QA is disabled(**Security reason)

Tommor topic : Collection continuation including all legacy classes.

sir can you please explain var ARGS one more time
varargs -> Introduced in jdk1.5V
Syntax: datatype ...variable
Note :internally the var arg variable will be used as an array only.

```
class Demo{
	public void add(int... x){//internally will be treated like int a[]
		//it will print the value of the arguments collected
		for(int data: x)
			System.out.println(data);
	}
}
class Test{
	public static void main(String[] args){
			Demo d = new Demo();
			d.add(10,20);
			d.add(10,20,30);
			d.add(10,20,30,40);
	}
}
```

Producer Consumer Problem

```
package in.ineuron.producerconsumer.main;

//producer thread operations
class Producer extends Thread {

	// Producer producing the data in StringBuffer
	StringBuffer sb;

	public Producer() {
		// StringBuffer object is created with a default capacity 16
		sb = new StringBuffer();
	}

	@Override
	public void run() {

		synchronized (sb) {

			for (int i = 1; i <= 10; i++) {
				try {
					sb.append(i + ": ");
					Thread.sleep(100);
					System.out.println("appending");
				} catch (InterruptedException e) {
					e.printStackTrace();
				}

			}
			// send the notification to the waiting thread
			sb.notify();
		}

	}
```

```
}

//Consumer thread operations
class Consumer extends Thread {

	// Creating producer object to get the produced data from StringBuffer
	Producer producer;

	// injecting the Producer Object into Consumer
	public Consumer(Producer producer) {
		this.producer = producer;
	}

	@Override
	public void run() {

		synchronized (producer.sb) {

			try {

				//wait till the notification is sent by producer
				producer.sb.wait();

				// consume the data produced by the producer
				System.out.println(producer.sb);

			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
	}
}

//Effecient way of interthread communication  using wait() and notify()
public class BetterCommunication {

	//Driving code where he start the other thread
	public static void main(String[] args) {

		Producer obj1 = new Producer();
		Consumer obj2 = new Consumer(obj1);

		Thread t1 = new Thread(obj1);// producer thread
		Thread t2 = new Thread(obj2);// consumer thread

		t2.start();// consumer should wait
		t1.start();// producer should start
	}
}


2 Threads wants to share the resource such that data inconsistency should not
happen
			"synchronized"


1st Thread ----> updating an object with some value
```

2nd Thread ----> get the value of the Object (wait)

wait() ----> lock is released from the Object and given to the Thread who wants to update the
value for the Object.

notify() ---> The thread now got the lock ,using the lock update the value for the Object,notify
to the waiting thread that hey u gave the lock i used and i have update plz take the
updated value and use it.

object level -> method should be just synchronized
class level -> method should be static synchornized

native -> code is not from java language, code is from other community languages like
c, c++, python

```
static synchronized {

}

synchronized
{

}
```