CO871 Assessment 2 – Report

## Introduction

To do this assessment, I took my own version of the Zuul project, since I designed it in a way that it could easily handle a Graphical User Interface.

That is the reason why I did not have to modify much of my previous code, the only part that I modified are mostly in the Room and Item class where I added a visual representation of those classes, as a String to load an Image. And some functions where renamed to get them to work with reflection.

## What did I add?

## List of: Exits, Room’s inventory, Player’s inventory, Actions, Players.

Since my Graphical User Interface contains 5 lists on where the user can click, I thought that, instead of creating 5 classes that would basically do the same thing but not show the same button, I could create a single class that is supposed to handle every list.

To do so I created a template class (ListOfClickableObjects) that inherit the JavaFx ScrollPane and that can receive its orientation (through a templated Box argument, that can either be an HBox or an VBox) and it’s data stream argument (through another templated Iterable argument).

This list’s goal is to know what the player wants to do, by clicking on the “Go” button it can tell its observers that the player wants to do the “Go” action.

To create this class I used the Model View Controller design pattern, because, since it is a huge class that handle a lot of logic, I thought it would be easier to read, understand and implement.

## The main view, RoomInfos.

This class’s main goal is to connect every list between each other and to add the other views that show the current state of the game in one BorderPane.

This class is also responsible for the refreshing of the view every time a new event occur, like if a player quit or if an user action have a consequence on the view (for an example: taking an object from a room will update the inventory lists).

Like ListOfClickableObjects, RoomInfos handles a lot of logic and functionalities, therefore this class implement the Model View Controller design pattern.

## RoomDescription and RoomRepresentation.

I added those two simple classes to handle the visual representation of the room as an image, and its name and description at the top of the screen. They are updated by RoomInfos every time the room changes.

## WorldLoader

To handle the world loading and the world loading options I created two classes: MainMenu and WorldChooserParametersForm.

MainMenu is responsible for creating the JavaFx’s FileChooser and to call WorldChooserParametersForm when the user choose a file, so it can asks the user about what options does he wants to create the game: Removing every room without exits, adding an item in every room, etc…

## Tutorial

This simple class handle a WebView that open a YouTube video to help the user understand how to play the ZuulGame.

## GraphicalUserInterfaceView

I had to replace my previous implementation of the Game (CommandLineInterfaceView) with the new GraphicalUserInterfaceView.

This class create the main Stage of the application and load the main RoomInfo view. And then the game starts.

## RoomParser

I already had a RoomParser that was handling JSON files, but I had to add another class that can handle another kind of world file as described in the subject. Now the RoomParser class detect the kind of file, and use the right implementation of the class with reflection in order to load the file correctly.