Joey Tatú

15015556 – joey.tatu@student.ncirl.ie

Distributed Systems

TBAA - 2021

Contents

[Question 1 3](#_Toc70783730)

[Part A 3](#_Toc70783731)

[Subscriber.java 3](#_Toc70783732)

[MqttSubscriber.java 3](#_Toc70783733)

[MqttSportsPublish.java 4](#_Toc70783734)

[MqttPoliticsPublish.java 5](#_Toc70783735)

[Output from MqttSubscriber.java (before Publishers are run) 6](#_Toc70783736)

[Output from MqttSportsPublish.java 7](#_Toc70783737)

[Output from MqttSubscriber.java (after MqttSportsPublish.java is run) 8](#_Toc70783738)

[Output from MqttPoliticsPublish.java 8](#_Toc70783739)

[Output from MqttSubscriber.java (after MqttPoliticsPublish.java is run) 9](#_Toc70783740)

[Parts B, C & D 10](#_Toc70783741)

[MySubscriberExtra.java 10](#_Toc70783742)

[Output from MySubscriberExtra.java 10](#_Toc70783743)

[Part E 11](#_Toc70783744)

[Example of durable subscription used in MqttSubscriberExtra.java 11](#_Toc70783745)

[Question 2 12](#_Toc70783746)

[Amazon Web Services 12](#_Toc70783747)

[Question 3 13](#_Toc70783748)

[Part A: The network is reliable 13](#_Toc70783749)

[Part B: The network is homogenous 13](#_Toc70783750)

[Question 4 14](#_Toc70783751)

[Part B 14](#_Toc70783752)

[RMI Overview 14](#_Toc70783753)

[How RMI Works 15](#_Toc70783754)

[Marshalling 15](#_Toc70783755)

[RMI Registry 15](#_Toc70783756)

[Use of middleware and registry 16](#_Toc70783757)

[Part C 16](#_Toc70783758)

[How do components communicate in a distributed system? 16](#_Toc70783759)

[Identify and compare … 17](#_Toc70783760)

[Comparison of RPC and Publish/Subscribe 18](#_Toc70783761)

[References 19](#_Toc70783762)

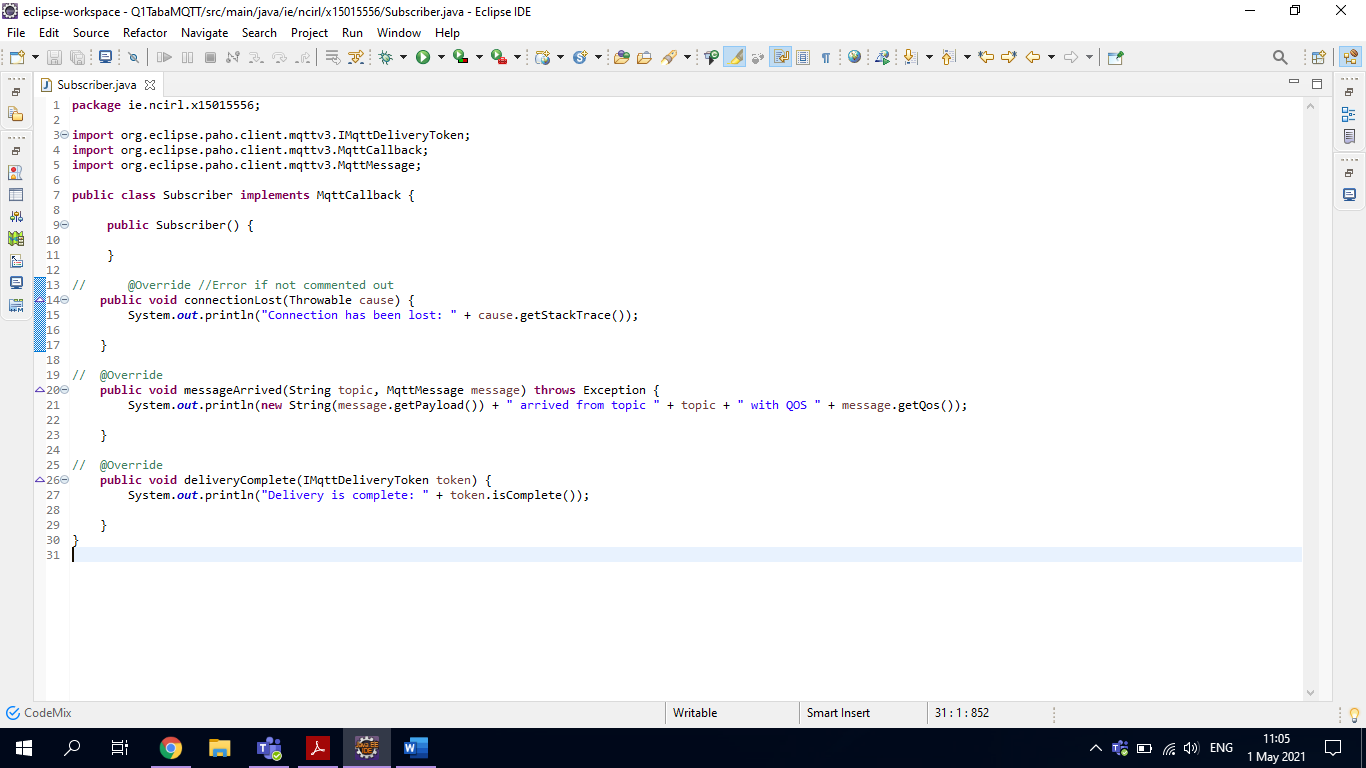
**Note: All code is uploaded to GitHub repo after submission deadline just in case there’s any issues with screenshots/images.**

**GitHub repo:** [**https://github.com/JoeyTatu/DS-TABA2021**](https://github.com/JoeyTatu/DS-TABA2021)**.**

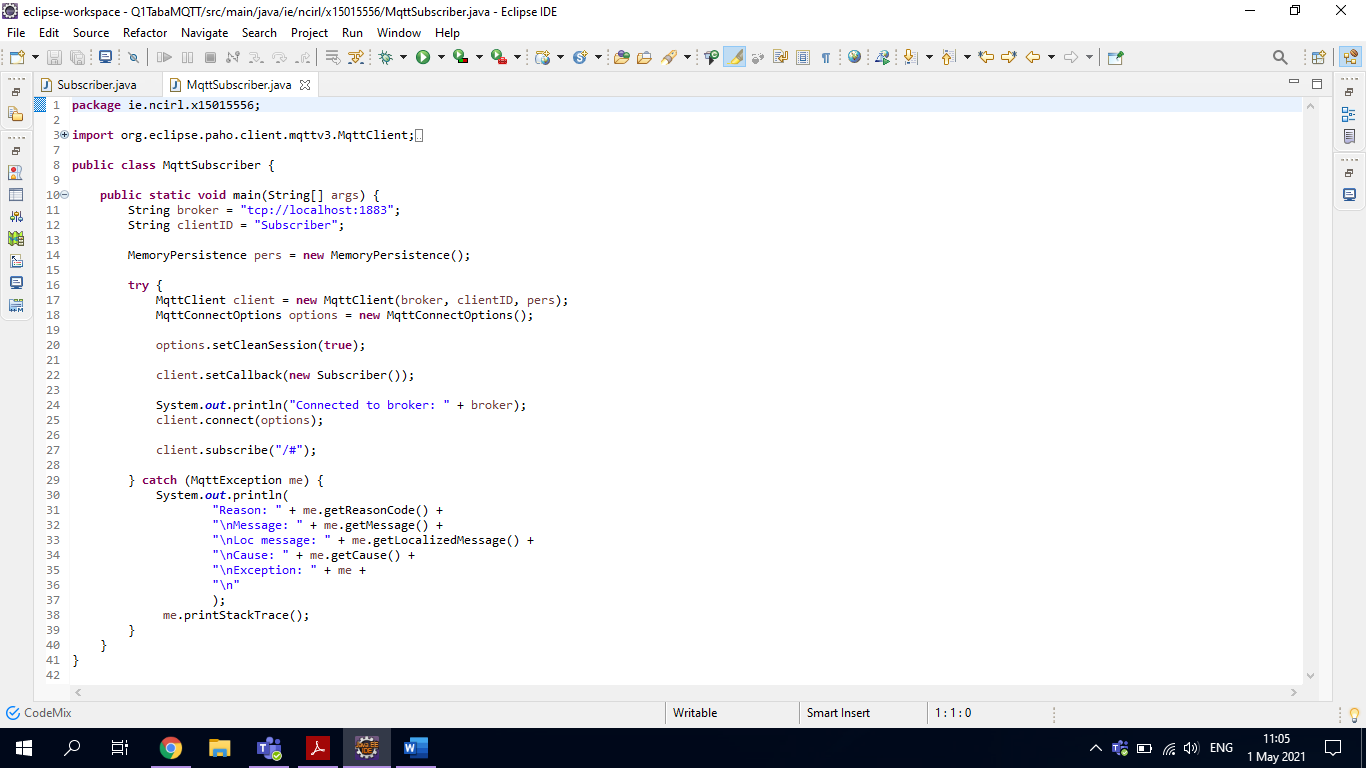
# Question 1

## Part A

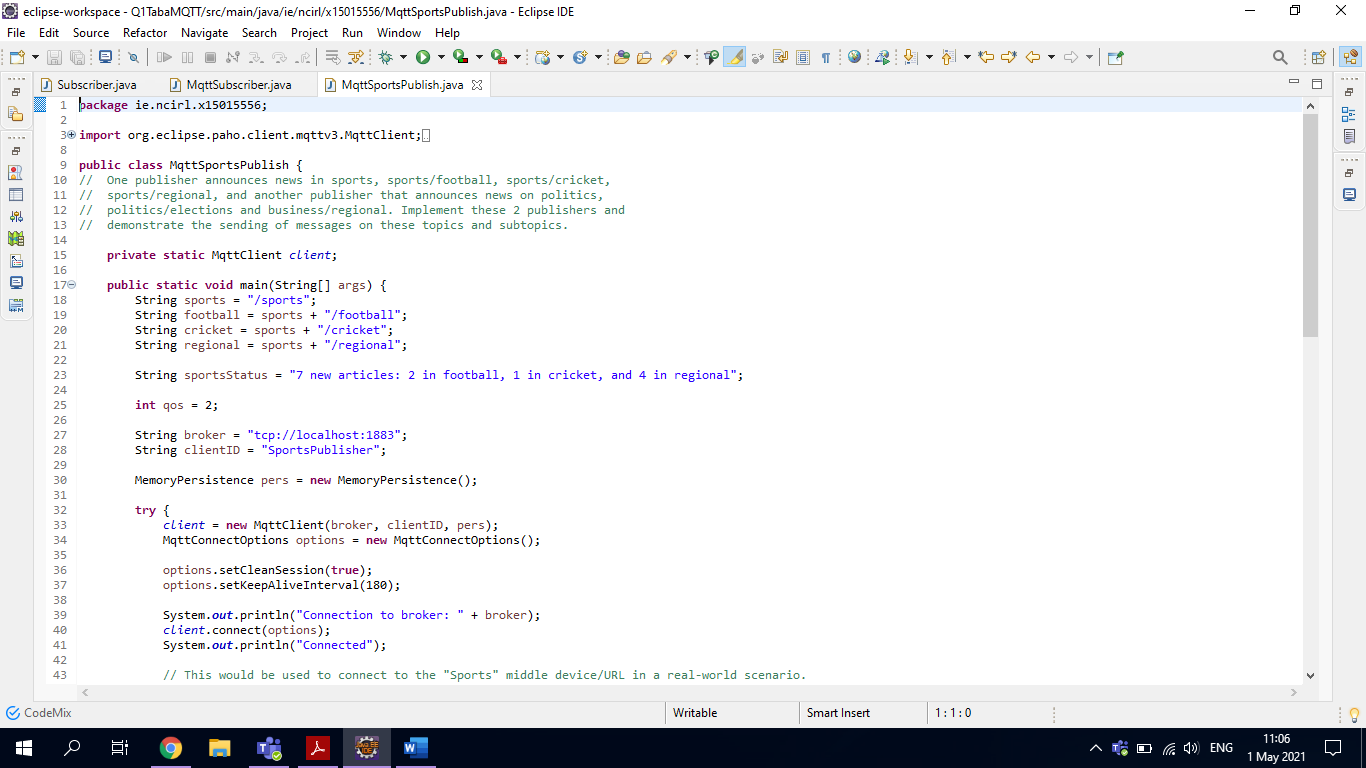
### Subscriber.java

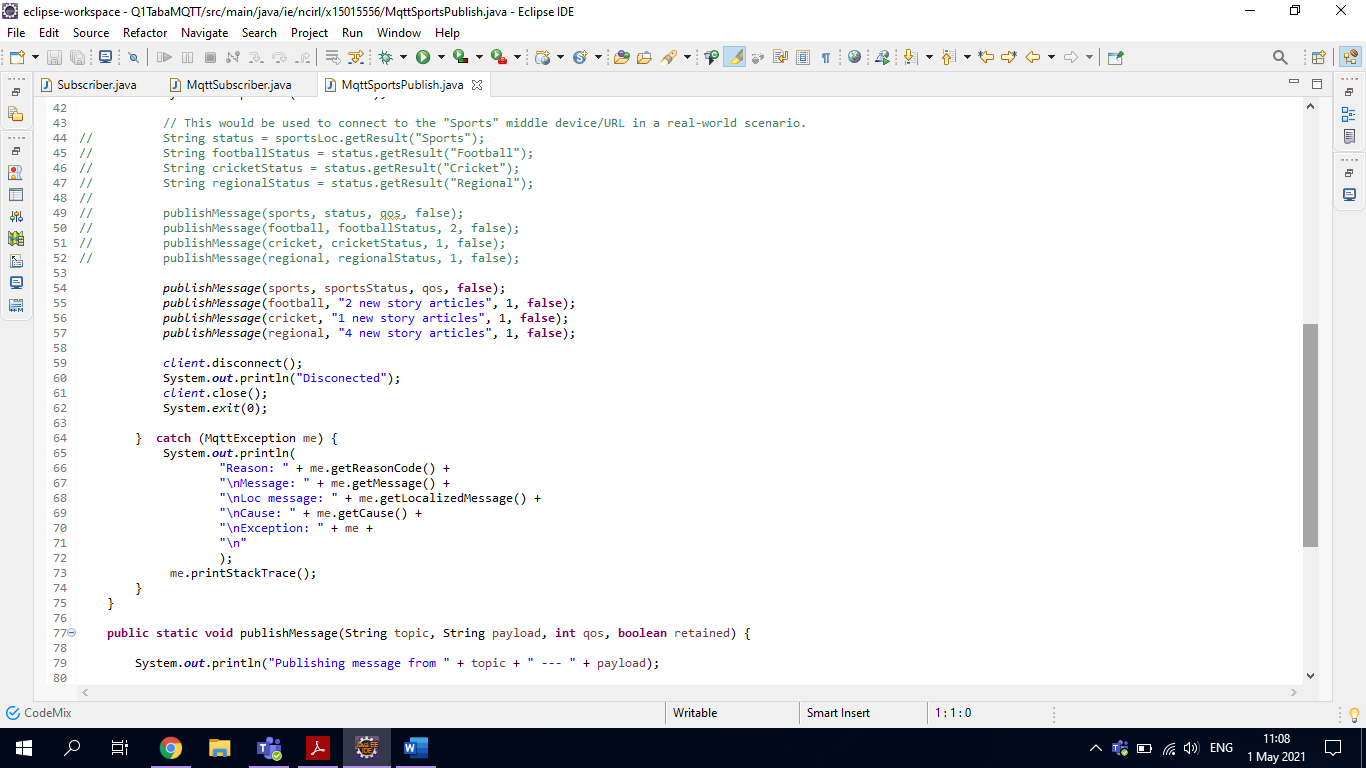


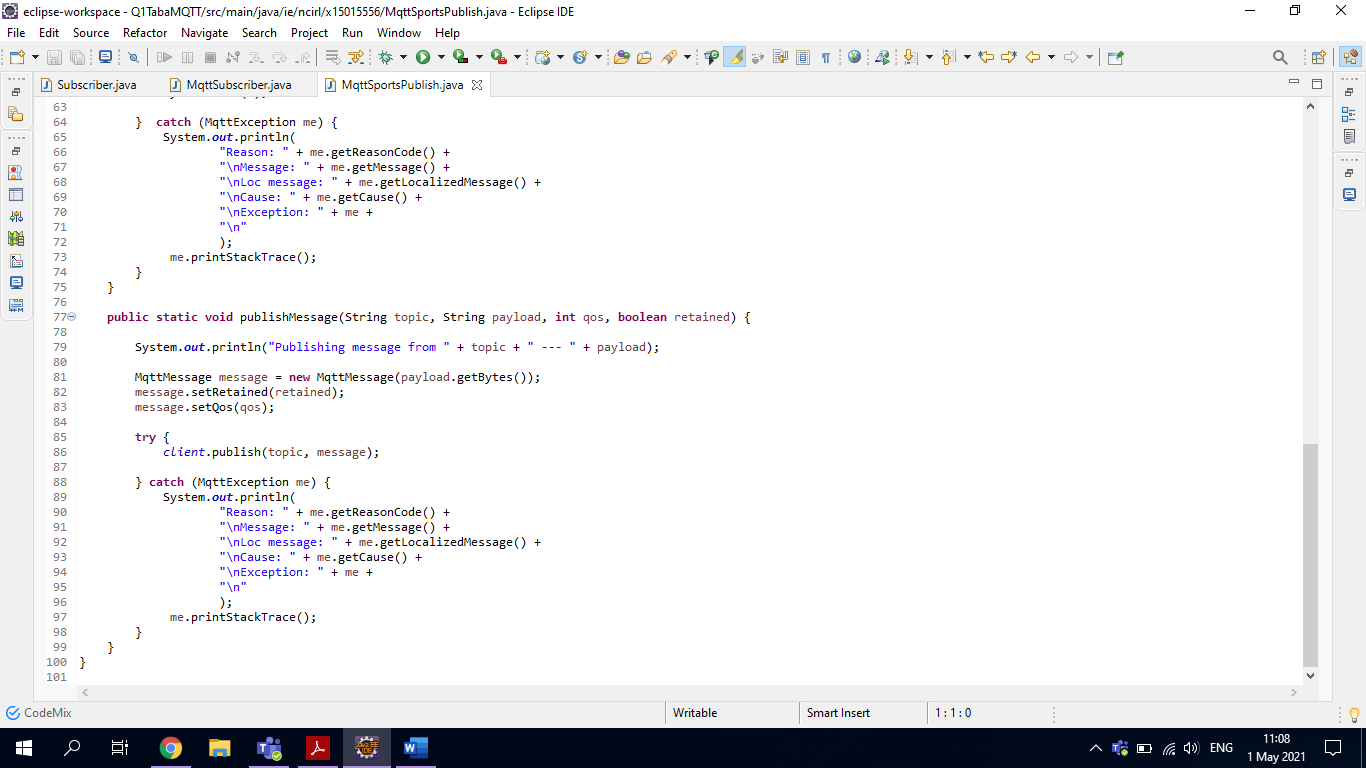
### MqttSubscriber.java



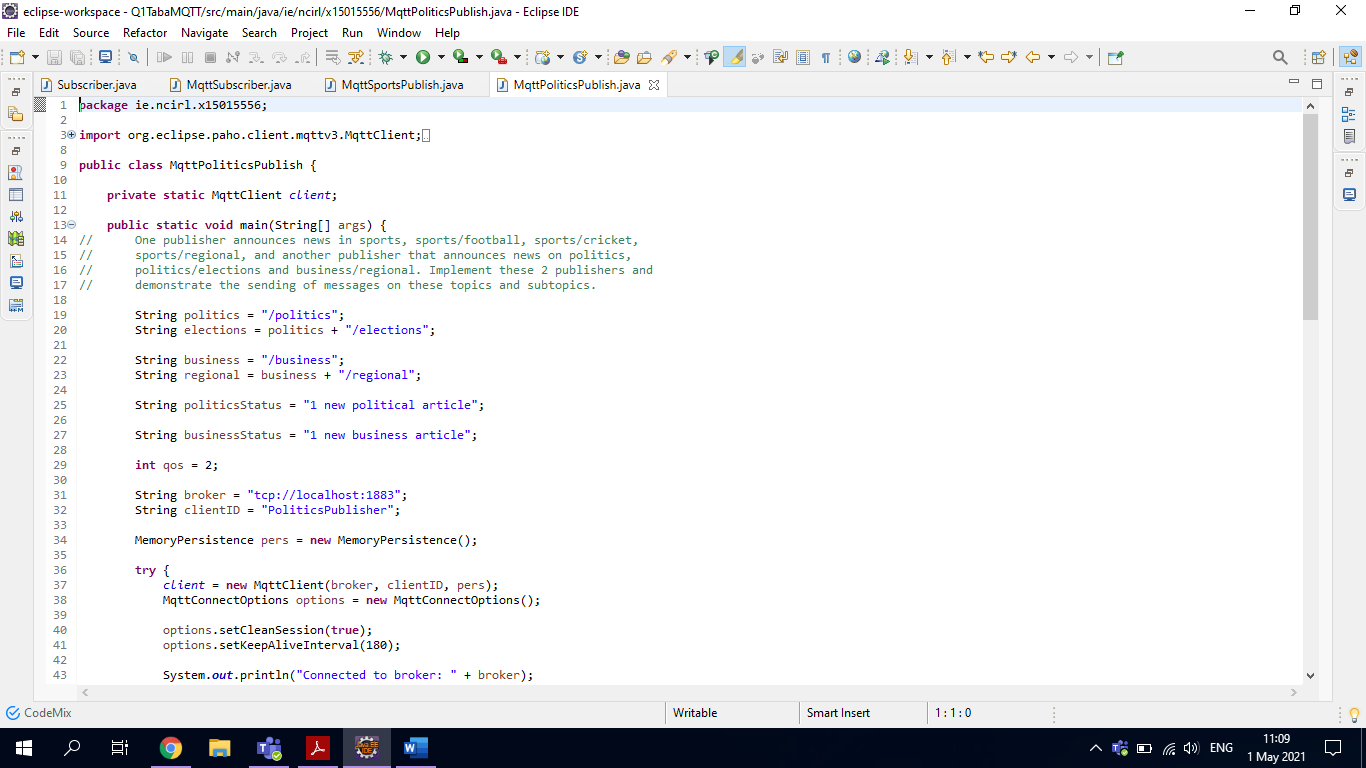
### MqttSportsPublish.java

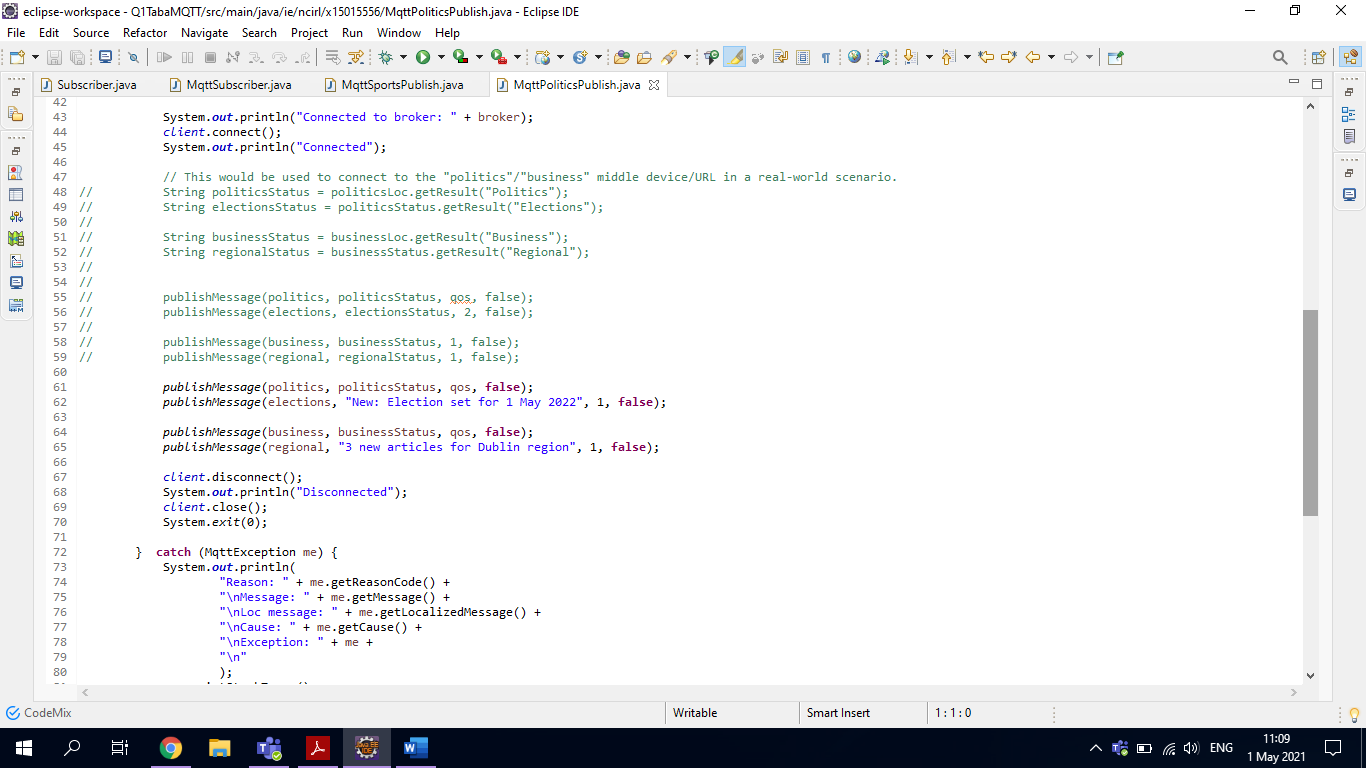


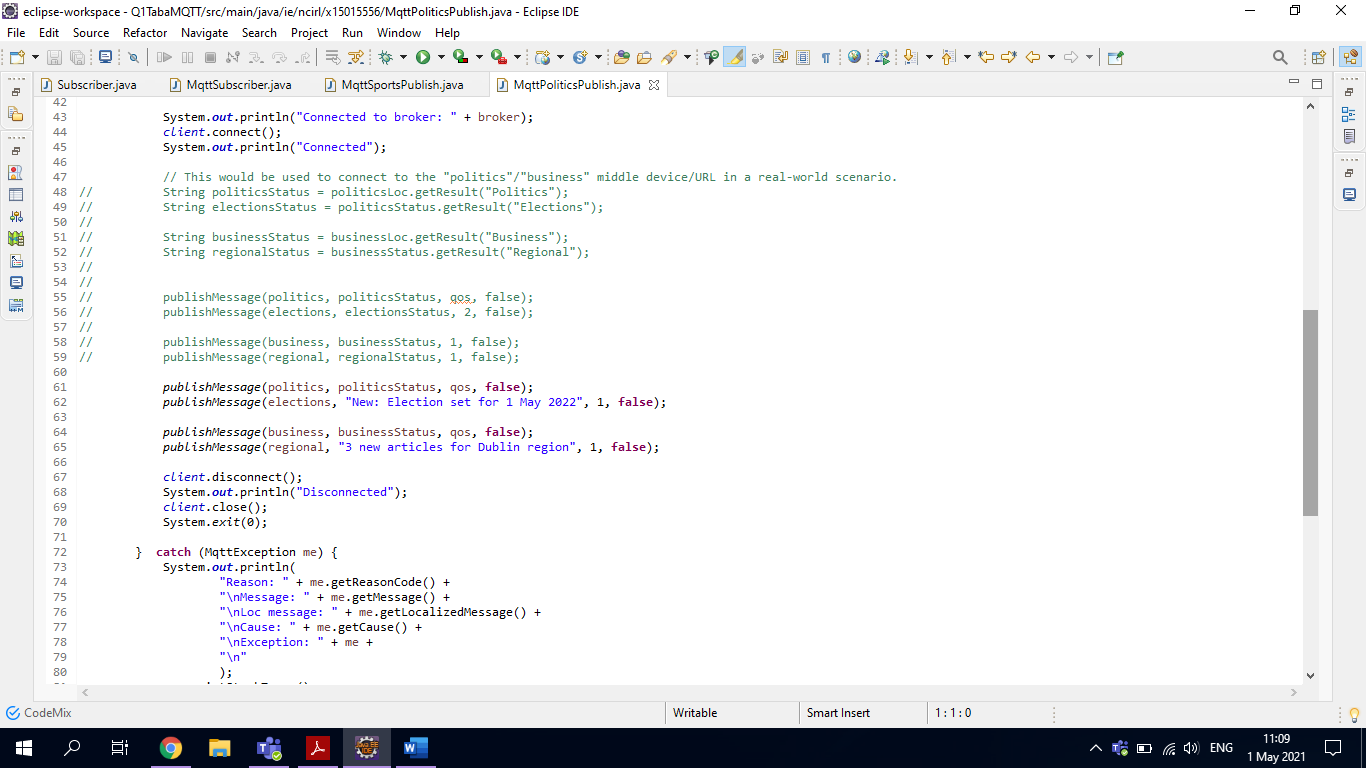




### MqttPoliticsPublish.java

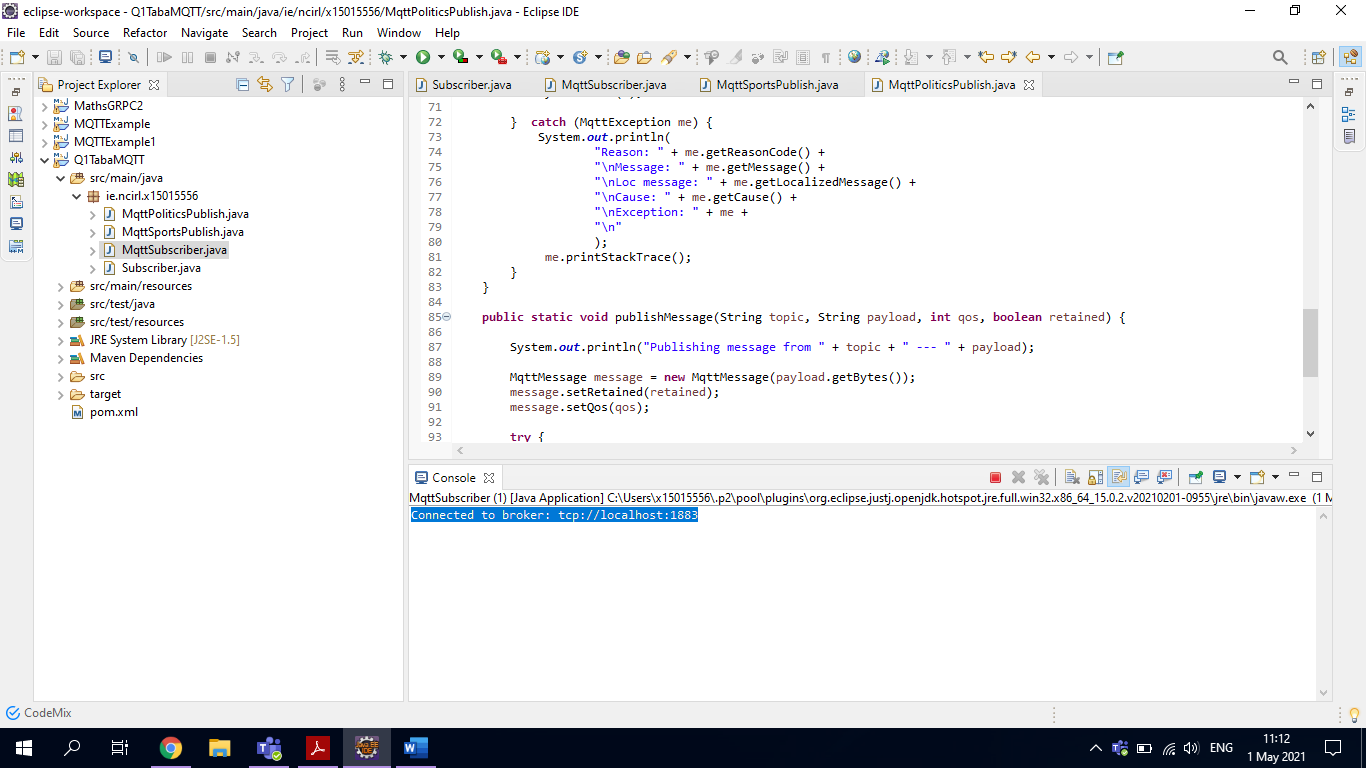






### Output from MqttSubscriber.java (before Publishers are run)

Connected to broker: tcp://localhost:1883



### Output from MqttSportsPublish.java

Connection to broker: tcp://localhost:1883

Connected

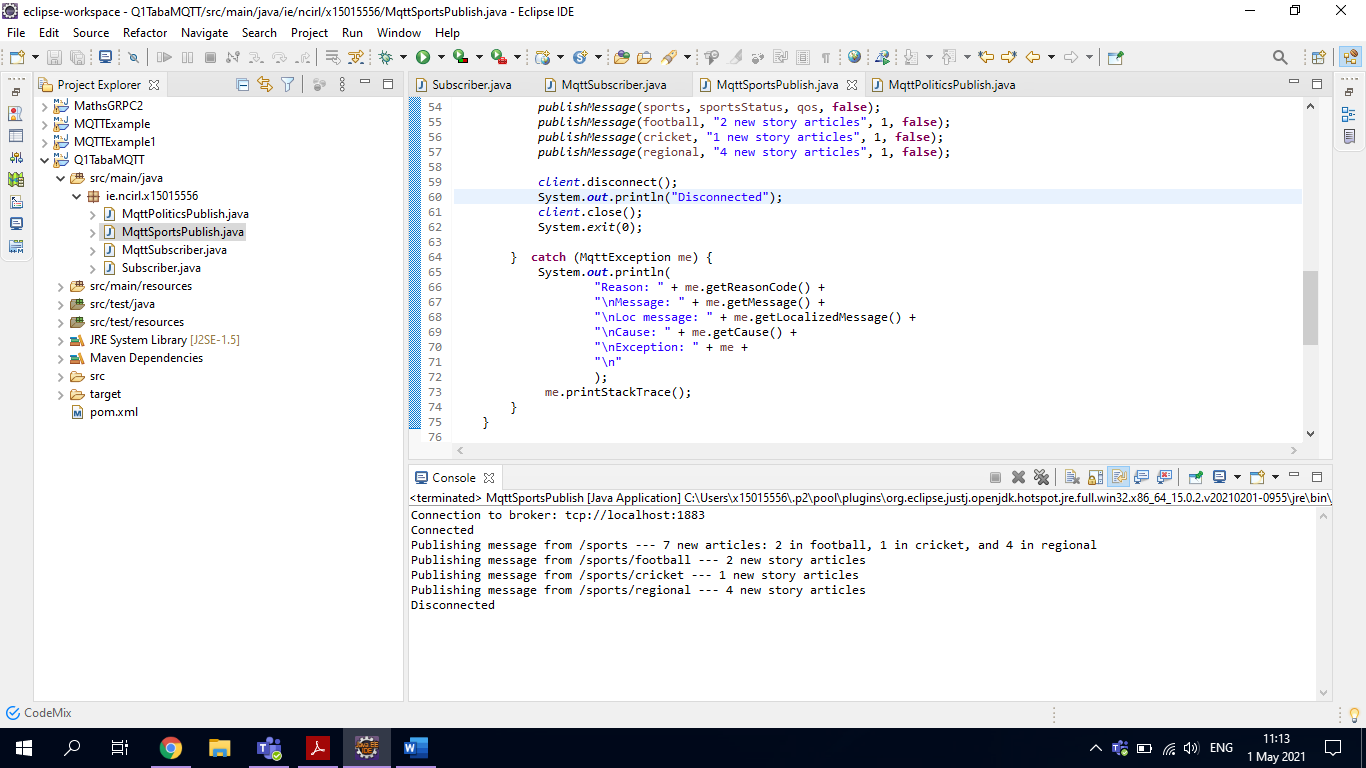
Publishing message from /sports --- 7 new articles: 2 in football, 1 in cricket, and 4 in regional

Publishing message from /sports/football --- 2 new story articles

Publishing message from /sports/cricket --- 1 new story articles

Publishing message from /sports/regional --- 4 new story articles

Disconnected



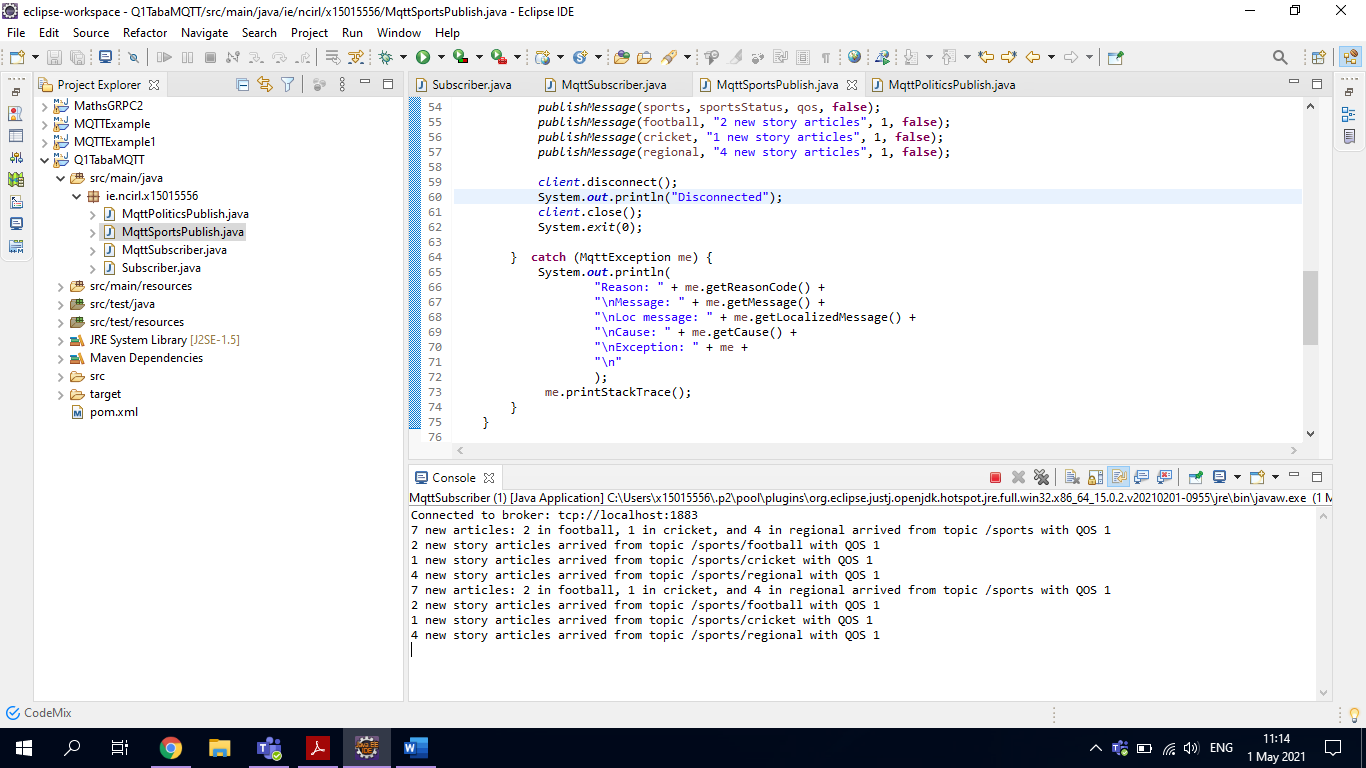
### Output from MqttSubscriber.java (after MqttSportsPublish.java is run)

7 new articles: 2 in football, 1 in cricket, and 4 in regional arrived from topic /sports with QOS 1

2 new story articles arrived from topic /sports/football with QOS 1

1 new story articles arrived from topic /sports/cricket with QOS 1

4 new story articles arrived from topic /sports/regional with QOS 1



### Output from MqttPoliticsPublish.java

Connected to broker: tcp://localhost:1883

Connected

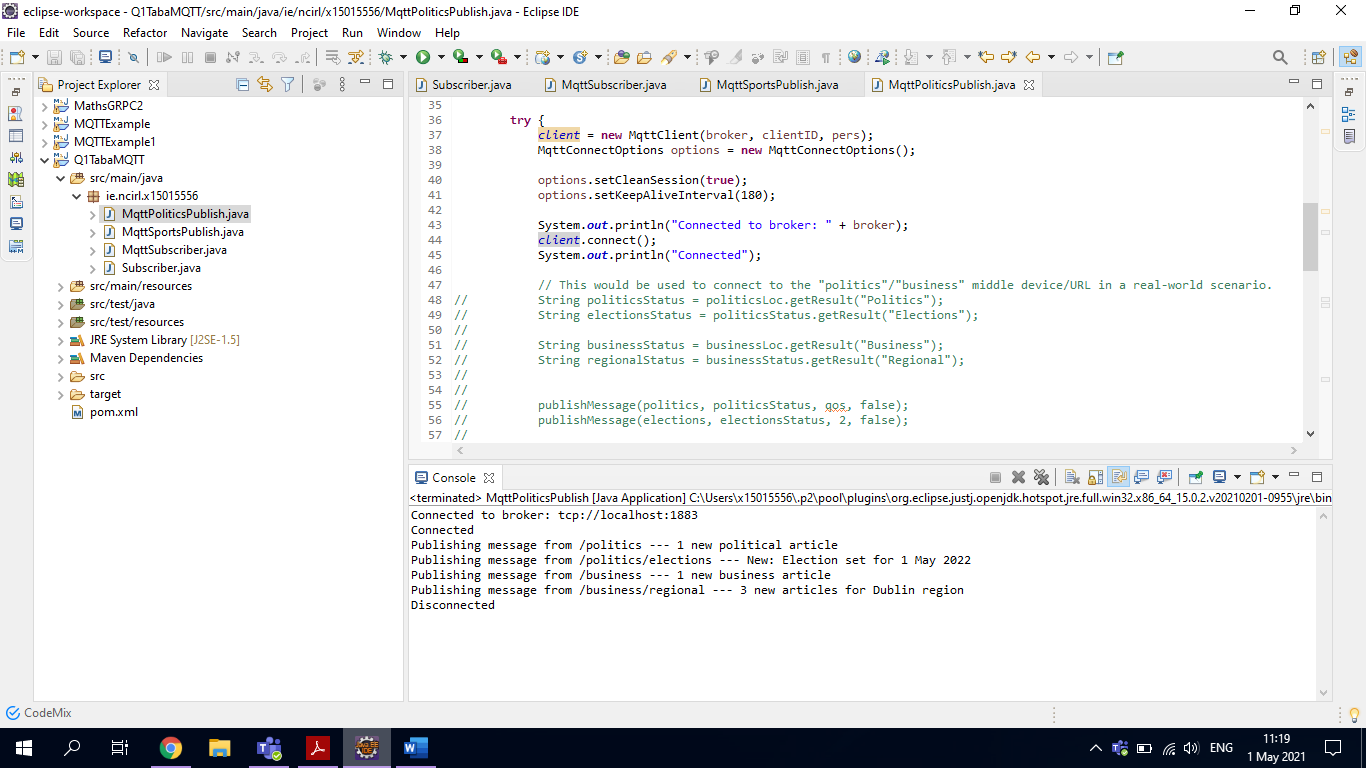
Publishing message from /politics --- 1 new political article

Publishing message from /politics/elections --- New: Election set for 1 May 2022

Publishing message from /business --- 1 new business article

Publishing message from /business/regional --- 3 new articles for Dublin region

Disconnected



### Output from MqttSubscriber.java (after MqttPoliticsPublish.java is run)

7 new articles: 2 in football, 1 in cricket, and 4 in regional arrived from topic /sports with QOS 1

2 new story articles arrived from topic /sports/football with QOS 1

1 new story articles arrived from topic /sports/cricket with QOS 1

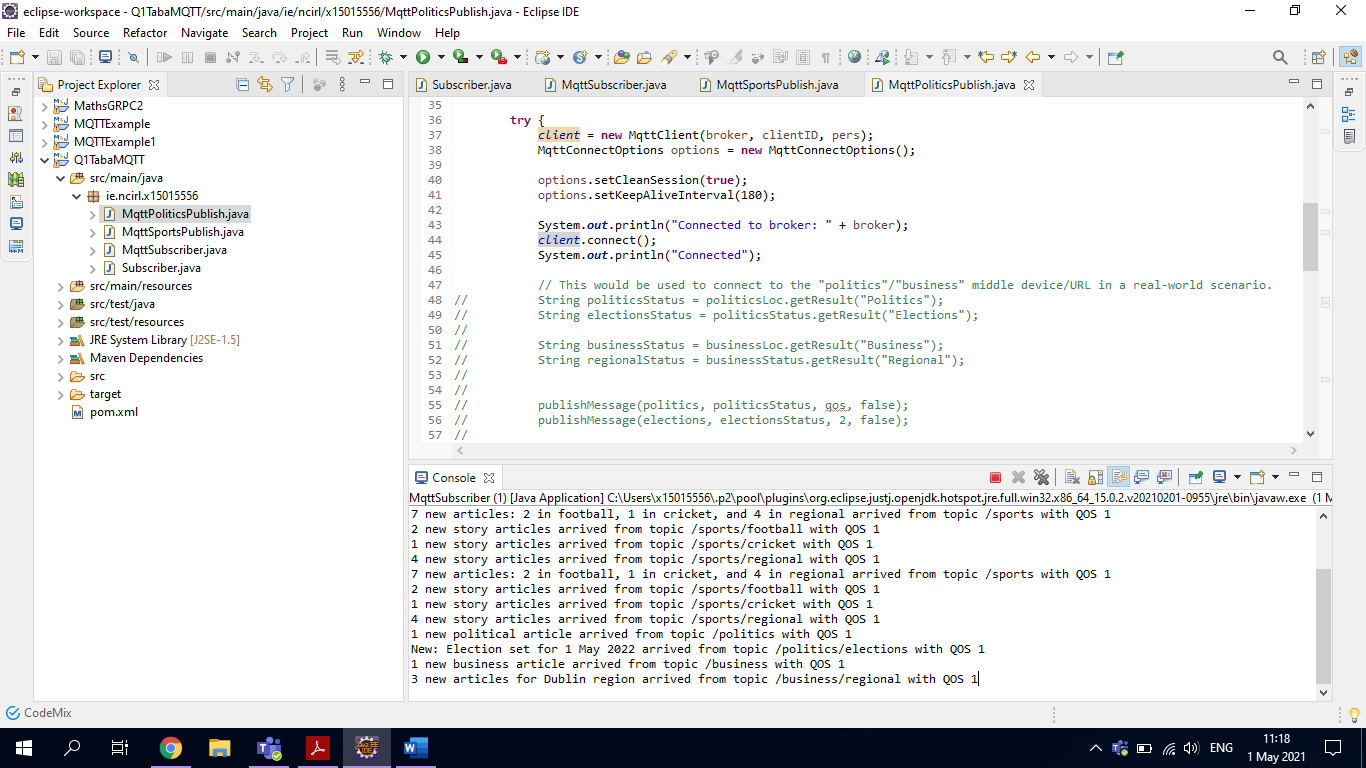
4 new story articles arrived from topic /sports/regional with QOS 1

1 new political article arrived from topic /politics with QOS 1

New: Election set for 1 May 2022 arrived from topic /politics/elections with QOS 1

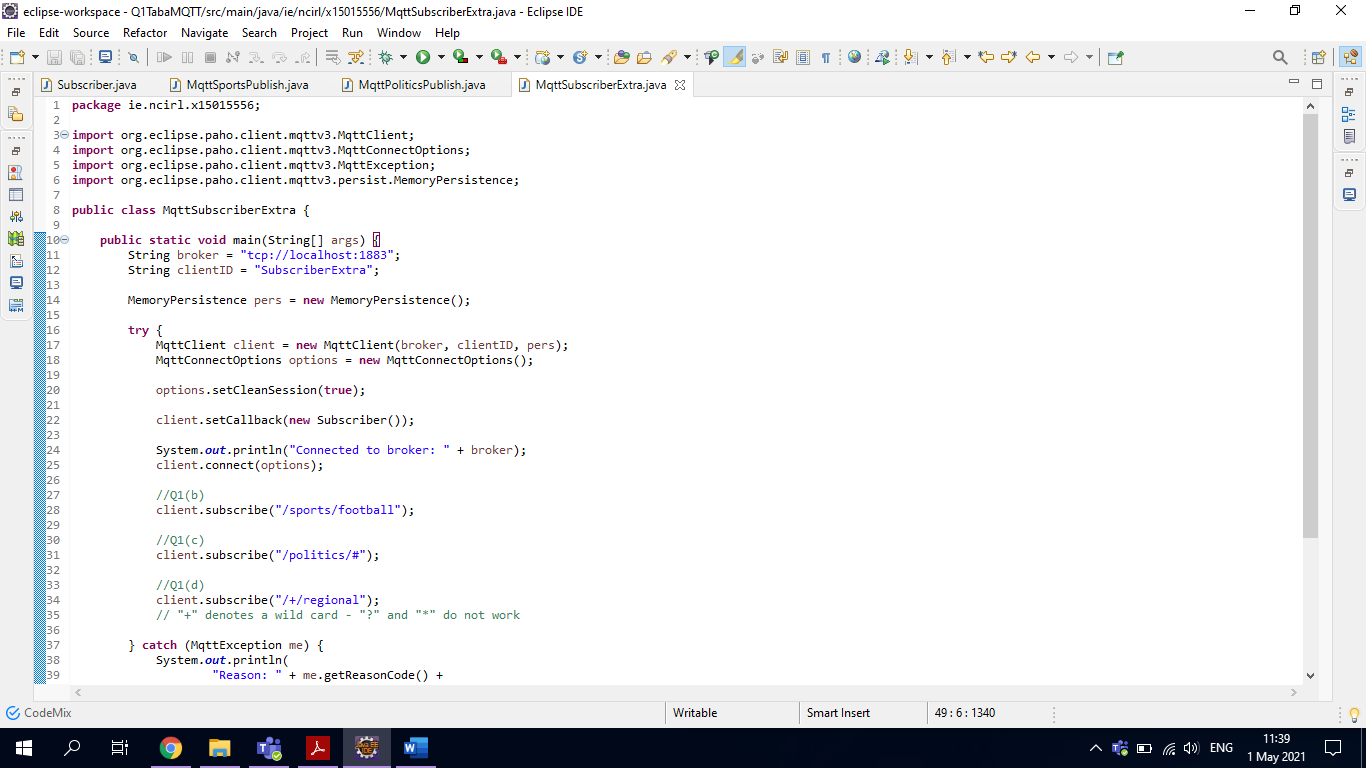
1 new business article arrived from topic /business with QOS 1

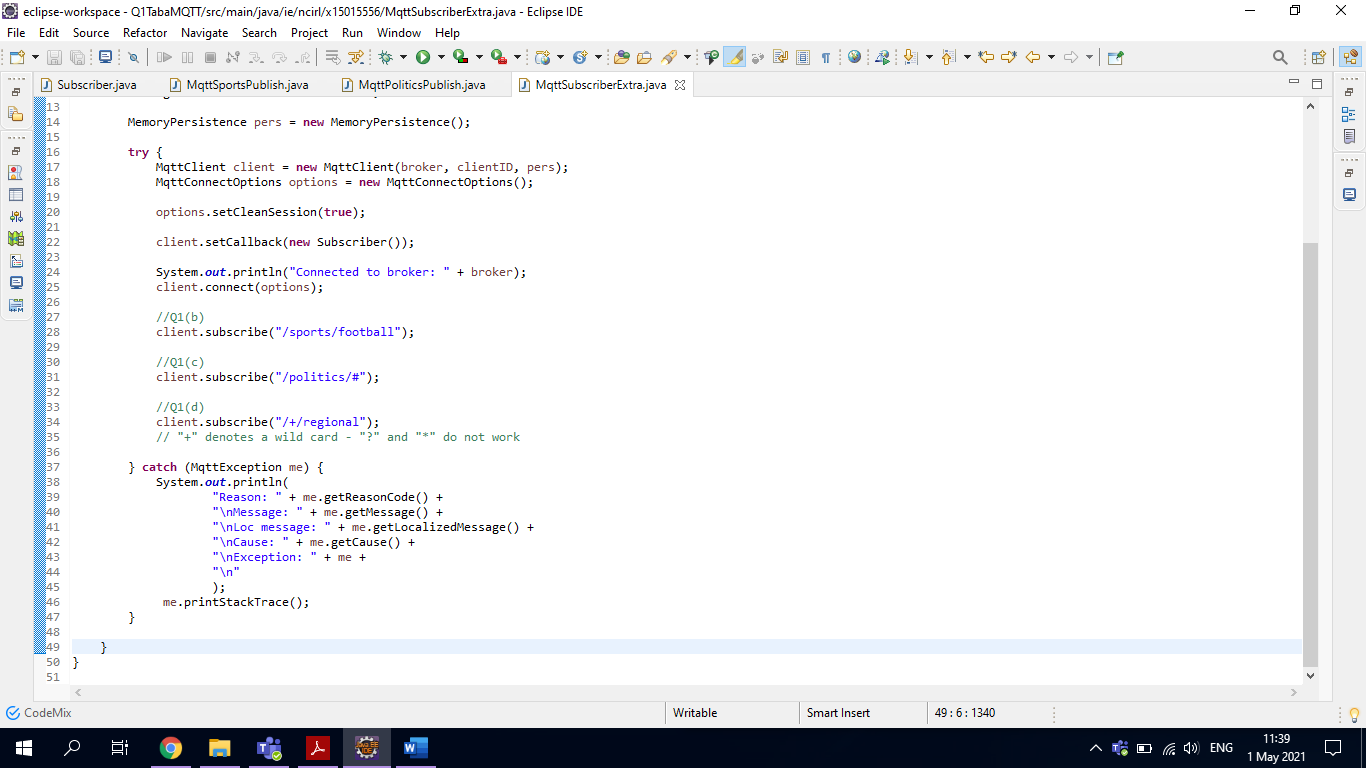
3 new articles for Dublin region arrived from topic /business/regional with QOS 1



## Parts B, C & D

### MySubscriberExtra.java





### Output from MySubscriberExtra.java

Connected to broker: tcp://localhost:1883

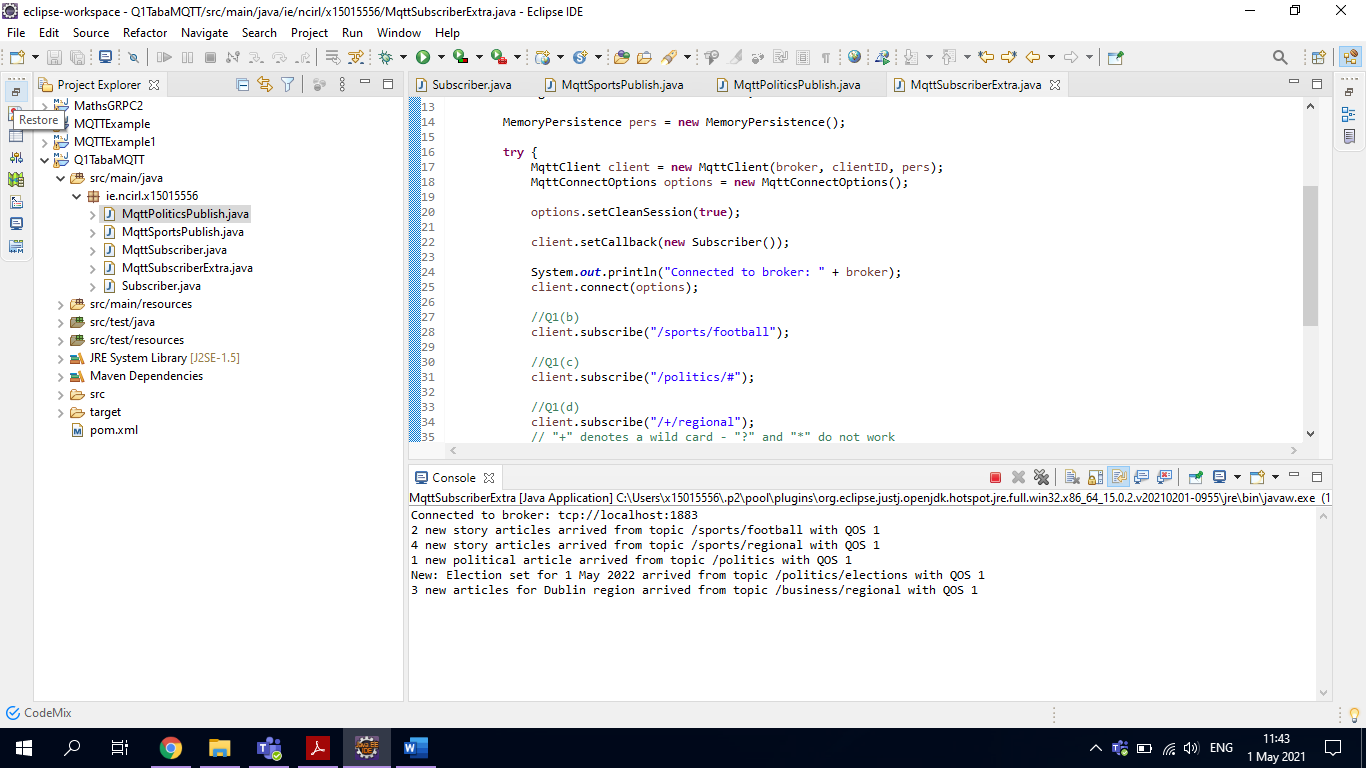
2 new story articles arrived from topic /sports/football with QOS 1

4 new story articles arrived from topic /sports/regional with QOS 1

1 new political article arrived from topic /politics with QOS 1

New: Election set for 1 May 2022 arrived from topic /politics/elections with QOS 1

3 new articles for Dublin region arrived from topic /business/regional with QOS 1



## Part E

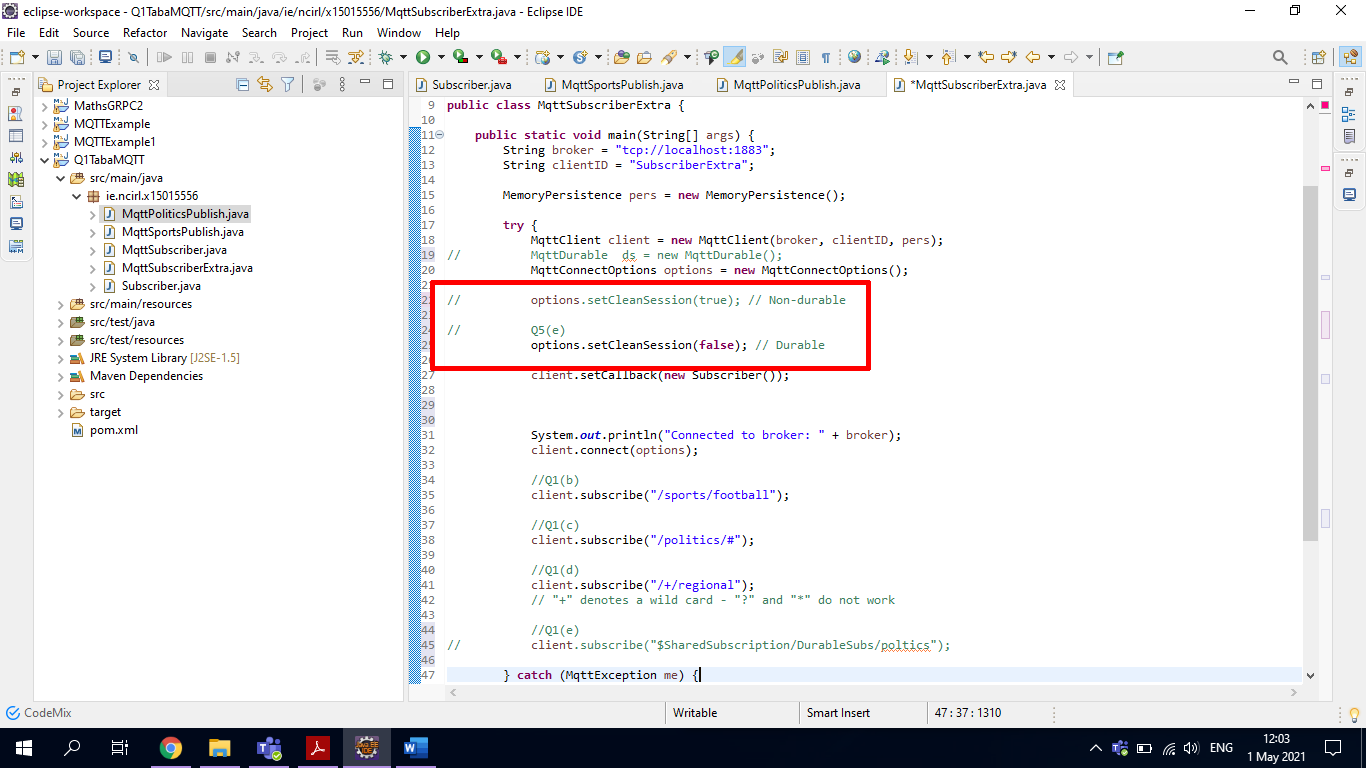
Non-durable:

options.setCleanSession(**true**);

Durable:

options.setCleanSession(**false**);

### Example of durable subscription used in MqttSubscriberExtra.java



# Question 2

## Amazon Web Services

The CAP in the CAP theorem stands for Consistency, Availability and Partition tolerance. In distributed systems, only two of the three are possible at one given time. PACELC is an extension to the CAP theorem and it stands for network partitioning, availability, consistency, else, (one must choose) latency or consistency. (Gilbert & Lynch, 2002) (Abadi, 2012) (Gall, 2019)

With Amazon Web Services, each virtual server (or Amazon Elastic Compute Cloud instances or EC2 for short) are located in an area called an “Availability Zone”. These zones are distributed over one or multiple data centres and servers. These data centres have mains power supply as well as backups to keep the power on. There are also connectivity and network availability that are also backed up in case of failure. (Brooker, 2020) (Brooker, et al., 2019)

To attempt to avoid partitions, duplication of data at different locations and on different servers is a must. The reproduced data is managed by a “fault-tolerant replication protocol”. (Brooker, 2020) This protocol is kept somewhat secret by Amazon. But Amazon does explain that this protocol is used for the likes of load balancing and queuing services. (Amazon Web Services, Inc, 2019)  
  
However, when a partition does ensue, the servers that are involved contact an amenity called the “configuration master”. The master hold small bits of data about where the partitions are, their size, and the process of the replication process. This is to ensure the information that is the newest is used. (Brooker, 2020) (Brooker, et al., 2019)

The configuration master is a large database called Physalia. To be more accurate, Amazon claim Physalia is a large database with “a collection of millions of tiny databases”. Physalia also implements failure corelation. Whether the customer of that particular service is having a problem is really not an issue. If it can be assured that if the service has issues and the customer of that service is also having issues, the failure methods can be synced, which would provide seamless availability to the customer, which is important. (Brooker, 2020) (Brooker, et al., 2019)

Physalia helps to resolve the CAP theorem by handling the partition part of CAP. It does this by knowing the data centre, knowing where everything is being stored and where there is and isn’t power using a “power topology”. In each cell of Physalia, the style of the database is using the idea that each cell is a “logical unit”. (Brooker, 2020) (Brooker, et al., 2019)

In Physalia, all sections in the Availability Zone contain its very own cell. In each of these cells, there are seven duplicates of the configuration files for that specific area. These configuration files are held within seven different servers. Each of these configuration files are called a node in the Physalia database. One particular server will contain thousands of these collections of configuration files, also known as nodes. (Brooker, 2020) (Brooker, et al., 2019)

Amazon Web Services’ customers depend on availability and this is how Amazon create and make features in their business. With availability, downtime should be kept to a minimum with decreasing the chance of problems to the smallest group of its customers as possible. Amazon calls this “blast radius reduction”. This is a main feature of Physalia, and it is a staple design principal. (Brooker, 2020) (Brooker, et al., 2019)

To resolve any issues that may arise, Physalia considers that the nodes should be nearby to reduce the chance they will be too far away to communicate effectively. Nonetheless, the nodes should not be too near to each other or to allow the same power supply on both nodes, as if this was the case two nodes could go down, which could cause issues to the cell. (Brooker, 2020) (Brooker, et al., 2019)  
  
With Physalia knowing the topology for the database, cells are kept to a minimum and are to be kept in the same localised area. This keeps consistency in the cell and safeguards that the data does not become unusable. (Brooker, 2020) (Brooker, et al., 2019)

# Question 3

## Part A: The network is reliable

Actuality: The network is not reliable.

The distributed system may be incorporating an external source resulting in a call over a network not being sent correctly. If one is making a query call to the system, one can simply try again.

Error handling is important to identify and fix calls over a network. The system becomes “non-deterministic” if error handling is not implemented. To manage all of these can become complicated.

To reduce the chances of network failure, one can capitalise the latest hardware, infrastructure and software.

(Carr & Staikopoulos, 2020)

## Part B: The network is homogenous

Actuality: The network is not homogenous.

A homogenous network consists of devices communicating that are using configurations and protocols that are alike, which is difficult to achieve. It’s important to guarantee that the parts of the system can accurately communicate with each other. Standard protocols help to achieve this over different systems.

To manage this, simple code formats should be used. Such examples include XML, JSON or protocol buffers. Using non-open source protocols will hinder applications and how they operate.

(Carr & Staikopoulos, 2020)

# Question 4

## Part B

### RMI Overview

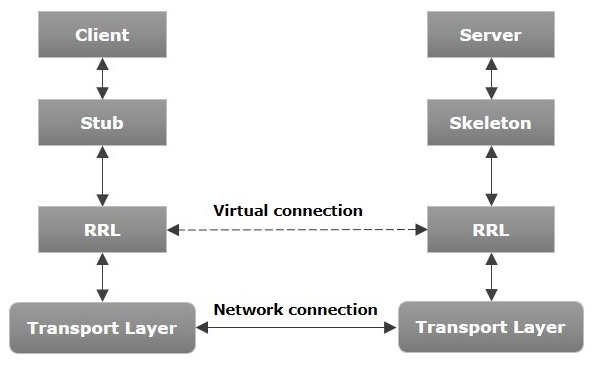


Image: (TutorialsPoint, 2021)

As far as the client is concerned, all data comes from the server and vice versa, but in fact this is not the case.

The **Transport Layer** is the actual network connection. It hands the current connections and creates new ones.

The **Remote Reference Layer (RRL)** is the layer that controls the references from the client to the server.

The **Stub** stays on the client side and is an entry point for the client code.

The **Skeleton** is an object that is on the server side to send data to the client side. The stub interacts with the skeleton to handle requests to the client.

(TutorialsPoint, 2021) (Oracle, 2021)

### How RMI Works

To explain briefly how RMI works –

* The client invokes a call to the server. The stub receives this call and sends it to the RRL.
* The RRL has the request, it invokes the object “remoteRef”. Then it sends this request to the server-side.
* The RRL on the server side then sends the request to the Skeleton
* The Skeleton invokes the necessary object, which is on the server.
* Then the result is sent in reverse back to the client.

(TutorialsPoint, 2021) (Oracle, 2021)

### Marshalling

In the above example of how RMI works, before being sent to the client, the object is marshalled. This means that the object, if a simple type, has a header attached to it. If the object is not a simple type, they are serialised.

On the server side, the object is “opened” and then the method is called. This is unmarshalling.

(TutorialsPoint, 2021) (Oracle, 2021)

### RMI Registry

This is a namespace. All of the server objects are added to the register. When the server makes and object, it registers it to the registry. A unique name is used.

To call a remote object, the client creates a reference for that object. Then the client gets the object from the registry using its unique name.

(TutorialsPoint, 2021) (Oracle, 2021)

An example of how the RMI Registry works is:

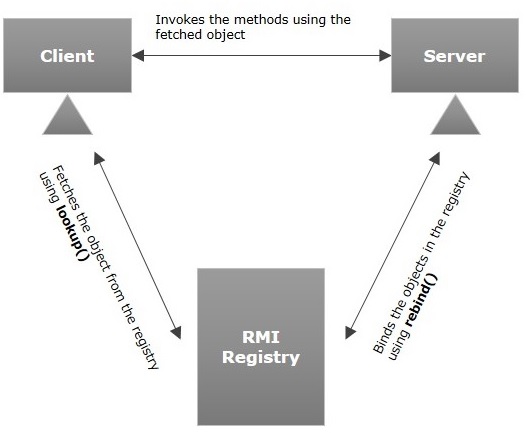


Image: (TutorialsPoint, 2021)

### Use of middleware and registry

The RMI Registry is sort of middleware and acts like a broker between the server and client. Without the registry, there would be nowhere for the client to call to as it cannot directly access the server.

## Part C

### How do components communicate in a distributed system?

Firstly, before two components can communicate with each other, their interfaces must be joined together. This is known as binding. Biding is generally language specific. (Pryce & Crane, 2018) In the example of the RMI above, the binding is between the stub and skeleton.

### Identify and compare …

… two different implementations that are based on different messaging paradigms (e.g. RPC, publish-subscribe)

#### RPC

The client-server model is used by RPC. RPC is synchronous meaning that when an operation starts, the operation is “stuck” until the results are returned. Nonetheless, multiple threads that use the same address can allow RPC operations in parallel. (Matturro, 2021)

Some advantages of using RPC include:

* Communication is done through procedure call in “high-level languages”
* Used in distributed or local environments
* Backs models oriented in processing and threading.
* Minimal coding required

(Matturro, 2021)

Some disadvantages include:

* Only suited for small amounts of data.
* Failure is prominent as it requires a communication system, another system and more than one process.
* No consistent standard, multiple implementations.

(Matturro, 2021)

#### Publish/Subscribe

This is a configuration of queue managers that are connected together. They can either be on the same system or on different servers. When the queue managers connected, one a subscriber can subscribe to one queue manager and receive messages from another queue manager. This other queue manager had message previously published to it. (IBM, 2021)

An example of how this could work is below:

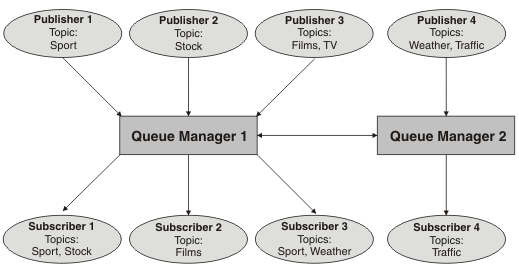


Image: (IBM, 2021)

## Comparison of RPC and Publish/Subscribe

With RPC, there are more tied relations between services but the request/response commination model is more simpler. When a service sends a request, it expects some kind or reply or it timeouts. With RPC with TCP, the session and communication are bidirectional. (Zv, 2017)

With Publish/Subscribe, a service sends a message. It is not interested in how the message will be received, just that it has been sent. When one service sends to just another one service, it doesn’t wait for a response. At the same time the reverse could be happening where the second server is sending a message to the first service. (Zv, 2017)

# References

Abadi, D. J., 2012. *Consistency Tradeoffs in Modern Distributed Database System Design,* New Haven, Connecticut: Yale University.

Amazon Web Services, Inc, 2019. *Fault-Tolerant Components on AWS,* Seattle, Washington, United States: Amazon Web Services, Inc.

Brooker, M., 2020. *Amazon EBS addresses the challenge of the CAP Theorem at scale.* [Online]   
Available at: https://www.amazon.science/blog/amazon-ebs-addresses-the-challenge-of-the-cap-theorem-at-scale  
[Accessed 1 May 2021].

Brooker, M., Chen, T. & Ping, F., 2019. *Millions of Tiny Databases,* Seattle, Washington, United States: Amazon Web Services, Inc..

Carr, D. & Staikopoulos, A., 2020. *Distributed Systems - Fallacies of Distributed Systems,* Dublin: National College of Ireland.

Gall, R., 2019. *The CAP Theorem in practice: The consistency vs. availability trade-off in distributed databases.* [Online]   
Available at: https://hub.packtpub.com/the-cap-theorem-in-practice-the-consistency-vs-availability-trade-off-in-distributed-databases/  
[Accessed 1 May 2021].

Gilbert, S. & Lynch, N., 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News,* 33(2), pp. 51-59.

IBM, 2021. *Distributed publish/subscribe networks.* [Online]   
Available at: https://www.ibm.com/docs/en/ibm-mq/8.0?topic=messaging-distributed-publishsubscribe-networks  
[Accessed 1 May 2021].

Matturro, B., 2021. *Remote Procedure Call (RPC).* [Online]   
Available at: https://searchapparchitecture.techtarget.com/definition/Remote-Procedure-Call-RPC  
[Accessed 1 May 2021].

Oracle, 2021. *Java Remote Method Invocation Distributed Computing for Java.* [Online]   
Available at: https://www.oracle.com/java/technologies/javase/remote-method-invocation-distributed-computing.html  
[Accessed 1 May 2021].

Pryce, N. & Crane, S., 2018. *Component Interaction in Distributed Systems,* London: s.n.

Rotem-Gal-Oz, A., 2019. *Fallacies of Distributed Computing Explained,* s.l.: Rotem-Gal-Oz, Arnon.

TutorialsPoint, 2021. *Java RMI - Introduction.* [Online]   
Available at: https://www.tutorialspoint.com/java\_rmi/java\_rmi\_introduction.htm  
[Accessed 1 May 2021].

Zv, V., 2017. *In microservices should i use pub/sub instead RPC to get more loosely couple architecture?.* [Online]   
Available at: https://stackoverflow.com/questions/44579396/in-microservices-should-i-use-pub-sub-instead-rpc-to-get-more-loosely-couple-arc  
[Accessed 1 May 2021].