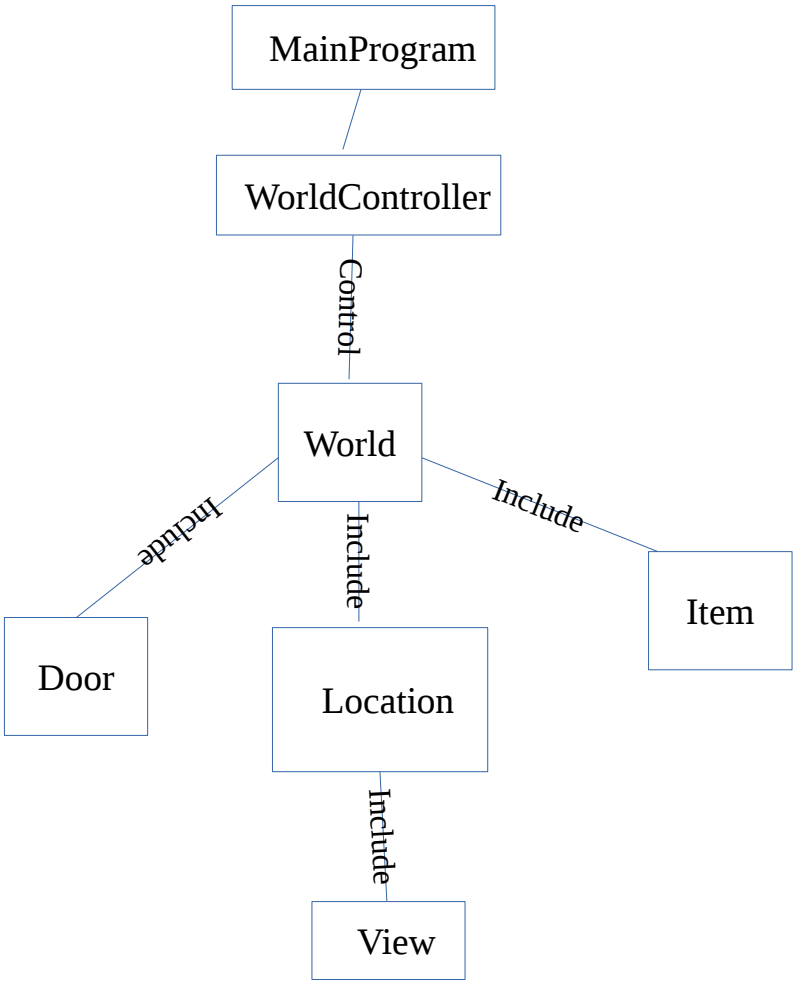
S1626297 Dian Yang

1.



There are 7 classes in total in my design. The core idea of whole design is to design a application that is easy for future potential users to use. For my design, the future user can easily achieve his/her virtual world by only changing the world.init() function.

View class is the first class that I built, which contains *image*, *index* and *locationId*. *Index* is the index in one particular location.

Location class is the collection of **views**. Hence **Location** class contains a *viewMap*(View HashMap), *name*, *positionX*, *positionY*. *PositionX* and *PositionY* are the coordinates on the logic map.

Item class is the portable item which includes *image*, *name*, *index*, *locationId*, *viewIndex*. *LocationId* and *viewIndex* is to save the locationId and viewIndex where we put down our item.

Door class is the relationship between location and location, which contains *currentPosition* and *nextPosition*. *CurrentPosition* is the position when you facing a door, and *nextPosition* is the position when you go through the door. *CurrentPosition* and *nextPosition* are Strings which combined the *locationId* and *viewId*. For example, if *locationId* is 1 and *viewId* is 2, the *currentPosition* is “12”.

World class concludes a *doorMap* (door HashMap), a *locationMap*(location HashMap), a *itemList*(item List), *currentView*(the view that shows on our application), *mapView*(the logic map). All data initialization is done in function **init()**, which create all views, all doors, all items and save

them into all HashMaps.

WorldController has following functions:

- ***initialise****: call world.init() to import data, set image for all imageViews and create a* ***hashMap*** *for saving items and itemViews*
- ***putDown****: put down a portable item if it's not be put down before and set the current locationId and current viewIndex to item.*
- ***pickUp*** *: pick up a portable item if it's in currentView, and do nothing if it's not in current view.*
- ***turnLeft****: return the view in this location with index ((currentIndex + viewMap.size - 1)%viewMap.size)*
- ***turnRight****: return the view in this location with index ((currentIndex + 1)%viewMap.size)*
- ***moveForward****: get currentPosition String using current Location id and view index, then find the nextPosition Strign in doorMap. Convert nextPosition to next locationId and viewIndex, then return the view with next locationId and viewIndex.*
- ***checkItem****: For all items, check if item is already put down and check if item is put down in current view. If it is, show item, if it is not, then hide it.*
- ***checkButton****: For each view, check if it can be forwarded to next location by looking in the doorMap. If it is, then enable the forward button. If it is not, then disable it.*
- ***checkPosition****:For each view, check what's position in logic map*
- ***printMessage****: For each view, print messages to show the current location and item locations.*

2.
(1) I put all data initialisation into the World class instead of WorldController. By this way, we do not need to change controller anymore when we try to import totally different data.
(2) I create Location class to store all views in current location instead of only create views.
By this way, I can easily add more views in one location and do not change the other functions in controller, such as turnLeft.

For example, I had 4 views in one location and want to add another one.
If I didnt create location class, I have to change the nextIndex calculation in turnLeft from **((currentIndex+ 3)%4)** to **(currentIndex +4)%5)**.
But in my implementation, I don't have to change the nextIndex calculation which is **((currentIndex + viewMap.size – 1)%viewMap.size)**

3. For my picture source, I create my house model from a website in http://home.by.me/en/ , hence there are watermarks in the pictures.

There is a door (see in right) which is not going forward to other position. It can be used to connect to another world.