```java
/*******************************************************************************
 * Copyright 2015 Maximilian Stark | Dakror <mail@dakror.de>
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 ******************************************************************************/


package de.dakror.wseminar.graph.generate;

import de.dakror.wseminar.Const;
import de.dakror.wseminar.WSeminar;
import de.dakror.wseminar.graph.DefaultGraph;
import de.dakror.wseminar.graph.Edge;
import de.dakror.wseminar.graph.Graph;
import de.dakror.wseminar.graph.WeightedEdge;

/**
 * @author Dakror
 */
public class GraphGenerator<V> {
  /**
   * @param params the bundle of generation parameters
   * @return the generated graph
   */
  @SuppressWarnings("unchecked")
  public Graph<V> generateGraph(Params<String> params) {
    long seed = params.get("seed");
    WSeminar.setSeed(seed);

    Graph<V> graph = new DefaultGraph<>();

    int nodeAmount = params.orElse("nodes", Const.nodeAmount);

    int nodes = (WSeminar.r.nextInt(nodeAmount / 2) + nodeAmount / 2) * (int)
    params.get("size");

    for (int i = 0; i < nodes; i++) {
      try {
        graph.addVertex((V) (Integer) i);
      } catch (Exception e) {
        throw new IllegalStateException("Generics not matching graph type!", e);
      }
    }

    System.out.println("Added " + graph.getVertices().size() + " nodes to the graph.");

    int edgesPlaced = 0;
    int edge_type = params.get("edge_type");

    for (int i = 0; i < nodes; i++) {
      int edges = Math.max(WSeminar.r.nextInt(Math.min(graph.getVertices().size() / 2 - 1,
      params.orElse("edges", Const.edgeAmount))), 1);

      for (int j = 0; j < edges; j++) {
        int index = i;
        do {
          index = WSeminar.r.nextInt(nodes);
        } while (index == i || graph.areConnected(graph.getVertices().get(i),
        graph.getVertices().get(index)));

        Edge<V> edge = new WeightedEdge<V>(graph.getVertices().get(i),
        graph.getVertices().get(index), WSeminar.r.nextInt(Const.edgesMaxCost));

        if (edge_type == 1 || (edge_type == 2 && WSeminar.r.nextFloat() >
        WSeminar.r.nextFloat())) edge.setDirected(true);

        graph.addEdge(edge);
      }

      edgesPlaced += edges;
    }

    System.out.println("Made " + edgesPlaced + " connections.");

    return graph;
  }
}
```