In [ ]:

```java
#Bellman-Ford Algorithm (Single Source Shortest Path)
#The Bellman-Ford algorithm is used to find the shortest path from a single source vertex to all other vertices in a weighted directe
#It works even for graphs with negative weight edges (unlike Dijkstra's algorithm) and can detect negative weight cycles

import java.util.*;

class BellamanFord
{
    public static void bellamanFord(int v,List<int[]> edges){
        int[] dist=new int[v];
        Arrays.fill(dist,Integer.MAX_VALUE);
        dist[0]=0;# start source as node 0

        for(int i=0;i<v-1;i++){ # loop through v-1 times(This is where it varies from Dijkstra's)
            for(int[] e:edges){
                int src=e[0];
                int dstn=e[1];
                int cost=e[2];

                # Relaxation same as Dijkstra
                if(dist[src]!=Integer.MAX_VALUE && dist[src]+cost<dist[dstn]){
                    dist[dstn]=dist[src]+cost;
                }
            }
        }

        #Print the result, and experiment with negative weights.
        System.out.println("Shortest distances are as follows");
        for(int i=1;i<v;i++){
            System.out.println("Distance between "+0+" and "+i+" is -> "+dist[i]);
        }
    }
    public static void main (String[] args)
    {

        int nodes=4;
        int edges=5;

        List<int[]> edgeList=new ArrayList<>();


        edgeList.add(new int[]{0,1,4});
        edgeList.add(new int[]{0,2,3});
        edgeList.add(new int[]{1,2,1});
        edgeList.add(new int[]{1,3,1});
        edgeList.add(new int[]{2,3,3});

        bellamanFord(nodes,edgeList);

        #BellamanFord is same as Dijkstra's Algorithm.
        #BF is costly than Dijkstra due to its TC being O(V*E) whereas Dijkstra's has O((V+E)logV)
        #Relaxation happens for every node, and BF doesnt work with Negative edges. So
        # Floyd warshall / other algorithms can be implemented over dijk/BF for negative weights.
    }
}
```