COMP 261 2016
Assignment 2
by Anna Lezhikova
300398605

Auckland road map: shortest path search, articulation points.

Main file for this app is MapDrawer.java

<u>Shortest path search</u>
To find the path click on Findroute button and follow the instructions in the output box. To remove the highlight click on the map anywhere.
The path will show the way according to active direction rules. It will never get you in oneway stree in wrong direction.
Main functions for this feature:

- route – for defining start and end points (MapDrawer, 106);
- getCloseNodes – to find the closest intersection to the click (NodeCollection, 76);
- findThePath – to find the segments to draw and print out (MapDrawer, 202);
- getThePath – get all nodes, marked as the parts of the shortest path (NodeCollection, 124);
- findShortestPath – A* algorithm (NodeCollection, 86).

Pseudo-code for findShortestPath function:

```
take the nodes collection
mark all node.visited = false, node.pathFrom = null, node.pathSegment = null

init fringe as a priority queue (curNode, prevNode, costToHere, estTotalCost, segment)
with estTotalCost as a priority
put the start node there
iterate though the queue, while loop (not empty)
    dequeue item
    if not curNode.visited then
        curNode.visited = true
        curNode.pathFrom = prevNode
        curNode.pathSegment = segment
        if node = goalNode
            return
        take node.neighboursOut and iterate segments
            nodeNeighbour = get segment other end
            if not nodeNeighbour.visited
                costToNeigh = costToHere + segment.length
                estTotalCost = costToNeigh + distance(nodeNeighbour and goalNode)
                fringe.enqueue neighbourNode, curNode, costToNeigh, estTotalCost, segment
```

<u>Articulation points</u>
To display all the articulation points click on the Critical points button. To remove the highlight click on the map anywhere.
The algorithm uses iteration function.
Main functions for this feature:

- findCriticalPoints (NodeCollection, 139);
- iterateArtPoints (NodeCollection, 164).

Pseudo-code for findArticulationPoints function:

```
Set each node.depth to infinity
Initialise articulationPoints set
for every node on collection
    init subTree
    if the node is not visited
    node.depth = 0
    for each node's neighbour
        if neighbour is not visited
            iterateArtPoints (neighbour, startNode, set)
        subTree ++
    if numSubtrees > 1
        set.add node
return set
```

Pseudo-code for iterateArtPoints function:

```
init stack of elements/fringe
create root element
create node element (firstNode, reachBack(1), parent(root), depth(0), childrenQueue(null))
add it to the fringe
while fringe is not empty
    fringeItem = fringe.peek()
    node = fringeItem.node
    if childrenQueue is null
        node.depth = fringeItem.depth
        fringeItem.reachBack = fringeItem.depth
        fringeItem.children = new Queue()
            for each node's neighbour
                if neighbour != fringeItem.root
                    fringeItem.childrenQueue.add(neighbour)
    else if childrenQueue is not empty
        child = childrenQueue.poll()
        if child is visited
            fringeItem.reachBack = min(fringeItem.reachBack, child.depth)
        else
            fringeStack.add (node(child), reachBack(nextNode.depth + 1), parent(fringeItem),
depth(0), childrenQueue(null))
    else
        if node != firstNode
            if fringeItem.reachBack >= fringeItem.parent.depth
                articulationPoints.add (fringeItem.parent.node)
             fringeItem.parent.reachBack = min( fringeItem.parent.reachBack,  fringeItem.reachBack)
        fringe.pop()
```