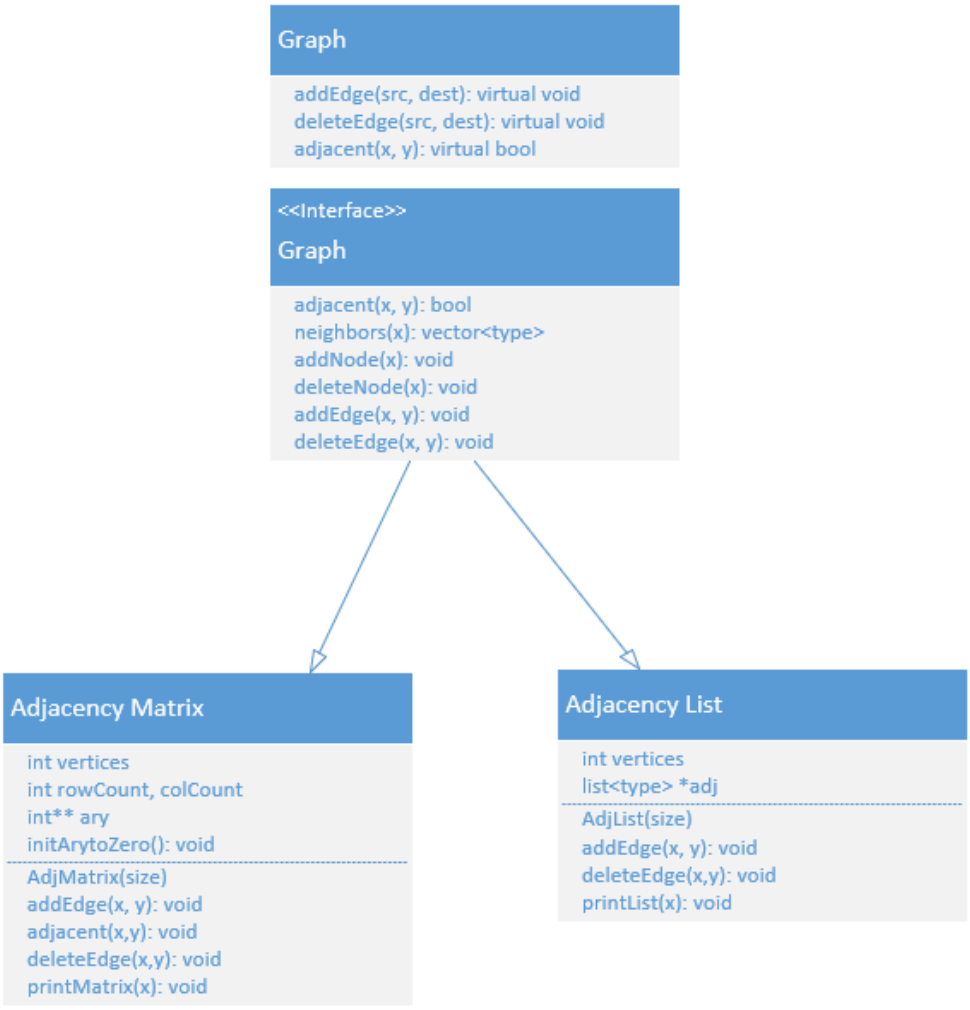
Chung-thuy Pham
00066242
cpham@my.athens.edu



Chung-thuy Pham
00066242
cpham@my.athens.edu

```
// OOP_Assignment1_GraphADT.cpp
//

#include "pch.h"
#include <iostream>
#include <vector>
#include <list>
#include <algorithm>
#include <iterator>


using namespace std;


//template <class type>
class Graph {
	// No need to declare private variables.
	// All about 'function'alities, not data.

public:
	//virtual bool adjacent(type x, type y) = 0;		// Is there a path from x to y
	//virtual vector<type> neighbors(type x) = 0;			// Return a vector
containing the nodes have an edge from x to elements in the vector
	virtual void addEdge(int src, int dest) = 0;
	virtual void deleteEdge(int src, int dest) = 0;
	virtual bool adjacent(int x, int y) = 0;
};

template <class type>
class AdjList : public Graph
{
private:
	int vertices;
	list<type> *adj;
public:
	AdjList(int size)
	{
		vertices = size;
		this->vertices = vertices;
		adj = new list<type>[vertices];
	}

	void addEdge(type x, type y) {
		adj[x].push_back(y); // Add y to x's list
	}

	void deleteEdge(type x, type y) {
		adj[x].erase(y);
	}

	void printList(type x) {
		for (list<int>::iterator it = adj[x].begin(); it != adj[x].end(); ++it)
			cout << *it << ' ';
	}

};

template <class type>
class AdjMatrix : public Graph
```

Chung-thuy Pham
00066242
cpham@my.athens.edu

```
{
private:
     int vertices; // Nodes
     int rowCount, colCount;
     int** ary; // Look up what ** does. Mayone one pointer to col, one to row (?)
     //int* ary = new int[aSize]; how to do 1D dynamic array
     void initArytoZero()
     {
          for (int r = 0; r < rowCount; r++) {
               for (int c = 0; c < colCount; c++)
                    ary[r][c] = 0;
          }
     }
public:
     // Constructor
     AdjMatrix(int size) {
          vertices = size;
          rowCount = colCount = vertices;
          ary = new type*[rowCount];  // Look up what this syntax is actually doing
          for (int i = 0; i < rowCount; i++)
               ary[i] = new int[colCount]; // Look up what this syntax is actually doing
          initArytoZero();
     }


     void addEdge(type x, type y) {
          ary[x][y] = 1;
     }

     void deleteEdge(type x, type y) {
          ary[x][y] = 0;
     }

     bool adjacent(type x, type y) {
          bool flag;
          if (ary[x][y] == 1)
               flag = true;
          /*else if (ary[y][x] == 1) //Add this if graph is undirected
               flag = true;*/
          else
               flag = false;

          return flag;
     }



     void printMatrix() {     // for debug
          for (int r = 0; r < rowCount; r++) {
               for (int c = 0; c < colCount; c++) {
                    cout << ary[r][c] << " ";
               }
               cout << endl;
          }
     }
};



int main() {
```

Chung-thuy Pham
00066242
cpham@my.athens.edu

```
     AdjMatrix<int> mat1(5);
     mat1.addEdge(0, 2);
     mat1.addEdge(0, 4);
     mat1.addEdge(1, 0);
     mat1.addEdge(3, 1);
     mat1.addEdge(4, 3);

     if (mat1.adjacent(4, 0))           // for debug
          cout << "Adjacent" << endl;
     else
          cout << "Not adjacent" << endl;

     mat1.printMatrix();

     /* AdjList<int> list1(5);
      list1.addEdge(0, 2);
      list1.addEdge(0, 4);
      list1.addEdge(1, 0);
      list1.addEdge(3, 1);
      list1.addEdge(4, 3);

      list1.printList(4);*/


     return 0;
}
```