Bilkent University

Department of Computer Engineering

Object-Oriented Software Engineering Course Project

CS319 Course Project: Virion

Final Report

Mert INAN, Ulugbek IRMATOV, Irmak YILMAZ

Supervisor: Uğur DOĞRUSÖZ

Final Report

Dec 24, 2016

This report is submitted to GitHub in partial fulfillment of the requirements of the Object-Oriented Software Engineering course CS319.

Contents

Final Report

CS319 Course Project: Virion

1 Introduction

In this report, the final remarks will be evaluated after the implementation process is complete, despite having incomplete points. The final object model is also discussed after some changes during the implementation process.

2 Implementation Process

The implementation part of the project was one of the challenging aspects of it, as the user interface elements started to take most of the efforts. The initial process started by writing the code for the highest level classes such as the game engine. However, we observed that this approach leaves a lot of empty methods; hence, we changed the strategy to a bottom-up one in which the lowest-level classes are written first, such as the game components.

In order to create the user interface screens more efficiently, the GUI form provided by the IntelliJ IDEA® was used. Testing of each user interface class and form were done individually, and testing of important methods of other classes were tested using main methods written in them.

3 Major Changes

The major change in our design is the Launcher class. During the implementation, we realized that there should be a class that connects user interface and controller classes. Launcher class has the main method that instantiates controller classes and UI components. Also, Launcher initiates the jar file. The updated UML Class Diagrams are given in the Figures 1-7 in the following pages which are done using Visual Paradigm Reverse Java Code© functionality.

Other changes in Design were in user interface part. We learned more about the UI during implementation. We added high score panel. We used forms provided by IntelliJ. So, each class is accompanied by form. Main methods were added panel classes in UI and new methods were auto generated by IntelliJ. Some method changes are as follows:

- **GameEngine Class : calculateDurability(int virusID, int proteinID) Method**
  In this method, the aim is to calculate durability according to the phases of the game. Like pre infection, post infection phases. However, it cannot be detected in a right way and since the implementation of the rest of the program could not be finished, the calculation of the durability according to the phases cannot be completed in all. There are boolean variables which are pre and post infection but they are only used in this method so the implementation cannot be done correctly for this method.
- **HighScoreManager Class: isHighScore(int score) Method**
  In this method, the aim is to return a boolean which is true if the integer parameter is the highest integer in the high score list document, else it is false. In this method, the document has strings and integers so ArrayList type should be string. However, in the implementation ArrayList type is made integer to compare the score integer.

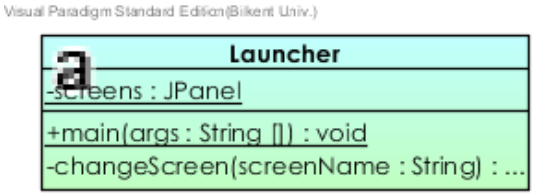

Figure 1 This is the updated Launcher class.

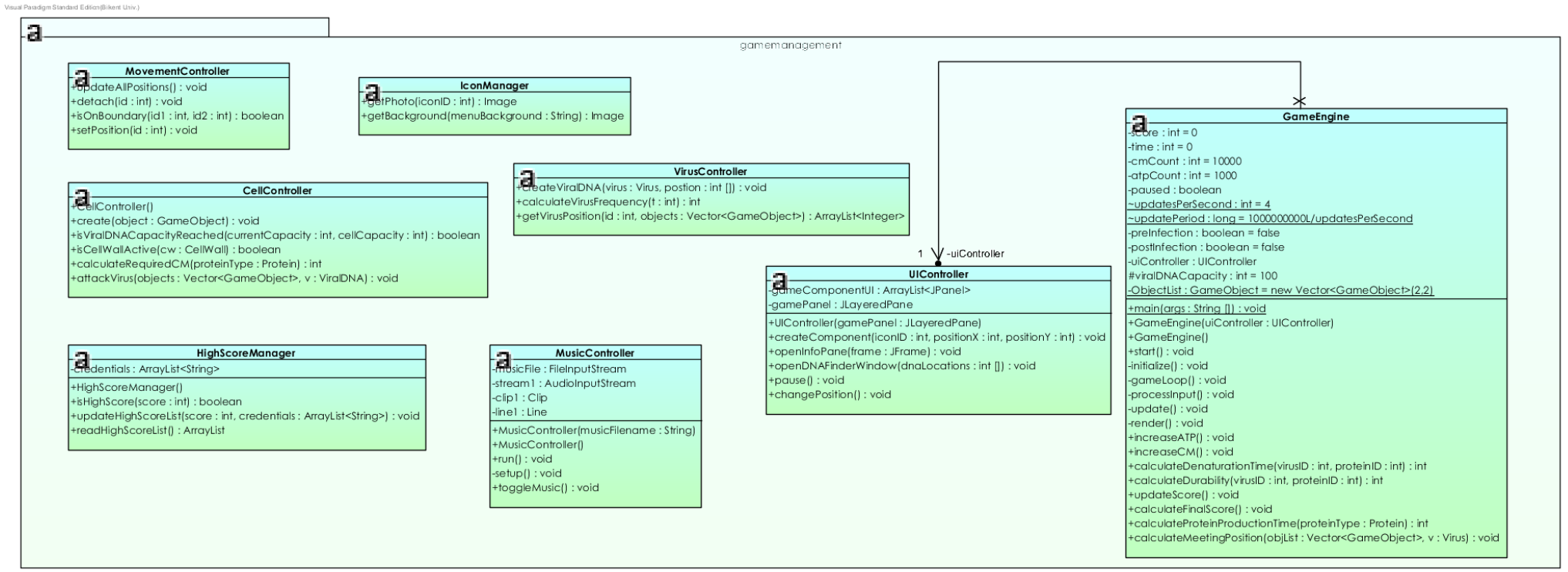

Figure 2 This is the class-package combined diagram of game management subsystem. Most of the classes in here are the same with the previous model.

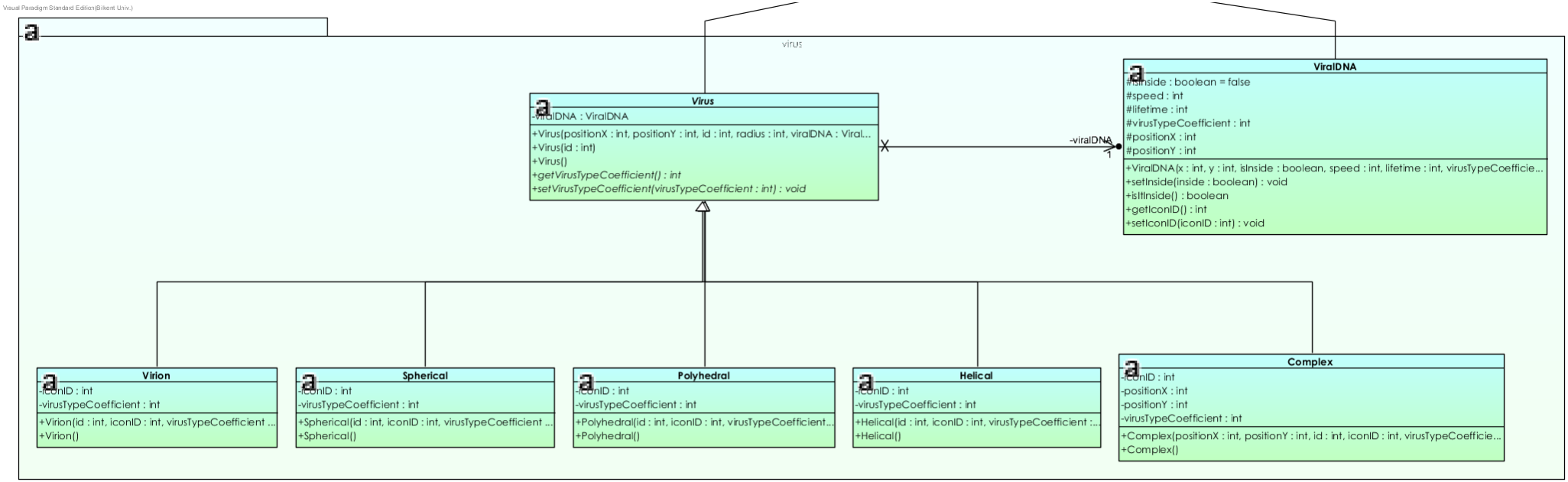

Figure 3 This diagram shows the updated classes of the virus package.

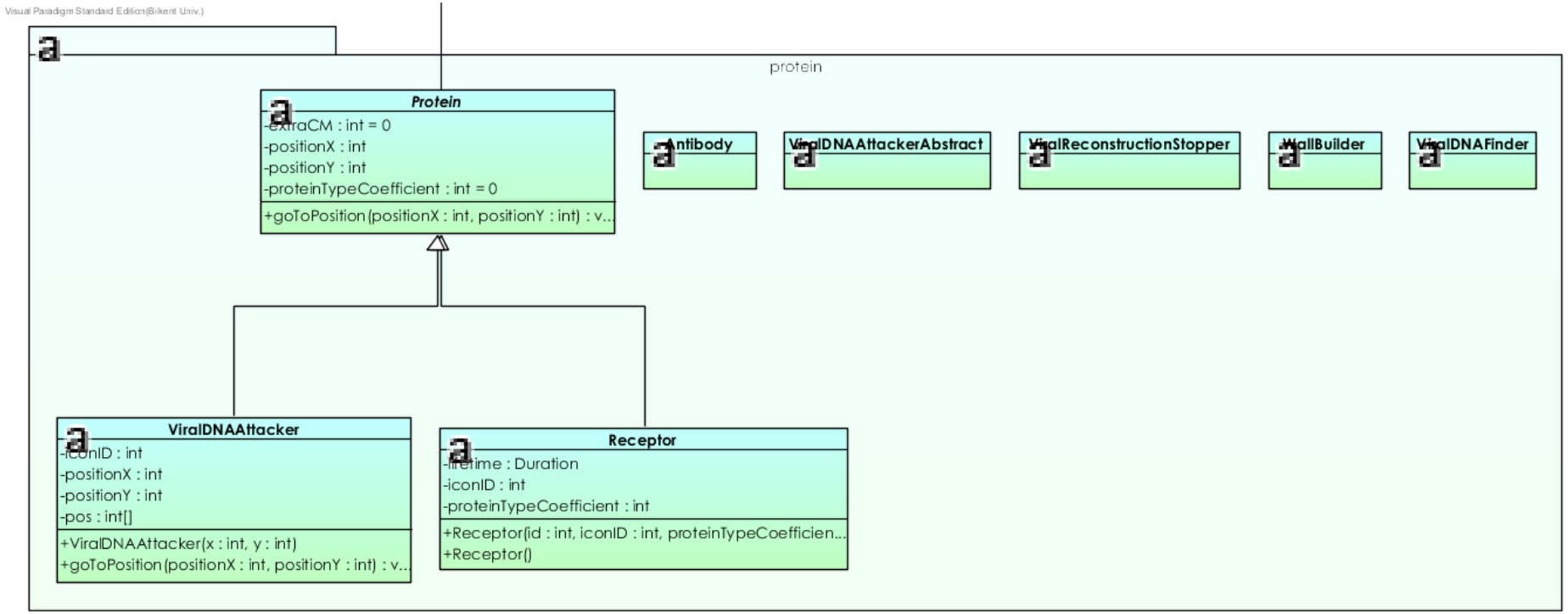

Figure 4 This is the diagram for the protein family classes.

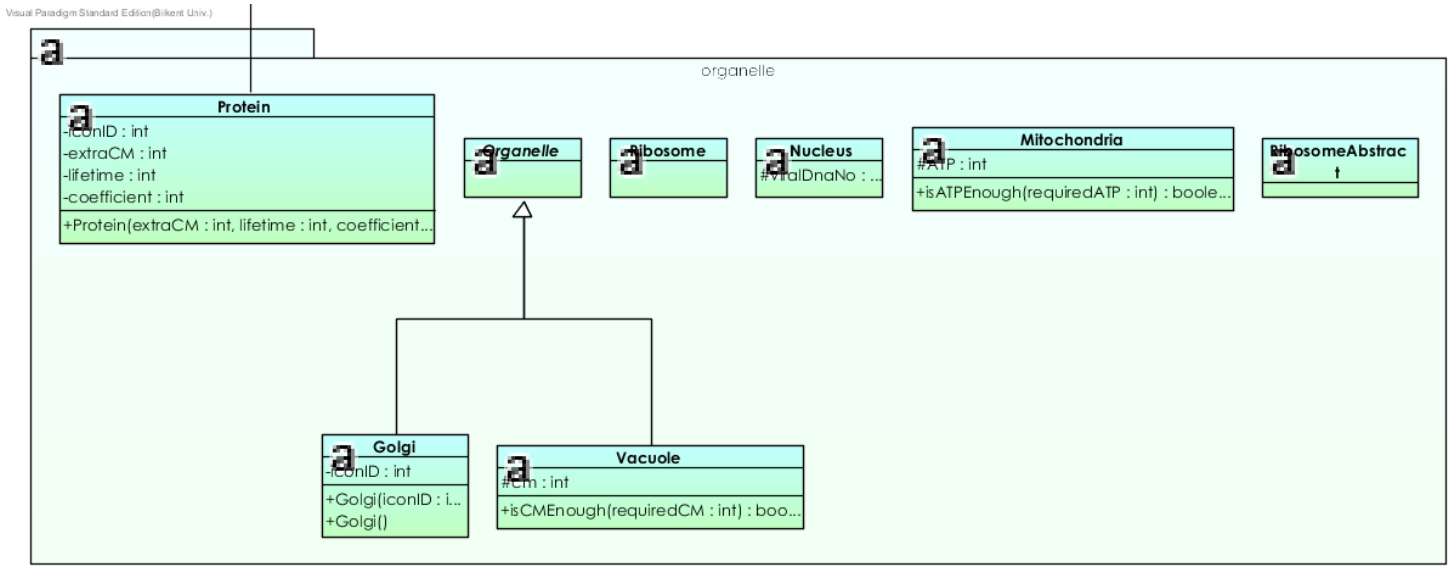

Figure 5 This is the class diagram for the organelle family.

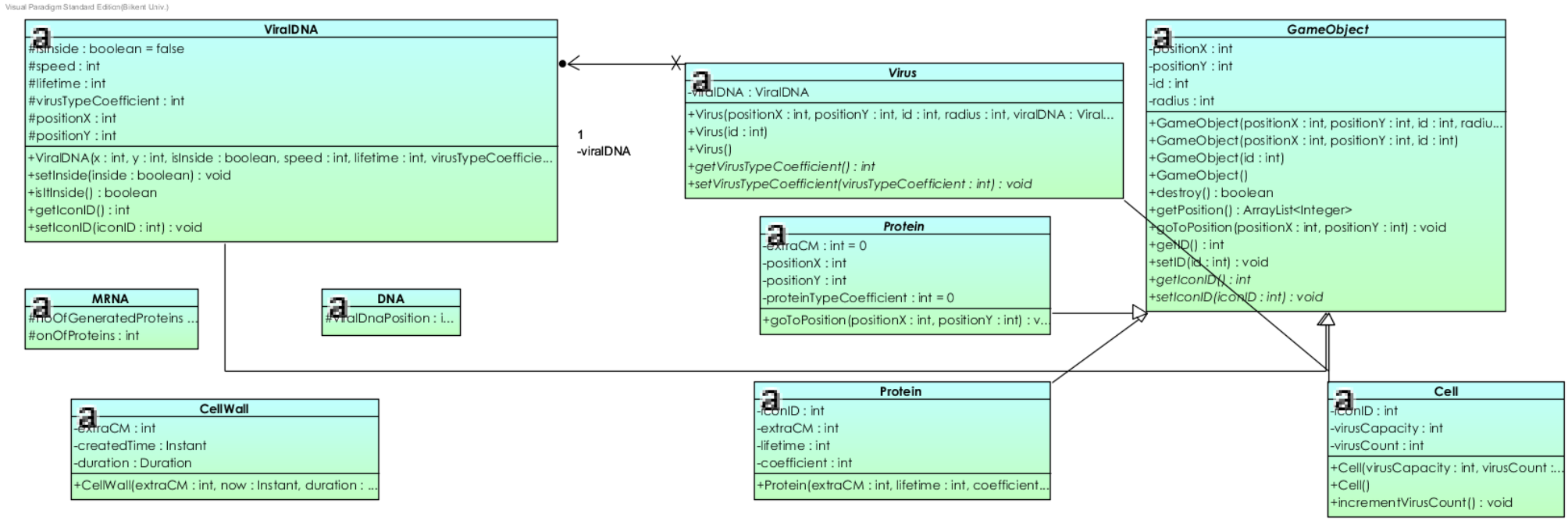

Figure 6 This is the UML diagram that shows the final interactions between the game objects and the abstract GameObject class.

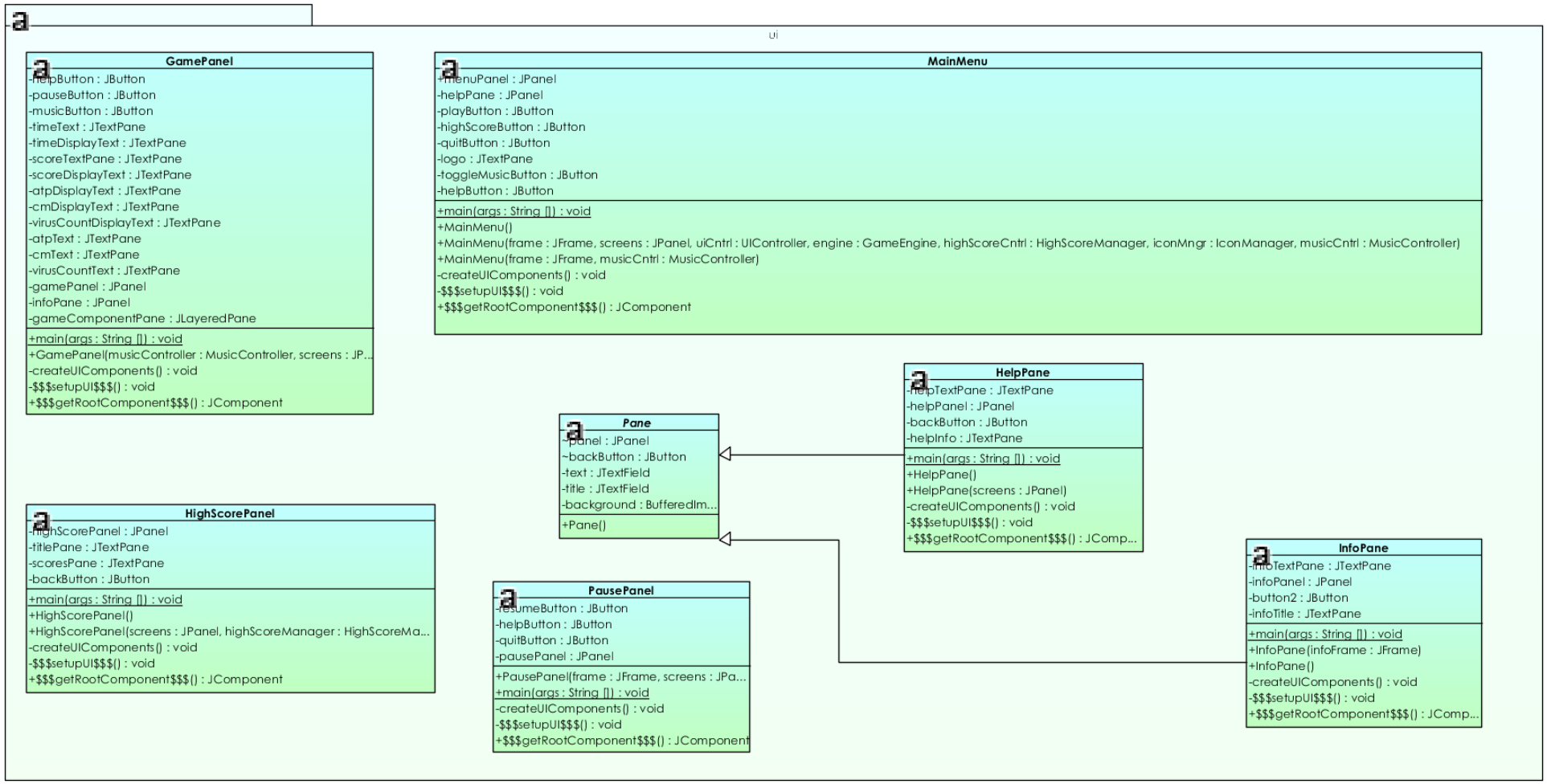

Figure 7 This figure shows the user interface classes, which had the major changes since the previous object model.

4 Status of Implementation

Most of the user interface classes are implemented, yet the incomplete parts of the implementation are in GameEngine and MovementController classes and their connections with the user interface classes. The main incomplete function is gameLoop. As it is one of the key methods, its absence renders the game unable to show the user interface of it.

Following methods of controller classes related to game loop are not complete.

- gameLoop() – what happens during game play.
- changePosition() – moves UI components during game play.
- getMegaOut() – gets the protein out after it becomes mega.
- checkRibosomes() – checks if the selected ribosomes are available.

Also, methods used in post infection phase couldn't get completed, as it relied on the understanding of the gameLoop:

- areAllViralDNAFound() - that tells whether user found all the viral DNAs in the post infection phase.
- openDNAFinderWindow() – opens new window after viral DNA enter nucleus.
- stopReconstruction() – stops the virus reconstruction in the post infection phase.
- sendFinder() – send finder protein to find viral DNAs in the nucleus.
- randomizeDNALocation()- randomizes the location of viral DNAs in nucleus.
- isAntibodyPresent() – tells if antibody is present after infection.

5 Appendix

In this section, relevant instructions and simple tutorials on the use and installation of the Virion system are given.

5.1 Installation

The final software is a ".jar" file, which increases its operating system independence and makes it more portable. In order to download this jar file, please visit: https://drive.google.com/file/d/0B-qaI9GmbW_1NWNGeVlQeThDQ00/view?usp=sharing

Once the software is downloaded, the user can start using it in no time by double-clicking the jar file and running it using Java Binary. The minimum requirement to run this software is JDK 1.8.

5.2 Simple User Guide

When the user double-clicks the jar file, it will open directly and the main menu will appear. The choices are very straight-forward and easy-to-grasp. Player can move through the menu items and select one of them by clicking on it. Play option, opens the main game screen and all the other options show information related to them if the user hovers over the button. The music logo toggles the music and the question mark logo opens the help screen.

When the player selects to play the game, the game will load certain elements by itself. The scores and relevant game information are displayed on the top of the screen. There are three buttons next to these information elements as well. These buttons are the regular music toggling and help buttons, in addition to the pause button, which opens the pause menu from which the user can end the game early.

During gameplay, the user can click on the game components and they will respond to them by either opening a dialog box or waiting for another selection.

By producing proteins which can be done by clicking on cellular organelles, the user can gain score and by defeating the viruses coming from the intracellular space, they will try to stop the viruses killing the cell. However, in the end, the cell always dies. The main is to try to set a high score by delaying the death of the cell.