```
import java.util.*;

public class BellmanFord {

    static class Edge {
        int source, destination, weight;

        public Edge(int source, int destination, int weight) {
            this.source = source;
            this.destination = destination;
            this.weight = weight;
        }
    }

    // Function to run the Bellman-Ford algorithm
    public static void bellmanFord(List<Edge> edges, int V, int start) {
        // Step 1: Initialize distances from start to all vertices as INFINITE
        int[] dist = new int[V];
        Arrays.fill(dist, Integer.MAX_VALUE);
        dist[start] = 0;

        // Step 2: Relax all edges V-1 times
        for (int i = 1; i < V; i++) {
            for (Edge edge : edges) {
                if (dist[edge.source] != Integer.MAX_VALUE && dist[edge.source] + edge.weight <
dist[edge.destination]) {
                    dist[edge.destination] = dist[edge.source] + edge.weight;
                }
            }
        }
```

```
        // Step 3: Check for negative-weight cycles
        for (Edge edge : edges) {
            if (dist[edge.source] != Integer.MAX_VALUE && dist[edge.source] + edge.weight <
dist[edge.destination]) {
                System.out.println("Graph contains negative weight cycle");
                return;
            }
        }

        // Print the distance array
        System.out.println("Vertex\tDistance from Source");
        for (int i = 0; i < V; i++) {
            if (dist[i] == Integer.MAX_VALUE) {
                System.out.println(i + "\t\tInfinity");
            } else {
                System.out.println(i + "\t\t" + dist[i]);
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        boolean tryAgain = true;

        // Loop to allow the user to try the algorithm multiple times
        while (tryAgain) {
            // Print decorative message with red color and big fonts using ANSI escape codes
            String decoration = "*".repeat(50); // 50 asterisks for decoration
            String introMessage = "\033[31m" + decoration + "\n" +
                                  "   BELLMAN-FORD ALGORITHM   \n" +
                                  decoration + "\033[0m"; // \033[31m sets text color to red, \033[0m resets it
```

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

// Take the number of vertices and edges as input
System.out.print("Enter the number of vertices: ");
int V = scanner.nextInt();
System.out.print("Enter the number of edges: ");
int E = scanner.nextInt();

// List to store all edges
List<Edge> edges = new ArrayList<>();

// Take edge input from the user
System.out.println("Enter the edges (source, destination, weight): ");
for (int i = 0; i < E; i++) {
    int source = scanner.nextInt();
    int destination = scanner.nextInt();
    int weight = scanner.nextInt();
    edges.add(new Edge(source, destination, weight));
}

// Take the source vertex as input
System.out.print("Enter the source vertex: ");
int start = scanner.nextInt();

// Run Bellman-Ford algorithm
bellmanFord(edges, V, start);

// Ask the user if they want to try again
System.out.print("Would you like to try the Bellman-Ford algorithm again? (yes/no): ");
String response = scanner.next();
if (response.equalsIgnoreCase("no")) {
```

```
                tryAgain = false; // Exit the loop if the user doesn't want to try again
            }
        }

        scanner.close();
    }
}
```