Aim : To implement BFS -Breadth First Search

```
import java.util.Scanner;
public class graph
{
   static final int MAX = 100;
   static int[] queue = new int[MAX];
   static int front = -1, rear = -1;
   static int[] visited = new int[MAX];

   // Enqueue Function
   static void enqueue(int vertex)
   {
      if (rear == MAX - 1)
      {
         System.out.println("Queue Overflow");
         return;
      }
      if (front == -1)
      {
         front = 0;

      }
      rear++;
      queue[rear] = vertex;
   }

   // Dequeue Function
   static int dequeue()
   {
      if (front == -1 || front > rear)
      {
         return -1;
      }
      int vertex = queue[front];
      front++;
      return vertex;
   }

   // BFS Function
   static void BFS(int[][] graph, int vertices, int start)
   {
      int i;
      // Initialize all vertices as unvisited
      for (i = 0; i < vertices; i++)
      {
         visited[i] = 0;
      }
```

```
        // Start BFS
        enqueue(start);
        visited[start] = 1;
        System.out.print("BFS Traversal: ");
        while (front != -1 && front <= rear)
        {
            int current = dequeue();
            System.out.print(current + " ");
            for (i = 0; i < vertices; i++) {
                if (graph[current][i] == 1 && visited[i] == 0)
                {
                    enqueue(i);
                    visited[i] = 1;
                }
            }
        }
        System.out.println();
        // Reset front and rear for subsequent tests if desired
        front = -1;
        rear = -1;
    }

    // Main Function
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int[][] graph = new int[MAX][MAX];
        int vertices, i, j, start;
        System.out.print("Enter number of vertices in the graph: ");
        vertices = sc.nextInt();
        System.out.println("Enter adjacency matrix:");

        for (i = 0; i < vertices; i++)
        {
            for (j = 0; j < vertices; j++)
            {
                graph[i][j] = sc.nextInt();
            }
        }

        System.out.print("Enter starting vertex for BFS (0 to " + (vertices - 1) + "): ");
        start = sc.nextInt();
        BFS(graph, vertices, start);
        sc.close();
    }
}
```

Output:

```
Enter number of vertices in the graph: 4
Enter adjacency matrix:
0 1 1 0
1 0 1 1
1 1 0 0
0 1 0 0
Enter starting vertex for BFS (0 to 3): 0
BFS Traversal: 0 1 2 3
```