ARCHITEKTUR VERTEILTER ANWENDUNGEN

ÜBUNG 1

VON YVES KIRCHER

INHALTSVERZEICHNIS

1. ALLGEMEINES

- **Programmiersprache**: JAVA
- **Entwicklungstool**: Eclipse for Java
- **Nachrichtenformat**: JSON/GSON
- **Betriebssystem**: Windows 7/8.1

- **Verwendete Libarys**:
  - Jung Libary
  - Jansi Libary
  - GSON

2. DER NETZWERK KNOTEN

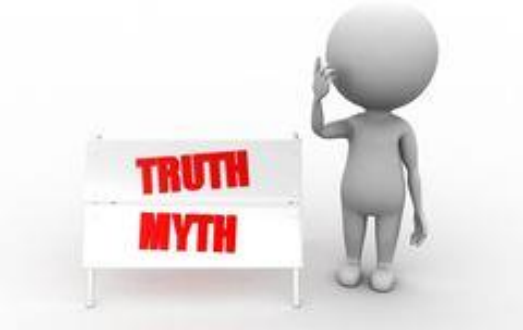
```
public class Node {
    private ArrayList<NodeSettings> friendNodeList = new ArrayList<NodeSettings>();
    private boolean                 initiator = false;
    private boolean                 NODE_ALIVE = true;
    private MythStorage             mythStorage;
    private final NodeSettings      nodeSettings;
    private Socket                  serverSocket;

    public Node(final NodeSettings nodeSettings, ArrayList<NodeSettings> friendNodeList){
        ...
        this.setListener();
    }

    private final void setListener(){
        ServerSocket server;
        try {
            server = new ServerSocket(nodeSettings.getPort());
            SimpleLogger.info("Listening on Port " + nodeSettings.getPort() + "...");
            serverSocket = null;
            while (NODE_ALIVE) {
                try {
                    serverSocket = server.accept();
                    SimpleLogger.info("Connected ... ");
                    incomingMessage();
                }
                catch (Exception e) {
                    SimpleLogger.error("Fatal error IO Exception " + e, "Node.setListener()");
                }
                catch (ClassNotFoundException e) {
                    SimpleLogger.error("Fatal error Message Class not found Exception " + e, "
                }
            }
        } catch (IOException e1) {
            SimpleLogger.warn("Port already in use.");
        }
    }
}
```

3. NACHRICHTEN

- **Enum**: MessageTypes
  (CLOSE, INITIATOR, MYTH, …)
- **NodeSettings:** des Sender-Knotens
- **Timestamp:** Zeitpunkt des Versands
- **Content Typen**:
  - Liste von NodeSettings
  - Plain Text
  - NodeSettings
- **Methoden**:
  - CompareMsg(Message msg)
  - toJson()

3. MYTHEN

- **MythStorage Klasse**
  - Verwaltet eine Liste von Mythen
  - Testet ob ein Mythos bereits bekannt ist
  - Fügt Mythen zur Liste hinzu
  - Auslesen eines Mythos durch das erhaltene Message Objekt (bzgl. Myth Status)



3. MYTHEN

```
public class Myth {
    private ArrayList<NodeSettings> mythReceivedByFriendsList;
    private Message mythContent;
    private boolean trusted = false;
    private final int TRUSTED_MYTH_NUMBER;

    public Myth ( Message msg, int tmn ) {
        this.mythContent = msg;
        this.TRUSTED_MYTH_NUMBER = tmn;
        this.mythReceivedByFriendsList = new ArrayList<NodeSettings>();
    }

    public void tryAddFriendToReceivedByList( NodeSettings ns ){
        if ( !checkIfFriendAlreadyInList(ns) ){
            addToReceivedByList(ns);
        }
    }

    private void checkIfMythTrusted(){
        if ( mythReceivedByFriendsList.size() >=  TRUSTED_MYTH_NUMBER){
            this.trusted = true;
            SimpleLogger.trusted(this);
        }else{
            SimpleLogger.mythStatus(mythReceivedByFriendsList.size(), TRUSTED_MYTH_NUMBER);
        }
    }
}
```

3. NACHRICHTENFORMAT

- **JSON mit GSON Libary**
- **Serialization:**

```
Gson gson = new Gson();
NodeSettings node = new NodeSettings(0, "127.0.0.1", 5000)
gson.toJson( node );
==> prints {"id":0,"address":"127.0.0.1", "port":5000}
```

- **Deserialisation:**

```
Gson gson = new Gson();
nodeJson = gson.toJson(node);
NodeSettings nodeFromJson = gson.fromJson(nodeJson, NodeSettings.class);
==> Objekt node entspricht einem äquivalent zu dem Objekt nodeFromJson
```

4. LOGGING

- **SimpleLogger Klasse**
  - Info()
  - Warn()
  - Debug()
  - Error()
  - Trusted() => Ausgabe wenn Mythos geglaubt wird
  - MythStatus => Ausgabe des aktuellen Status des Mythos
  - LogSendingMessage() => Zu sendende Nachricht
  - LogReceivedMessage() => Empfangene Nachricht

4. LOGGING UND JANSI

- **JANSI Libary**
  - Schmal gehaltene Java Libary
  - Erlaubt ANSI escape code zum formatieren von Konsolenausgaben
  - Funktioniert in Windows Shell
  - **Farben** der Schrift oder des Hintergrundes
  - Ermöglicht leeren des Screens

```
WhiteOnBlue : Hello world

Bold : Press return...Normal text and bold text.
Normal yellow text and bold text.
Normal text and bold text.
Normal yellow text and bold text.
and uses its Internet connection to post them on the web. From there, it's
this concludes the Jansi demo
[oracle@soabpm-vm 510.Jansi.fusesource]$
        10,10 reverse : Hello world
```

4. LOGGING UND JANSI

```java
// Import Jansi
import org.fusesource.jansi.AnsiConsole;

// Escape String to Reset ANSI Settings
private static final String ANSI_RESET = "\u001B[0m";
// Escape String for Yellow Color
private static final String ANSI_YELLOW = "\u001B[33m";

// Enabling jansi ANSI support
AnsiConsole.systemInstall();

// Example: Warning from SimpleLogger
public static final void warn(String log){
    System.out.println(ANSI_YELLOW + "WARNING:\t" + log + ANSI_RESET);
}

// Disable jansi ANSI support
AnsiConsole.systemUninstall();
```

5. GRAPHENGENERIERUNG

- **GraphGeneration Klasse:**
  - **generateGraph()**:
    Lädt Graph aus *node* und *graph* Dateien.
  - **newRandomGraph(int nodesCount, int EdgesCount)**:
    Erzeugt einen neuen *zufälligen Graphen* mit der geforderten Anzahl an Knoten und Kanten (wenn eingabe Möglich) und *schreibt generierten Graph in die Dateien.*
  - **drawGraph**():
    Zeichnet den entstandenden Graphen mit Hilfe der *JUNG Libary.*

6. JUNG LIBARY

- **Java Universal Network/Graph Libary**
- **Aktuelle Version**: JUNG 2.0.1 24.01.2010
  (Google Trends : Suchvolumen zu gering .. Laut Seite: 272 Downloads/Woche)
- **Open Source (BSD license)**
- Entwickelt von 3 Studenten
- **Def**: Stellt eine erweiterbare Sprache für die Modellierung, Analyse und Visualisierung von Daten (Graphen oder Netzwerke).
- **Verwendung**:
  Social Networks, Web Graph Analysis, Graph Theory
- **Mehr Infos**: http://jung.sourceforge.net/

6. JUNG LIBARY

- Unterstützt **gerichtete-**, **ungerichtete** und Hyper-**Graphen** (auch Graphen mit parallelen Kanten).
- Knoten und Kanten können **beschriftet** und graphisch **formatiert** werden.
- **Zusätzliche Features:**
  - Algorithmen für Graphentheorie, Data-Mining und zur Analyse von Sozialen Netzwerken.
  - Zufallsgraphenerzeugung
  - Graphen Berechnungen: z.B. der Entfernung zweier Knoten.
  - Graphen Optimierung
  - … vieles mehr das ich noch nie gehört habe ☺

6. JUNG LIBARY

- **Einfacher ungerichteter Graph:**

```
// Graph<V, E> where V is the type of the vertices
// and E is the type of the edges
Graph<Integer, String> g = new SparseMultigraph<Integer, String>();
// Add some vertices. From above we defined these to be type Integer.
g.addVertex((Integer)1);
g.addVertex((Integer)2);
g.addVertex((Integer)3);
// Add some edges. From above we defined these to be of type String
// Note that the default is for undirected edges.
g.addEdge("Edge-A", 1, 2, EdgeType.UNDIRECTED);
g.addEdge("Edge-B", 2, 3, EdgeType.UNDIRECTED);
// Let's see what we have. Note the nice output from the
// SparseMultigraph<V,E> toString() method
System.out.println("The graph g = " + g.toString());
```

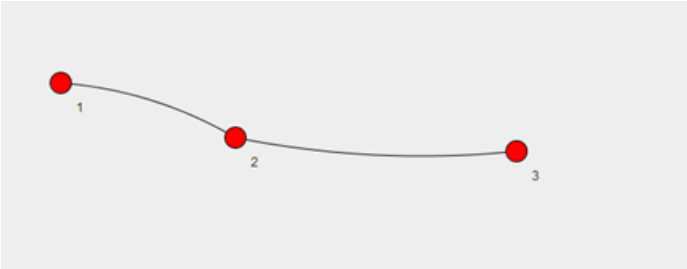
6. JUNG LIBARY

- **Shortest Path (ungerichtet):**

```
DijkstraShortestPath<Node,Edge> alg = new DijkstraShortestPath(graph);
List<Edge> l = alg.getPath(n1, n4);
System.out.println("The shortest unweighted path from" + n1 +
" to " + n4 + " is:");
System.out.println(l.toString());
```

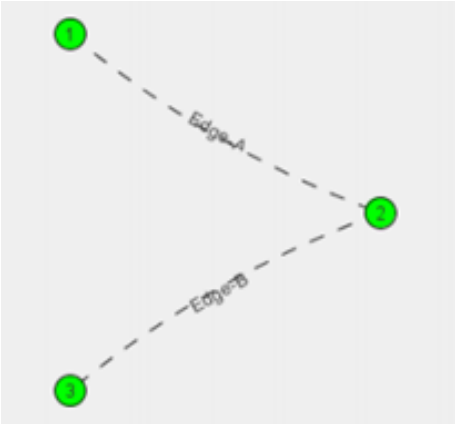
- **Visualisierung:**

```
Layout<Integer, String> layout = new CircleLayout(graph);
layout.setSize(new Dimension(300,300)); // sets the initial size of the space
BasicVisualizationServer<Integer,String> vv = new BasicVisualizationServer<Integer,String>(layout);
vv.setPreferredSize(new Dimension(350,350)); //Sets the viewing area size
JFrame frame = new JFrame("Simple Graph View");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.getContentPane().add(vv);
frame.pack();
frame.setVisible(true);
```

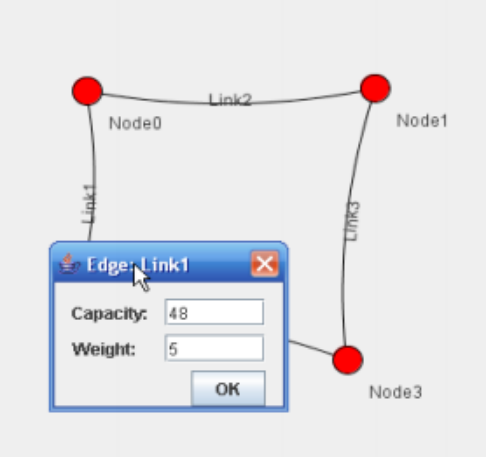


6. JUNG LIBARY

- **Custom Format:**



- **Zooming, Rotation,**
  Mouse Interaction, Key Listener, …



7. VORFÜHRUNG

Vielen Dank für Ihre Aufmerksamkeit !

Noch Fragen ?