# IPPO assignment 2

Yuwen Heng, UUN: s1572869, Exam Number: B083523

23 November 2019

## 1 Design Changes

Several changes are made based on part A. The class **Locations** is renamed to **Room** and **Direction** class is renamed to **Wall** to match with the design introduced in Lecture 3. The class **Map** is renamed to **WorldMap** and is designed to initialise, store and get Room objects by name. The **Item** class is re-designed to be an abstract class to support multiple kinds of objects. The subclass **ImageItem** is added to represent the items with an image.

To present the class definition clearly, packages are used to organise related classes. In order to load the model from an external JSON file, the library **Gson** is used since it is easy to override the deserialise method and parse the JSON file in the same hierarchy as the objects: WorldMap → Room → Wall → Item. Since the Item class is designed in the way that items are only present in the current direction, the JSON format is edited to place items with particular direction. To comply with the MVC design pattern, a class named **Model** is also created on top of the **WorldMap** and **Player** objects.

The classes which are responsible for transforming the command acquired from user interface, in the form of text input, to the methods which are delegated by the **Model** class, are grouped together in the **command** package. The abstract **Command** class defines the structure of the command, which is combined with two words. The first word decides which Command subclass object should fire their "execute" method. The second word fulfils the command. For example, the first word "Pickup" means that the PickItemCommand object should do the job, and the second word "ballon" means that ballon item should be picked up. In this way, the user interface can be easily extended to command line interface or using other GUI library. These classes are inspired by the textbook "Objects First with Java", Chapter 8.

The CacheProxy class and LocalService class are adapted from assignment 1 to fetch images from resources folder and can be extended to support web service.

To explain the responsibilities for main classes, the class diagram is draw in Figure 1. The classes related to parse the JSON file is shown in Figure 2.

## 2 Additional Design

In this exercise, multiple players and multiple user interfaces are well prepared. To support multiple players in the application, a Map object with player name and Player object pair is created in the **Model** class. A field named **currentPlayer** is defined to trace which player we are playing with. When a Command object is executed, the Model object will choose the right player to deal with. To switch between Player objects, a Command class named **SwitchPlayerCommand** is designed and the command word "Switch player" is attached to a Menu named "Player". The WorldMap object is passed to each Player object to support multiple players in a common room.

The command word is defined as String and the only event that the interface can fire is the "select" method in AppController object. In this way, View class can be extended as we want. Observer pattern is not used here since the application should be simple and won't cost much time to redraw everything. A simple solution is used to support multiple views by adding View objects to a Set object and update every View objects in the Set when an interaction happens. However, since time limits, no extra View object is implemented.

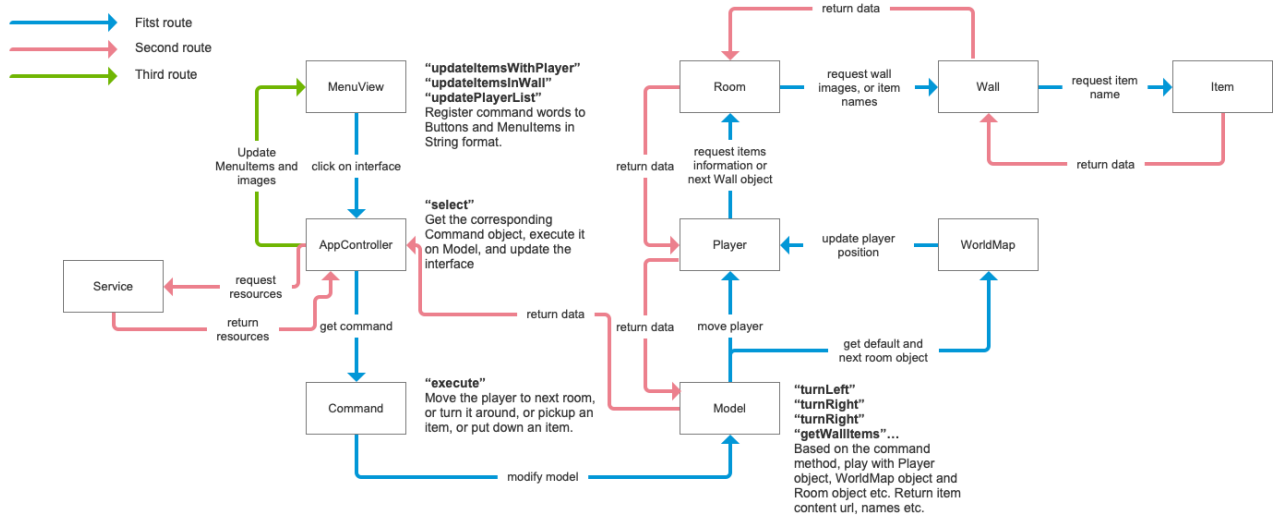


Figure 1: Main classes to run the application (MVC model)

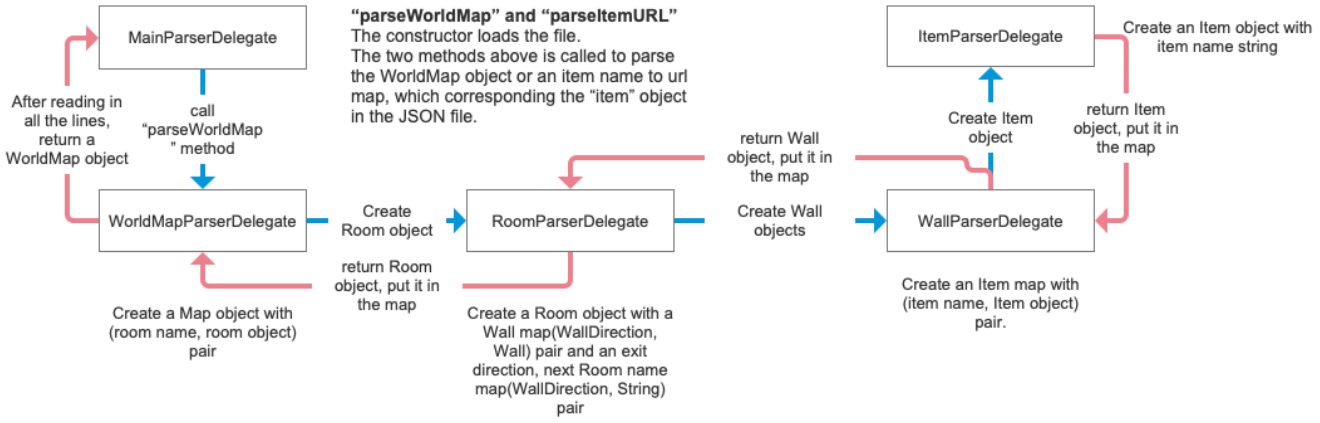


Figure 2: Classes to parse the JSON file

## 3 Acknowledge

Thanks to Paul and Chris, the feedback of the first part task helped me with the MVC pattern design and the JSON parser design. It is impossible for me to keep improving my code without the topics that my classmates have posted. The Gson design strategy is proposed by Norman Peitek at https://futurestud.io/tutorials/gson-advanced-custom-deserialization-basics.

It is wonderful to study the IPPO course, thanks.