**Pattern-Based Software Development**

**Team ID: B1**

**Group Assessment**

**Project Specification:**

In this project we were asked to create a Java application from scratch in order to learn how to correctly apply software design patterns and their practical implications. However, finding a good domain for an application that would be able to utilise at least eight different design patterns proved to be a more challenging task than previously thought. We had to put thought into the practical use cases of the design patterns and the advantages that come with them before deciding the best candidate for our project. At the end, the team decided to develop a variation of the popular arcade game Pacman for desktop.

The game follows the typical Pacman structure. There is a board that resembles a maze that contains simple dots, fruit enhancements, ghosts and of course, the Pacman. The user can select on run time between three different board sizes with unique layout, different number of ghosts and different number of fruit enhancements. As the game progresses and the player collects more points, based on the board size, the difficulty of the game will change. There is a total of three different game levels which change the behaviour of the ghosts. In the first level, the ghosts move completely random, in the second level the ghosts calculate the Euclidean distance between their current location on the grid and Pacman’s and lastly, in the last level (level 3) the ghosts move based on the Manhattan distance metric which is specifically designed to work on 2D grids. Moreover, the fruit enhancements are self-explanatory. There are four different kinds of fruits which offer unique bonuses to the gameplay. One gives extra points, another stops the ghosts from moving for a fixed amount of amount, one other gives the Pacman immortality and now it is able to pass through and “eat” ghosts without dying and the last one is a composite fruit which offers a random combination of two of the previously mentioned functionalities.

In order to complete the game, you have to complete all three levels of difficulty and then a success screen is shown, and the game is terminated. If you manage to lose all three of your available lives before completing the game, then a screen is shown prompting you to return to the game board selection in order to start over again.

Lastly the game can be played by either using two different sets of keys (arrows or wasd) which move the Pacman to the respective direction or by using the mouse where the left and right clicks as well as scrolling move the Pacman to the right direction.

**Pattern Specification:**

For the project purposes we had to use the following six patterns:

1. Model-View-Controller
2. Composite
3. Strategy
4. Factory
5. Singleton
6. Adapter

Also, we have included in our implementation the following two patterns:

1. Iterator
2. State

As an architectural pattern, Model-View-Controller stands at the core of the implementation which there are three different packages, the model, view, controller which communicate through the controller without any direct interaction between the view and the model. The rest of the software design patterns were used as follows and will be more thoroughly described in the rest of the report:

1. Composite – fruit functionalities and their composition in the composite fruit
2. Strategy – based on the level of the game, the ghosts move by the use of a different movement algorithm
3. Factory – different board sizes with different attributes can only be initialized through the factory to make it easier and correct every time
4. Singleton – singleton makes sure only one factory is created during runtime, and every other need of instantiation comes through the same, initial factory object
5. Adapter – combines the keyboard and mouse interfaces for user input
6. Iterator – iterate through the list of ghosts in order to move them all at the same time, every time
7. State – Different views comprise the different state of the game. The states are self-explanatory: Selection, Play, GameOver, Success