public class MapReader {

double minLat =0;

double maxLat =0;

double minLon =0;

double maxLon=0;

int roadCounter = 0;

ArrayList<Nd> arrayAllNodes = new ArrayList<Nd>();

public ArrayList<Road> roads = new ArrayList<Road>();

NodeList allNodes;

MapReader(String docString)

private void setBounds(NodeList bounds)

public ArrayList<Road> getRoads()

private void getListOfNd(NodeList wayList)

public void makeTrafficLightList(Element elem)

private boolean isRoad(Element showElement):

private Document getDocument(String docstring)

private void makeNodeList()

private void sortAllNodes()

private Nd makeNd(long ref)

void setIntersections()

============================================

public class GraphBuilder

ArrayList<Road> roads = new ArrayList<Road>();

ArrayList<Nd> stops = new ArrayList<>();

GraphBuilder(ArrayList<Road> roads)

private void makeIntersectNode(int i, int j, Indexes fIJ)

private Indexes findNodeIndex(long ref,int roadsIndex, int interIndex)

private Nd findNode(long ref,int roadsIndex, int interIndex)

public void buildStops()

public ArrayList<Nd> getStops()

public ArrayList<Road> getRoads()

public Nd findNode(double x, double y)

============

private class Indexes{

int i =0;

int j =0;

}

public class Simulation {

GraphBuilder gb;

int tickCounter =0;

private ArrayList<Auto> cars = new ArrayList<Auto>();

private Renderer display;

private double minLat=0;

private double minLon=0;

private double maxLon=0;

private double maxLat =0;

Simulation()

Simulation(String fileName)

public void openMap(String fileName)

public void saveMap()

public void loadMap()

public void startRenderer(double scale)

public void setScale(double sca)

public void demo()

public void getSimInfo()

public void SetCars(int totalCars)

public Direction makeDirection()

public Directions makeDirection(nd start)

public void step(double velocity, double stepSize)

private Boolean checkStop(int carIndex)

public void updateRenderer()

============================================

public class Renderer extends JFrame{

int playSpeed = 1;

DrawStuff drawer = new DrawStuff();

ArrayList<Road> roads = new ArrayList<>();

ArrayList<Auto> cars = new ArrayList<>();

ArrayList<Shape> roadShapes = new ArrayList<Shape>();

ArrayList<Shape> intersectShapes = new ArrayList<Shape>();

ArrayList<Shape> carShapes = new ArrayList<Shape>();

NodeInfoFrame nodeInfoFrame;

double mouseLong=0;

double mouseLat=0;

int mouseOffSetX=0;

int mouseOffSetY=0;

double scale = 1;

final int scale1 = 100000;

Shape testCar = new Rectangle2D.Double(0,0,0,0);

double minLat=0;

double minLon=0;

double maxLon=0;

double maxLat =0;

int latRange=0;

int lonRange=0;

boolean nodeButtonOn;

Renderer(double scale, ArrayList<Road> roads, double minLat, double maxLat, double minLon, double maxLon)

public void setScale(double scale)

public void stepCars(ArrayList<Auto> cars)

private double latToGrid(double lat)

private double longToGrid(double lon)

public void setMap()

public void reSizewindow()

public void drawWindow()

private double xToLong(double lon)

private double yToLat(dobuel lat)

boolean nodeSearchPlease;

Boolean showClickSpot;

private class Mopuse Handler

Private class DrawStuff

============================================

Public class Nd{

private long ref = 0;

private double longitude = 0;

private double latitude = 0;

boolean isIntersection = false;

boolean isStop = false;

boolean isStopLight=false;

double trafficLoad = 0; //lets measure this in seconds, average car loiter time

double speedLimit = .00016; // 44 feet per second, 30 mph. default

Road parentRoad = new Road();

Auto parentAuto;

ArrayList<Nd> cars = new ArrayList<Nd>();

ArrayList<Nd> connections = new ArrayList<Nd>();

ArrayList<Double> weights = new ArrayList<Double>();

ArrayList<Nd> stopQ = new ArrayList<Nd>();

Public void addConnection(Nd node, double weight)

Nd()

Nd(Nd node)

Public void setRef(long ref)

Public long getRef()

Public void setLong(double longitude)

Public void set Lat(double latitude)

Public long getLong()

Public long getLat()

Public void addcar(Nd carNode)

Public void removeCar(Nd carNode)

Public double calcDistance(Nd node2)

============================================

public class IntersectNd extends Nd {

private long secondaryRef = 0;

ArrayList<Nd> connections = new ArrayList<Nd>();

ArrayList<Double> weights = new ArrayList<Double>();

IntersectNd()

Public void addEdge(Nd node)

void setSecondary(long ref)

long getSecondary()

============================================

public class Road {

private int id = 0; //identification # of road

ArrayList<Nd> nodeList = new ArrayList<Nd>(); // Nodes that make up road

ArrayList<IntersectNd> intersections = new ArrayList<IntersectNd>();

ArrayList<Double> distList = new ArrayList<Double>(); // distances between nodes

boolean oneWay = false;

public double distance =0;

Road()

Public void buildLocalIntercets()

Public void setID(int id)

Public int getID()

Public void addNode(Nd node)

Public ArrayList<Nd> getNodes(0

Public void addIntersection(IntersectNd nd)

Public void setIntersections(ArrayList<IntersectNd> list)

Public ArrayList<IntersectNd> getIntersections()

Public void buildRoads()

Private double calcDistance(Nd node1, Nd node2)

============================================

public class Auto {

Directions directions = new Directions();

Nd posNode = new Nd();

Nd waypointNode = new Nd();

Nd lastWaypointNode = new Nd();

double distNext = 0;

double velocity=0;

double maxVelocity=0;

double acceleration =.000025;

boolean stop = false;

static int carID=0;

boolean accelerationOn=true;

int justGo =0;

double carSpacing =.00024;

double stopSpacing=0;

int justGo =0;

boolean stoppedOnce=false;

Auto()

Auto(Directions dir)

Private void setNodeParent()

Public void setDirections(Directions directions)

Public void setPos(Nd node)

Public void setPos(double lon, double lat)

Public Nd getPos()

Public void ping()

Public void calcPos(double newVelocity, double timeIncrement)

Public Boolean nextWaypoint()

Public void step(double maxVelocity, double timestep)

Public double calcDistance(Nd node1, Nd node2)

Public Nd getCurrentNd()

Public void debug()

============================================

public class Directions {

Nd start;

ArrayList<Integer> directions = new ArrayList<Integer>();

public double score;

stepIndex =0;

private boolean empty = false;

Directions()

Directions(Directions dir)

Directions(int start)

Directions(Nd start)

Public int getNextIndex()

Public Boolean inProgress()

Public Boolean hasStarted()

Public int next()

Public void add(int dir, double score)

Public void setEmpty()

Public Boolean isEmpty()

Public Directions findRoute(ArrayList<Road> roads, Nd start, Nd end)

Private Boolean wasVisited(long ref, ArrayList<Long> visited)

Private ArrayList<Directions> sortQueue(ArrayList<Directions> queue)

Public int compareTo(Directions d)

Public int compareTo(Double score)

============================================

public class NodeInfoFrame extends JFrame{

Nd node= new Nd();

JTextField refField = new JTextField(10);

JTextField longField = new JTextField(10);

JTextField latField = new JTextField(10);

JTextArea carArea = new JTextArea(7,12);

JTextArea conArea = new JTextArea(7,12);

JRadioButton stopYes = new JRadioButton("true");

JRadioButton stopNo = new JRadioButton("false");

ButtonGroup stopButtons = new ButtonGroup();

============================================

NodeInfoFrame()

Public void setNode(Nd node)

Public void setFields()

private class RadioButtonListener implements ActionListener

