Mobile Agents Design:

Node k k = 1, 2,.,m

MobileAgents

Agent

BaseStation

* **Objects:**
* **MobileAgents:**

This object reads in the config file and creates the graph by first reading the nodes and then adding the edges to the neighbor list of the nodes. Then it will create add the first agent and the program will be started. It also has an update method for the observer pattern. This method is invoked whenever we want to update the screen.

* **Agent:**

The agent is basically created on a yellow or blue node neighbor to a yellow node. The agent clones itself to keep this positioning whenever a node turns red and its agent dies. The node does this by cloning itself to eligible neighbors whenever one of its neighbors dies and its node turns yellow.

It only behaves differently at the start. At the start it will randomly walk through the nodes until it finds a yellow nodes and stays there (and clones itself).

* **Node:**

The node provides all the required methods to do all the tasks that agent will need or are needed to propagate the yellow and red nodes. It also has the functionality to send ID, clone, walk, update display and make the nodes red.

* **BaseStation:**

The base station is almost like the node, it’s main difference is that it gets, display and stores the agent location and id.

* **Methods Used:**
* **MobileAgents:**

1. *main()* – launches the program.
2. *stop()* – stops the program, when user exit the application.
3. *start(Stage primaryStage)* – Display element, reads the graph using scanner, stores the graph internally and draws the graph on the screen. It also runs the simulation, baseStattion findPaths in the graph, agent start wandering from baseStation, and onFire node spreads the fire.

* **Node:**

1. *run()* – runs the node thread,
2. *getStatus()* – gets the state of the node.
3. *getBurner()* – gets the fire spread thread.
4. *getX()* – gets the x position of specific node.
5. *getY()* – gets the y position of specific node.
6. *setState(Status status)* – change the state of the node to yellow or red.
7. *passAgent()* –passes the agent to a random neighbor (which doesn't have an agent already) and set the agent to null (Not cloning).
8. *recieveAgent(Agent agent)* – receives the agent on a specific node if the agent is not already there.
9. *setID(int id)* – creates the id for the node.
10. *makeAndSendAgentID()* –
11. *addNeighbor(Node node)* – this node adds neighbor node to the neighbor list.
12. *sendCloneAgent()* – clones the agents and sends the clone of the agent to live nodes that are blue or yellow and do not already have an agent.
13. *recieveClone(Agent clone)* – can make it private!!!!!
14. *scream()* – this node lets its neighbors know that he’s dead and fire is spreading.
15. *findPaths(LinkedList<Node> path,Node caller)* –
16. *sendID(int id, int x, int y)* –
17. *passID(int id, int x, int y, LinkedList<Node> path, LinkedList<Node> returnPath)* –
18. *passIDFromQueue(int id, int x, int y, LinkedList<Node> path, LinkedList<Node> returnPath)* –
19. *returnID(int id, int x, int y,boolean status, LinkedList<Node> path,LinkedList<Node> returnPath)* –
20. *returnIDFromQueue (int id, int x, int y,boolean status, LinkedList<Node> path,LinkedList<Node> returnPath)* –

* **Agent:**

1. run() –
2. kill() –

* **BaseStation:**

1. findPaths() –
2. passID(int id, int x, int y, LinkedList<Node> path, LinkedList<Node> returnPath) –
3. sendID(int id, int x, int y) –
4. printIDs() –