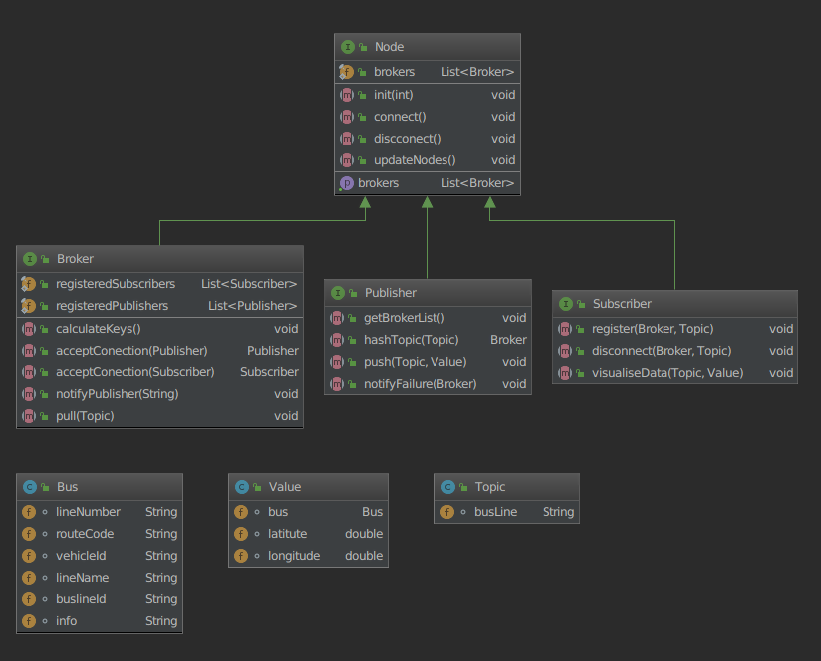
**PROJECT ON FOR A DISTRIBUTED SYSTEMS EXPERT**

****

**It has to change to publisher node?**

For the publish/subscribe model we need to build three components, the publisher node, the broker nodes and the consumer node explained at the end

**Implementation of a publish/subscribe system on Java 8 [3]**. The publish/subscribe mode will be as follows:

1. At each broker node there will be an instance of the system that will properly forward the data.

2. Each instance (consumer node, broker nodes, consumer nodes) will listen to a predefined port for connections.

3. When a broker node receives a query from the user, initially it examins whether it is already connected with it (if the consumer node has been registered with that broker). If a connection already exists, it simply makes sure to wait properly until there are available data that it can send. If not, then a new connection is created and at the same time the list of other brokers and for which keys are responsible, is returned to the consumer. Once the data is available, the broker sends them appropriately to the consumer. If it does not, it returns an appropriate message informing the user.

4. Each broker will forward the data for the specific data range for which it is responsible.

5. When the consumer node receives the information, it visualizes it appropriately.

**Android Application**

**You will deploy an application that will run on Android operating system devices and display the bus locations that the user requests in real time on the map. The application will be implemented on the Android platform and will benefit from the publish/subscribe framework that will run independently.**

Android app will run as follows:

1. This app's basic screen will include a Google Maps screen. On this screen the user will be able to select the bus lines for which they want information.

2. By doing so, the application will be addressed to the appropriate broker node asking for the necessary information if it exists, otherwise it will wait for an appropriate time until it is available.

3. Follow the procedure outlined in the part of implementation of the transmission and management of publisher data by brokers. If there is no information available at that time and a reasonable time (seconds) has passed, then an appropriate message ("not found") is returned about the line's information to the mobile while sending information about the other bus lines.

4. When it receives the information from the broker, it must properly "paint" the bus positions on the map.

**Deliverables:**

The project will be delivered in two phases:

**Deliverable A: (Date of delivery: 18/4/2019)**

In this delivery, you must have completely completed the publish/subscribe system just as you have been requested so that it can be used in the next phase of the project.

**Deliverable B: (Date of Delivery: 29/4/2019)**

This is the android application described earlier. In this phase the system should be fully functional, with all its components functioning properly.

**References - Useful links:**

[1] <https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html>

[2] Android. URL : <http://code.google.com/android/http://code.google.com/android/>

[3] Android SDK: <http://developer.android.com/sdk/index.html>

[4] Android Studio <http://developer.android.com/sdk/index.html>

1. **Publisher node:** It is responsible for taking the data from the appropriate source. This node sends the data to the brokers in an appropriate manner and at an appropriate time. Essentially what it does is push the data for the keys for which it is responsible, appropriate to the brokers who are responsible for the particular keys. The publisher node at startup should know for which keys it is responsible, as well as all the necessary information for the brokers. It needs to know about each available broker for which keys it is responsible for. This will be accomplished at the time each broker start to operate. For this reason, **some form of synchronization should be made** when each broker informs the consumer of which keys are responsible. So when the data is transmitted, then those will be transferred to the respective brokers and not to all. If for any reason the publisher node does not send results (for example, because the sensor has failed and does not send data), then the publisher node returns an appropriate message that should be transferred to the broker and after be handled appropriately then be promoted to the consumer nodes.

**2. Broker Nodes:** These nodes are responsible for a range of topics (in our case: bus lines). In order to make an equal distribution of how many topics each broker handles, some kind of hashing is needed. Then using a function hash (like SHA1 or MD5) we get the Hash of String BusLineID and its Hash of IP + Port. So every broker will be responsible for the records hashes which are smaller than the hash of his IP + Port. (Attention to which hashes correspond to the broker with the smallest hash, you'll need to use mod). These nodes properly inform publisher nodes about which keys they are responsible for and then open connection to them (with publisher nodes) so they can get the information when it becomes available and promote it to the appropriate consumers who have been registered on them. Also, in every new connection of a new consumer they inform him properly about who the other brokers are and for which topics they are responsible for. A very important element of brokers is that the information they send to the consumer nodes they should send it to all at the same time. For example, if we have two consumer nodes that are interested in the same topic (the same bus line) then the data should be sent simultaneously to both using the multi-threaded programming principles.

**3. Consumer Node:** The Consumer Node is essentially the mobile that receives the information about the bus line and its corresponding location. Accepts from the broker nodes tuples of the form:

<busLine, (Bus LatLng)>. The consumer node is permanently connected to the broker responsible for the bus line of interest at each time. Because, however, this is not always known, the consumer promptly asks all available brokers about who is responsible and for which keys. Therefore, during the first contact of a consumer node with one of the broker, it receives from him the information about who the other brokers are and for which keys. An object of the form Info {ListOfBrokers {IP, Port}, <BrokerId, ResponsibilityLine>} is returned. Based on this object, therefore, the consumer then is easy to choose the appropriate broker who will return to him the corresponding information for a given bus line.

1. Publisher

자료를 수집하는 역할을 담당합니다. 이 노드는 적절한 방식으로 적절하게 브로커에 데이터를 보냅니다. 한마디로 어떤 키에 해당한 자료를 그 키를 관리하는 Broker에게 전달하는것이다.

처음에 Broker가 어떤 키들을 관리하는가를 알고 있어야 한다. 이것은 모든 Broker가 봉사를 시작할때 실행된다.

이러한 리유로 모든 Broker가 Consumer에게 어떤 키를 담당하고 있는가를 알릴 때 어떤 형태의 동기화가 이루어져야합니다.

따라서 데이터가 전송되면 모든 데이터가 전송되는 것이 아니라 해당 브로커로 전송됩니다.

어떤 원인으로 Publisher가 결과를 보내지 않으면 (예 : 센서가 실패하고 데이터를 보내지 않음) Publisher는 Broker로 전송되어야하는 적절한 메시지를 반환하고 적절히 처리 한 후에 consumer에게 전달한다.

1. Broker

이 노드는 다양한 주제를 다루고 있습니다 (우리의 경우 : 버스 라인).

모든 Broker에게 부하를 균일하게 주기 위해 해싱이 필요합니다. 함수 해시 (예 : SHA1 또는 MD5)를 사용하여 BusLineID 와 IP + Port 의 Hash코드를 얻는다. 그러므로 모든 Broker는 자기 IP + Port의 해시보다 작은 레코드 해시를 담당하게됩니다. (해시가 가장 작은 브로커에 해당하는 경우 mod를 사용해야합니다. )

이러한 노드는 Publisher에게 관리하는 키를 알리고 Publisher에 접속하고 정보를 사용할 수있게되면 정보를 가져와 등록 된 해당 Consumer에게 알립니다. 또한 새로운 Consumer가 접속되면 다른 Borker와 그것들이 관리하는 키를 알려준다.

Broker에서 가장 중요한것은 Consumer에 보내는 정보를 동시에 모든 요청된 Consumer에게 보내야한다는 것입니다. 예를 들어 동일한 주제 (동일한 버스 라인)에 관심이있는 두 개의 소비자 노드가있는 경우 다중스레드를 리용하여 자료를 동시에 전송해야합니다.

1. Consumer

소비자 노드는 본질적으로 버스 라인 및 해당 위치에 대한 정보를 수신하는 모바일입니다. 브로커 노드에서 다음 형식의 튜플을 수락합니다.

<busLine, (버스 LatLng)>. 소비자 노드는 매번 관심있는 버스 라인을 담당하는 브로커에 영구적으로 연결됩니다. 그러나 이것이 항상 알려지지는 않았기 때문에 소비자는 모든 브로커에게 책임자와 키에 대해 신속하게 요청합니다. 따라서 소비자 노드가 브로커 중 하나와 처음 접촉하는 동안 다른 브로커가 누구이며 어떤 키가 있는지에 대한 정보를 제공받습니다. Info {ListOfBrokers {IP, Port}, <BrokerId, ResponsibilityLine>} 형식의 개체가 반환됩니다. 따라서 이 객체를 기반으로 한 소비자는 특정 버스 라인에 대한 해당 정보를 반환 할 적절한 중개인을 쉽게 선택할 수 있습니다.

Broker정보를 txt화일로 가지고 있다가 Publisher 와 Consumer가 기동하면 리용한다.

* Publisher

처음으로 기동할 때 Broker정보들을 모두 읽어야 한다.

자료를 읽어 해당한 Broker에게 통지해 준다.

BusLine에 대한 SHA1코드를 얻고 이것을 IP\_PORT Hash코드와 비교하여 Broker를 선택한다.

* Broker

기동하면 Publisher와 Consumer에서 들어오는 요청을 기다린다.

Publisher가 접속되면 Socket를 창조하고 해당한 자료를 읽어 들인다.

Consumer가 접속되면 등록해놓고 해당한 정보(뻐스상태, 없으면 잠시 기다리다가 없다는 통지를 날린다.) 를 제공한다. 또한 다른 Broker가 관리하는 Line정보들도 같이 보낸다.

* Consumer

기동하면 로그인하여 Broker에 접속한다. 어떤 BusLine정보를 얻기 위하여 값을 입력하면 그 정보를 관리하는 Broker를 찾아 값을 반환받아 현시한다.