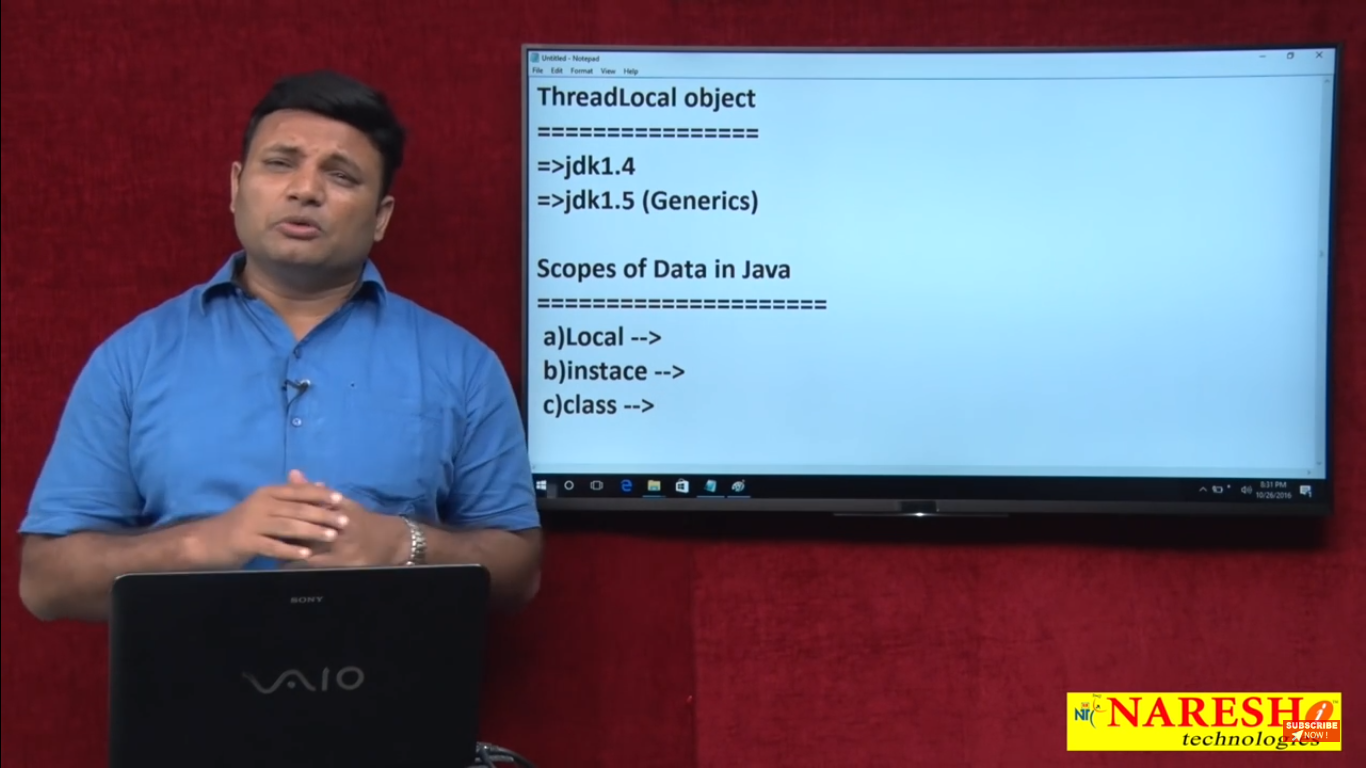
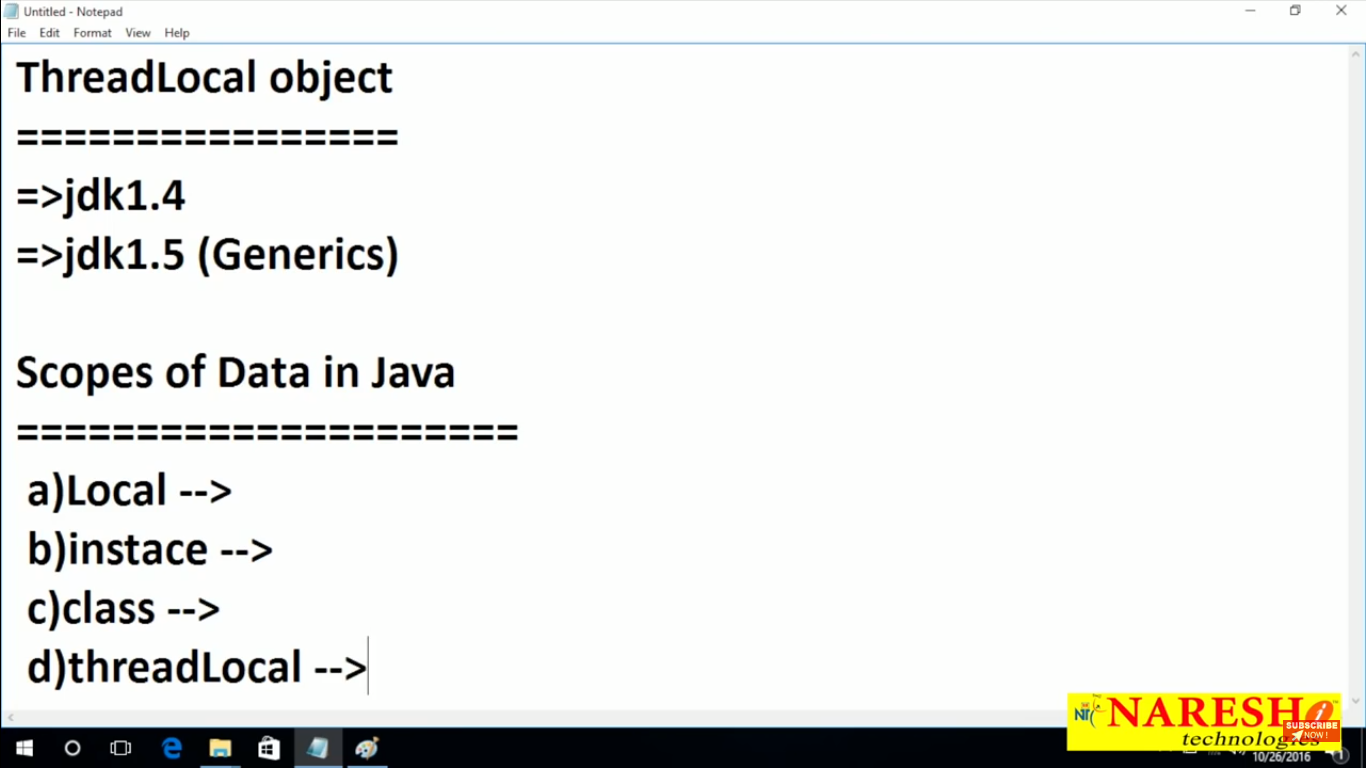
Thread Local https://youtu.be/lFsX-BknckU





Thread Local : It will be local scope for each thread that can be visible local as well as global, Only the owner of that thread can access this data not other thread can access

Thread Local: Specific to each thread

Ex : Suppose your using login application and user name and pass

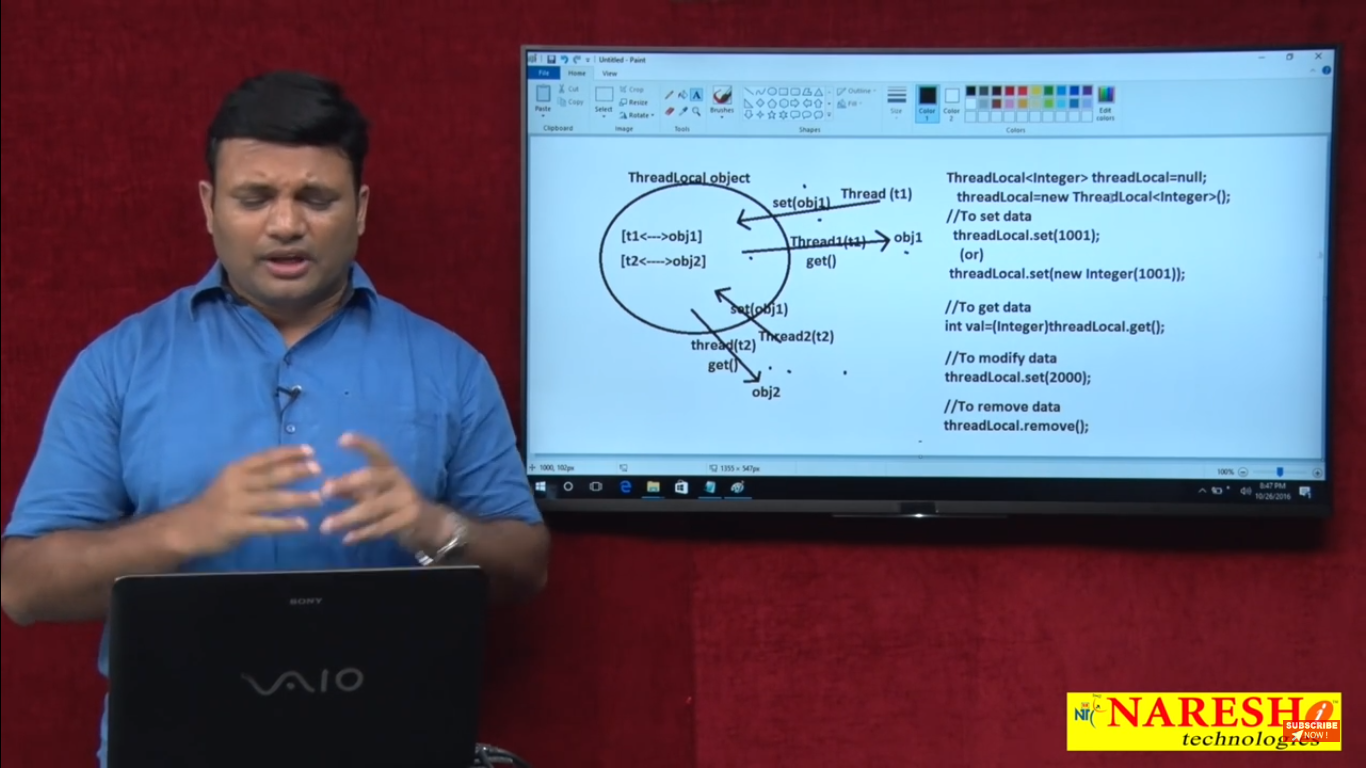
You can save in threadlocal because once user will be login one specific thread will be created and you can use that data for specifically and you can use this data to the particular thread

Once thread job is over data will also destroy .

Here in example each thread have separate thread object

Note : With synchronized you can use this to do thread safety you want any thing to be save place those data to particular to thread.

Example : In real time application each request create new thread then you may think of this implementation



Here it does not matter where you create this thread local object if create as instance, or class level or inside in method

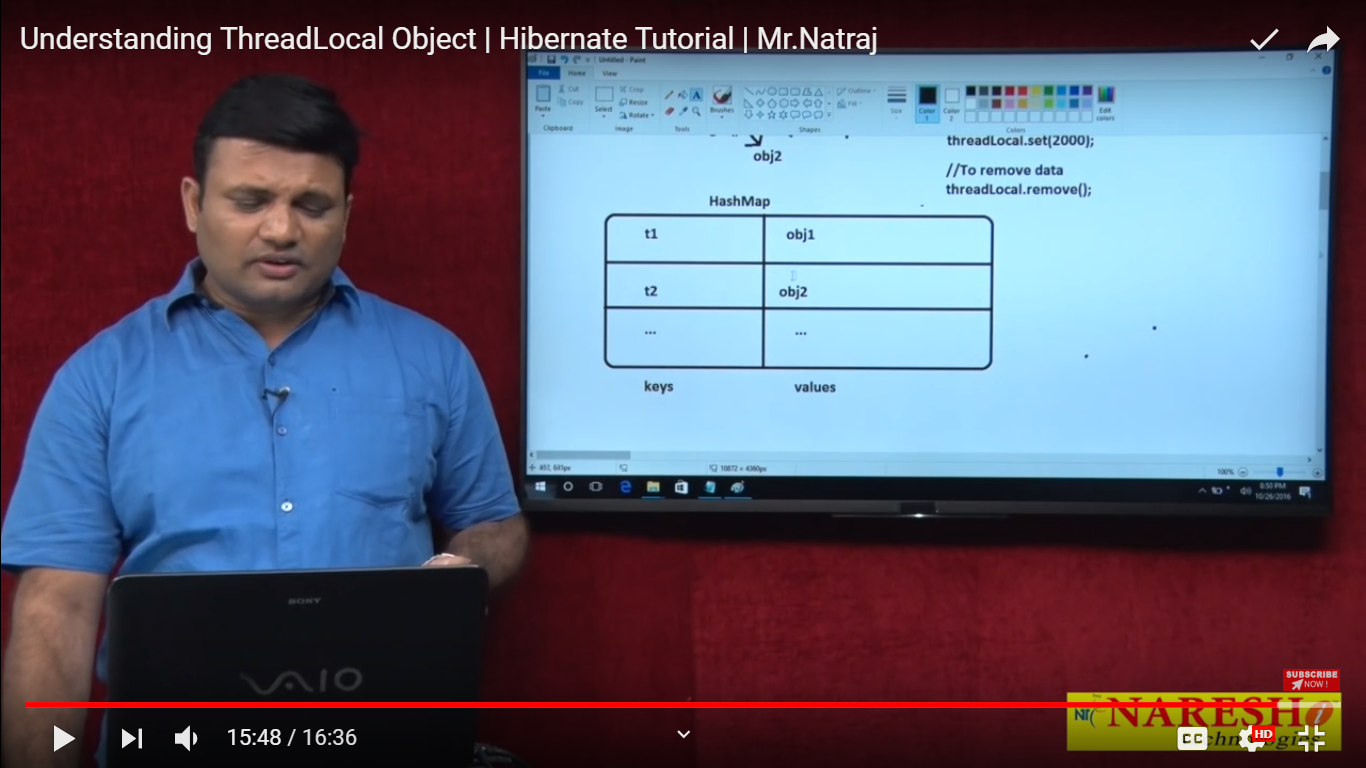
Still that will be specify to thread each thread…….

🡪If you call threadlocal.get() from t1 then only Obj1 will get not Obj2 value you will get.

* You can modify that value to specific to thread

How these value is maintaining ?

These value is mintend in Hastable like key is thread(cuurent thread) and value will be Object so key will unique and value will different if you call current thread example t2 then it will fetch t2 value object fro hasmap



<https://youtu.be/rMRQ8UFa_DQ>

Developing HibernateUtil class :

Why we creating HiberUtil class ?

Only for creating SessionFactory and Sesssion Objects



Its not good practice to write persistence logic in deo layer

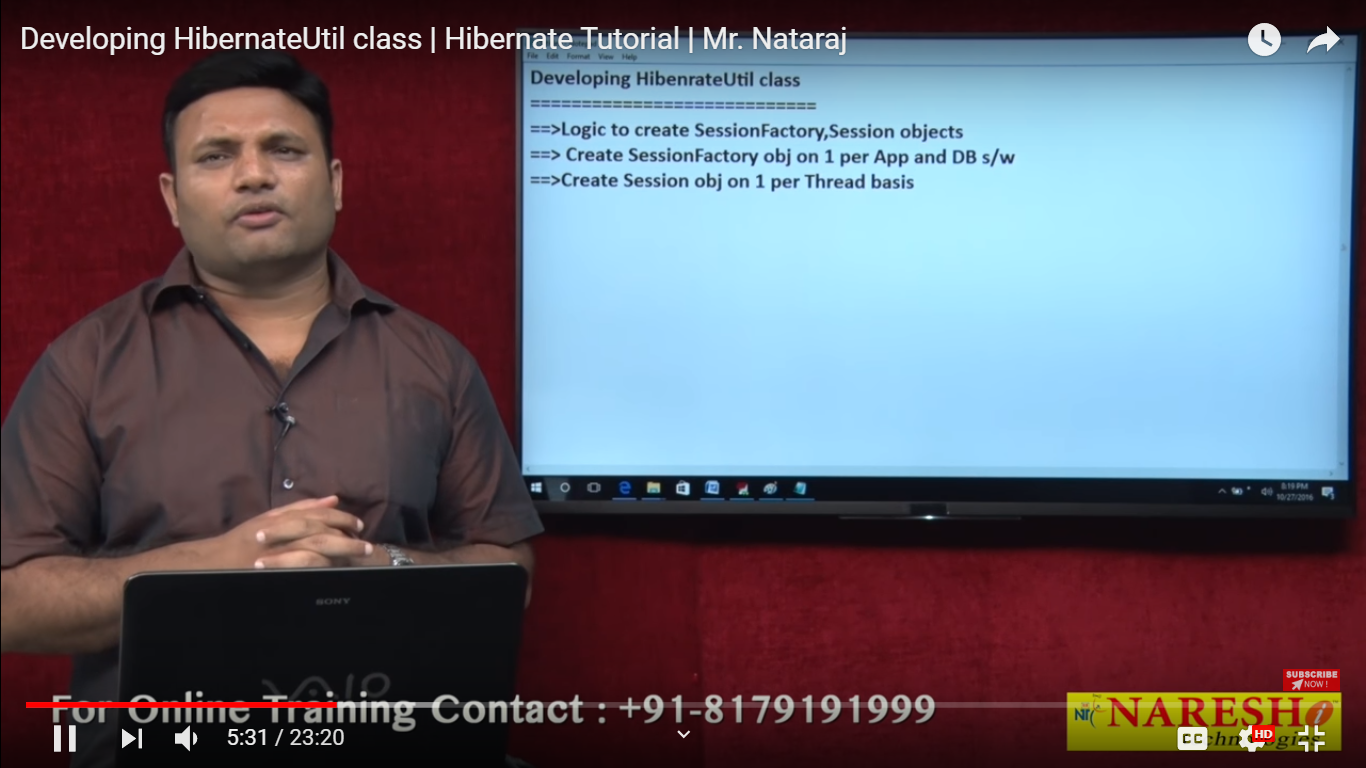
Or any where else who want to talk to database s/w

Note : if you want to write persistence logic keepit at util class so other more than one deo class or other application may use this util class to talk to database software, Use deo for only implementation of persitance logic.

Note: Just we are keeping things at one place and that we can write closing session and cleanup code at one place we will not forgot and we are not spreading code over the project

**Session factory**- one per application and one perdatabase

**Session :** one per Thread suppose we have 10 thread then 10 thread should create here Threadlocal is ultimately best on work on perThread based

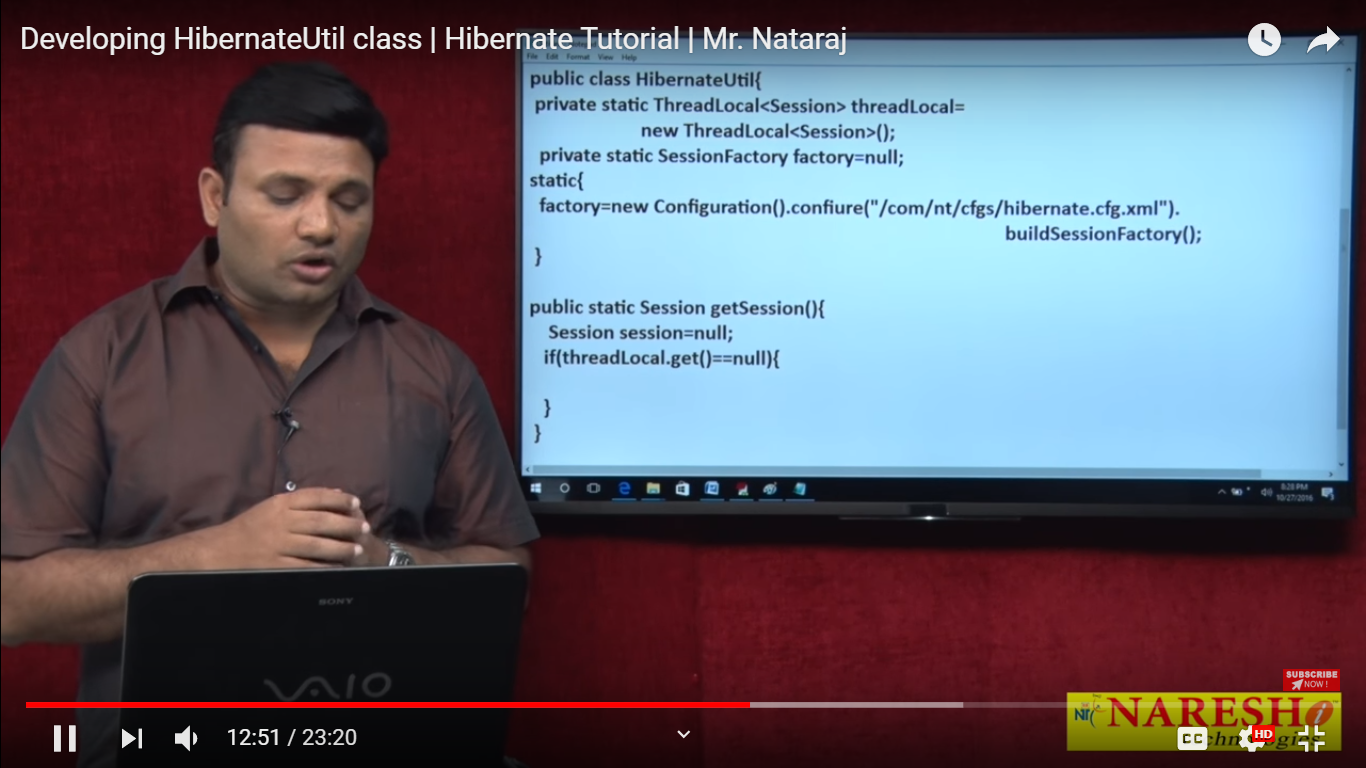




We are using lots of static because static is free from instance and I want load at time of application is loaded

So I used here static black

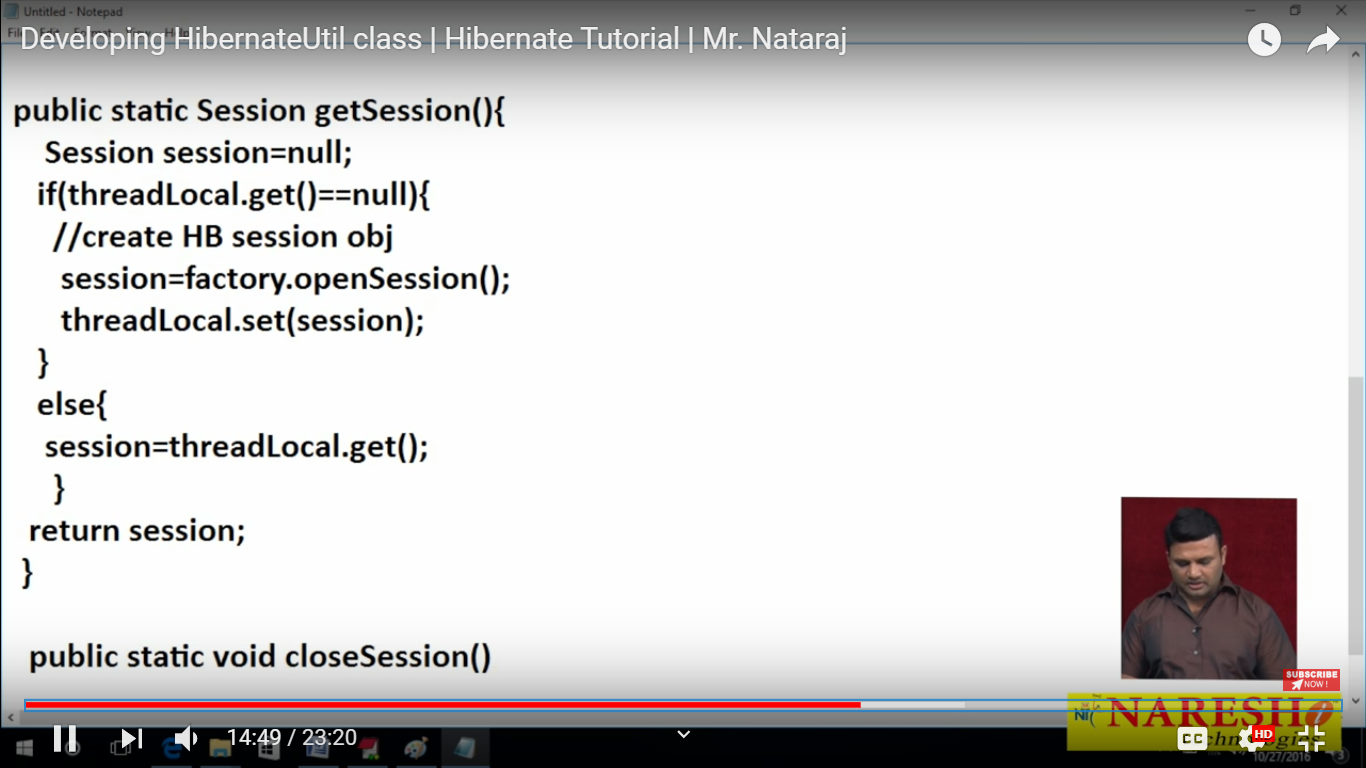
Note : Threadlocal<Session> only allowed value is session



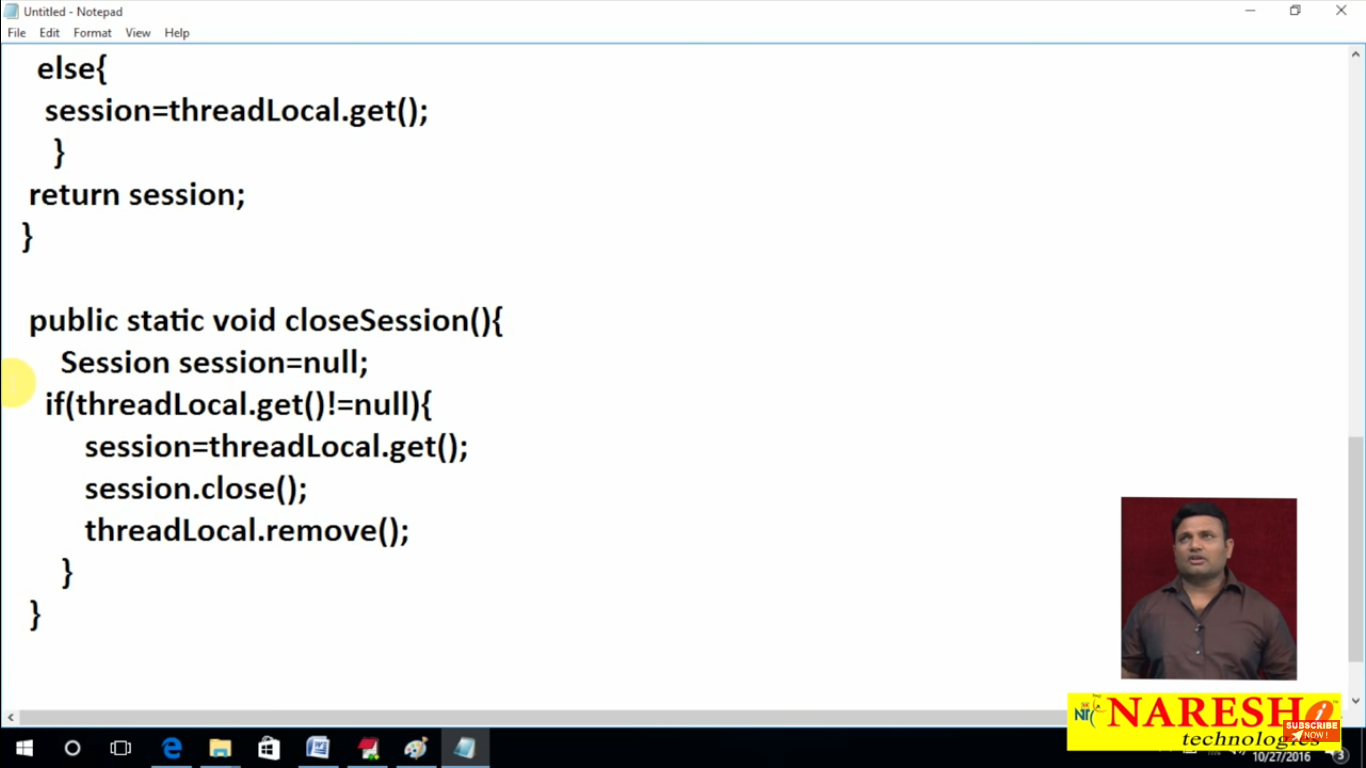
1. Static block which will load sessionfactory at the time of application load and readconfigration from the class path
2. getSession() we will keep per session per thread base

so first we will check whether session is in ThradLoacl or not if not then create and place in the Threadlocal

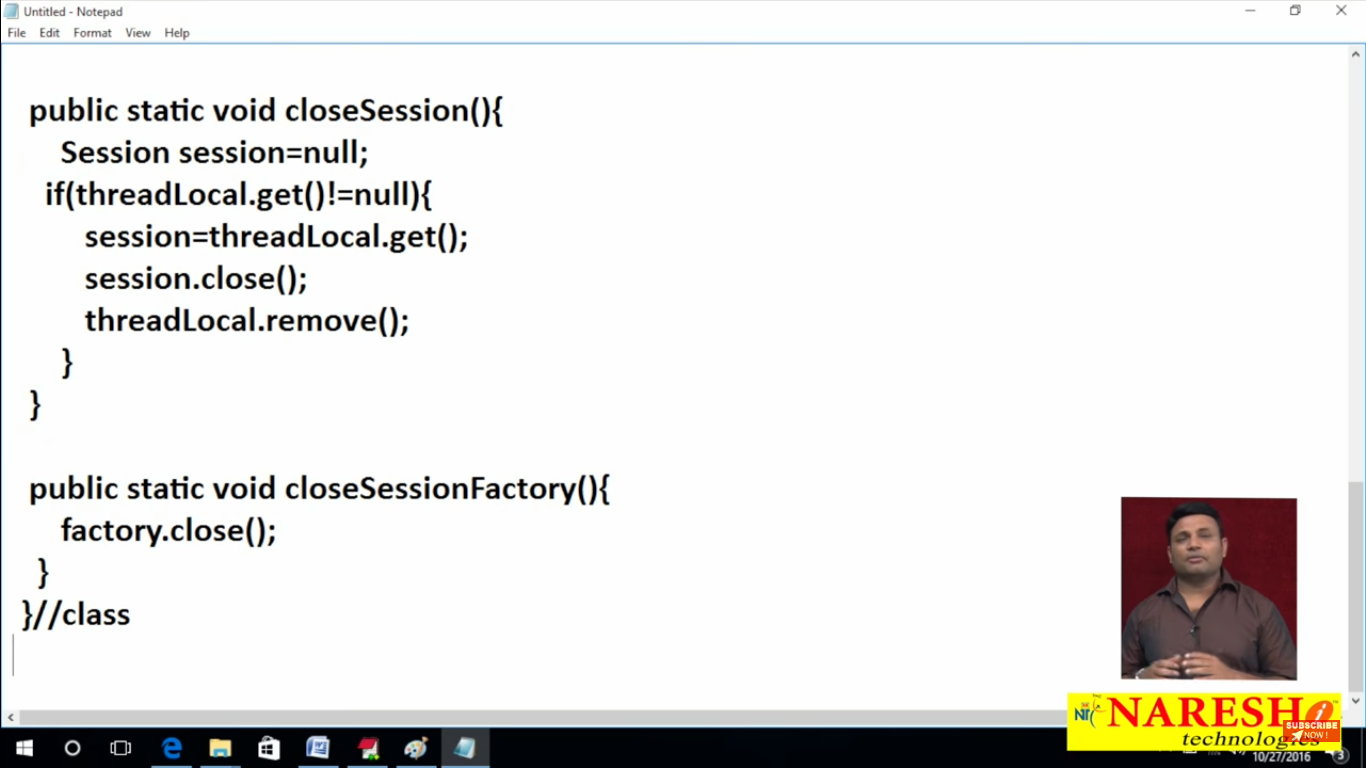
* if its present then return the session.

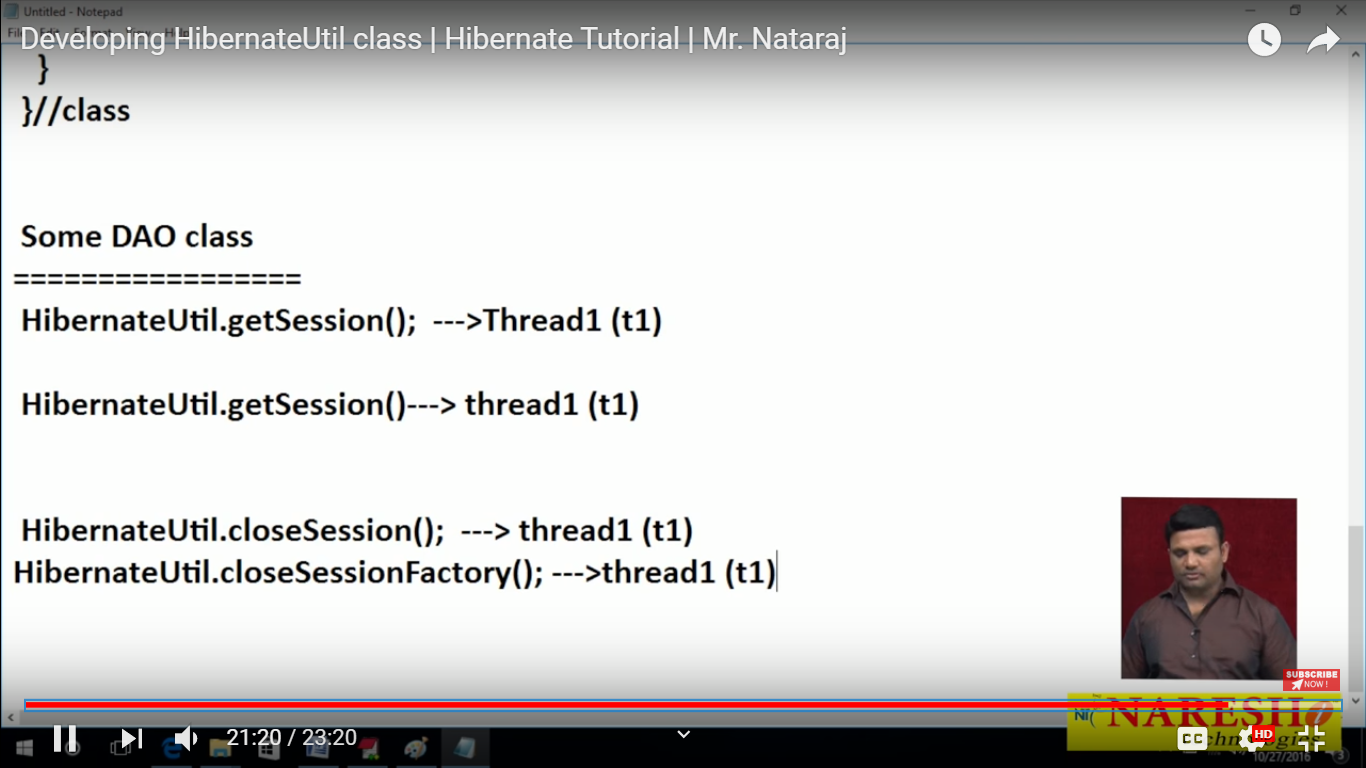


Here we are going to write cleanup code:



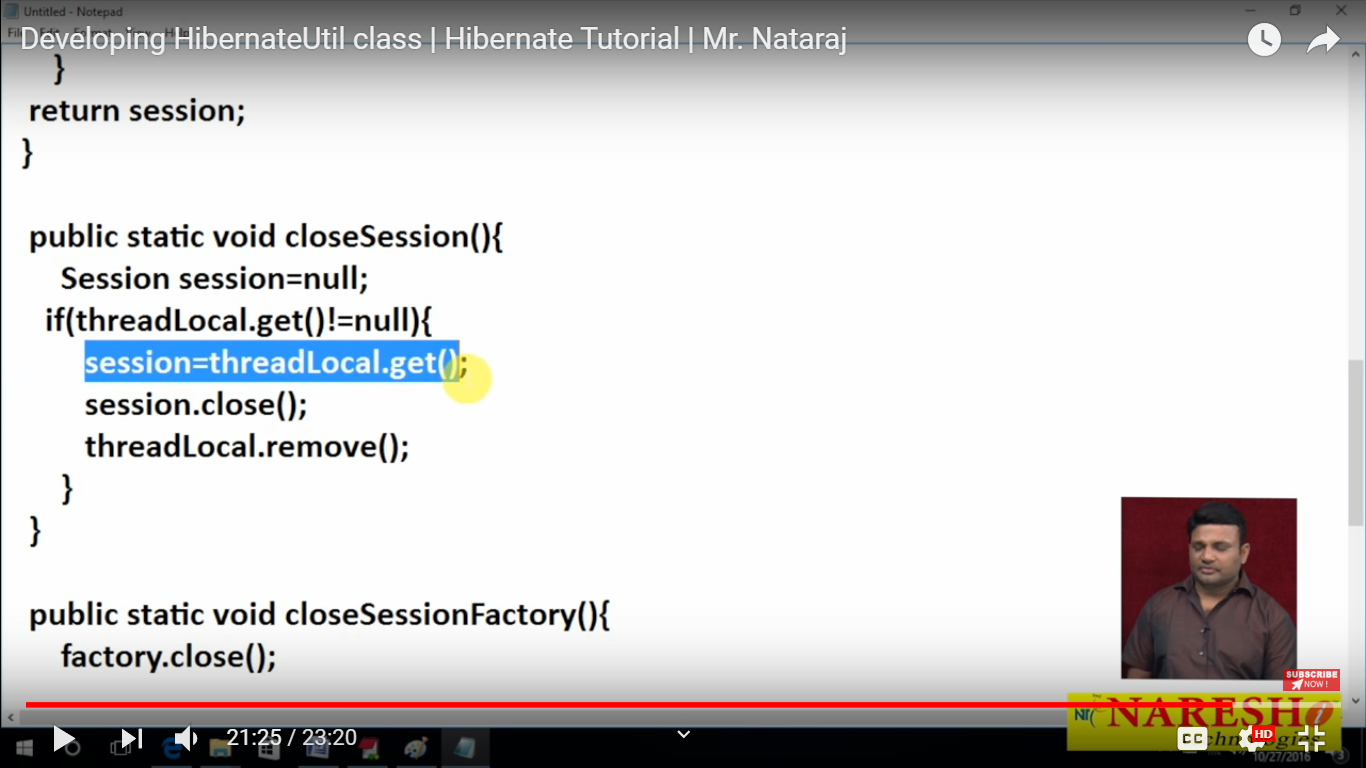
Here we are checking whether current seesion available in threadlocal or not if its there then only get the session and closes the session and remove from threadlocal



**UseCases :** 

If you call getsession at first time it will go and check whether session is ThreadLoacl or not if its not there it will create and place to ThreadLoacl -- and second time if we will call that method it will go check wheter in localthread or not it will find its there then it will go to else part and return the session from ThreadLocal for the partuculalar thread.

This pics of code here



For closing session :

First it will check whether particular thread t1 session is present in threadlocal object or not if present then it will close this and remove from from threadlocal

Note : These class member is static so all load at the time of class loading and identify from top to bottom this rule will applicable here