**private** SightingsDAO dao;

**private** List<String> stati;

**private** Graph<String,DefaultEdge> grafo;

**public** Model() {

**this**.dao = **new** SightingsDAO();

}

**public** List<AnnoCount> getAnni(){

**return** dao.getAnni();

}

**public** **void** creaGrafo(Year anno) {

**this**.grafo = **new** SimpleDirectedGraph<String,DefaultEdge>(DefaultEdge.**class**);

**this**.stati = **this**.dao.getStati(anno);

Graphs.*addAllVertices*(**this**.grafo, **this**.stati);

//soluzione semplice--> doppio ciclo, controllo esistenza arco

**for**(String s1: **this**.grafo.vertexSet()) {

**for**(String s2: **this**.grafo.vertexSet()) {

**if**(!s1.equals(s2)) {

**if**(**this**.dao.esisteArco(s1,s2, anno)) {

**this**.grafo.addEdge(s1, s2);

}

}

}

}

System.***out***.println("Grafo creato!");

System.***out***.println("#vertici "+ **this**.grafo.vertexSet().size());

System.***out***.println("#archi "+ **this**.grafo.edgeSet().size());

}

**public** **int** getNVertici() {

**return** grafo.vertexSet().size();

}

**public** **int** getNArchi() {

**return** grafo.edgeSet().size();

}

**public** List<String> getStati() {

**return** **this**.stati;

}

**public** List<String> getSuccessori(String stato){

**return** Graphs.*successorListOf*(**this**.grafo, stato);

}

**public** List<String> getPredecessori(String stato){

**return** Graphs.*predecessorListOf*(**this**.grafo, stato);

}