Nonlinear Data Structure

http://smartlead.hallym.ac.kr

Instructor: **Jin Kim**
010-6267-8189(033-248-2318)

jinkim@hallym.ac.kr

Office Hours:
Office : 자연대 7618

Lab(Graph)

Instructor: Jin Kim
010-6267-8189(033-248-2318)

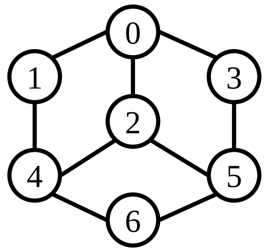
jinkim@hallym.ac.kr
Office Hours:

Graph

GraphMatrix.java

Adjacency Matrix

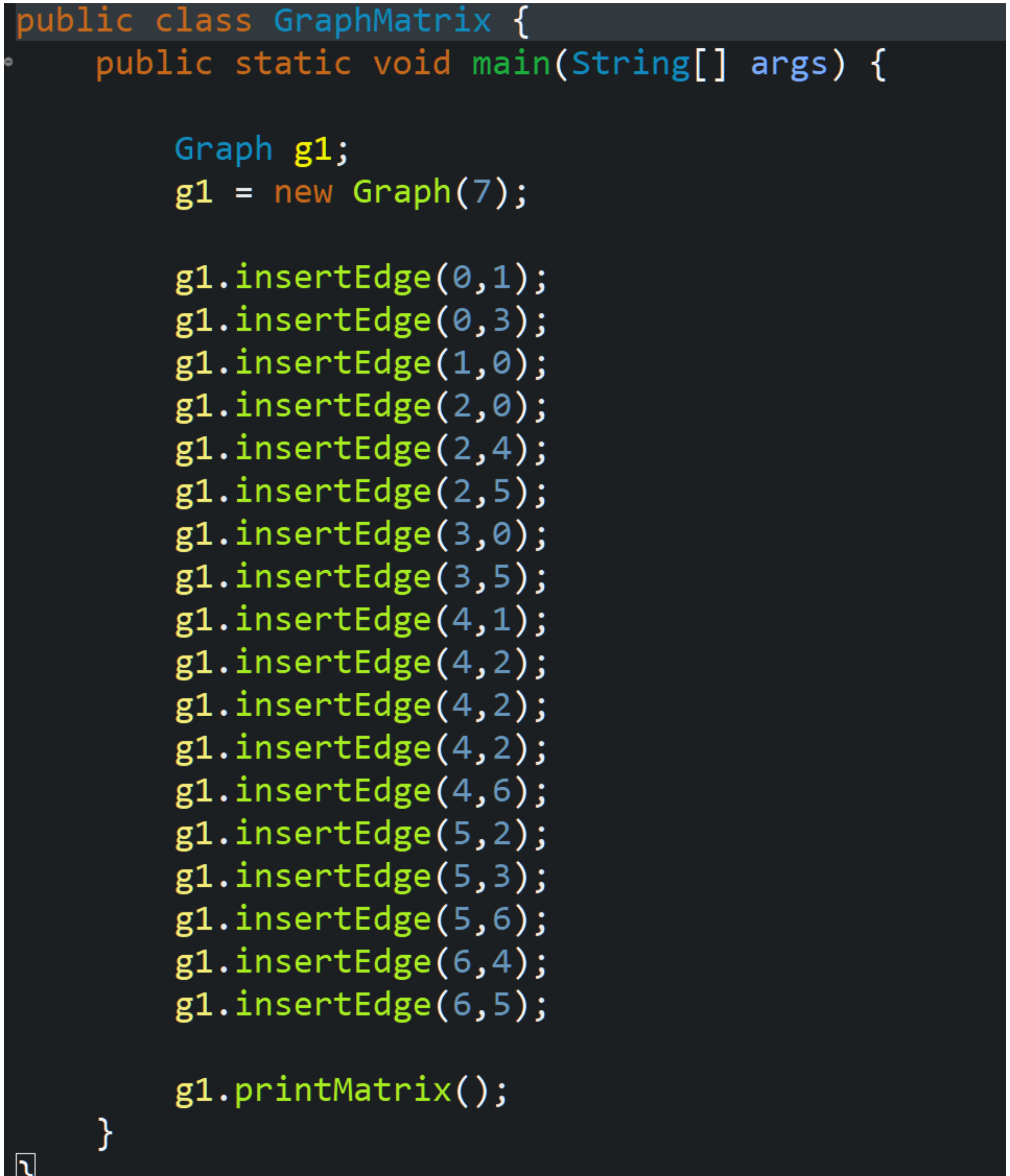
- Represent the graph G1 by adjacency matrix(그래프 g1을 인접행렬로 표현하라.



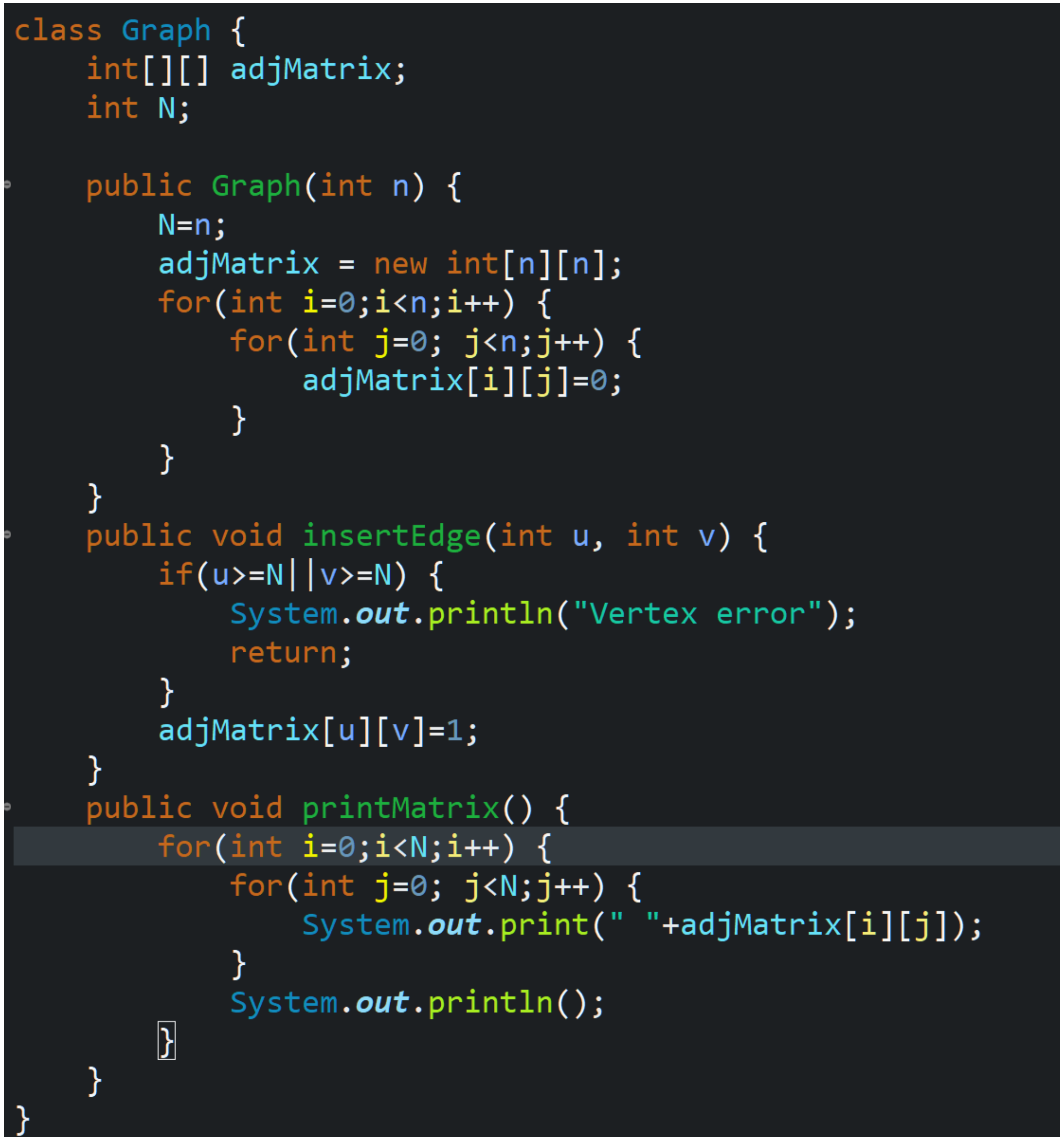
탐색을 위한 그래프 G

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	0	1	0	1	0	0	0
1	1	0	0	0	1	0	0
2	1	0	0	0	1	1	0
3	1	0	0	0	0	1	0
4	0	1	1	0	0	0	1
5	0	0	1	1	0	0	1
6	0	0	0	0	1	1	0

GraphMatrix.java



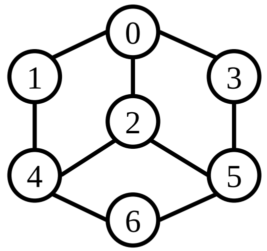
```
public class GraphMatrix {
    public static void main(String[] args) {

        Graph g1;
        g1 = new Graph(7);

        g1.insertEdge(0,1);
        g1.insertEdge(0,3);
        g1.insertEdge(1,0);
        g1.insertEdge(2,0);
        g1.insertEdge(2,4);
        g1.insertEdge(2,5);
        g1.insertEdge(3,0);
        g1.insertEdge(3,5);
        g1.insertEdge(4,1);
        g1.insertEdge(4,2);
        g1.insertEdge(4,2);
        g1.insertEdge(4,2);
        g1.insertEdge(4,6);
        g1.insertEdge(5,2);
        g1.insertEdge(5,3);
        g1.insertEdge(5,6);
        g1.insertEdge(6,4);
        g1.insertEdge(6,5);

        g1.printMatrix();
    }
```



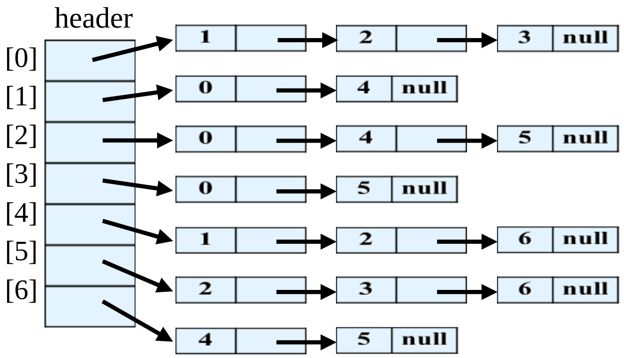
```java
class Graph {
    int[][] adjMatrix;
    int N;

    public Graph(int n) {
        N=n;
        adjMatrix = new int[n][n];
        for(int i=0;i<n;i++) {
            for(int j=0; j<n;j++) {
                adjMatrix[i][j]=0;
            }
        }
    }
    public void insertEdge(int u, int v) {
        if(u>=N||v>=N) {
            System.out.println("Vertex error");
            return;
        }
        adjMatrix[u][v]=1;
    }
    public void printMatrix() {
        for(int i=0;i<N;i++) {
            for(int j=0; j<N;j++) {
                System.out.print(" "+adjMatrix[i][j]);
            }
            System.out.println();
        }
    }
}
```

GraphList.java

- Represent the graph G1 by adjacency list(그래프 g1을 인접리스트로 표현하라)

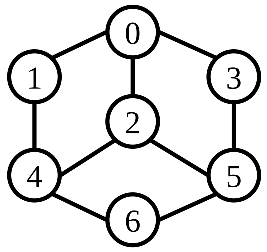


탐색을 위한 그래프 G

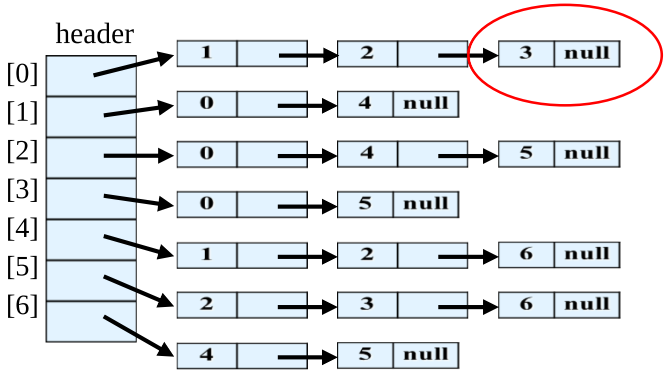


그래프 G에 대한 인접 리스트 표현

tmpNode = new Node(3)

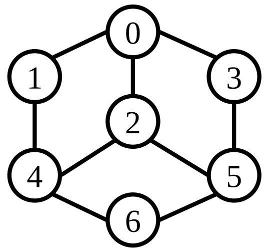


탐색을 위한 그래프 G

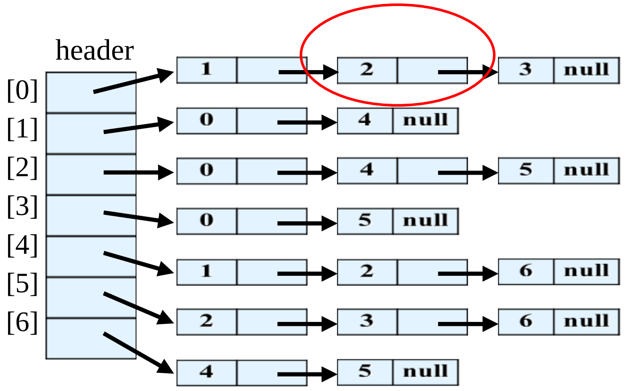


그래프 G에 대한 인접 리스트 표현

tmpNode1= new Node(2, tmpNode)

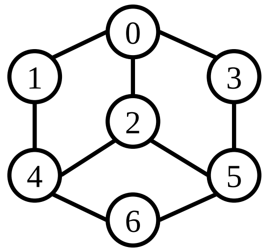


탐색을 위한 그래프 G

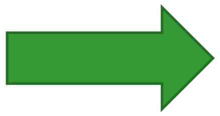


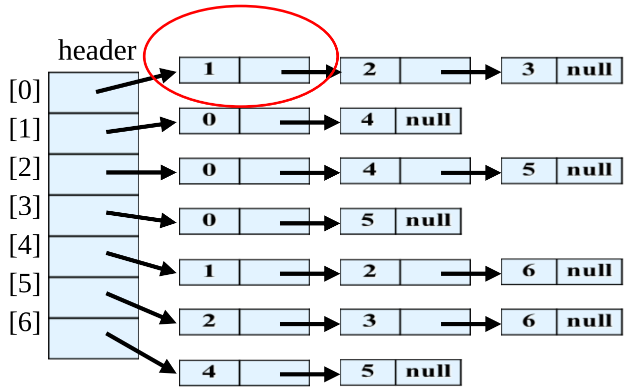
그래프 G에 대한 인접 리스트 표현

tmpNode2= new Node(1, tmpNode1)



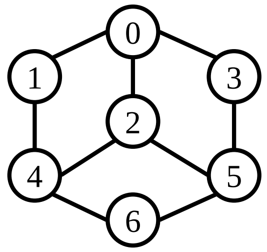
탐색을 위한 그래프 G



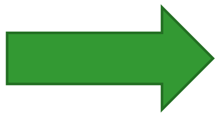


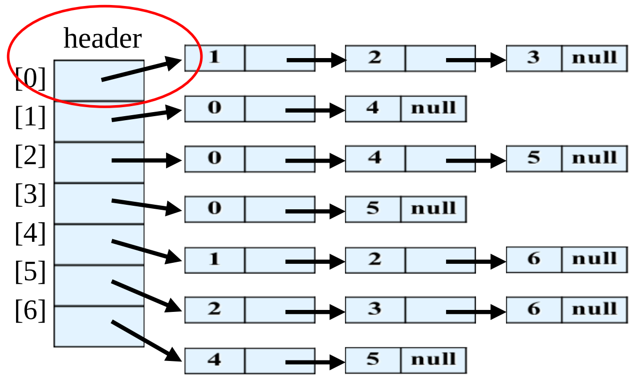
그래프 G에 대한 인접 리스트 표현

header[0]= tmpNode2



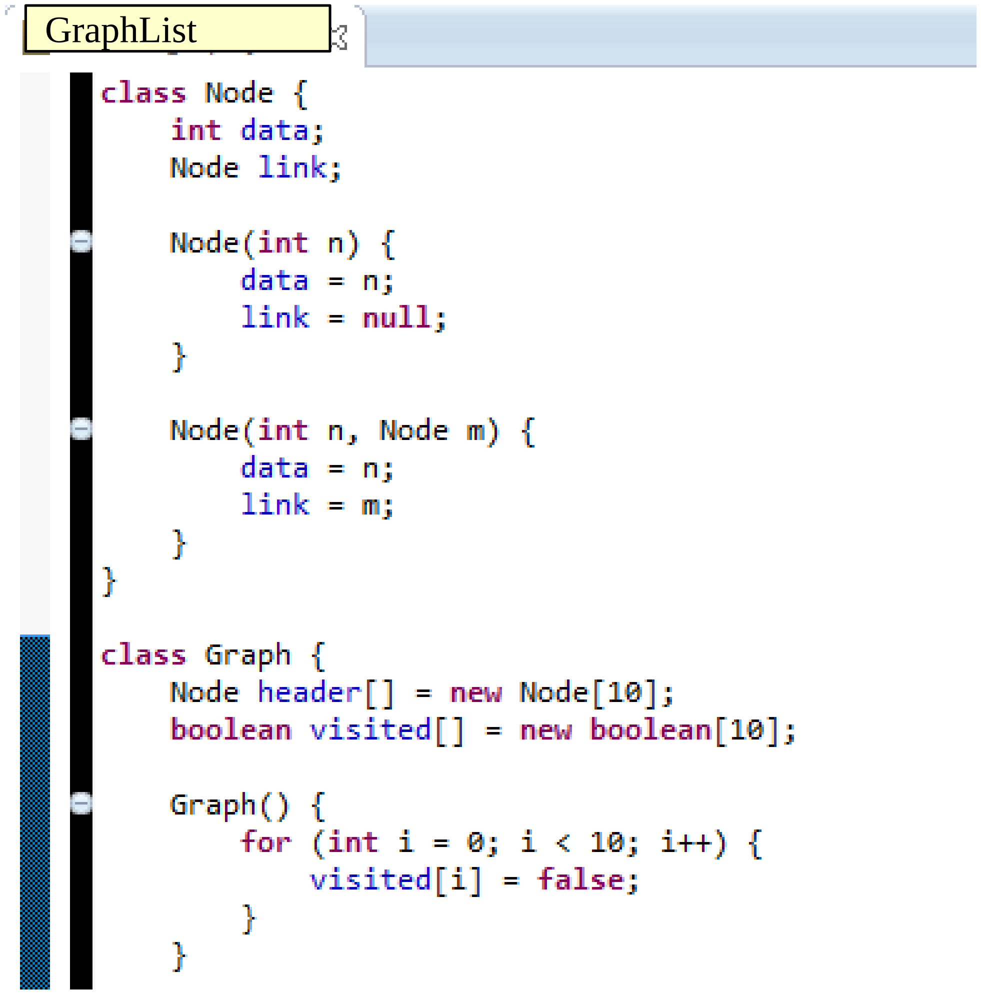
탐색을 위한 그래프 G





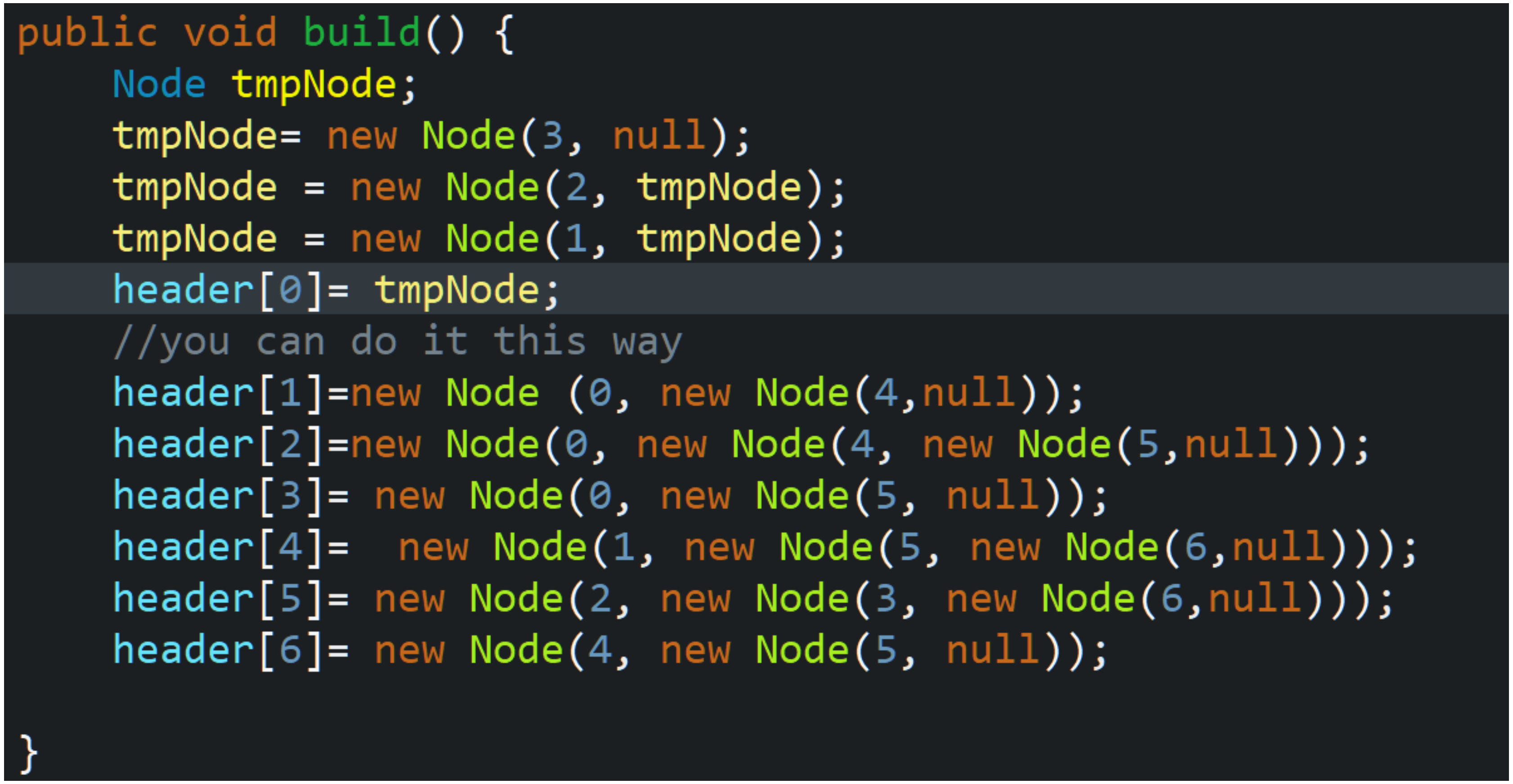
그래프 G에 대한 인접 리스트 표현

GraphList.java

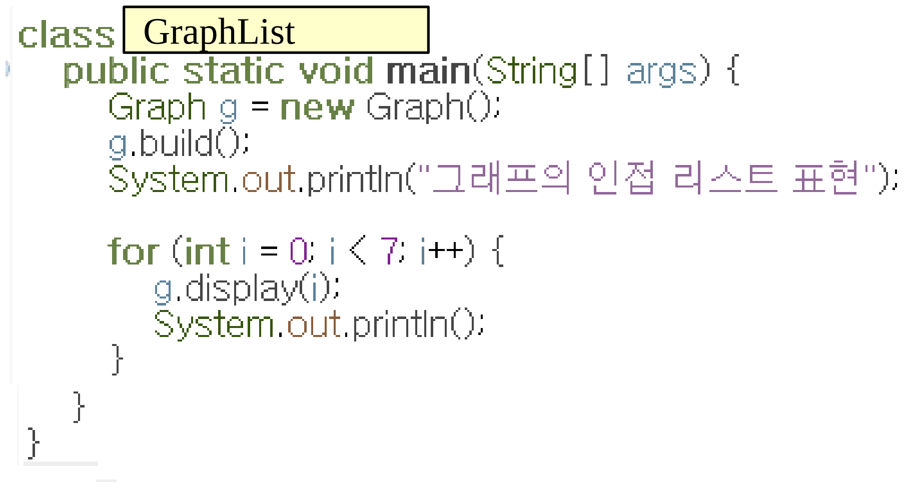


GraphList

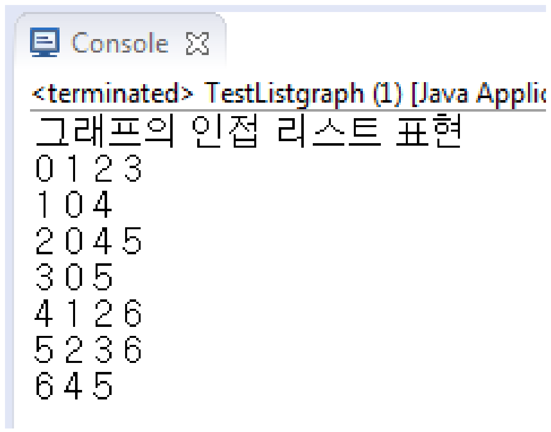
```
class Node {
    int data;
    Node link;

    Node(int n) {
        data = n;
        link = null;
    }

    Node(int n, Node m) {
        data = n;
        link = m;
    }
}

class Graph {
    Node header[] = new Node[10];
    boolean visited[] = new boolean[10];

    Graph() {
        for (int i = 0; i < 10; i++) {
            visited[i] = false;
        }
    }
```



```java
public void build() {
    Node tmpNode;
    tmpNode= new Node(3, null);
    tmpNode = new Node(2, tmpNode);
    tmpNode = new Node(1, tmpNode);
    header[0]= tmpNode;
    //you can do it this way
    header[1]=new Node (0, new Node(4,null));
    header[2]=new Node(0, new Node(4, new Node(5,null)));
    header[3]= new Node(0, new Node(5, null));
    header[4]=  new Node(1, new Node(5, new Node(6,null)));
    header[5]= new Node(2, new Node(3, new Node(6,null)));
    header[6]= new Node(4, new Node(5, null));

}
```



```
class GraphList
    public static void main(String[] args) {
        Graph g = new Graph();
        g.build();
        System.out.println("그래프의 인접 리스트 표현");

        for (int i = 0; i < 7; i++) {
            g.display(i);
            System.out.println();
        }
    }
}
```

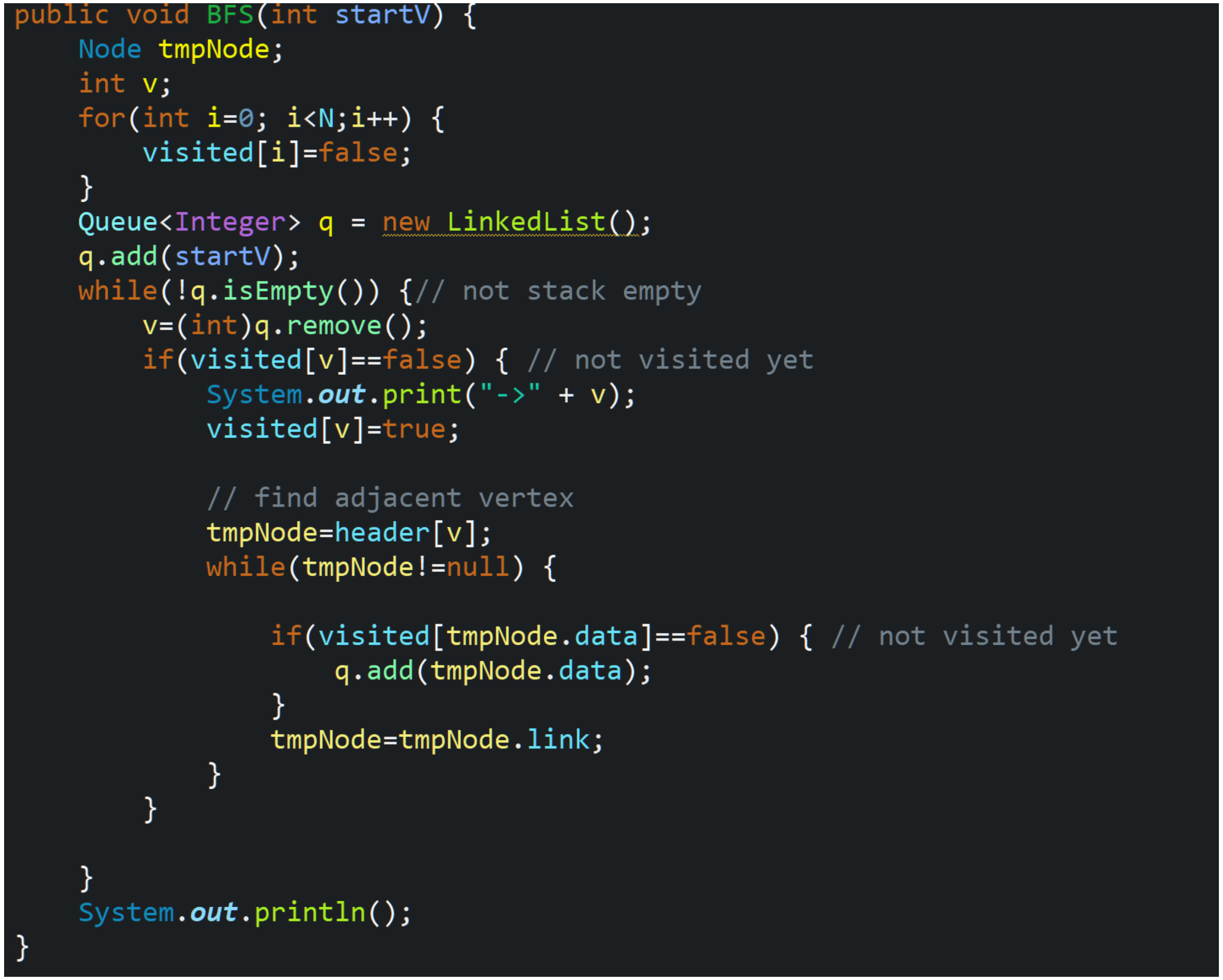


```
<terminated> TestListgraph (1) [Java Applic
그래프의 인접 리스트 표현
0 1 2 3
1 0 4
2 0 4 5
3 0 5
4 1 2 6
5 2 3 6
6 4 5
```

GraphTraversal.java

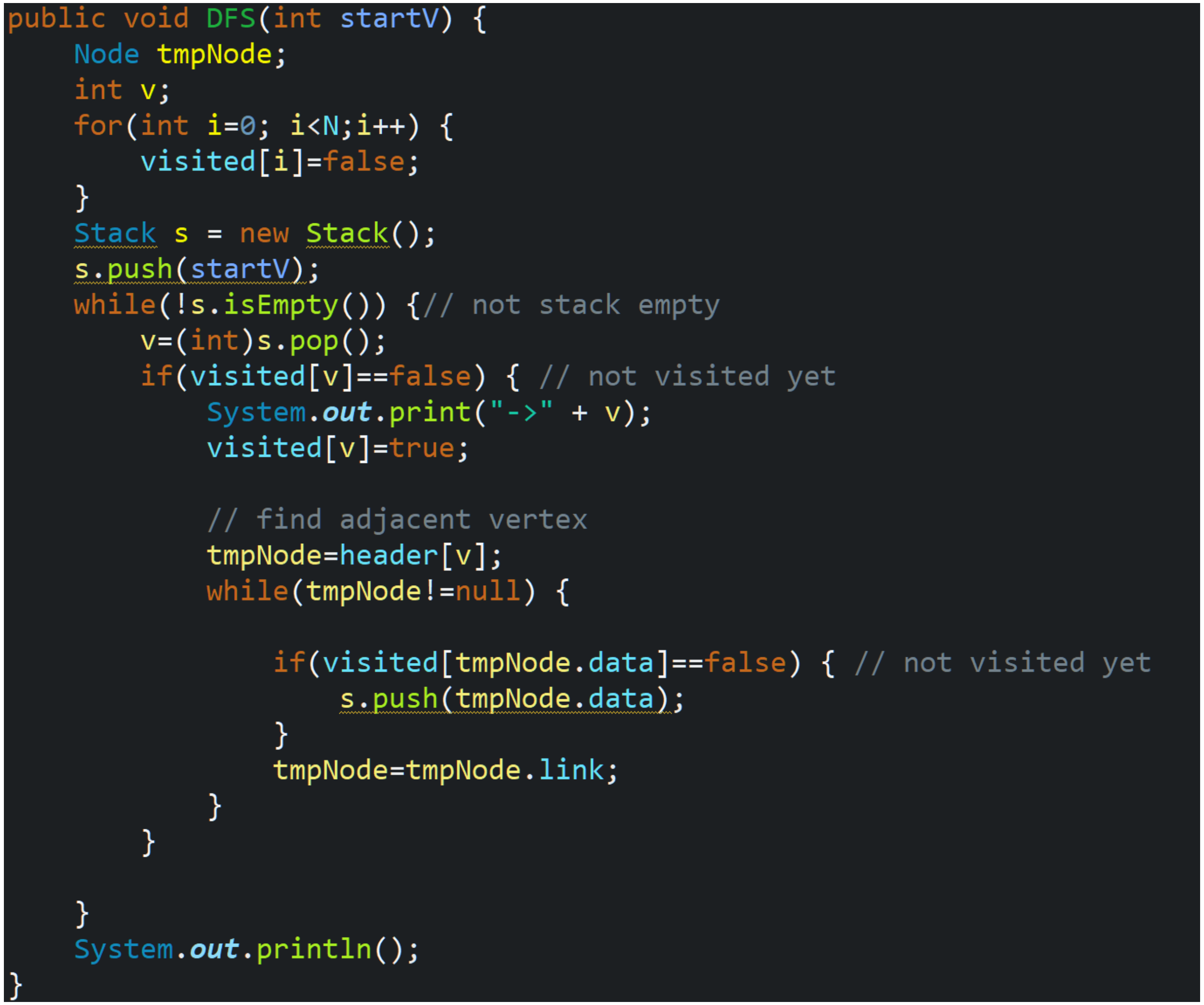
- Breadth First Search(BFS너비우선탐색): Queue
- Depth First Search(DFS깊이우선탐색) : Stack

GraphTraversal.java(BFS)



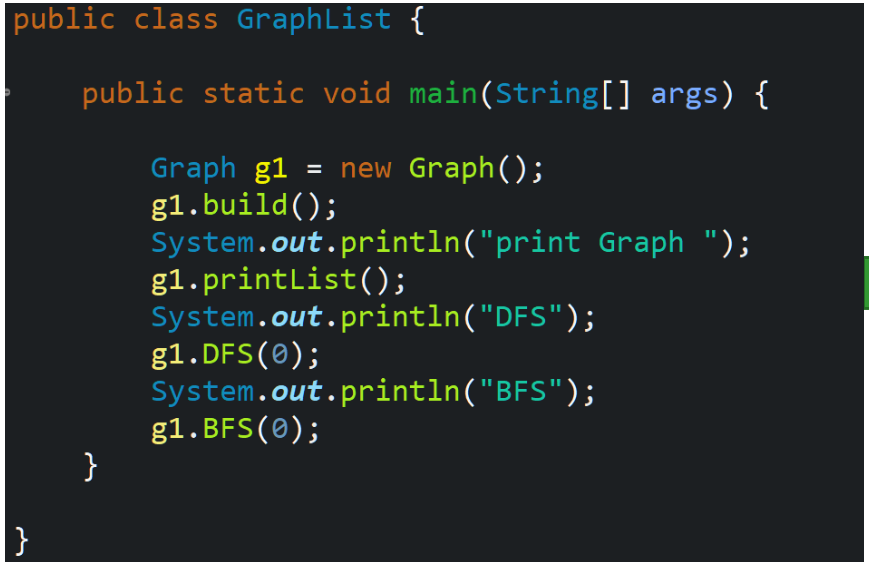
```java
public void BFS(int startV) {
    Node tmpNode;
    int v;
    for(int i=0; i<N;i++) {
        visited[i]=false;
    }
    Queue<Integer> q = new LinkedList();
    q.add(startV);
    while(!q.isEmpty()) {// not stack empty
        v=(int)q.remove();
        if(visited[v]==false) { // not visited yet
            System.out.print("->" + v);
            visited[v]=true;

            // find adjacent vertex
            tmpNode=header[v];
            while(tmpNode!=null) {

                if(visited[tmpNode.data]==false) { // not visited yet
                    q.add(tmpNode.data);
                }
                tmpNode=tmpNode.link;
            }
        }

    }
    System.out.println();
}
```

GraphTraversal.java(DFS)

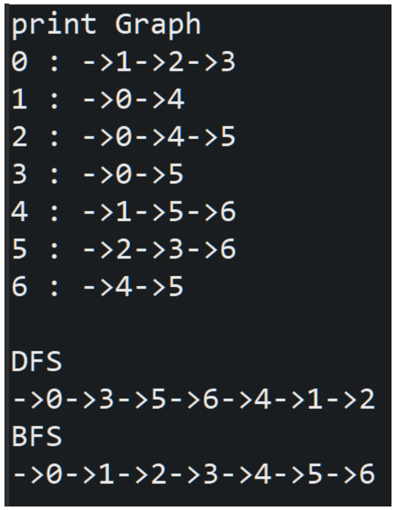


```java
public void DFS(int startV) {
    Node tmpNode;
    int v;
    for(int i=0; i<N;i++) {
        visited[i]=false;
    }
    Stack s = new Stack();
    s.push(startV);
    while(!s.isEmpty()) {// not stack empty
        v=(int)s.pop();
        if(visited[v]==false) { // not visited yet
            System.out.print("->" + v);
            visited[v]=true;

            // find adjacent vertex
            tmpNode=header[v];
            while(tmpNode!=null) {

                if(visited[tmpNode.data]==false) { // not visited yet
                    s.push(tmpNode.data);
                }
                tmpNode=tmpNode.link;
            }
        }

    }
    System.out.println();
}
```

GraphTraversal.java



```java
public class GraphList {

    public static void main(String[] args) {

        Graph g1 = new Graph();
        g1.build();
        System.out.println("print Graph ");
        g1.printList();
        System.out.println("DFS");
        g1.DFS(0);
        System.out.println("BFS");
        g1.BFS(0);
    }

}
```

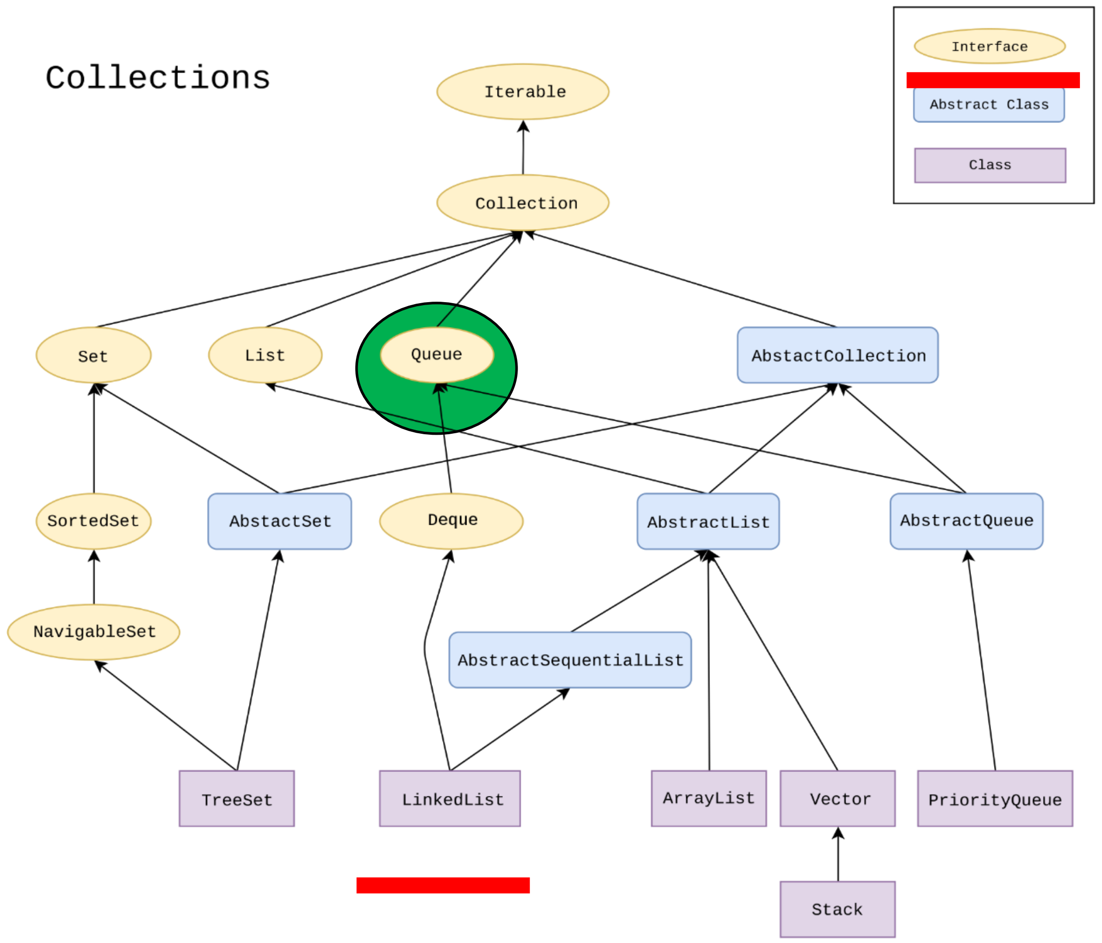


```
print Graph
0 : ->1->2->3
1 : ->0->4
2 : ->0->4->5
3 : ->0->5
4 : ->1->5->6
5 : ->2->3->6
6 : ->4->5


DFS
->0->3->5->6->4->1->2
BFS
->0->1->2->3->4->5->6
```

Java Queue

- https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html
- java.util
- **Interface Queue<E> // Queue는 class가 아니라, Interface이다. 따라서 Queue로 객체를 생성할 수 없고, Queue의 내용을 구체화해야 한다.**

Java collections frameworks



Methods in Queue

- **add()-** enqueue. This method is used to add elements at the tail of queue. More specifically, at the last of linkedlist if it is used, or according to the priority in case of priority queue implementation.
- **peek()-** This method is used to view the head of queue without removing it. It returns Null if the queue is empty.
- **element()-** This method is similar to peek(). It throws *NoSuchElementException*when the queue is empty.
- **remove()-** Dequeue. This method removes and returns the head of the queue. It throws *NoSuchElementException* when the queue is empty.
- **poll()-** Dequeue. This method removes and returns the head of the queue. It returns null if the queue is empty.
- **size()-** This method return the no. of elements in the queue.

QueueExample Program

```
import java.util.LinkedList;
import java.util.Queue;
public class QueueExample {
  public static void main(String[] args)   {
    Queue<Integer> q = new LinkedList<>();
    // Adds elements {0, 1, 2, 3, 4} to queue
    for (int i=0; i<5; i++)
     q.add(i);
    // Display contents of the queue.
    System.out.println("Elements of queue-"+q);
    // To remove the head of queue.
    int removedele = q.remove();
    System.out.println("removed element-" + removedele);
    System.out.println(q);
    // To view the head of queue
    int head = q.peek();
    System.out.println("head of queue-" + head);
  int size = q.size();
    System.out.println("Size of queue-" + size);
  }
}
```

Interface Queue

```
0.  Queue<Integer> q = new LinkedList<>();  // 찍을 수없음
1.  Queue<Integer> q = new LinkedList<>();  // ??
2. LinkedList<Integer> q = new LinkedList<>();
```

자바에서 제공하는 LinkedList 빵틀에서 Queue를 구체화함. 따라서 LinkedList 클래
Queue 인터페이스에 있는 모든 메소드를 사용할 수 있다.

1번 명령문은 Queue interface에 있는 메소드만 사용할 수 있다는 의미.

1. GraphMatrix.java
2. GraphList.java
3.GraphTraversal.java

Zip all your programs(학번+name.zip) and upload to smartlead.hallym.ac.kr