```
====================
SAAGA Design Pattern
====================

=> It is a design pattern to manage complex transactions in microservices

=> In Microservices architecture multiple services will be involved

=> Every service will maintains its own data and every service is responsible to to perform specific
operation

Ex:
                                order_service  : will place order
                                payment_service : will process payment
                                order_fullfill_service : will confirm/fail order

=> First user will place order and will proceed for payment.

=> If payment is succesful then order should be confirmed else order should be failed

=> Based on payment event order_fullfill_service will take action

Note: To complete user request co-ordination is required between above services

=> As these 2 serviecs using 2 diff databases, managing tx commit and rollback is very challenging.

Note: We need to implement Distributed Transaction to handle this scenarion

### SAGA design pattern is used to manage distributed transactions in the application.

=> SAGA Transactions means sequence of transactions

=> SAGA pattern is used to simplify distributed tx management in microservices architecture

=> SAGA Pattern we can implement in 2 ways

                1) Choreography   (event based)

                2) Orchestration  (command based / controller)

=> Choreography patterns works based on events (We will use message broker to publish & subscribe
events)



==============
Project Setup
==============

1) Create Maven Multi Module Project

2) Create Maven Module (Common-Bindings)

3) Create Maven Module (Order-Service)

4) Create Maven Module (Payment-Service)

5) Add required dependencies in parent pom.xml

====================================
Common-Bindings Project Development
====================================
```

1) Create Entity Classes

- PurchaseOrder.java (purchase_orders_tbl)
- PurchaseOrderPayments.java (purchase_orders_payments_tbl)
- UserBalance.java (user_bal_tbl)

2) Create Repositories for entity classes

3) Create Required Request & Response Binding Classes

- OrderRequestDto.java
- OrderResponseDto.java
- OrderStatusEnum.java
- PaymentStatusEnum.java

4) Add Common-Bindings project as a dependency in order-service & payment-service

==========
Order-Service Project Development
==========

1) KafkaProduceConfig.java

2) KafkaConsumerConfig.java

3) RestController
- createOrder
- getOrders

4) Service
- createOrder (save in db & publish msg to order-topic)
- getAllOrdersFromDB
- consumePaymentsTopicMsg (check pay status & confirm/cancel order)

==========
Payments-Service Project Development
==========

1) KafkaProduceConfig.java

2) KafkaConsumerConfig.java

3) Service
- handleOrderPayment (listen to orders-topic)
  - check order amt & user balance
  - if user having sufficent bal then deduct bal and update payment completed
  - produce payment status to payments-topic

==========
Kafka Setup
==========

1) Run Zookeeper server

2) Run Kafka Server

3) Open Kafka tool and connect to it

4) Run Order Service (order-topic will be created)

5) Run Payment Service (payments-topic will be created)

6) Send Post req to order service and verify data

Git Hub Repo : https://github.com/ashokitschool/SAGA_Choreography_Implementation