Specifications:

The UI class has the following methods:

Attributes:

__n : int, the number of vertices in the graph.
__m : int, the number of edges in the graph.
__graph: calls the Graph class

Methods:

printMenu(self):

Prints out the list of available options for the user
Returns nothing

addVertex(self):

Adds the vertex to the graph using the Graph object's add_vertex method
Increments the number of vertices
Prints out "Vertex was added" if the vertex was successfully added or "Vertex already exists" if the vertex already exists in the graph
Throws ValueError if the input vertex number is negative

removeVertex(self):

Removes the vertex from the graph using the Graph object's remove_vertex method
Decrements the number of vertices
Prints out "Vertex was removed" if the vertex was successfully removed or "Vertex does not exist" if the vertex does not exist in the graph
Throws ValueError if the input vertex number is negative

addEdge(self):

Asks the user to input two vertex numbers and a cost
Adds an edge between the two vertices using the Graph object's add_edge method
Increments the number of edges if the edge is successfully added
Prints out "Edge was added" if the edge was successfully added, "This edge already exists" if the edge already exists, or "One or both of the vertices you entered do not exist" if one or both of the vertices do not exist in the graph
Throws ValueError if the input vertex numbers or cost are negative

removeEdge(self):

Removes an edge between the two vertices using the Graph object's remove_edge method
Decrements the number of edges if the edge is successfully removed
Prints out "Edge was removed" if the edge was successfully removed or "Edge does not exist" if the edge does not exist in the graph
Throws ValueError if the input vertex numbers are negative

getNumberOfVertices(self):

Prints out the number of vertices in the graph using the Graph object's get_number_of_vertices method
Returns nothing

printVertices(self):

Prints out the set of vertices in the graph using the Graph object's get_vertices method
Returns nothing

isEdge(self):
Prompts the user to input two vertex numbers
Verifies if there is an edge between the two vertices using the Graph object's is_edge method
Prints out "There is an edge between the vertices" if there is an edge or "There is no edge between the vertices" if there is no edge
Throws ValueError if the input vertex numbers are negative

GetInDegreeOfVertex(self):
Prompts the user to input a vertex number
Gets the in-degree of the vertex using the Graph object's get_in_degree method
Prints out the in-degree of the vertex
Throws ValueError if the input vertex number is negative

GetOutDegreeOfVertex(self):
Gets the out-degree of the vertex using the Graph object's get_out_degree method
Prints out the out-degree of the vertex
Throws ValueError if the input vertex number is negative

getOutboundEdgesOfAVertex(self):
Prompts the user to input a vertex number
Parses the set of outbound edges of the vertex using the Graph object's parse_outbound_edges method
Prints out the set of outbound edges of the vertex

getInboundEdgesOfAVertex(self):
Prompts the user to input a vertex number
Parses the set of inbound edges of the vertex using the Graph object's parse_inbound_edges method
Prints out the set of inbound edges of the vertex

costOfEdge(self):
This function takes two vertices as input and prints the cost of the edge between them.

changeCostOfEdge(self):
This function takes two vertices and a new cost as input and changes the cost of the edge between the two vertices.

readFromFile(self):
This function takes the name of the input file as input, reads the file and creates a graph object with the data.

writeToFile(self):
This function takes the name of the file where the graph will be saved as input and writes the graph data to the file.

generateRandomGraph(self):
This function generates a random graph with a given number of vertices and edges.

Class Graph: A class to represent a directed graph.

Attributes:

__number_of_vertices : int, the number of vertices in the graph.
__vertices : list, a list of all vertices in the graph.
__list_of_predecessors : list of lists, a list containing for each vertex in the graph the list of its predecessors.
__list_of_successors : list of lists, a list containing for each vertex in the graph the list of its successors.
__cost : dict, a dictionary containing for each edge in the graph its cost.

Methods:

is_vertex(vertex: int) :
bool
Checks if a vertex is in the graph.

is_edge(vertex_1: int, vertex_2: int) :
bool
Checks if there is an edge from vertex_1 to vertex_2 in the graph.

add_vertex(vertex: int) :
int
Adds a vertex to the graph.

add_edge(vertex_1: int, vertex_2: int, edge_cost: float) :
int
Adds an edge from vertex_1 to vertex_2 with a given cost to the graph.

remove_edge(vertex_1: int, vertex_2: int) :
int
Removes the edge from vertex_1 to vertex_2 from the graph.

remove_vertex(vertex_to_remove: int) :
int
Removes a vertex from the graph.

get_edge_cost(vertex1: int, vertex2: int) :
float
Returns the cost of the edge from vertex1 to vertex2.

set_edge_cost(vertex1: int, vertex2: int, new_cost: float) :
int
Changes the cost of the edge from vertex1 to vertex2 to a new value.

get_number_of_vertices() :
int
Returns the number of vertices in the graph.

get_vertices() :
    list
    Returns a list of all vertices in the graph.

get_predecessors() :
    list of lists
    Returns a list containing for each vertex in the graph the list of its predecessors.

get_successors() :
    list of lists
    Returns a list containing for each vertex in the graph the list of its successors.

get_cost() :
    dict
    Returns a dictionary containing for each edge in the graph its cost.

service_in_degree(vertex: int) :
    Tuple[int, List[int]]
    Returns a tuple containing the in-degree of the vertex and a list of its predecessors.

service_out_degree(vertex: int) :
    Tuple[int, List[int]]
    Returns a tuple containing the out-degree of the vertex and a list of its successors.

service_outbound_edges(vertex: int) :
    List[int]
    Returns a list of all vertices that can be reached from the given vertex.

service_inbound_edges(vertex: int) :
    List[int]
    Returns a list of all vertices that can reach the given vertex.

service_change_cost_of_edge(vertex_1: int, vertex_2: int, new_cost: float) :
    None
    Changes the cost of the edge from vertex_1 to vertex_2 to a new value.

get_list_of_costs() :
    dict
    Returns a dictionary containing for each edge in the graph its cost.

This is a Python implementation of a directed graph data structure with various methods for adding and removing vertices and edges, as well as retrieving information about the graph such as the number of vertices, vertices themselves, and the costs of edges.
The **Graph** class has an initializer **__init__** which takes two arguments, **n** and **m**, representing the number of vertices and edges in the graph respectively. This implementation allows for the creation and manipulation of directed graphs with the ability to add and remove vertices and edges, as well as retrieve information about the graph and its properties.