Introduction:

At the beginning of this documentation, I need to mention that my teammate dropped the course without doing anything special, and I did all the jobs by myself. This documentation is organized as follows: first, each class and its property is being briefly explained, and then the fellow of the program is being explained in the second part

1- Key Classes of the Game:

There are four key classes in this project, GameRunner class, EventHandler class, Statistics, and Galaxy class, each of which play its own role in the game.

Galaxy Class:

It is responsible for storing all the ships, and star systems inside the game and their properties inside itself. In other words, galaxy is a container which somehow plays the role of the main data structure for us. At the heart of the galaxy class, there are two vectors defined as private members, galaxy_ships, and galaxy_star. First one is used to store the cargo ships and their properties such as location and engine type to name but two, and the second one is used to store the start systems and their attributes. This class has been inherited from the iGalaxy, but it has some extra member function such has getGalaxyStar(), which outputs the vector of star systems.

EventHandler class:

This class, as its name suggests, is designed to manage the logical parts of the events that happens inside the game. In other words, it is responsible to report the events, which happens in the game such as creation of a new cargo ship, to the main UI of the program. It is dose not handle the graphic event.

GameRunner Class:

This class, which entirely implemented by the course staff, has the responsibility to run the game by using both Galaxy and EventHandler classes. It has three main functions spawnedShip( ), creatAction( ), and doAction( ) which are used to create cargo ships in galaxy and make them to do some action. it also connect the events inside the program connected to EventHandler.

Statistics class:

This class has been inherted from iStatistics class. It is an interface for keeping the track of game states. It contains a number of the simple functions which change or return some private variables. We test these function in StatisticsTest class.

StatisticsTest class:

This class contains a private variable name test_statistics_ which is a shared pointer to Statistics class mentioned above. It also contains some member function by which we test all the Statistics class member functions. In this implementation, I used QCompare( ) to check all member functions. I also used QVERIFY_EXEPTION_THROWN( ) for exception handling in reducedPoints( ) and reducedCredits( ).

MainWindow class:

MainWindow is responsible for creation of game menu. It is consists of two buttons: Play Game, and Quite. By clicking the Play Game button, the main window become hidden and the dialog which is responsible for showing the game appears, and if you press the Quite button the MainWindow object will be closed without anything happens. I used the signal and slots in the designer form to build these properties. In addition, this class has a setter function named setDialog( ) to take the address of the dialog created in the main function. This pointer to the dialog, consequently, is being used in a private slot to show the playing window of the game. In other words, dialog dose not use show( ) function in them main function.

Dialog class:

This is the main playing window for the game. It has several function inside it. In its constructor, it sets up Ui, creates a QGraphicsScene and adds it to a graphicViewer.

It has member function called setUiInfo( ) to transfer objects from Galaxy (the container) and GameRunner.

It has a function named implementUI( ) which is used to first draw start systems as yellow ellipses in the view and use GameRunner object to creates 30 random cargo ships in galaxy and sets a timer for them. The timer becomes connected to a private slot moveCargoShip which is defined inside the Dialog class.

Dialog has a method named drawNewShip( ). This method is being called by the EventHandler object to draw some red squares as cargo ships in the graphicView.

2- Fellow of the Program:

The program starts by main function in the Main.cc file. It first creates three objects of EventHandler, Galaxy, and GameRunner. Then, it creates objects of MainWindow, and Dialog,

sets up the dialog and shows the MainWindow object. In this function, EventHandler object, handler, receives a pointer from the Dialog to establish the connection. In the Dialog, 30 ships are being created by GameRunner, and handler calls the function drawNewShip( ) to make some red squares.

I did not managed to move ships in this implementation, but I wrote a slot function to call createAction( ) and doAction( ) as the timer is being timed out continuously.