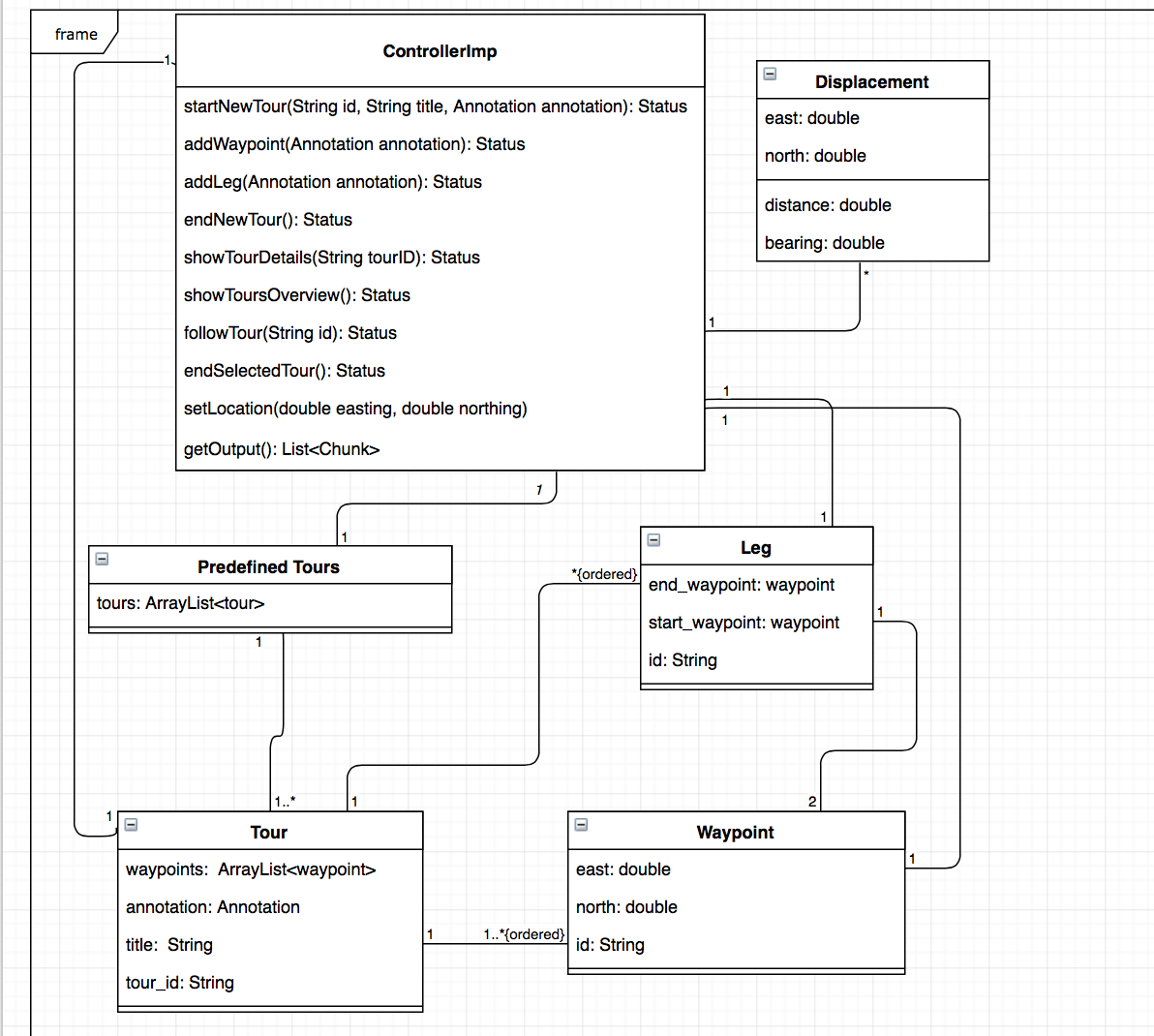
Report

Zach Whitelaw s1688197

Fangan Yi s1610548

**UML Class diagram**



**High-level design description**

* Create tour mode

Author create a new tour by entering this new tour id, title and annotation. Then author would walk the tour routine in the mean while add all the waypoints and legs with or without an annotation along the tour via using *setLocation* function to get current position. All the waypoints and the waypoints in legs will be stored in *waypoints: ArrayList<waypoint>* one by one*.* Since we use arraylist to store tour routine, we don’t need to worry about the size of waypoints in advance and waypoints in the routine will keep in ordered as well. Once this tour is finished, click to end the new tour in order to finish this mode.

* Browser tour mode

Once user calls *showToursOverview()*, all the predefined tours will come up with a brief introduction. If user is interested in one of the tour from predefined tour and calls *showTourDetials(String tourID)*, more information about this specific tour will shown to help user select one tour. Once the user has his decision, he will call *followTour(String id)*, then this mode would end and follow tour mode will stand by.

* Follow tour mode

If the user is in the range of the first waypoint of the tour detected by *setLocation*. Follow tour mode would start right away. ID of the waypoint will help to detect whether this waypoint is just a waypoint or it’s one of the waypoint of a leg. If it’s just a waypoint, annotation will pop up until user leave the waypoint region if applicable. If it’s the start waypoint of a leg, annotation will pop up and keep there if applicable. If it’s the last waypoint of a leg, annotation will disappear. In the mean while, functions in displacement class will be called to get direction and distance to next waypoint of the tour based on user’s current waypoint(*setLocation*) and next waypoint of the tour. The applicable annotation and output of displacement will be gathered by *getOutput* function. A loop would help to get waypoints of the tour one by one, this loop will only end when user reach the last waypoint of the tour. When the loop end, this follow tour mode would end.

* Difference between cw2 and this work

We use one displacement class instead of distance class and direction class. We write all the functions in *controllerImp* instead of separate them in all classes.

**Implementation decision**

We use ArrayList class to store tours in predefined tours and waypoints in tour. Then we could forget about the size of those ArrayList. And element in the arraylist will keep in order. In *ControllerImp* when we call *followTour* function, *distance* and *bearing* functions will be kept calling until this mode is end. So we will have one-to-many relationship between *ControllerImp* and *Displacement*.