Cosmic Application

The CosmicApp class will serve as the entry-of the program and initializes View, Controller, and Model. The design pattern is MVC based. The Model will consist of a ResearchStation that manages multiple RadioDish Objects by using a Map. The View will consist of a user interface that communicates with the user, as well as takes in user input and communicates it with the View by primarily restricting to an enumeration for valid commands (although not entirely). The Controller will process the input from View and communicate between View and Model.

The following classes exist: CosmicApp, View, Control, Model, ResearchStation, Map, RadioDish.
The following interfaces exist: MapADT.

The RadioDish class will contain classes relevant to a Radio Dish object. Specifically accessor and mutator methods for name and calibration, as well as a listen method.

The Map class contains Map ADT methods, and will serve as the underlying data structure for our Research Station.

The view class contains a method for the terminal for the user to interface with the Research Station through Controller to Model to Research Station and back, satisfying MVC and Single-Responsibility Principle as much as possible.

The controller class contains a method called begin that begins the simulation with respect to the rules we've dictated.

The model class houses all the important data (and can retrieve it), also performs logical operation on data as requested by controller. It communicates back to the controller directly.

With our setup, we can easily extend the project as much as we want, and ensure that information hiding, single responsibility principle, and polymorphism is satisfied. This approach satisfies the SOLID principles to a decent degree, the only issue being the processInput method in Controller giving us some issues for retrieving data to View as it only returns Boolean.

A initial iteration UML Diagram will be included in the repository, as well as a revised UML Diagram after the implementation has occurred. The latter of which will be automatically generated and can be used to compare the pre-implementation and post-implementation differences in design.

The simulation guides the user through the UI on how to interact with it, which is sufficient for the user to be able to utilize the simulation.

Note: in future iterations, a GUI might be implemented, additional rules and properties may be imposed, and the work may be publicized as a simulation and or video game. I quite like the idea behind this project, hence I may or may not further on it in the future, whether it is far or soon.