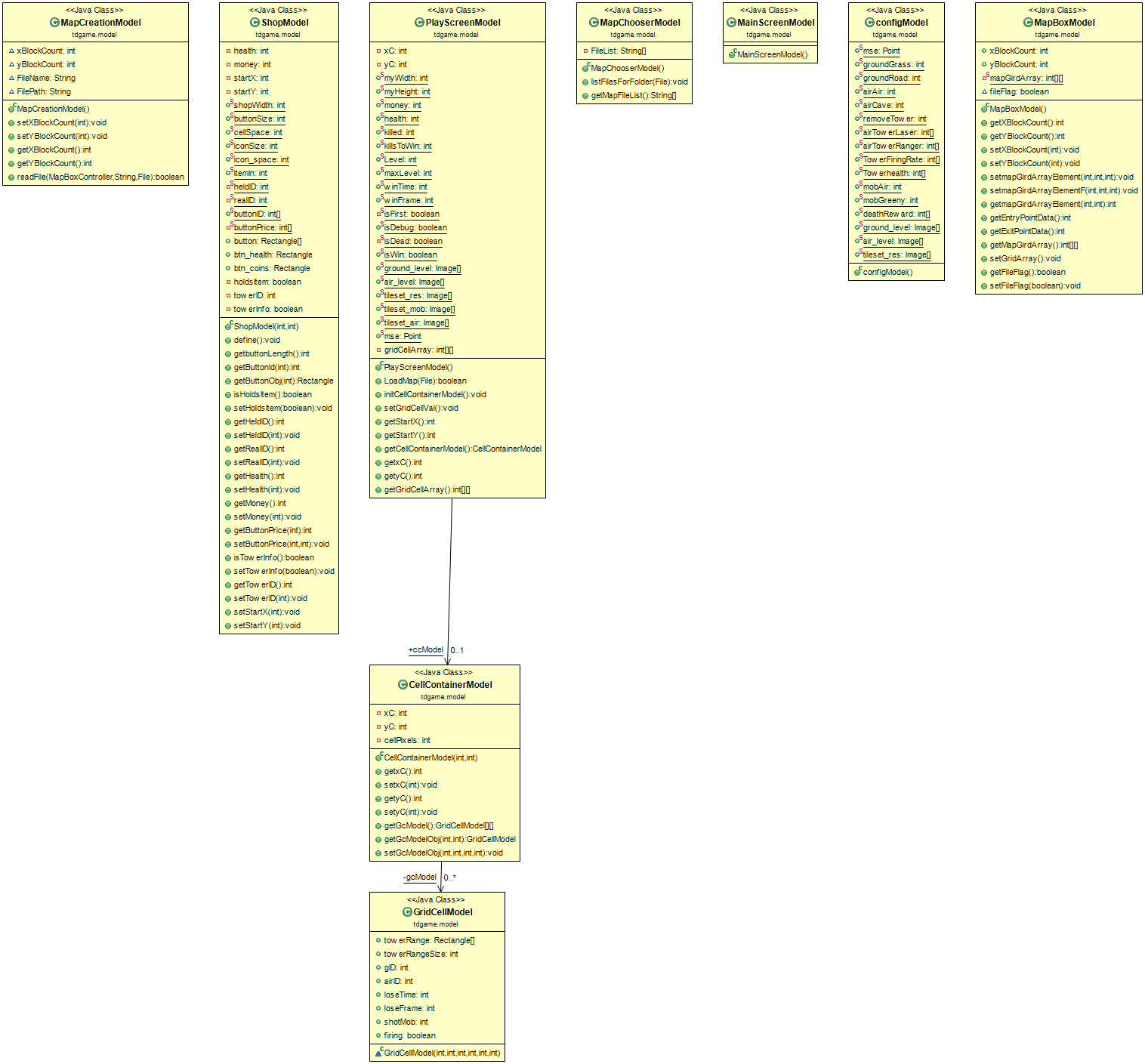


**UML DIAGRAMS:**

**CONTROLLER:**

****

**MODEL:**

****

**VIEW:**

