Main.java

* Regex to read from galaxy.txt and proper line format
* Regex to read from pilots\_routes.txt and the proper line format
* Main
  + Set up 2 Scanners to read from galaxy.txt, pilot\_routes.txt, and a PrintWriter for patrols.txt.
  + WeightedGraph graph = ReadGalaxy(Scanner galaxy)
  + String[] validLine = ReadPilots(scanner pilots)
  + ArrayList<String array of length 3> pilots
  + While validLine is not null
    - Pilots.add({validLine[0] , findLength(graph,validLine[1]), isValid(graph, validLine[1]))});
  + All Pilots have been grabbed, all items have been calculated, so formatAndPrint(Printwriter, pilots);
  + Close scanners and printwriter
  + exit
* Boolean isValid(WeightedGraph graph,String vertecies)
  + int[] verticies = vertecies.split(“ ”)
  + for(int i = 1; I < vertecies.length; i++)
    - if NOT graph.getNeighbors(graph.getIntex(i)).contains(i-1)
      * return false
  + Since all of the verticies have neighbors with another verticie it is paired with, then return true
* int findLength(WeightedGraph graph, String vertecies, )
  + if isValid(String vertecies) is not valid
    - return 0;
  + else
    - int[] verticies = vertecies.split(“ ”)
    - int total
    - for (int i = 1; I < vertecies.length; i++)
      * total += graph.getWeight(vertecies[i], verticies[i-1])
    - return total
* Void formatAndPrint(Printwriter, ArrayList<String[]> output)
  + Accepts the String, formats the information into an acceptable line to print out, and prints it out through the printWriter.
  + First it reorders the ArrayList into proper printable order based upon validity, then length, then name
  + Then Prints out each item ArrayLIst to printWriter in proper formatting
* String[] readFile(Scanner)
  + String line = nextLineFromScanner
  + While line doesn’t match the regex for pilotLineFormat
    - If there is still another line in the Scanner
      * Then Line = nexLineFromScanner
      * Else return null as there is nothing left
  + So line must contain a valid line now, so return String[]{line name parsed, rest of vertecies};
* WeightedGraph ReadGalaxy(Scanner)
  + Takes the scanner
  + ArrayList<String> stuff;
  + Int largestNum
  + For each line in the scanner
    - Determines if the format of the line matches
      * If it doesn’t, move on to next line
      * If it does match, then add the String to stuff
      * Also check to see if current vertex number is larger than largestNum, and if it is, set largestNum to be this number
  + Now that stuff contains all of the valid lines in the Galaxy, create ArrayList<WeightedEdge> edges;
  + For every item in stuff
    - For every edge contained within that String
      * Add a new WeightedEdge to edges containing currentVertex (taken from String stuff), connectingVertex, and weight
  + Then return weightedGraph(edges, stuff.size())