High-level design description:

The Displacement2 class is to calculate distance and bearing between two points (two Displacement). It has a constructor “Displacement (Displacement d1, Displacement d2)” which takes two Displacement as arguments and then used for the further calculation. The “distance()” method gives you the distance between d1 and d2, and the “bearing()” method gives the bearing of d2 from d1.

The ListLocation class is used to record the locations (which are objects of Displacement) of a tour. The locations are added to the ArrayList <Displacement> when a waypoint is added while creating a tour, and this ArrayList <Displacement> is stored as ListLocation when a tour is created. The method “out()” is simply return the ArrayList<Displacement> stored in the class.

Stages class is what I used to associate fixed stages to a tour. It has attributes “numWaypoint”, “numLeg”, “stage” which tells the total number of waypoints, legs and stages of a tour. The attributes “stageAndLeg” and “stageAndWaypoint” tells you the annotation of a leg and waypoint of a tour for the given stage. It has method “increaseStage ()” which is used to increase the number of stages while creating a tour and then return the total number of stages when a tour is created. The methods “addWaypoint(Annotation waypoint)” and “addLeg(Annotation leg)” are used to record the annotation of waypoints and legs for a certain stage.

ControllerImp is the class where we put all classes we created together. In the controllerImp class, it has a private double attributes call “mode”, which represents the current mode (1 = browse mode, 2 = creating mode, 3 = follow mode). When a Controller object is created, it automatically sets the mode to browse mode, and attributes “waypointRadius” and “waypointSeperation” is set up when the Controller is created. All the methods except “setLocation” or “getOutput” in ControllerImp return Status, where Status.Ok indicate it’s available to do such an action, and Status.Error indicate such an action can’t be done.

The method “startNewTour” is only available when mode is browse mode and it will sets the mode to creating mode. The attribute “mode” can also be equal to 2.1 and 2.2, where mode 2.1 for adding leg and mode 2.2 for adding waypoints.

The method “showToursOverview” and “showTourDetails” have mode 1.0 and 1.1 respectively, where mode 1.0 display overview for all tours and mode 1.1 shows detailed information for the chosen tour.

The method “followTour” sets the mode to follow mode(mode 3), it is also only available when the current mode is 2(browse mode)

* note: This high-level description is very different from the one from Coursework 2 as many different class and methods were used.

Implementation decisions:

All my work is mainly based on the given code. The only classes I created are Stages, ListLocation, Displacement2.

As displacement class can only help to calculate distance and bearing of a point from origin. A Displacement2 was created to calculate the distance and bearing between two locations, i.e. from a position to a waypoint.

A lots of HashMap and ArrayList were used. For example, in the Stage class, I want something that can display the leg and waypoint annotation at the current stage, therefore I used HashMap since personally I think HashMap is a convenient way to achieve this goal. Stage class was created to store detailed information of a tour, including: total number of legs, waypoints and stages, also the information of leg and waypoint corresponding to each stage.

In the controllerImp class, it will create a HashMap in the form of (HashMap <String, ArrayList <Displacement>>), and A problem initializing of ArrayList<displacement> was encountered. To solve this problem, ListLocation class was created to avoid the initializing of the ArrayList. ListLocation class can store the ArrayList <Displacement> first, and it is then used in the HashMap.

All these implementation decisions are trying to minimize the changes to the skeleton code. Since the skeleton code is enough to achieve most of the requirements, therefore only few new classes were added.