Team Athens Design Doc

Requirements:

- Maps are randomly generated in MapGenerator
- Four different terrain types and resources
  - Dirt, Water, Gold, Food
  - Water is impassable
  - Maps do not fit in a single window scroll bars are needed
  - Maps are 2D arrays of tiles, which have a number to indicate
    - What resource is on that tile

- Player starts with 3 agents
- Agents needs are hunger, thirst and taxes
- If needs aren't met health declines until the agent becomes unplayable as a
  Philosopher
- Player can only request an Agent to do a task, wont be done unless the needs
  Are satisfied.
- Player may have 4 agents for every barracks built
- Agents have limited storage capacity before their resources must be deposited
  To a storehouse building

- Players can interact with the agents through buttons to build buildings or
  Gather resources
- Two buildings available are a storehouse and a barracks
- Building buildings require resources either on the Agent or stored in a building
- Game plays in real time with little to no delays
- Game uses simple sprites to show the agents, map terrains and all other objects
- Player wins by collection 100 gold

Design Patterns:

Singleton – Map and Images

The Map class uses the singleton design pattern so only one randomly generated map instance is used throughout the program. Almost every class uses an instance of map and its corresponding map array

Images uses the singleton instance so that images for the tiles are only read in once then those images can just be grabbed by the Tiles that need it

MVC –

We use the MVC pattern to separate data and user interface. However our implementation is modified in we don't have a controller aspect. The action listeners are just integrated into the view class.

Factory -- Buildings

We use the factory pattern to build out buildings. We have the abstract building factory class which is implemented by the concrete build building class. This class takes a string, a type of building, and returns a building of that type.

Iterator –

Iterator is used throughout but most heavily in our dijkstra class which is our algorithm that finds a route for agents to traverse to items on the map.

Composite— Swing

Composite design pattern is hard to avoid in swing. We use swing for all of our graphics and user interface. JButtons are in Panels, Panels are in scroll panes, and everything is added to a Jframe.

Observer/Observable –

We use the observer observable to update our map as the tiles change. When an agent gathers a resource or builds a building the tiles number is changed, these numbers dictate what image is drawn. Observer is called and notifies the GUI to redraw the map to its Jpanel with the updated terrain.