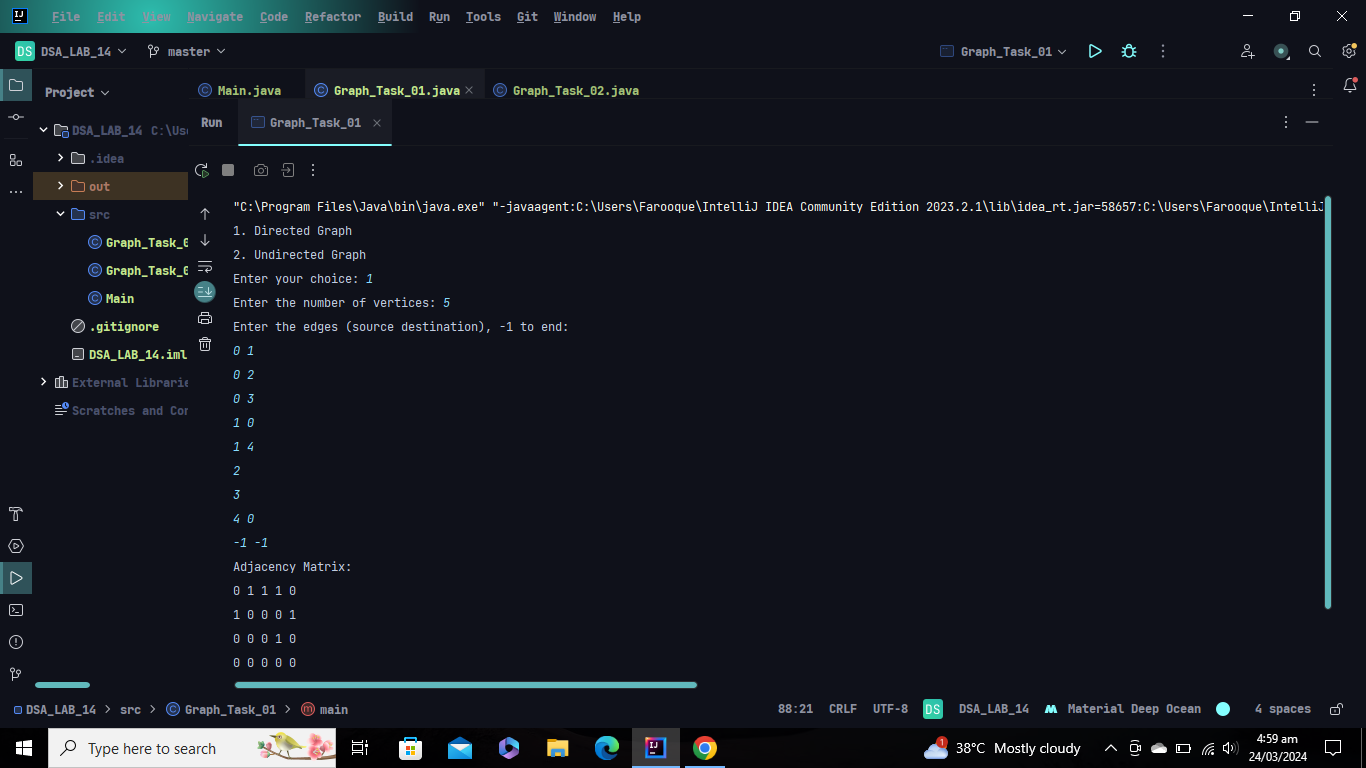
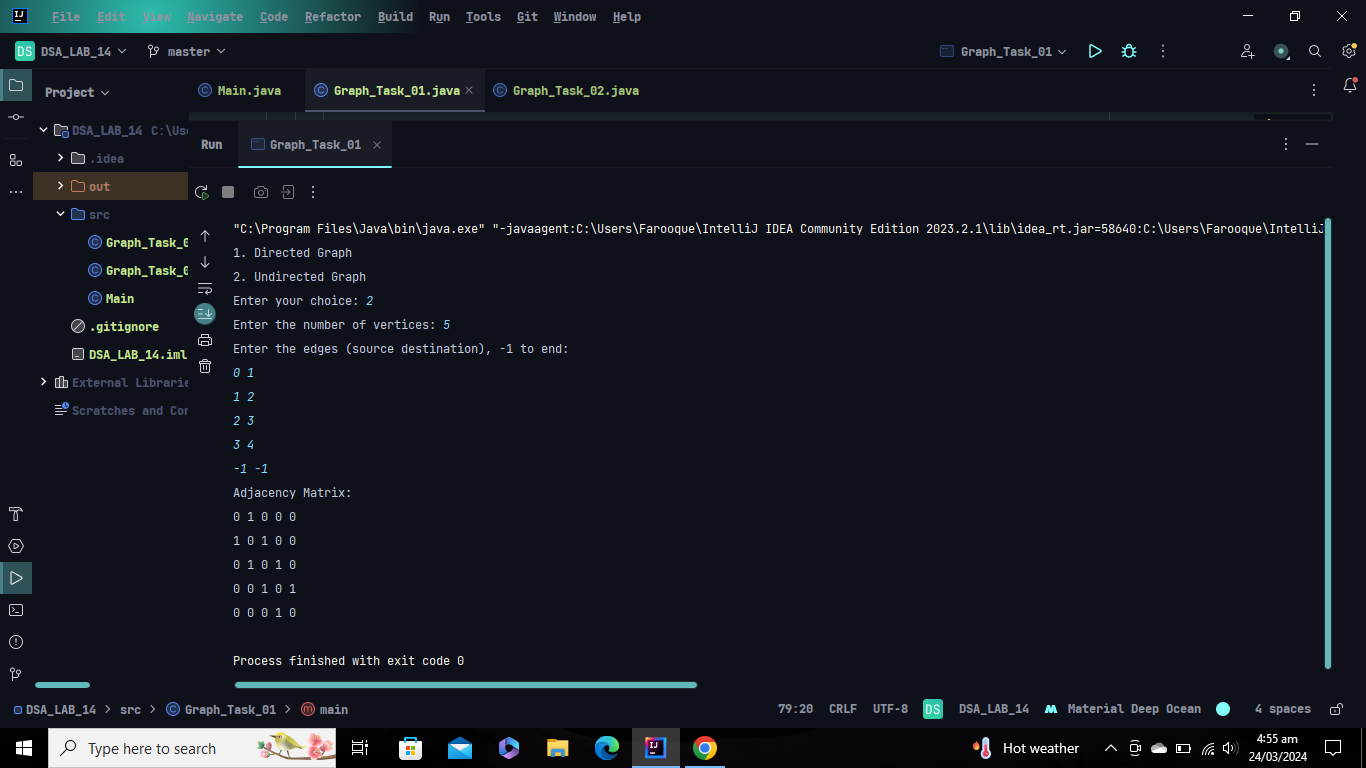
|  |  |
| --- | --- |
| ***Roll No*** | ***22SW040 --> section: 01*** |
| ***Subject*** | ***DSA LAB*** |
| ***LAB 14*** | ***Tasks*** |
| ***Teacher*** | ***Ma’am Muskan*** |

**Task # 1: Implement Graph as an adjacency matrix for a directed, undirected and unweighted graph.**

***Code***

***import* java.util.Scanner;**  
  
***public class* Graph\_Task\_01 {**  
 ***private boolean*[][] adjacencyMatrix;**  
 ***private int* numVertices;**  
  
 ***public* Graph\_Task\_01(*int* numVertices) {**  
 ***this*.numVertices = numVertices;**  
 **adjacencyMatrix = *new boolean*[numVertices][numVertices];**  
 **}**  
  
 ***public void* addEdge(*int* source, *int* destination) {**  
 ***if* (source >= 0 && source < numVertices && destination >= 0 && destination < numVertices) {**  
 **adjacencyMatrix[source][destination] = *true*;**  
 **} *else* {**  
 **System.*out*.println("Invalid edge: " + source + " -> " + destination);**  
 **}**  
 **}**  
  
 ***public void* printGraph() {**  
 ***for* (*int* i = 0; i < numVertices; i++) {**  
 ***for* (*int* j = 0; j < numVertices; j++) {**  
 **System.*out*.print(adjacencyMatrix[i][j] ? "1 " : "0 ");**  
 **}**  
 **System.*out*.println();**  
 **}**  
 **}**  
  
 ***public void* createDirectedGraphFromUserInput() {**  
 **Scanner scanner = *new* Scanner(System.*in*);**  
 **adjacencyMatrix = *new boolean*[numVertices][numVertices];**  
 **System.*out*.println("Enter the edges (source destination), -1 to end: ");**  
 ***while* (*true*) {**  
 ***int* source = scanner.nextInt();**  
 ***int* destination = scanner.nextInt();**  
 ***if* (source == -1 || destination == -1) {**  
 ***break*;**  
 **}**  
 **addEdge(source, destination);**  
 **}**  
 **}**  
  
 ***public void* createUndirectedGraphFromUserInput() {**  
 **Scanner scanner = *new* Scanner(System.*in*);**  
 **adjacencyMatrix = *new boolean*[numVertices][numVertices];**  
 **System.*out*.println("Enter the edges (source destination), -1 to end: ");**  
 ***while* (*true*) {**  
 ***int* source = scanner.nextInt();**  
 ***int* destination = scanner.nextInt();**  
 ***if* (source == -1 || destination == -1) {**  
 ***break*;**  
 **}**  
 **addEdge(source, destination);**  
 **addEdge(destination, source);**  
 **}**  
 **}**  
  
 ***public static void* main(String[] args) {**  
 **Scanner scanner = *new* Scanner(System.*in*);**  
 **System.*out*.println("1. Directed Graph");**  
 **System.*out*.println("2. Undirected Graph");**  
 **System.*out*.print("Enter your choice: ");**  
 ***int* choice = scanner.nextInt();**  
 ***switch* (choice) {**  
 ***case* 1:**  
 **System.*out*.print("Enter the number of vertices: ");**  
 ***int* numVertices1 = scanner.nextInt();**  
 **Graph\_Task\_01 directedGraph = *new* Graph\_Task\_01(numVertices1);**  
 **directedGraph.numVertices = numVertices1;**  
 **directedGraph.createDirectedGraphFromUserInput();**  
 **System.*out*.println("Adjacency Matrix:");**  
 **directedGraph.printGraph();**  
 ***break*;**  
 ***case* 2:**  
 **System.*out*.print("Enter the number of vertices: ");**  
 ***int* numVertices2 = scanner.nextInt();**  
 **Graph\_Task\_01 undirectedGraph = *new* Graph\_Task\_01(numVertices2);**  
 **undirectedGraph.numVertices = numVertices2;**  
 **undirectedGraph.createUndirectedGraphFromUserInput();**  
 **System.*out*.println("Adjacency Matrix:");**  
 **undirectedGraph.printGraph();**  
 ***break*;**  
 ***default*:**  
 **System.*out*.println("Invalid choice!");**  
 **}**  
 **}**  
**}**

***Output***



**Task # 2: Implement Graph as an adjacency list for directed, undirected and unweighted graph. (You can use built in linked list class of java).**

***Code***

***import* java.util.\*;**  
  
***public class* Graph\_Task\_02 {**  
 ***private final Map*<Integer, *List*<Integer>> adjacencyList;**  
 ***private int* numVertices;**  
  
 ***public* Graph\_Task\_02(*int* numVertices) {**  
 ***this*.numVertices = numVertices;**  
 **adjacencyList = *new* HashMap<>();**  
 ***for* (*int* i = 0; i < numVertices; i++) {**  
 **adjacencyList.put(i, *new* LinkedList<>());**  
 **}**  
 **}**  
  
  
 ***public void* addDirectedEdge(*int* source, *int* destination) {**  
 **adjacencyList.get(source).add(destination);**  
 **}**  
  
 ***public void* addUndirectedEdge(*int* source, *int* destination) {**  
 **adjacencyList.get(source).add(destination);**  
 **adjacencyList.get(destination).add(source);**  
 **}**  
  
 ***public void* printGraph() {**  
 ***for* (*int* vertex : adjacencyList.keySet()) {**  
 **System.*out*.print("Vertex " + vertex + " is connected to: ");**  
 ***for* (*int* neighbor : adjacencyList.get(vertex)) {**  
 **System.*out*.print(neighbor + " ");**  
 **}**  
 **System.*out*.println();**  
 **}**  
 **}**  
  
 ***public static void* main(String[] args) {**  
 **Scanner scanner = *new* Scanner(System.*in*);**  
 **System.*out*.print("Enter the number of vertices: ");**  
 ***int* numVertices = scanner.nextInt();**  
 **Graph\_Task\_02 graph = *new* Graph\_Task\_02(numVertices);**  
 **System.*out*.println("Choose the type of graph:");**  
 **System.*out*.println("1. Directed");**  
 **System.*out*.println("2. Undirected");**  
 **System.*out*.print("Enter your choice: ");**  
 ***int* choice = scanner.nextInt();**  
 ***switch* (choice) {**  
 ***case* 1 -> *createDirectedGraph*(graph, scanner);**  
 ***case* 2 -> *createUndirectedGraph*(graph, scanner);**  
 ***default* -> {**  
 **System.*out*.println("Invalid choice!");**  
 ***return*;**  
 **}**  
 **}**  
 **System.*out*.println("Adjacency List:");**  
 **graph.printGraph();**  
 **}**  
  
 ***private static void* createDirectedGraph(Graph\_Task\_02 graph, Scanner scanner) {**  
 **System.*out*.println("Enter the edges (source destination): ");**  
 ***while* (*true*) {**  
 ***int* source = scanner.nextInt();**  
 ***int* destination = scanner.nextInt();**  
 ***if* (source == -1 || destination == -1) {**  
 ***break*;**  
 **}**  
 **graph.addDirectedEdge(source, destination);**  
 **}**  
 **}**  
  
 ***private static void* createUndirectedGraph(Graph\_Task\_02 graph, Scanner scanner) {**  
 **System.*out*.println("Enter the edges (source destination): ");**  
 ***while* (*true*) {**  
 ***int* source = scanner.nextInt();**  
 ***int* destination = scanner.nextInt();**  
 ***if* (source == -1 || destination == -1) {**  
 ***break*;**  
 **}**  
 **graph.addUndirectedEdge(source, destination);**  
 **}**  
 **}**  
**}**

***Output***

