JavaCode\GraphDriver.java

```java
import java.util.Scanner;

public class GraphDriver
{

    /**
     * Displays a menu of options and handles user input to perform graph operations
     * @param input Scanner for reading user input
     * @param siteAmount Number of sites in the graph
     * @param SitesArray Array containing all site objects
     * @param edges 2D array representing connections between sites with weights
     */
    public static void Menu(Scanner input, int siteAmount, Sites[] SitesArray, int[][] edges)
    {
        int menuChoice = 0;
        String site;
        String site1;
        String site2;
        int weight;
        while(menuChoice != 5){
            System.out.println("Menu:");
            System.out.println("1. Search for a site");
            System.out.println("2. Insert a connection between two sites");
            System.out.println("3. Display all connections for a site");
            System.out.println("4. Find the closest site to a given site");
            System.out.println("5. Exit");

            menuChoice = input.nextInt();
            input.nextLine(); // Consume newline

            switch(menuChoice)
            {
            case 1:
                System.out.println("Enter site name to search:");
                site = input.nextLine();
                System.out.println(Search(site, siteAmount, SitesArray));
                break;
            case 2:
                System.out.println("Enter site1 name:");
                site1 = input.nextLine();
                System.out.println("Enter site2 name:");
                site2 = input.nextLine();
                System.out.println("Enter weight:");
                weight = input.nextInt();
                Insert(site1, site2, weight, siteAmount, SitesArray, edges);
                break;
            case 3:
                System.out.println("Enter site name to display all connections:");
                site = input.nextLine();
                Allcons(site, siteAmount, SitesArray, edges);
                break;
```

```
            case 4:
                System.out.println("Enter site name to find the closest site:");
                site = input.nextLine();
                Closest(site, siteAmount, SitesArray, edges);
                break;
            case 5:
                System.out.println("Exiting the program.");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
                break;
            }
        }
    }


    /**
     * Searches for a specific site by name and returns its details
     * @param site Name of the site to search for
     * @param siteAmount Total number of sites in the array
     * @param SitesArray Array containing all site objects
     * @return String containing site details or "Site not found"
     */
    public static String Search(String site, int siteAmount, Sites[] SitesArray)
    {
        for(int i = 0; i < siteAmount; i++)
        {
            // Case-insensitive comparison to find the site
            if(site.equalsIgnoreCase(SitesArray[i].getName()))
            {
                return SitesArray[i].toString();
            }
        }
        return "Site not found";
    }

    /**
     * Creates a connection between two sites with a specified weight
     * @param site1 Name of the first site
     * @param site2 Name of the second site
     * @param weight The weight/distance of the connection
     * @param siteAmount Total number of sites
     * @param SitesArray Array containing all site objects
     * @param edges 2D array to store the connection weights
     */
    public static void Insert(String site1, String site2, int weight, int siteAmount,
Sites[] SitesArray, int[][] edges)
    {
        int site1Index = -1;
        int site2Index = -1;

        // Find the indices of both sites in the array
        for (int i = 0; i < siteAmount; i++)
        {
```

```
105
106             if(SitesArray[i].getName().equalsIgnoreCase(site1))
107             {
108                 site1Index = i;
109             }
110             else if(SitesArray[i].getName().equalsIgnoreCase(site2))
111             {
112                 site2Index = i;
113             }
114         }
115
116         // Check if sites are different and both exist
117         if(site1Index != site2Index)
118         {
119             if(site1Index != -1 && site2Index != -1)
120             {
121                 // Create the connection with the specified weight
122                 edges[site1Index][site2Index] = weight;
123             }
124             else
125             {
126                 System.out.println("Sites not found ");
127             }
128         }
129         else
130         {
131             System.out.println("Cant create a connection to same site");
132         }
133     }
134
135     /**
136      * Displays all connections for a specific site
137      * @param site Name of the site to show connections for
138      * @param siteAmount Total number of sites
139      * @param SitesArray Array containing all site objects
140      * @param edges 2D array containing connection weights
141      */
142     public static void Allcons(String site, int siteAmount, Sites[] SitesArray, int[][]
    edges)
143     {
144         int rowIndex = 0;
145
146         for(int i = 0; i < siteAmount; i++)
147         {
148             if(site.equals(SitesArray[i].getName()))
149             {
150                 rowIndex = i;
151                 break;
152             }
153         }
154
155         // Check all possible connections from this site
156         for(int col = 0; col < siteAmount; col++)
157         {
```

```
158             if(col != rowIndex && edges[rowIndex][col] != 0)
159             {
160                 System.out.println(SitesArray[col].toString());
161             }
162         }
163     }
164 
165     /**
166      * Finds and displays the closest connected site to a given site
167      * @param site Name of the site to find closest neighbor for
168      * @param siteAmount Total number of sites
169      * @param SitesArray Array containing all site objects
170      * @param edges 2D array containing connection weights
171      */
172     public static void Closest(String site, int siteAmount, Sites[] SitesArray, int[][]
edges)
173     {
174         int rowIndex = 0;
175         int minDist = Integer.MAX_VALUE;
176         int closestIndex = -1;
177 
178         for(int i = 0; i < siteAmount; i++)
179         {
180             if(site.equals(SitesArray[i].getName()))
181             {
182                 rowIndex = i;
183                 break;
184             }
185         }
186 
187         // Find the connection with minimum weight
188         for(int col = 0; col < siteAmount; col++)
189         {
190             if(col != rowIndex && edges[rowIndex][col] != 0)
191             {
192                  if(col != rowIndex && edges[rowIndex][col] != 0 && edges[rowIndex][col] <
minDist)
193                 {
194                     minDist = edges[rowIndex][col];
195                     closestIndex = col;
196                 }
197             }
198         }
199         System.out.println(SitesArray[closestIndex].toString());
200 
201     }
202 
203     public static void main(String[] args)
204     {
205         final int siteAmount = 10;
206 
207         // Initialize data structures
208         Sites[] sitesArray = new Sites[siteAmount];
209         int[][] edges = new int[siteAmount][siteAmount]; // Adjacency matrix for graph
```

```
210         Scanner input = new Scanner(System.in);
211
212         // Initialize all edges to 0 (no connection)
213         for(int row = 0; row < siteAmount; row++)
214         {
215             for(int col = 0; col < siteAmount; col++)
216             {
217                 edges[row][col] = 0;
218             }
219         }
220
221         // Set up predefined connections between sites
222         edges[0][1] = 3;
223         edges[1][0] = 3;
224         edges[1][2] = 2;
225         edges[2][1] = 2;
226         edges[1][3] = 2;
227         edges[3][1] = 2;
228         edges[2][3] = 4;
229         edges[3][2] = 4;
230         edges[3][4] = 1;
231         edges[4][3] = 1;
232         edges[3][5] = 1;
233         edges[5][3] = 1;
234         edges[4][5] = 2;
235         edges[4][9] = 9;
236         edges[5][4] = 2;
237         edges[5][6] = 2;
238         edges[5][8] = 2;
239         edges[6][5] = 2;
240         edges[6][7] = 3;
241         edges[7][6] = 3;
242         edges[7][8] = 1;
243         edges[8][5] = 1;
244         edges[8][7] = 1;
245         edges[8][9] = 5;
246         edges[9][4] = 9;
247         edges[9][8] = 5;
248
249         // Create site objects with names and coordinates
250         sitesArray[0] = new Sites("Whitehouse", 167, 98);
251         sitesArray[1] = new Sites("Treacys", 201, 134);
252         sitesArray[2] = new Sites("D-Bar", 89, 123);
253         sitesArray[3] = new Sites("Holohans", 156, 78);
254         sitesArray[4] = new Sites("The Baileys", 234, 167);
255         sitesArray[5] = new Sites("The Antique", 189, 45);
256         sitesArray[6] = new Sites("Stamps", 67, 234);
257         sitesArray[7] = new Sites("Rackards", 45, 23);
258         sitesArray[8] = new Sites("Dawsons", 78, 156);
259         sitesArray[9] = new Sites("Donohoes", 123, 89);
260
261         // Start the interactive menu
262         Menu(input, siteAmount, sitesArray, edges);
263
```

```
264         }
265 }
```