

"Concept & Coding" YT Video Notes

- ThreadLocal class provide access to Thread-Local variables.
- This 'Thread-Local' variable hold the value for particular thread.
- Means each Thread has its own copy of Thread-Local variable.
- We need only 1 object of ThreadLocal class and each thread can use it to set and get its own Thread-variable variable.

```
public static void main(String args[]) {  
  
    ThreadLocal<String> threadLocalObj = new ThreadLocal<>();  
  
    //main thread  
    threadLocalObj.set(Thread.currentThread().getName());  
  
    Thread thread1 = new Thread() -> {  
        threadLocalObj.set(Thread.currentThread().getName());  
        System.out.println("Task1");  
    };  
  
    thread1.start();  
  
    try{  
        Thread.sleep( mills: 2000);  
    }catch (Exception e){  
  
    }  
  
    //here we have main thread  
    System.out.println("Main thread: " + threadLocalObj.get());  
}
```


Output:

```
public static void main(String args[]) {  
  
    ThreadLocal<String> threadLocalObj = new ThreadLocal<>();  
  
    ExecutorService poolObj = Executors.newFixedThreadPool( nThreads: 5 );  
  
    poolObj.submit(() -> {  
        threadLocalObj.set(Thread.currentThread().getName());  
    });  
  
    for(int i=1; i<15; i++){  
        poolObj.submit(() -> {  
            System.out.println(threadLocalObj.get());  
        });  
    }  
}
```

```

null
pool-1-thread-1
null
pool-1-thread-1
null
null
null
null
null
pool-1-thread-1
null
null
null

```

```
public static void main(String args[]) {

    ThreadLocal<String> threadLocalObj = new ThreadLocal<>();

    ExecutorService poolObj = Executors.newFixedThreadPool( nThreads: 5);

    poolObj.submit(() -> {
        threadLocalObj.set(Thread.currentThread().getName());
        //my work completed, now clean up
        threadLocalObj.remove();
    });

    for(int i=1; i<15; i++){
        poolObj.submit(() -> {
            System.out.println(threadLocalObj.get());
        });
    }

}
```

Output:

[illegible]

Virtual Thread vs Platform Thread

Moto of Virtual Thread: To get Higher throughput not Latency

```
Thread th1 = Thread.ofVirtual().start(RunnableTask);  
Or  
ExecutorService myExecutorObj = Executors.newVirtualThreadPerTaskExecutor();  
myExecutorObj.submit(RunnableTask)
```

