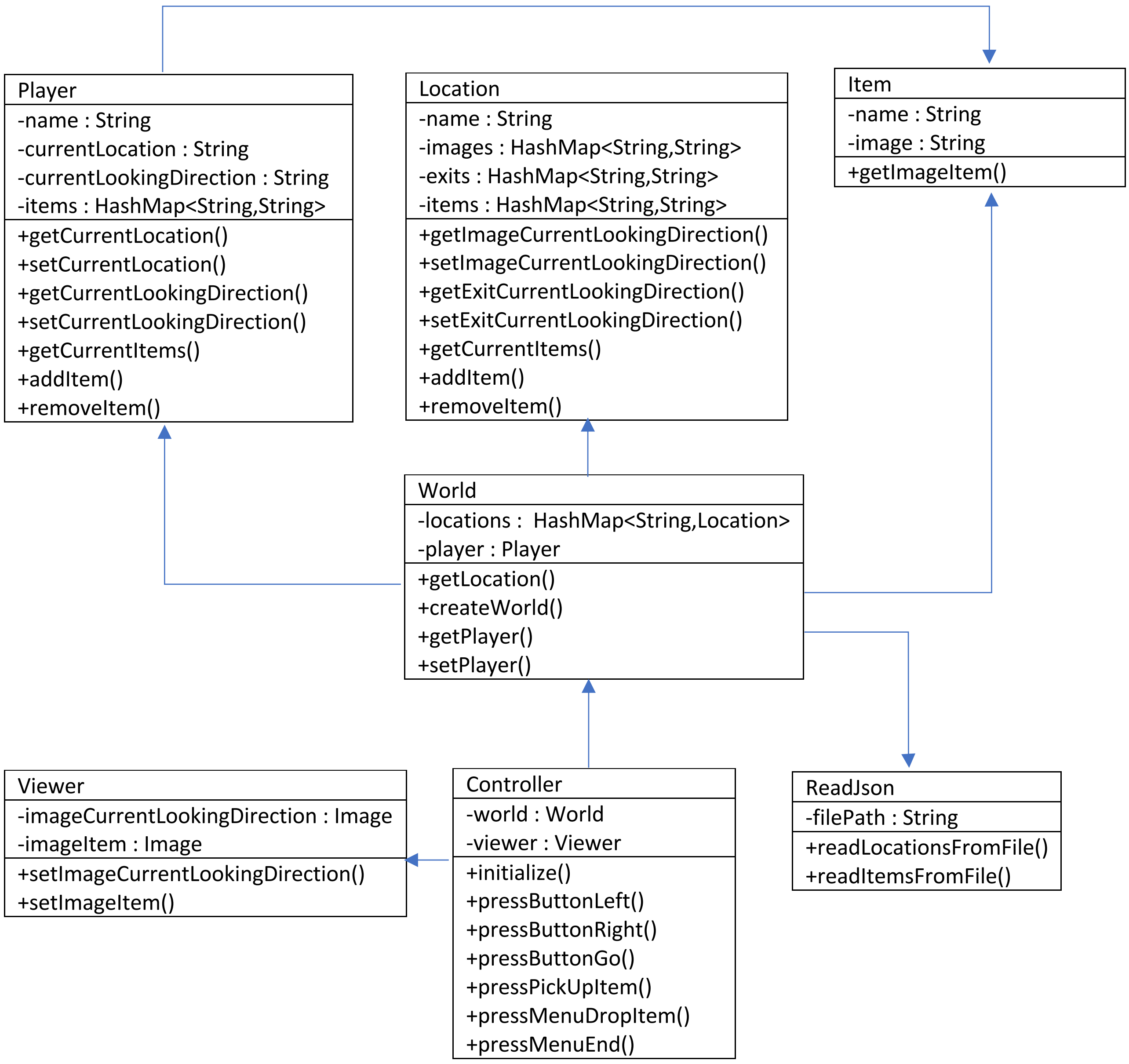
Class diagram



Description

In the following section I will briefly describe the classes of the assignment 2 app. The purposes of the classes and their relations to each other. First and foremost, all the classes have private attributes, that means the access to the classes attributes is just possible through their public methods.

Class `World`

The class `World` is containing the whole world of the app. That means it is responsible for creating and holding all `Location` and `Player` instances. The instances of the `Location` class are created by the method *`createWorld()`* and stored in a `HashMap`. In order to create the world, the data is provided by the instance of the `ReadJson` class. `Locations` are accessible through the method *`getLocation()`*. For now, it is planned to have only one `Player` instance that is interacting in the app's world, that is why there is only one attribute of the class `Player`. The `Player` instance is created by *`setPlayer()`* and accessed by *`getPlayer()`*. This form of setting and accessing the attributes can be seen throughout the whole design.

Class `Location`
The class `Location` is holding all the relevant information of the locations. Each `Location` has a `HashMap` (`images`) containing the four compass directions with the associated image paths, a `HashMap` (`exits`) containing the compass directions with the associated name of the exit `Location` (neighbour `Location`) and a `HashMap` containing the `Items`. The class methods are for accessing the class attributes.

Class `Player`
The `Player` Class is responsible for having the information at which `Location` the `player` currently is in (`currentLocation`), what direction he or she is looking at (`currentLookingDirection`) and what items the `player` is currently caring (`items`). `Items` carried by the `player` are stored in a `HashMap` and can be picked up on a `Location` by the method *`addItem()`* and dropped to the `location` with the method *`removeItem()`*.

Class `Item`
This Class holds two attributes about `Items`, the `Item` name and the image path of the corresponding `Item`. The only method returns the image path to the `Item`.

Class `Controller`
The `Controller` class is the class that handles the interaction between the user and the app. It is responsible for initialising the app logic, creating the GUI and handling the whole user interaction. The `world` attribute is an instance of the `World` class and hence the connection to the app logic since the `World` class has instances of the `Locations` and the `Player`. Furthermore, its second attribute `viewer` is an instance of the `Viewer` Class responsible for viewing the right images at the right `Location`. At the end it is also managing the actions of the buttons and the menu with the methods *`press*()`*.

Class `Viewer`
This class is responsible for viewing the actual images. The attribute `imageCurrentLookingDirection` contains the image that is currently displayed to the user and its corresponding method *`setImageCurrentLookingDirection()`* is passing the image to JavaFX to show it on the GUI. Analogue to this, the `imageItem` and its method are responsible for showing the `item` on the GUI.

Class `ReadJson`
This Class has one attribute called `filePath` for holding a `String` with the file path of the Json file. The main task of this class is to open and phrase a Json file containing data about the locations and items. The method *`readLocationsFromFile()`* is reading all the data about the locations and returning a `HashMap` to the `World` class. Similarly the method *`readItemsFromFile()`* is reading and returning a `HashMap` with all the `items`.

Design choices

While designing the first design for the app I put the attribute `currentLocation` in the `World` class while putting the attribute `currentLookingDirection` in the `Player` class. First it made sense since the `World` class contains all the instances of the `Location` class and they can be directly accessed through this class. But thinking through the perspective of a person that is going to read the code for the first time it is not logical. The purpose of the class `Player` is to contain all the information relevant to the player. Hence it makes sense that the `Player` class contain the players current location, the players looking direction and the items carried by the player. The `World` class contains an instance of `Player,` as such It able to access the information `Player` is containing.