Volatile → keyword

ex:

```
class processor extends Thread
{               volatile
  private boolean running = true;
  public void run(){
      while(running){
        sysout("Hello");

        try{
          Thread.sleep(100);
        }catch(InterruptedException e)
        {e.printStackTrace();
        }

    }

public void shutdown(){
```

```
            running = false ;
        }
    }


        class App {


            public static void main(String[] args)
            {
            processor proc1 = new processor();
            proc1. start() ;

            Sysout ("press Enter to Stop..);
           Scanner scan = new Scanner(Sys.in);
            sc. nextLine();

            proc1. shutDown();
    }
}
```

Explanation:

Total 2 threads running

✓ main-thread

✓ processor-thread

w/o - Volatile Keyword ⇒

To change/modify value in processor thread, from main method thread it may not work in all the systems. Also its not a good practice.

[illegible]

eg:- running - keyword in above program.

With-Volatile Keyword :⟹

If one update "running" value. it will work in all systems & also its a good practice.

2/ **Synchronized Keyword :-**

Every object in java has intrinsic lock
(or)
Monitor lock
(or)
Mutex.

✓ If we call Synchronized method of an Object here, we have to acquire the intrinsic lock before the call.

✓ if Thread-1 acquires the intrinsic lock & runs this method, and if another thread Thread-2 call same method at same time. Thread 2 should wait untill first thread execution completes.

✓ intrinsic locks can acquire by only one thread at a time.

Ex:-

```
public class App {

    private int count =0;
```

```
public synchronized void increment()
 {   Count ++;

 }

public static void main(String[] args)
{

 App   app  =  new  App();
 app.doWork();

}


public void doWork() {


Thread  t1=new Thread(new Runnable()
    {
       public void run() {
```

```
                    for(int i=0;i<10000;i++){
                    increment();
                  }
               }
           };


        Thread t2=new Thread(new Runnable()
         {
              public void run(){


                for(int i=0;i<10000;i++){
                increment++;
                }
         }
        });

        t1.start()
```

```
t2.start()

try {
    t1.join();
    t2.join();
} catch ( )
{
}

Sysout ("Count = " );
```

Explanation:

w/o Synchnized,
Count value changes.

with synchized,

[illegible]

Count value 80,000.

only one thread can access the method:

=★=