## Class Design

**1. Picture**

Every picture used in the MapViewer application is a class Picture. A Picture contains an image which can be used to display in the ImageView of Javafx.

*Picture()*: Create an empty Picture

*Picture(Image image)*: Create a picture from an Image of Javafx

*setImage(Image image)*: Set the image of the Picture from an Image

*image()*: Return the image of the Picture

**2.0 Service**

A service interface which is used to get Picture for MapViewer application. We can extend the service to whatever you want, such as LocalService or InternetService.

*serviceName()*: Return the service name

*getPicture(String location, String direction)*: Get a Picture from the service

**2.1 LocalService**

The local service for the MapViewer application. We can get the Picture from the local directory. The name of Pictures stored in the local directory should follow the format "LocationDirection.png", such as "AppletonEast.png".

*serviceName()*: Return the local service name

*getPicture(String location, String direction)*: Get a Picture from the local service

*getIcon(String icon)*: Get an icon Image from the local service

**3 Icon**

One Location contains one Icon which stores icon's types and their directions. It can be used to implement portable items.

HashSet<String> iconTypes: Store icons(Dog, Panda or Rabbit) which can be added to the Location. So, we can add different types of icons to the MapViewer application.

HashMap<String, String> iconDirections: Store icons of different directions in the Location. The first string represents direction, and the second is icon's type.

*Icon()*: Create an empty Icon

*getIconTypes()*: Get icon types

*getIconDirections()*: Get icons' directions and their types

*addIcons(String direction, String nameIcon)*: Add an icon to one direction of the Location

*removeIcons(String direction)*: Remove an icon in one direction of the Location

*hasIcons(String direction)*: To judge whether the direction has an icon

**4 Location**

One Location in the MapViewer application. A Location represents one location in the whole Map. One Location has at least 1 nearby Location and at most 4 nearby Locations, and it stores nearby locations as well as their direction.

String nameLocation: Name of the Location

HashMap<String, Location> nearbyLocations: Store the Location's nearby Locations. For example, East->Location1, West->Location2

HashSet<String> directions: Every Location has 5 default directions: North, South, West, East, Down

Icon icons: Store each direction's Icon

*Location(String nameLocation)*: Create a Location whose name is nameLocation
*getNameLocation()*: Return the name of the Location
*setNearbyLocations(String direction, Location location)*: Define nearby Locations for the Location
*getNearbyLocations(String direction)*: Get the nearby Location of given direction
*hasNearbyLocation(String direction)*: To judge whether the Location has the nearby location
*setDirections(String direction)*: Add a new direction to the directions of the Location
*getDirections()*: Get directions of the Location
h*asDirection(String direction)*: To judge whether the Location has the direction
*getIcons()*: Get icons

**5 Map**
The main class of MapViewer application. You can move Location to another Location and change the direction of Location, so you are able to view pictures from different Location and different direction. The Map class creates 7 different Locations, and each Location has 5 default directions.
*Map()*: Create a Map which contains 7 Locations
*createLocations()*: Create all the Locations in the Map, and connect with their nearby Locations
*moveLocation(String direction)*: Move from one Location to another Location
*getCurrentLocation()*: Get current Location
*getCurrentDirection()*: Get current direction
*getNameLocation()*: Get the name of the Location
*getNumberIcon()*: Get the number of icons which can be placed

**6 MapViewer**
Start the application according to the fxml file. Set the title, the size of the window screen, and the display screen.

**7 Controller**
Define the operation of the user interface. The current Location and Direction are displayed at the top of the screen and the corresponding image is displayed in the middle of the screen. If the user wants to go outside the MapViewer boundaries, a warning will appear on the interface. In terms of portable items, the number of items available for placement is displayed in the bottom right corner of the screen.

## Significant Choice

Is there a need for a separate class for Icon?
When I first started writing the Location class, I saved the icon in a HashMap<String, String> in the Location, like nearbyLocations. However, when I wanted to extend the icon (for example, by adding new kinds of icons), it was very complicated. After that, I wrote a separate class for Icon and saved the type of the icon in a HashSet. Whenever I want to add a new kind of icon, I just add the new icon type to this HashSet, so my MapViewer application is able to display many different kinds of icons without changing too many codes.