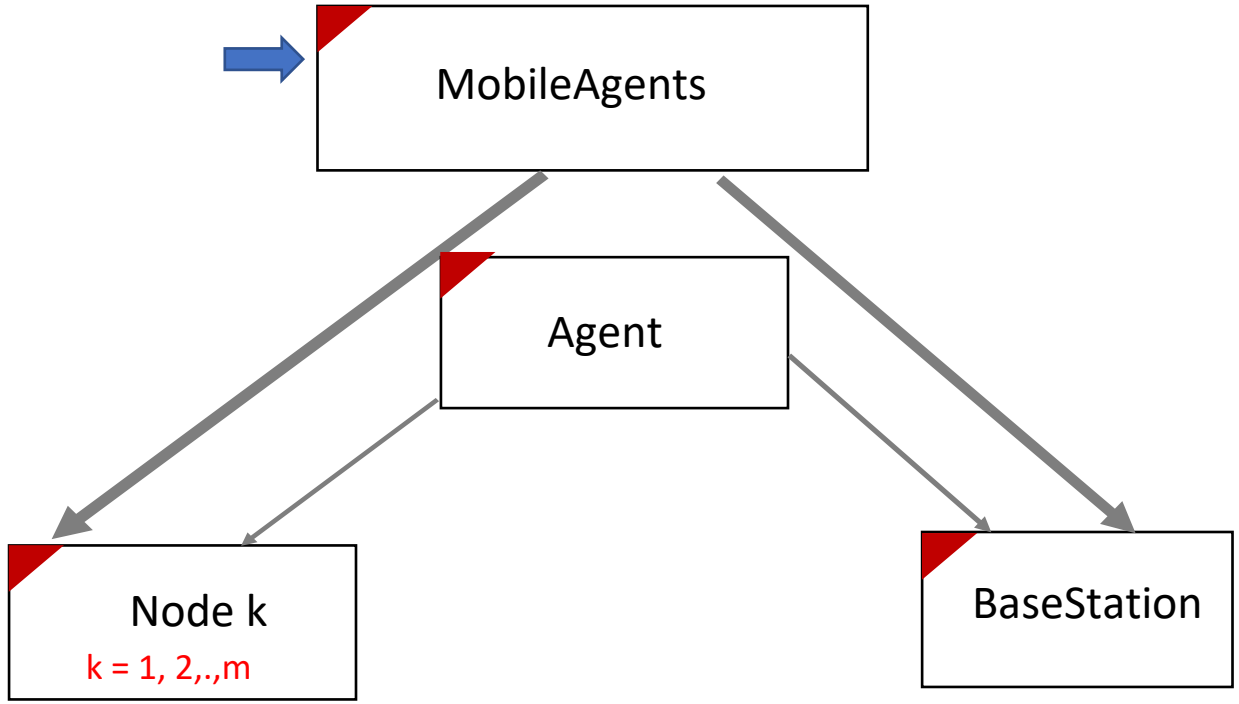
Mobile Agents Design:



Design Details

- **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.

Design Details

- **Methods Used:**
- **MobileAgents:**
  I. *main()* – launches the program.
  II. *stop()* – stops the program, when user exit the application.
  III. *start(Stage primaryStage)* – This reads the config file and creates the node objects. It also creates the display elements. The display consists of circles as nodes in x,y of config file. The edges are lines from x1,y1 to x2,y2. IMPORTANT NOTE: The circle objects which are nodes should be passed to node objects when they are being created. The size of canvas is: width=MAX(c*x)+20 ans height=MAX(c*y)+20. where c is a constant. The graph is formed by detecting and adding the edges to the neighbors of the nodes. At the end we add an agent to the basestation and the program is started. At the end, it will terminate, because now other threads are doing everything. 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.

Design Details

- **Node:**
  I. *run()* – This function will check for the status of the node and if the node is yellow it has a timer to wait for 2 seconds and then turn red. If it turns red it should notify other neighbors. It also updates the color of Circle object. (Note than in the future we might use BlockingQueues, so run will handle them).
  II. Kill() Kills the node and agent.
  III. *getStatus()* – gets the state of the node.
  IV. *getBurner() – gets the thread that spreads the fire.*
  V. *getX()* – gets the x position of specific node.
  VI. *getY()* – gets the y position of specific node.
  VII. *setState(Status status)* – sets the state of the node and calls functions to update screen.
  VIII. *passAgent()* –passes the agent to a random neighbor (which doesn't have an agent already) and set the agent to null (Not cloning).
  IX. *recieveAgent(Agent agent)* – receives the agent on a specific node if the agent is not already there.
  X. *setID(int id)* – sets the id for the agent.
  XI. *makeAndSendAgentID() – Make a unique id for the agent and pass it...*

Design Details

XII. *addNeighbor(Node node)* – this node adds neighbor node to the neighbor list.

XIII. UpdateScreen() : This function updates the screen element using the observer pattern.

XIV. *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.

XV. *scream()* – this node lets its neighbors know that he's dead and fire is spreading.

XVI. *findPaths(LinkedList<Node> path,Node caller) – This functions is used in the beginning of the program, to find all possible paths to the base station.*

XVII. *sendID(int id, int x, int y) – This one uses the paths to send an id to the next node via calling the pass id function.*

XVIII. *passID(int id, int x, int y, LinkedList<Node> path, LinkedList<Node> returnPath) –This one adds a message to a blocking queue to be passed.*

XIX. *passIDFromQueue(int id, int x, int y, LinkedList<Node> path, LinkedList<Node> returnPath)* – This function will be called from the queue to pass id to the next node. It checks to see if the current path is available, then it sends to next node. If not available, it will return false.

XX. *returnID(int id, int x, int y,boolean status, LinkedList<Node> path,LinkedList<Node> returnPath) –This adds return status to the queue.*

Design Details

XXI. *returnIDFromQueue (int id, int x, int y,boolean status, LinkedList<Node> path,LinkedList<Node> returnPath)* – This returns the success or not success of the function.

- **Agent:**
  I. run() –This function does all the agent related tasks. For the first agent, it goes through the nodes randomly to find a yellow node and stays there. If a node turns yellow, its agent clones itself to the eligible neighbors.
  II. kill() –*This one kills the agent.*

- **BaseStation:**
  I. findPaths() – This function finds the path from any node to this node and save them there. Note that it is only invoked at the start of the program.
  II. passID(int id, int x, int y, LinkedList<Node> path, LinkedList<Node> returnPath) –*This function will store the returned id and location of the agents*

  III. sendID(int id, int x, int y) –This one save the id because it don't need to save it.

Design Details

IV. printIDs() –*Prints all ID's*

V. *makeAndSendAgentID(): This one makes and send agent ID.*