Ochre Assignment 03 Petition

Why doesn't it work and which design requirements it does not satisfy?

The assignment does not work from an implementation standpoint because there is a significant amount of the code base missing or left unimplemented. There is also code that is just wrong in terms of how the program should function. In this assignment we want a single server side Model that many local Controller/Views will connect to. In their Forager Server class, they use a Map to Map Controllers to ForagerModels, meaning that in their implementation each local Controller/View has its own Model. This goes against the functionality to have many players playing on the same board. I will elaborate on this more in later sections. The design of the function also does not meet design requirements. The two design patterns chosen by the previous group were the Observer pattern and the Decorator pattern, however the UML does not show a proper usage of either pattern. There is an Observer in the UML but no class that extends the Observable. They talk about using the Observer pattern over the Java RMI interface which would not work so even if the ForagerModel class extended the Observable class the design would still be broken. In the context of the Decorator pattern, they do not have an AbstractClass that an AbstractDecorator or ConcreteDecorator could implement/extend. In the UML all they have is a "PlayerDecorator" class that points to the Player class but that is not in fact a Decorator. For these reasons, even if implemented perfectly, this UML would not satisfy the design requirements because it would result in non-functional code and an incorrect usage of the design patterns.

List of features and functionalities that work:

- Local View with blank game board that connects to local Model and Controller
- Single player represented by a blue dot on the game board
- Local player movement on the local view that is connected to a local Model and Controller

List of features and functionalities that are not present or don't work:

- Implemented RMI interface that the class voted on
- The ability for the View to represent Resrouces
- The ability for the Model to hold or update the resource state based on the forager game logic
- The ability for the Model to read in some sort of text file to establish a starting state of resources on the board
- The ability to have a Model in the server that Controllers can connect to
- The ability for many players to connect to a distributed game
- The ability for many players to move on a distributed game
- The ability for the Model/game to run on a server
- The Controller to call methods for the ModelInterface to communicate with a distributed object

List of classes/interfaces in the project, explaining for each class/interface what it does or doesn't do:

- Controller:
  - The Controller should be the communication point between the local client and the server. However, there is no functionality here to communicate or pass data to a distributed object or call RMI interface methods
- ForagerInterface
  - The interface voted on from class. It has everything there as an interface
- ForagerModel
  - Where the data of the game data is housed. It can currently locally deal with single player movement but has no board resource growth logic and doesn't seem to be able to keep track of multiplayer movement. It also doesn't have a game clock/update itself and uses a Point class instead of the Pair class used in the RMI interface
- ForagerServer
  - A class to hold a Map<Conteroller, ForagerModel>. As discussed earlier, this means that each Controller has its own model meaning each player is playing their own game rather than together. It seems to be a remote object but has no real built in functionality from the RMI interface as all the methods are to be implemented
- Player
  - A player object to hold player data. This in some ways conflicts with the agreed upon RMI interface and also holds player movement inside of it which is truly data that should be allocated to the Model rather than held locally
- PlayerDecorator
  - A class that extends Player. Seems to add some functionality but isn't really a decorator from a design standpoint and just adds some rudimentary functions like setting the ID. Don't really see the point of this one
- Tile
  - A class to represent a tile on the gameboard
- View
  - The view which can display players and blank gameboard tiles. No way to display resources

Design requirements that are broken or missing or malfunctioning (including Java RMI interface):

This is discussed pretty thoroughly in the first paragraph, but to summarize:

- The Observer pattern is over the RMI interface which will not work and is missing the Observable in the design
- The Decorator pattern is just incorrect and doesn't follow the structure of a Decorator pattern