Lab 12:

Exercise 1: Security

Provided is an OAuth2 Authentication service.

Change the in-memory users so that we have 3 users:
One with the role customer, one with the roles customer and employee, and one with the roles customer, employee and manager.

Then create 2 microservices A, B and C.
C contains salary data
B contains employee contact data (phone)
A is out actual company service that is used by customers, employees and managers
In A you can call productdata that is accessible by all customers, employees and managers
In A you can call employee contact data that is accessible only by employees and managers.
A will call B to get the actual employee contact data.
In A you can call salary data that is accessible only by managers. A will call C to get the actual salary data.

Exercise 2: Kafka.

Starting Kafka:

Go to **C:\architecturetraining\kafka_2.11-1.1.0** and double-click **startzookeeper.bat**

Then double-click **startkafka.bat** and wait till kafka is started.

Import the given project **KafkaProject** into Eclipse and run it.

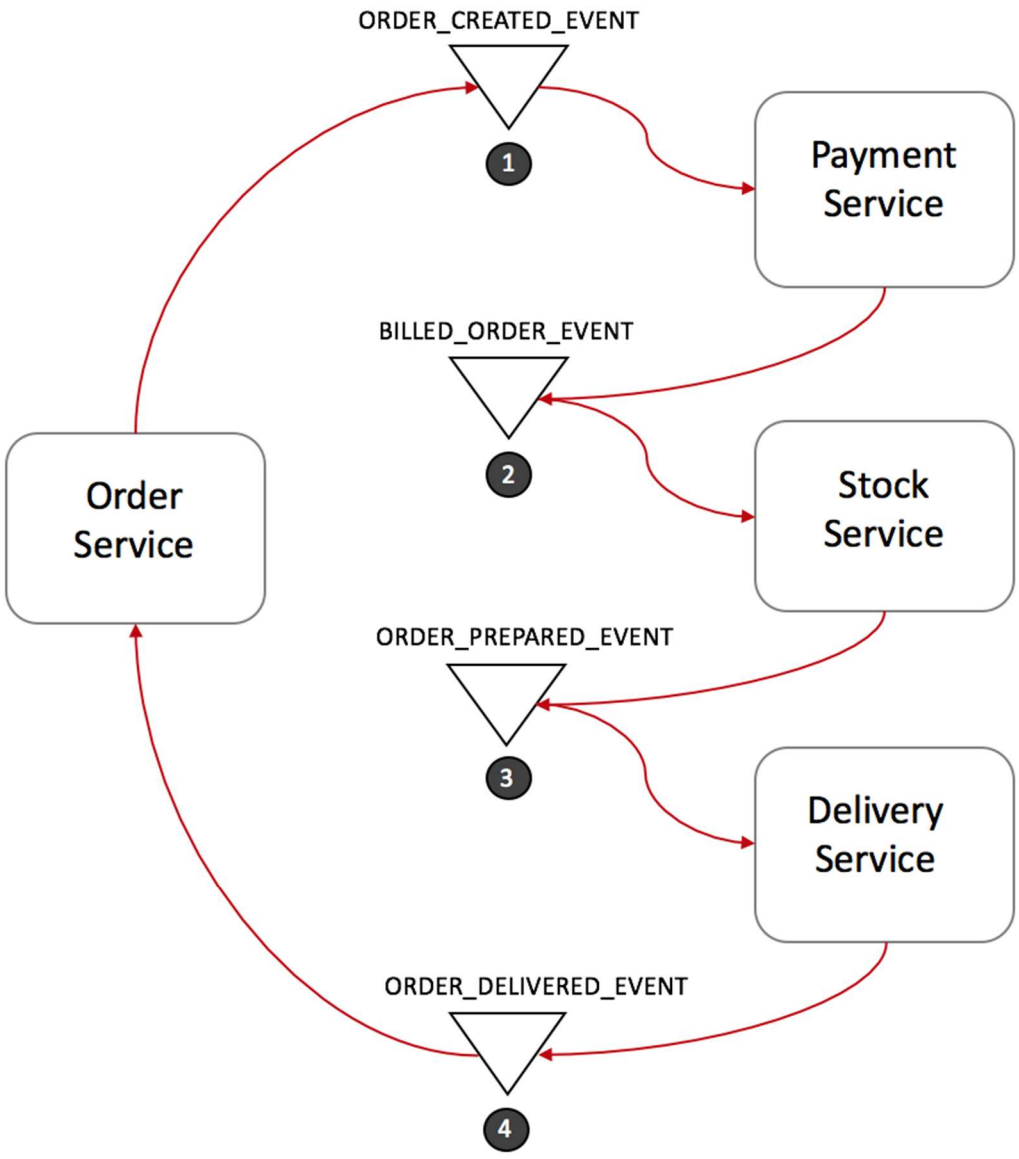
If everything works correctly, you should see the following output:

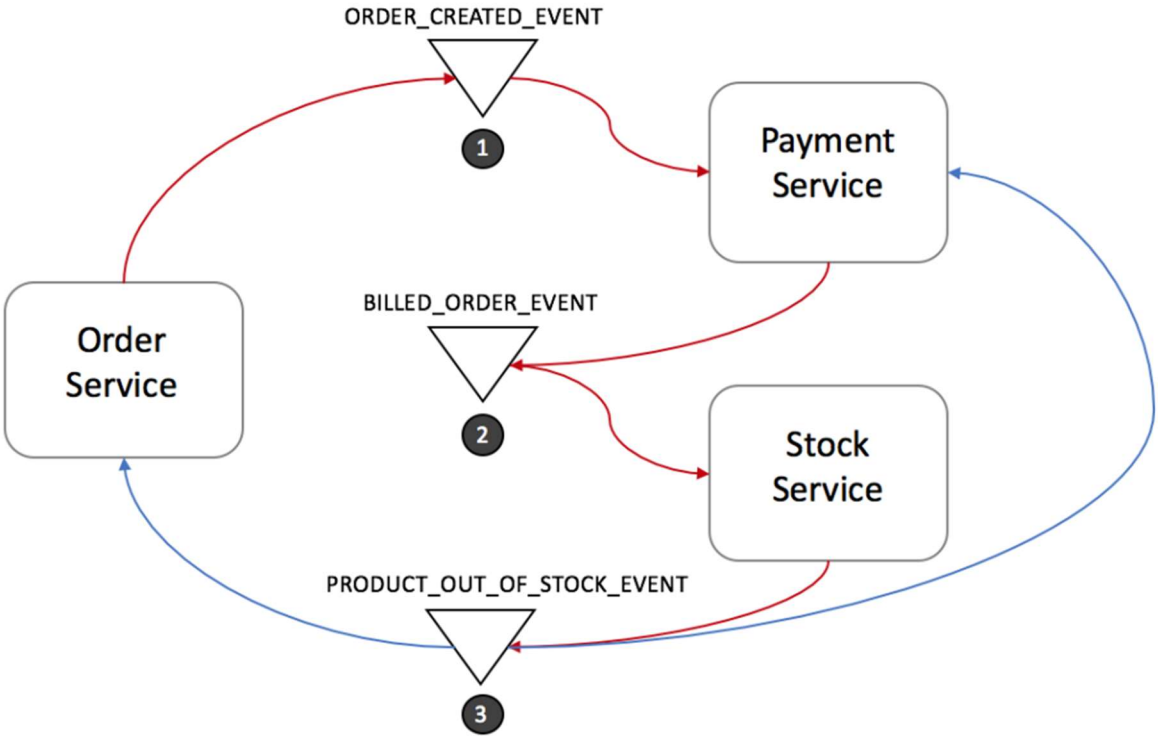
```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.1.RELEASE)

sending message=Spring Kafka and Spring Boot Configuration Example to topic=greetingtopic
received message=Spring Kafka and Spring Boot Configuration Example
kafka_offset : 0
kafka_consumer : org.apache.kafka.clients.consumer.KafkaConsumer@189bb91
kafka_timestampType : CREATE_TIME
kafka_receivedMessageKey : null
kafka_receivedPartitionId : 0
kafka_receivedTopic : greetingtopic
kafka_receivedTimestamp : 1531252357917
```

Now create 4 different projects, 2 producers that produce different content to the same topic, and 2 consumers that listen to this topic. Play with it so that you see that:

- Messages remain in the topic
- Every consumer has its own offset.

Now implement an event driven microservice architecture that implements the following scenarios:





Only implement the Order, Payment and Stock service (NOT the Delivery service)

You can keep the microservices as simple as possible. We only send messages that contain Strings.