Java Code

```
1  /*
2   * Author(s): Javier Reyes & Konrad Skoczylas
3   *
4   * Submission Date: 09/December/2025
5   *
6   * Purpose: Create an executable program that manages and query a simple map of sev
en tourist
7   * sites in Tokyo. It models the sites and the distances between some of them using
arrays and
8   * an adjacency matrix.
9   *
10  * Methods:
11  * Search: search for a site by its code or name to return its details and coordina
tes
12  * Insert: inserts or updates the distance between two sites
13  * All Connections(allCons): Display direct neighbors from a specefied site along w
ith the distances
14  * Closest: find the closest directly connected site to a specified site
15  *
16  * Component    ----------------------> Description
17  * Class Tokyo ----------------------> The main class that encapsulates the data an
d methods for the tokyo site
18  *
19  * siteCode[]   ----------------------> An Array of String site codes(i.e "A", "B",
"C").
20  *
21  * siteName[]     ----------------------> An array of String site names(i.e "Shibu
ya Crossing", "Meiji Shrine")
22  *
23  * x[] , y[]      ----------------------> Arrays of int representing the (x,y) coo
rdinates of each site.
24  *
25  * dist[][]           ----------------------> A 7×7 integer array (adjacency matri
x) storing the distance between sites.
26  *                                                                              A va
lue greater than 0 indicates a direct connection and the
27  *                                                                              dist
ance (in KM). A value of 0 indicates no direct connection is defined
28  *                                                                              .It
is symmetrical (dist[i][j] == dist[j][i]).
29  * initMap()    ----------------------> Initializes the dist array with the predef
ined direct
30  *                                                                              conn
ections and distances between the 7 sites.
31  *
32  * menu()         ----------------------> Displays the interactive main menu and
handles user
33  *                                                                              inpu
t for selecting program functionality.
34  *
35  * search(String searchIndex) --------->   Searches for a site by its code or name
and prints its
```

```
36    *                                                                                cod
e, name, and coordinates. Returns the array index of the
37    *                                                                                site
if found, otherwise returns -1.
38    *
39    * insert(int i, int j, int distance)-->Sets the distance between the site at index
i and the site at index j
40    *                                                                                to d
istance in the dist matrix (bidirectional).
41    *
42    * allCons(int i)  --------------------> Prints a list of all sites directly connec
ted to the site at
43    *                                                                              index i
and the distance to each.
44    *
45    * closest(int i)  --------------------> Finds and prints the name and distance of
the closest site that is
46    *                                                                                dire
ctly connected to the site at index i.
47    *
48    * */
49   import java.util.Scanner;
50
51   public class Tokyo {
52       private String[] siteCode = {"A", "B", "C", "D", "E", "F", "G"};
53       private String[] siteName = {"Shibuya Crossing","Meiji Shrine","Tokyo Tower","A
kihabara","Senso-Ji Temple","Ueno Park","Tokyo Skytree"};
54       private int[] x = {3, 4, 5, 8, 9, 6, 11};
55       private int[] y = {7, 1, 3, 8, 7, 2, 2};
56       static Scanner sc = new Scanner(System.in);
57
58       private int[][] dist = new int[7][7];
59
60       public Tokyo() {
61           initMap();
62       }
63
64       private void initMap() {
65           dist[0][1] = 4; dist[1][0] = 4;
66           dist[1][2] = 3; dist[2][1] = 3;
67           dist[0][5] = 2; dist[5][0] = 2;
68           dist[5][4] = 2; dist[4][5] = 2;
69           dist[4][6] = 6; dist[6][4] = 6;
70           dist[2][6] = 5; dist[6][2] = 5;
71           dist[5][3] = 3; dist[3][5] = 3;
72           dist[3][6] = 4; dist[6][3] = 4;
73       }
74
75       public void start() {
76           menu();
77       }
78
79       private void menu() {
80           int choice = 0;
81           do {
```

```
82              System.out.println("----------------------------------------");
83              System.out.println("1. Search");
84              System.out.println("2. Insert");
85              System.out.println("3. All Connections");
86              System.out.println("4. Closest");
87              System.out.println("0. Exit");
88              System.out.print("Enter a choice: ");
89              choice = sc.nextInt();
90              System.out.println("----------------------------------------");
91              switch(choice) {
92                  case 1 -> doSearch();
93                  case 2 -> doInsert();
94                  case 3 -> doAllCons();
95                  case 4 -> doClosest();
96                  case 0 -> System.out.println("Exit the program");
97                  default -> System.out.println("Enter a valid choice");
98              }
99          } while(choice != 0);
100     }
101
102     //-------------------------------------------------------------------------------
--------------------//
103
104     private void doSearch() {
105         sc.nextLine();
106         System.out.print("Enter search index: ");
107         String searchIndex = sc.nextLine();
108         search(searchIndex);
109     }
110
111     private int search(String searchIndex) {
112         for (int i = 0; i < 7; i++) {
113             if (siteCode[i].equals(searchIndex) || siteName[i].equals(searchIndex))
{
114                 System.out.println("Found " + siteCode[i] + "  " + siteName[i]);
115                 System.out.println("Coordinates (" + x[i] + "," + y[i] + ")");
116                 return i;
117             }
118         }
119         System.out.println("Site not found");
120         return -1;
121     }
122
123     //-------------------------------------------------------------------------------
--------------------//
124
125     private void doInsert() {
126         sc.nextLine();
127         System.out.print("Enter site index i (0-6): ");
128         int i = sc.nextInt();
129         System.out.print("Enter site index j (0-6): ");
130         int j = sc.nextInt();
131         System.out.print("Enter Distance: ");
132         int distance = sc.nextInt();
133         insert(i, j, distance);
```

```
134         }
135
136         private void insert(int i, int j, int distance) {
137             if (i >= 0 && i < 7 && j >= 0 && j < 7) {
138                 dist[i][j] = distance;
139                 dist[j][i] = distance;
140                 System.out.println("Connection added " + distance + "KM");
141             } else {
142                 System.out.println("Invalid site indices");
143             }
144         }
145
146         //------------------------------------------------------------------------------
--------------------//
147
148         private void doAllCons() {
149             sc.nextLine();
150             System.out.print("Enter site index (0-6): ");
151             int index = sc.nextInt();
152             allCons(index);
153         }
154
155         private void allCons(int i) {
156             if (i >= 0 && i < 7) {
157                 System.out.println("Connections from " + siteName[i] + ":");
158                 boolean hasConnections = false;
159                 for (int j = 0; j < 7; j++) {
160                     if (dist[i][j] > 0) {
161                         System.out.println(siteName[i] + " -> " + siteName[j] + ": " +
dist[i][j] + "KM");
162                         hasConnections = true;
163                     }
164                 }
165                 if (!hasConnections) {
166                     System.out.println("No connections found");
167                 }
168             } else {
169                 System.out.println("Invalid site index");
170             }
171         }
172
173         //------------------------------------------------------------------------------
--------------------//
174
175         private void doClosest() {
176             sc.nextLine();
177             System.out.print("Enter site index (0-6): ");
178             int index = sc.nextInt();
179             closest(index);
180         }
181
182         private int closest(int i) {
183             if (i >= 0 && i < 7) {
184                 int minDistance = 999;
185                 int closestIndex = -1;
```

```
186                 for (int j = 0; j < 7; j++) {
187                     if (dist[i][j] > 0 && dist[i][j] < minDistance) {
188                         minDistance = dist[i][j];
189                         closestIndex = j;
190                     }
191                 }
192                 if (closestIndex != -1) {
193                     System.out.println("Closest: " + siteName[closestIndex] + " at the
distance of " + minDistance + "KM");
194                 } else {
195                     System.out.println("No connections found from " + siteName[i]);
196                 }
197                 return closestIndex;
198             } else {
199                 System.out.println("Invalid site index");
200             }
201             return -1;
202         } }
```