**Desigin Pttren**

**Enum**

**Reacative programming**

**Kafaka**

**Design Pattern:**-jab hum software design krate hain to bahut sari problem ati hain to un problem ko bahut se developer apne -2 tarike se solve krte hain but jaroori nahi ki unki approach sahi ho to unhi problem ko solve krne ke liye design pattern use kiya jata hain

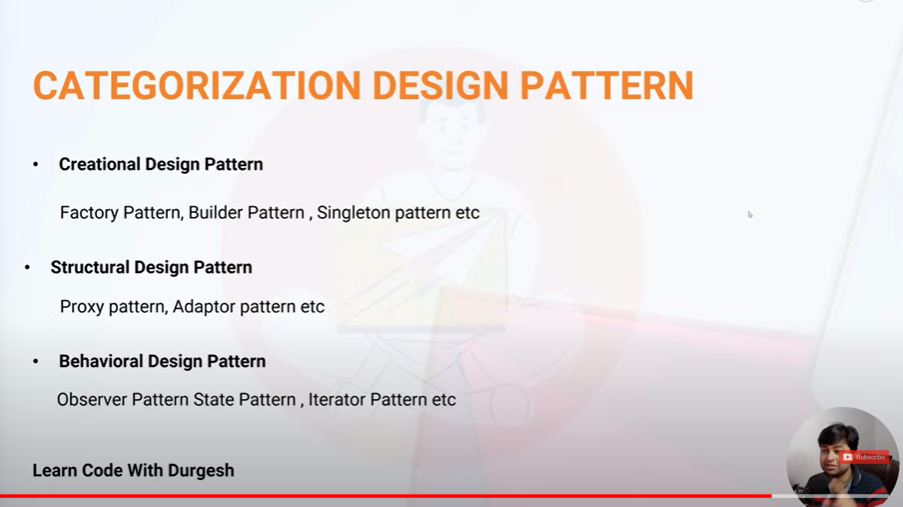
Ye isliye use krte hain ye common tarika hota hain isme ye pata hain ki agar is type ki problem ayi to ye pattern use hoga

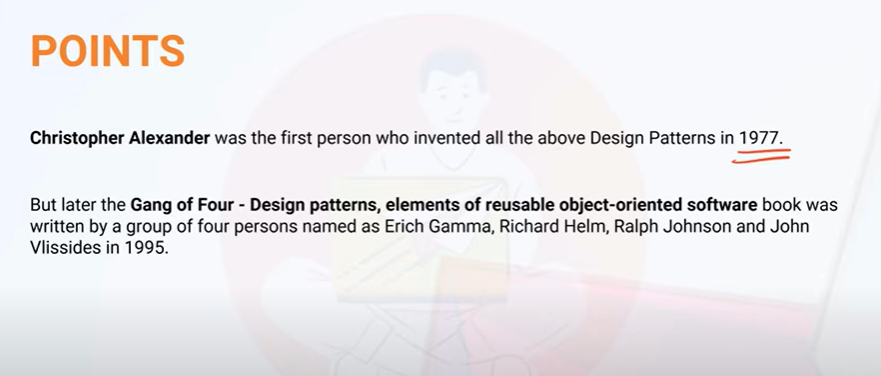


Creational Design pattern :-object create se realated hote hain

Structural Design:ye class se realated hote hain

Behavioral Design patten: ye object ke behaviour per hota hain

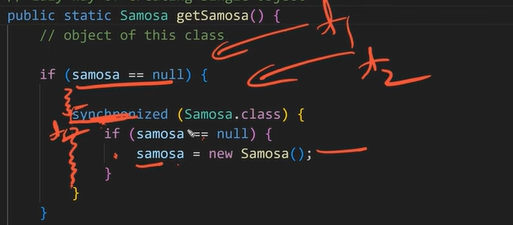




**Singleton Design Pattren:-**isme hum object ko ek baar create krte hain agr hum chahate hain ki naya object na mile same object mile baar -2 to hume ise use krte hain

**Docker,Jenkins,kafaka,**

* *Lazy of creating single object-jab app call kroge tab hi call hoga isme hum syncoronized block laga dete hain thread safety ke liye*

****

**1.Reflection Api to break singleton pattern**

-> if object is there =throw exception from inside constructor

**Solution-**1-use enum

**2.DeSerilization-**break singletion pattern

**Soloution:**implementing readResolve method

**3.Cloning krke bhi break kr sakte hian**

**Singleton Design Pattern**

The Singleton design pattern is a creational design pattern that restricts the instantiation of a class to a single object and provides global access to that instance throughout the application. This pattern ensures that only one instance of a class is created and provides a global point of access to it.

**Singleton Object**

Singleton object are the object which are instantiated only once for project (jvm). If we try to get the object then we get same object again and again.

**lets create object using singleton pattern using java**

Lazy way of creating singleton object

class Example{

private static Example ob;

public static Example getExample(){

if(ob==null){

ob=new Example();

}

return ob;

}

}

Copy

**calling the singleton object**

class Main

{

public static void main(String args[]){

Example ob=Example.getExample()

//using the object

}

}

}

Copy

Eager way of creating Singleton object

class Example{

private static Example ob=new Example();

public static Example getExample(){

return ob;

}

}

Copy

Accessing object

class Main

{

public static void main(String args[]){

Example ob=Example.getExample()

//using the object

}

}

}

Copy

note: for multithreaded environment we use syncronized block for creating singleton object.

class Example{

private static Example ob;

public static Example getExample(){

if(ob==null){

syncronized(Example.class){

if(ob==null)

{

ob=new Example();

}

}

}

return ob;

}

}

Copy

**Breaking Singleton Design Pattern**

**There are three ways to break singleton design pattern . Lets talk about it. I am also going to tell you about the solution of these problems.**

**1. Using Reflection API**

With the help of relfection api we can call private constructor  and create multiple object by calling private constructor.

**lets see how we can call private constructor**

Constructor<Example> constructor=Example.class.getDeclaredConstructor()

//changing the accessibility to true

constructor.setAccessible(true)

Example example=constructor.newInstance();

Copy

**solution**

**we can do the soultion in two ways.**

1. **using ENUM**

public enum Example{

INSTANCE

}

Copy

1. **check the object in private constructor if the object exists then throw exception to terminate the execution.**

private Exmaple(){

if(ob!=null)

{

throw new RuntimeExcepiton("you are trying to break singleton pattern")

}

}

Copy

**2. Using Deserialization**

when we serialize and deserialize the singleton object then singleton pattern automatically got destroyed and provide us different object.

ObjectOutputStream oos = new ObjectOutputStream(new

FileOutputStream("abc.ob"));

oos.writeObject(ob);

System.out.println("serialization done..");

ObjectInputStream ois = new ObjectInputStream(new FileInputStream("abc.ob"));

Example s2 = (Example) ois.readObject();

System.out.println(s2.hashCode());

Copy

**solution:**

just implement readResolve() method

public Object readResolve() {

return ob;

}

Copy

**3. Using cloning**

when we clone then also we get different object.

**solution**

just override clone method and return the same instance.

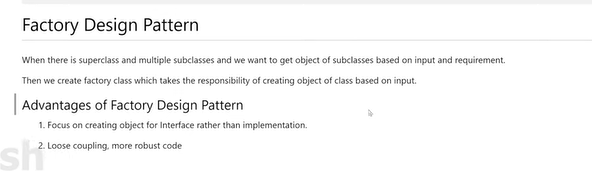
@Override

public Object clone() throws CloneNotSupportedException {

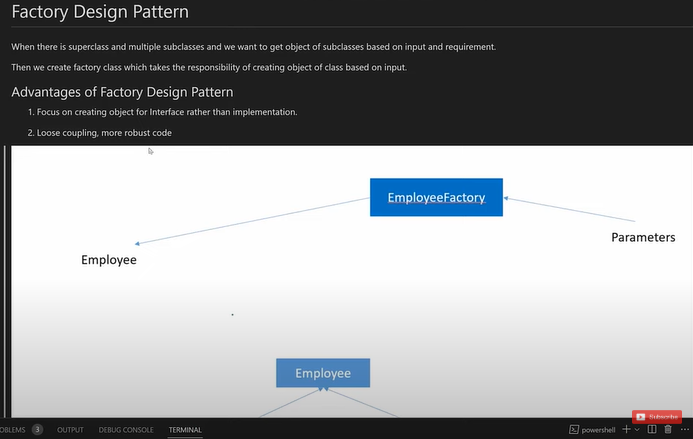
return samosa;

}

**Factory Design Pattren:-**

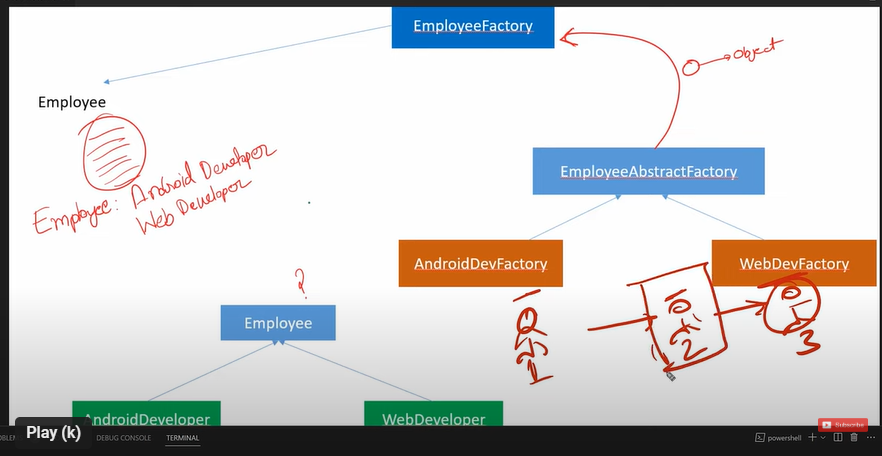
****

**Abstract Design Pattren:-**

****

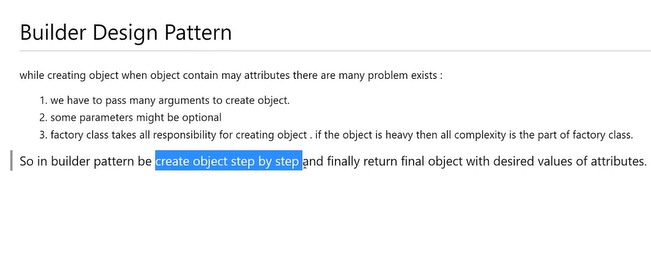
**Abstract:-**

****

****

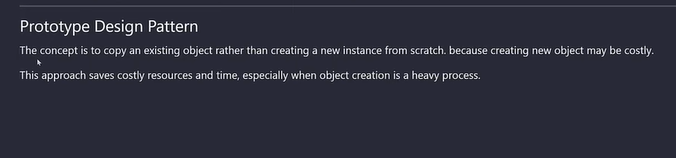
**Isme ek layer mid main jud jati hain jise ye aur bhi flexible ho jati hain**

**Builder Design Pattren:-**

****

**Jadatar builder pattern ka user immutable object create krne ke liye hota hain aur build method se object create hota hain**

**Prototype Design Pattren:-**

****

**Isme hum object phir se use kr sakte hain**

**Isme hum cloning krte hain**

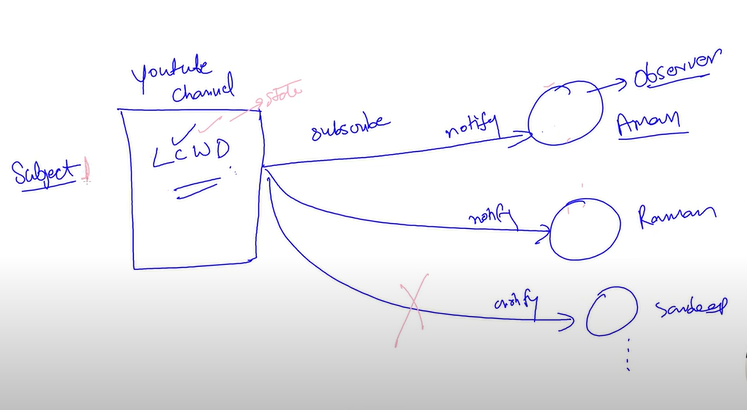
**Shallow copy**:-ek object hain aur use hm copy kre aur use object main student ka object hain kyuki student list type ka -to waha per student object ka refernce copy hoga jise problem ye hoga ki agar hum agar copy kiye object main jo refrence diya tha usme agar hum kuch change krenge to original refernce main bhi changes ho jayenge

**Deeep Copy:-ye shallow copy ki problem ko solve krne ke liye aya hain**

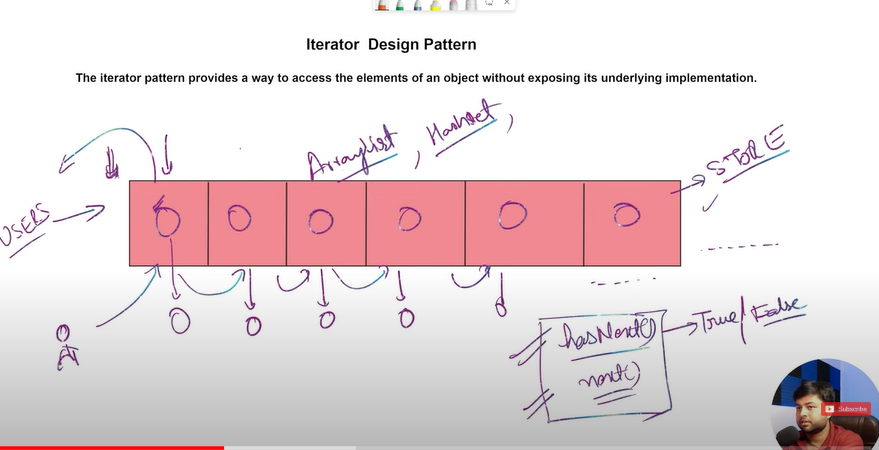
Isme hum jo refernce hain usme hum ek naya refence bana lenge to copy krne per olde refence per koi frak nahi padega naya jo banay use main change honge

**Observer Pattren:-**

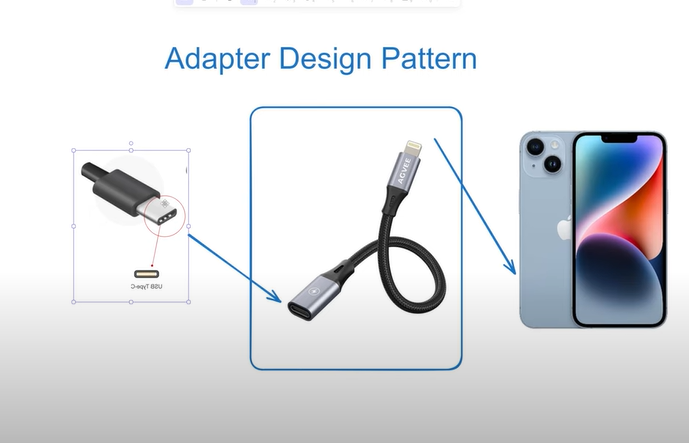
****

****

**Iterator Design Pattren:-**

****

**Adapter Design Pattren:-**

****

**Generic:-**Generic hum type safety ke liye use krte hain for example agar humne ek arraylist banaya aur uska type string kr diya to ab usme hum sirf String hi value de sakate hain

It is used for type safety. For example I made a object of arraylist and this object made type of String then afterthat we can give a string value in this object of arraylist then it is called generic.

**Enum:-** enum keyword hota hain jaise aap interface ,class banate hain ho waise hi enum banate hain

**Java 1.5**

**Enum is the constant or the named constant**

**Fix value ko samabhal ke rakhane ke liye hum enum ka use krte hain**

**Hum class main bhi rakh sakate hian but usme jada likhan padega**

**For example**

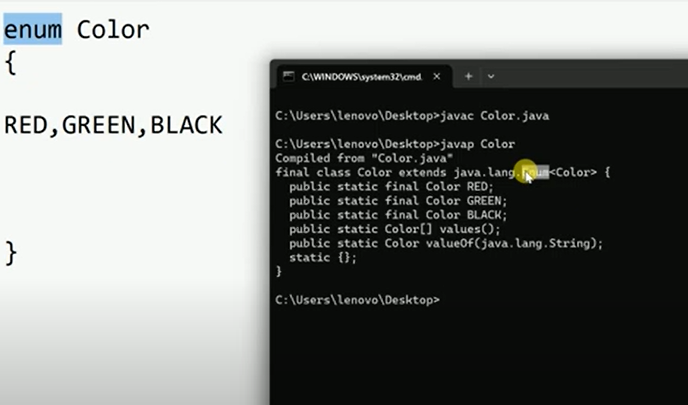
**Class color{  
public static final String Red=”red”**

**}**

**enum color{**

**RED,GREEN**

**}**

****

**enum ek keyword and Enum class ko inherite krati hain**

**Enumaration ek interface hota hain and find the java.util.\*;**

**Made for enum Not allowed:**

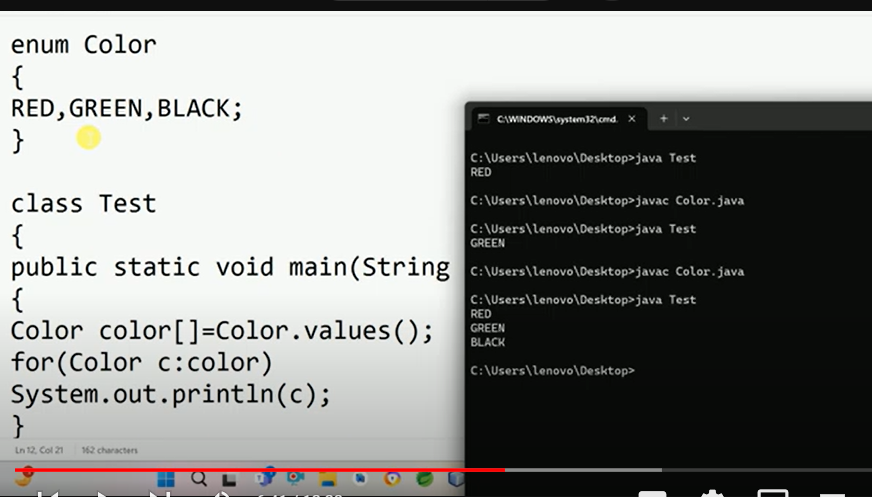
**Final,private ,protected**

**Made for enum allowed:**

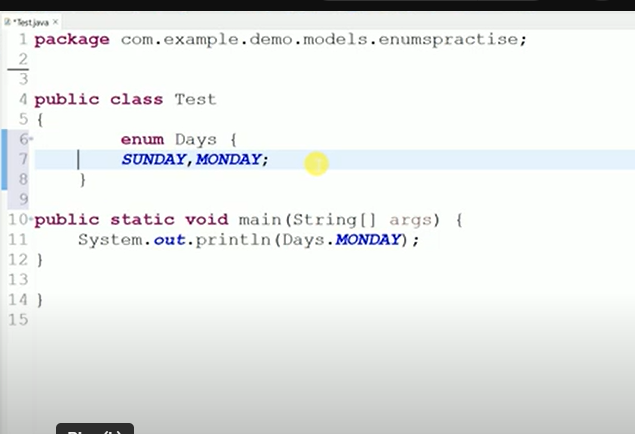
**public**

**enum created two ways:-**

**outside the class:-**

****

**inside the class :-**

****

**Ye per private ,protected bhi banasakate ho**

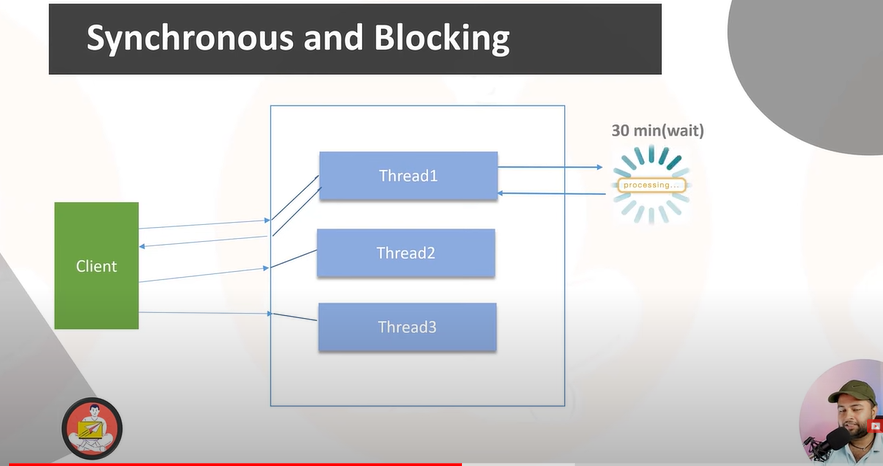
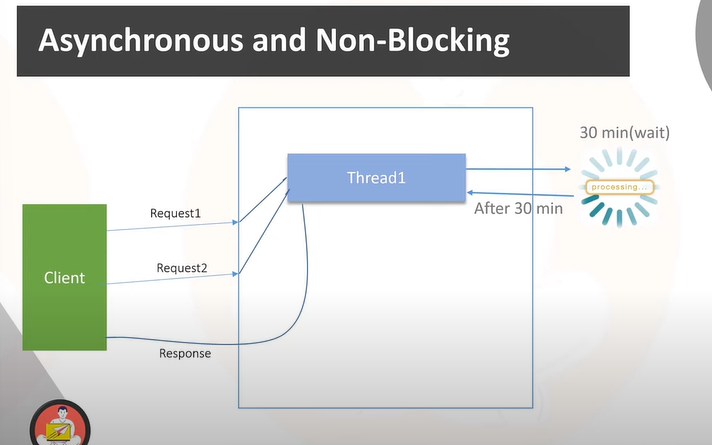
**Enum ke andar constructor ,variable ,method bhi bana sakte ho**

**Enum can not create abstract kyuki class ke sath ye final bana gaya hota hain**

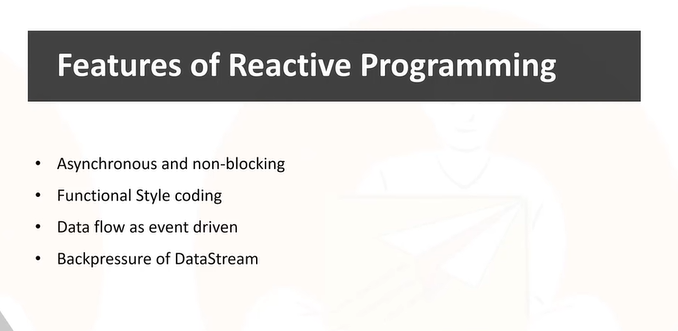
**Enum cannot create object,inherit,**

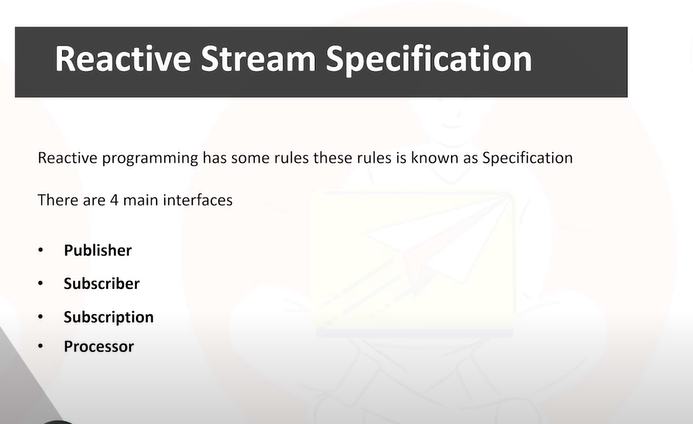
**SpringBoot Reactive Programming:-**

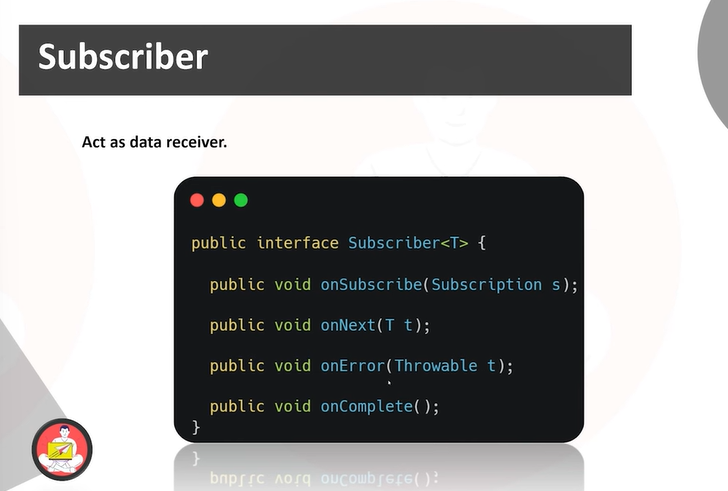
**Reactive Programming:-**

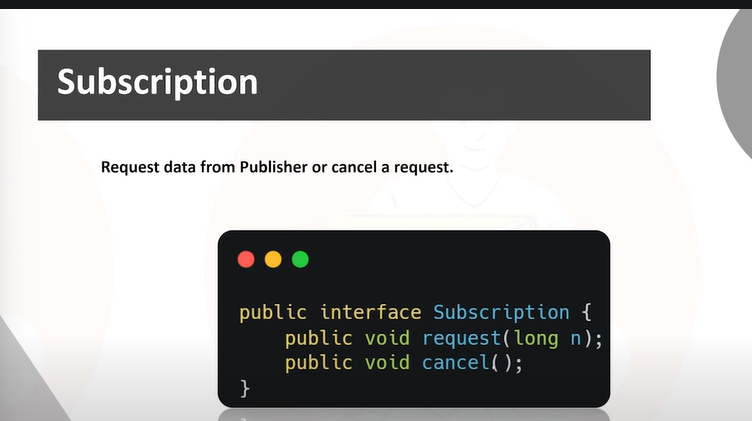
* Ek tarika hota hain jisme hum use krte hai**n asynchronous and non** blocking data streams to handle data events
* ****
* **Isme client ek request lega aur uske baad usko reponse deke dusrea request lega**
* ****

**Ye jitani bhi request hogi wo leta rehega aur jitana time lagega ek-2 krke sab reponse dega**

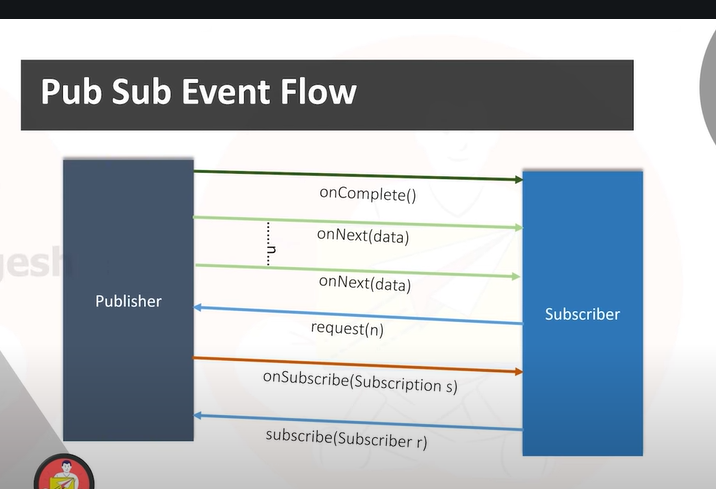
****



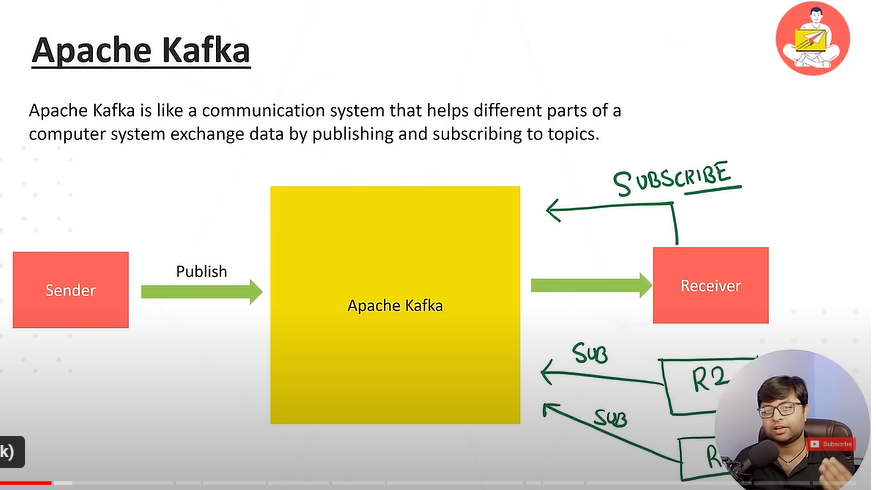




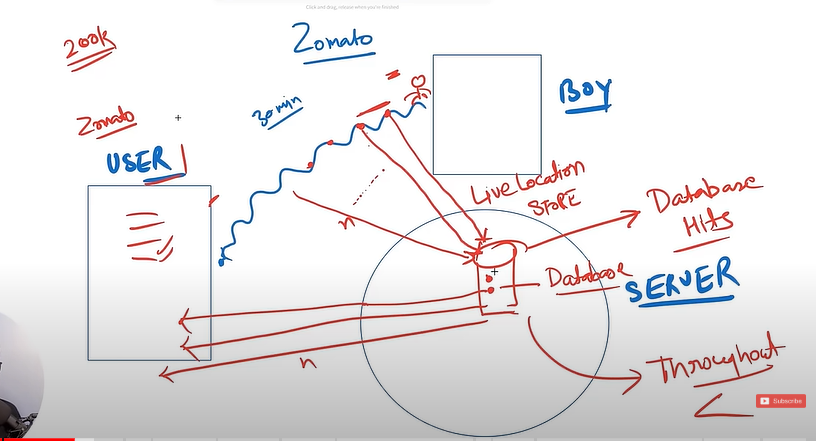




**Apache Kafaka:-**

****

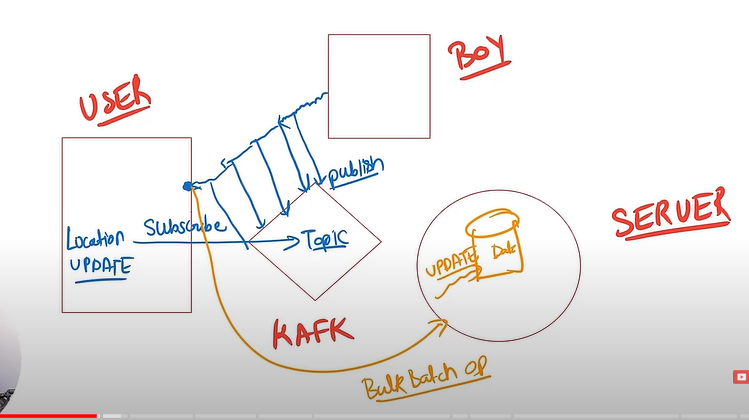
**Database hits jada hota hain to usi jagah per hum kafaka use krte hain**

****

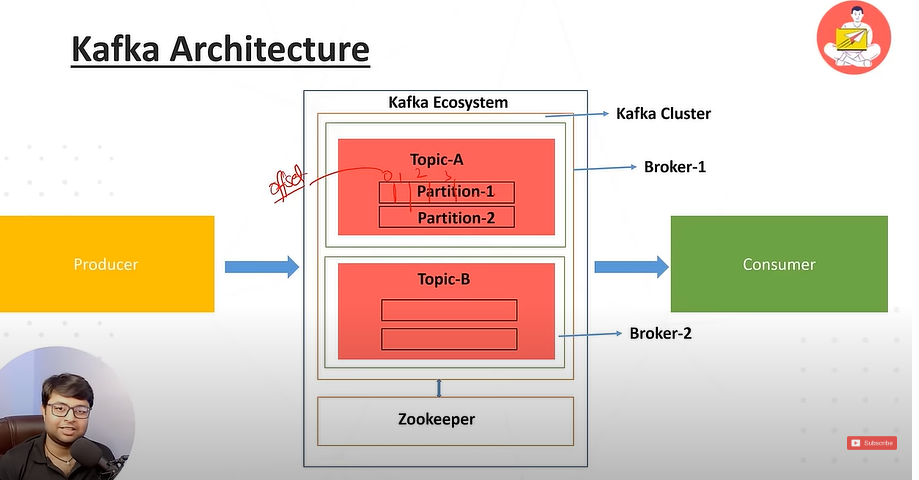
**In this example –**

* isme ek end user hain aur ek zimati boy
* apne ek parcle mangaya aur wo 30 min a jayega location dikha raha hain
* to wo jaise -2 aapki taraf ayega to uski location track hogi
* wo server ke pass data jayega pahle wo server user ko response de dega ki kaha hain
* her time per database se data read hoga to database main problem a jayegi
* kyuki database main data store karane ke liye hota hain
* usi problem ko solve krne ke liye kafaka ka use hota hain

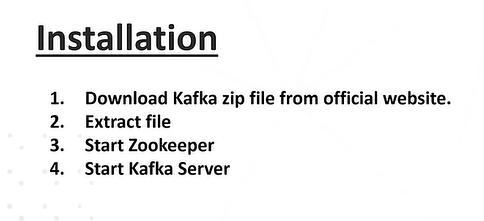
**Apache Kafaka:-**

****

****

****

**Zookeeper kafaka ko mange krta hain**

****

**C:\Kafaka\kafka\_2.13-3.7.0>bin\windows\kafka-topics.bat –create**

**--topic user-topic --bootstrap-server localhost:9092**

**Created topic user-topic.**

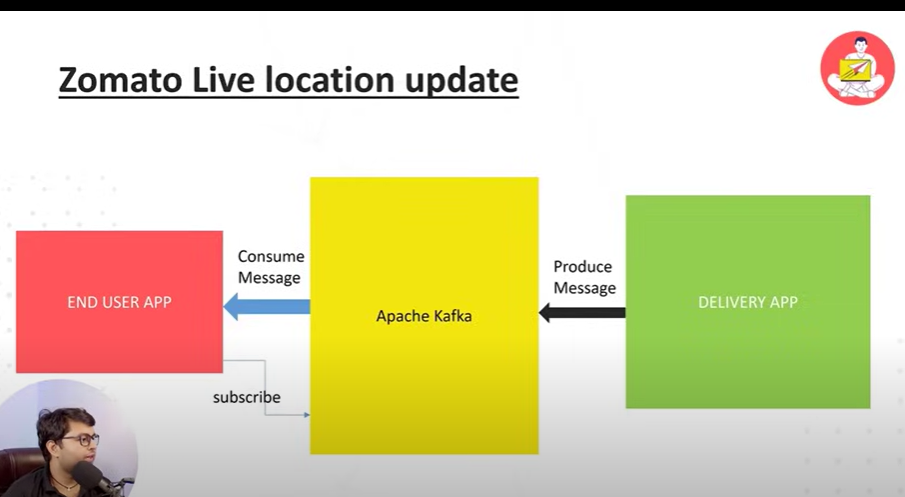
**C:\Kafaka\kafka\_2.13-3.7.0>bin\windows\kafka-console-producer.bat --topic user-topic --bootstrap-server localhost:9092**

**>hi**

**>this is my message**

**>**

**C:\Kafaka\kafka\_2.13-3.7.0>bin\windows\kafka-console-consumer.bat --topic user-topic --from-beginning --bootstrap-server localhost:9092**

****