Pseudocode:

When running the program, the main method bellow is executed

- Main() throws IOException
  - graphInput file as a String
  - directDistance as a string
  - While (true)
    - Try {
      - Print out “Enter city name …etc”
      - Print out “To quit the program, just enter ‘quit’.”
      - If (user enters ‘quit’)
        - Program terminates
      - If (length of user input == 1)
        - Create a new Graph object with input (graphInput, directDistance);
      - Creates currentCity user input as an upper case character.
      - Calls shortestPathFinder() with graph and user, input, and algorithm number city name as parameters
      - Which finds the shortest path from user input city to the destination city Z.
      - After it is done, “New Session:” message is printed out to user
    - Catch (InputMismatchException | NullPointerException | EmptyStackException e)
      - If one of the above exceptions is caught, it prints out “the input entered is not a city in graph”

- shortestPathFinder()

  * **Input:** the Graph that was built in the main() method, and a Character userCity, and either heuristic algorithm 1 (based on shortest direct distance only, or algorithm 2 (based on the shortest: direct distance + weight.
  * **Output:** Prints out the shortest distance from the input city and final destination Z, and also prints out the shortest path to get to Z, as well as all cities that were considered in the path.

  - and algorithm number.
  - City CurrentCity = city (of type class City) that corresponds to user input city name
  - ArrayList to store cities visited in citiesVisited variable
  - Add user input to citiesVisited as the first City Object of the array list citiesVisited
  - Creates a stack of visiting Cities to store all cities that have been visited by current City
  - While current city is not final destination Z:
    - Add current City to the stack of cities Visited
    - Flag current City to be visited (true)
    - Set current City to be have the shortest distance to Z (true)
    - Create ArrayList to store all edges of the current City
    - Create a temporary object nextCity of type City, and set it to null value
    - For loop over current City edges and chose the shortest distance one (using either algo1 or 2):
      - If adjacent city of current Edge was visited:

- Continue to check the next Edge (go to next iteration)

- If temporary variable next City null:
  - Update next City to be current adjacent City
- Else
  - If algorithm input number was 1:
    - Execute algorithm 1 by checking if direct distance to Z of current City is bigger than direct distance to Z of adjacent City: if so, current City is updated to be adjacent City
  - Else:
    - Execute algorithm 2 by doing the same thing as algorithm 1, except for when comparing the two distances, the weight is also added to both values before comparing them (using a helper class to tie up the winning City and its total of weight + direct distance to Z.

- If next City is still null:
  - Pop() the last city in the stack
  - Flag the popped City to be false as the shortest distant City
  - Pop() the last City in the stack and assign it to next City
- Add name of next City to the Array List of cities visited
- Update current City to be next City

- If name of current City is Z:
  - Set it to be the shortest distant City (setItTheShortest(true).)
- Build the output String to be printed out
- Prints out the String results.