SYNCHRONIZATION

Synchronization

- Synchronized is the keyword applicable for methods and blocks but not for classes and variables.
- If a method or block declared as the synchronized then at a time only one Thread is allow to execute that method or block on the given object.

Synchronization

- If multiple thread are trying to operate simultaneously on the same java object then their may be chance of data inconsistency problem.
- To overcome this problem we should go for synchronized.

Synchronization

- The main advantage of synchronized keyword is we can resolve data inconsistency problems
- But the main disadvantage of synchronized keyword is it increases waiting time of the Thread and effects performance of the system.
- Hence if there is no specific requirement then never recommended to use synchronized keyword.

Synchronization

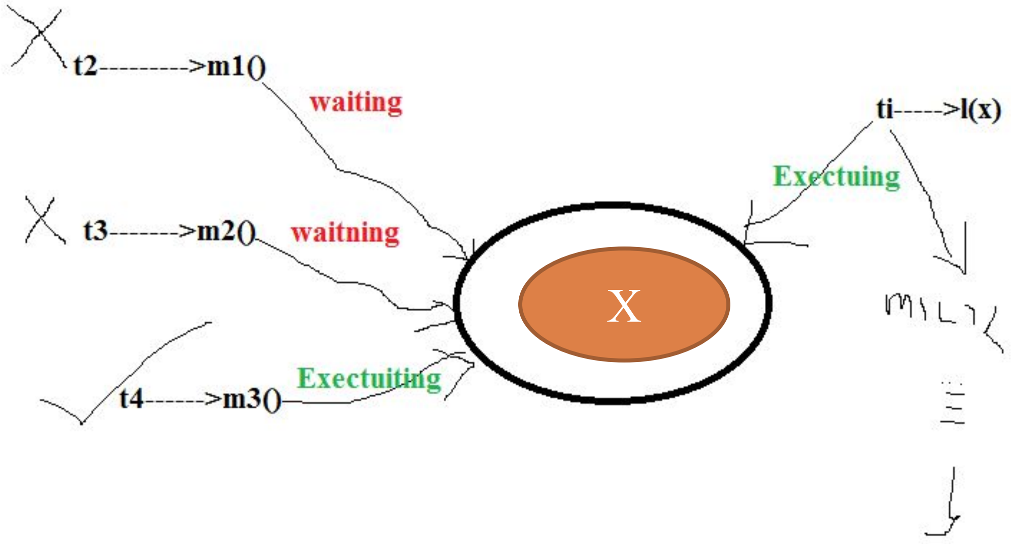
- Internally synchronization concept is implemented by using lock concept.
- Every object in java has a unique lock. Whenever we are using synchronized keyword then only lock concept will come into the picture.

Synchronization

- If a Thread wants to execute any synchronized method on the given object 1st it has to get the lock of that object. Once a Thread got the lock of that object then it's allow to execute any synchronized method on that object. If the synchronized method execution completes then automatically Thread releases lock.
- Acquiring and Releasing lock internally takes care by jvm and program not responsible.

Example

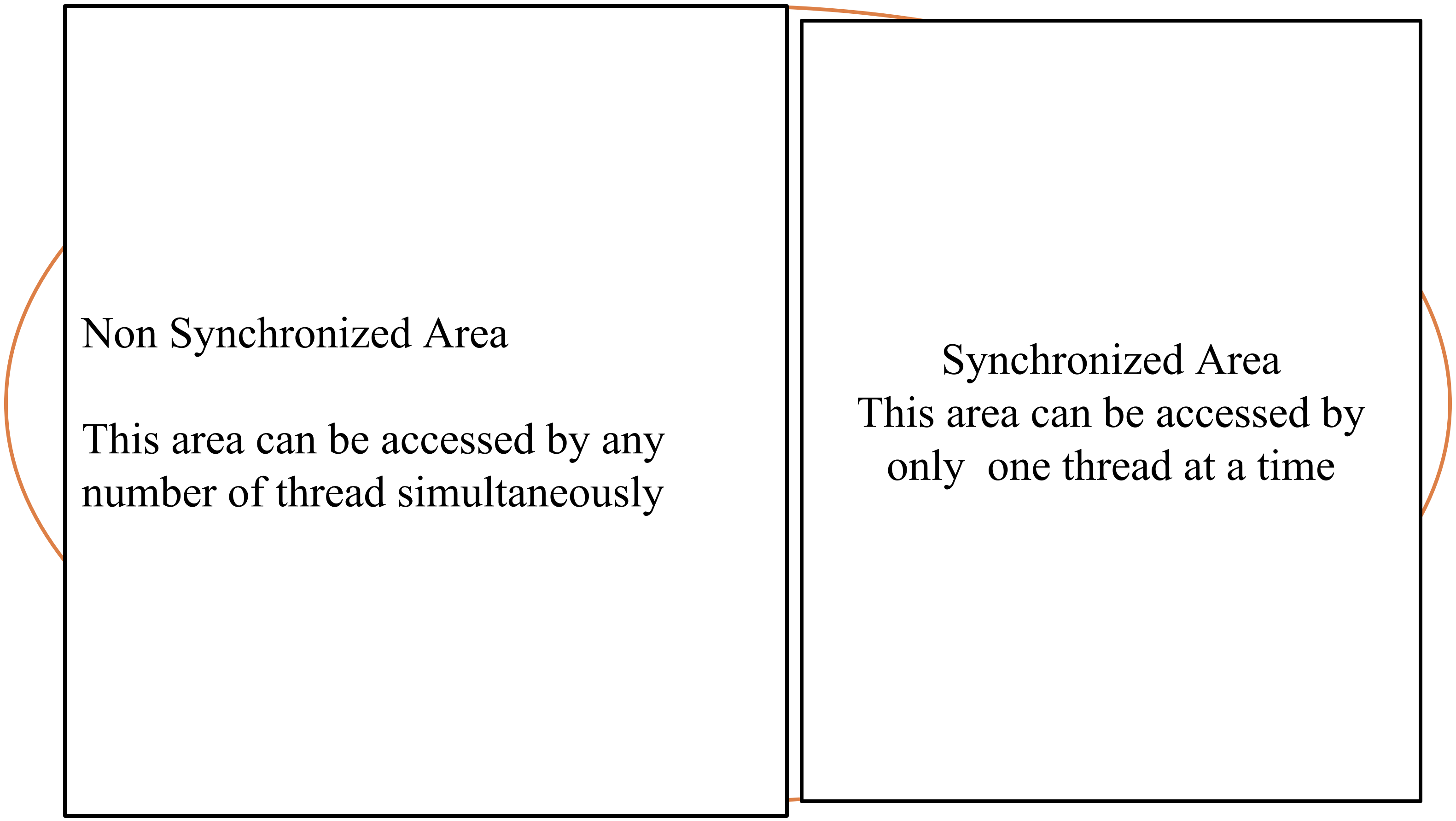
```
class X{
syncronized m1(){
}
syncronized m2(){
}
m3(){
}
}
```



Synchronization

- While a Thread executing any synchronized method the remaining Threads are not allowed execute any synchronized method on that object simultaneously. But remaining Threads are allowed to execute any non-synchronized method simultaneously. [lock concept is implemented based on object but not based on method].

Object have 2 State

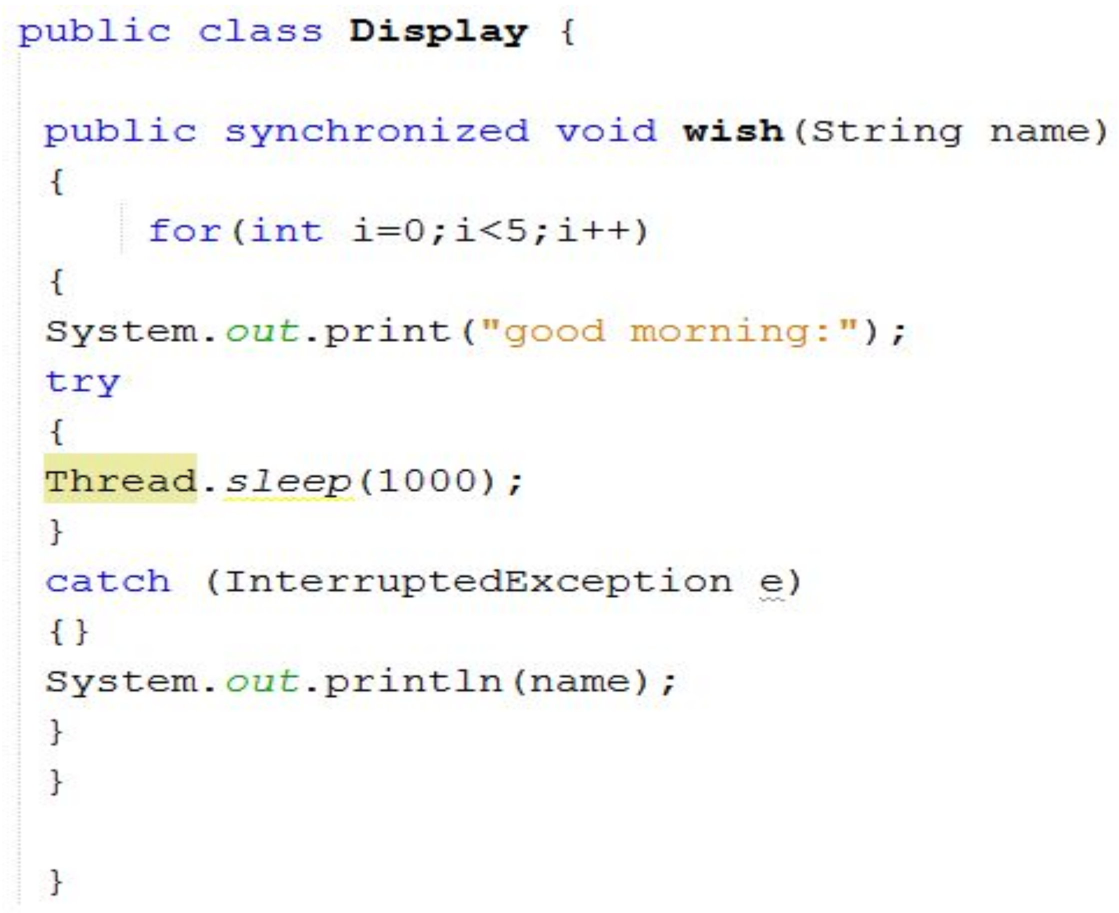


Non Synchronized Area

This area can be accessed by any number of thread simultaneously

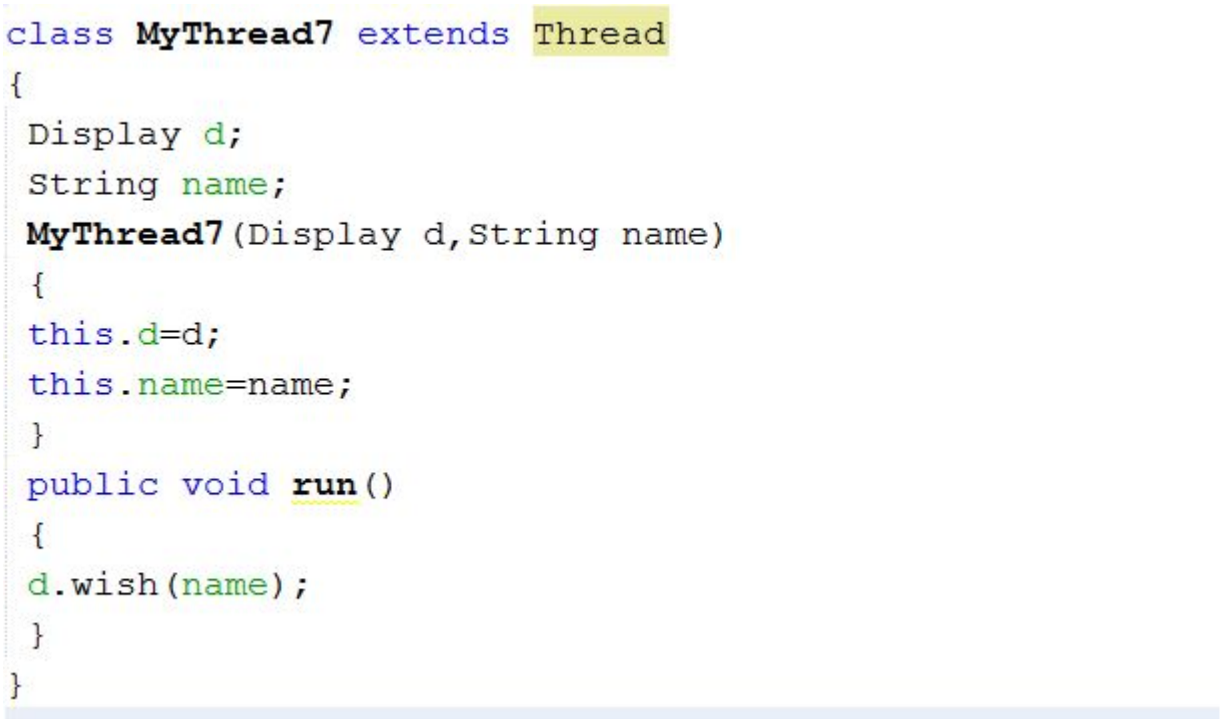
Synchronized Area
This area can be accessed by only one thread at a time

Display.java



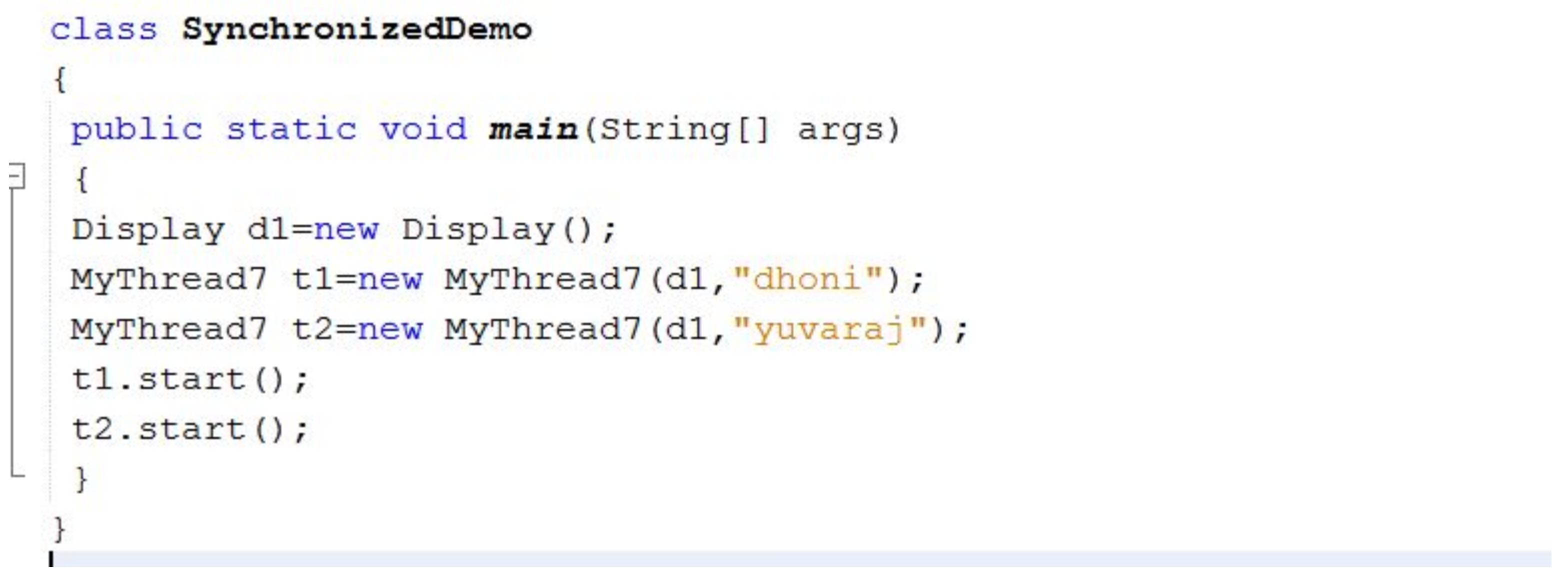
```
public class Display {

 public synchronized void wish(String name)
 {
     for(int i=0;i<5;i++)
 {
 System.out.print("good morning:");
 try
 {
 Thread.sleep(1000);
 }
 catch (InterruptedException e)
 {}
 System.out.println(name);
 }
 }


 }
```

MyThread7.java



```
class MyThread7 extends Thread
{
 Display d;
 String name;
 MyThread7(Display d,String name)
 {
 this.d=d;
 this.name=name;
 }
 public void run()
 {
 d.wish(name);
 }
}
```

Synchronized Demo



```
class SynchronizedDemo
{
 public static void main(String[] args)
 {
 Display d1=new Display();
 MyThread7 t1=new MyThread7(d1,"dhoni");
 MyThread7 t2=new MyThread7(d1,"yuvaraj");
 t1.start();
 t2.start();
 }
}
```

Conclusion

- If we are not declaring wish() method as synchronized then both Threads will be executed simultaneously and we will get irregular output.
- If we declare wish()method as synchronized then the Threads will be executed one by one that is until completing the 1st Thread the 2nd Thread will wait in this case we will get regular output which is nothing but

Synchronization

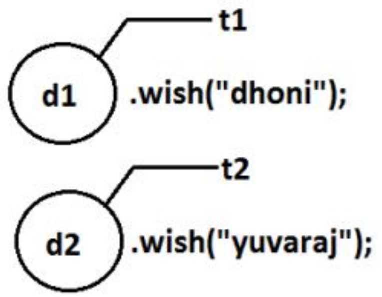
Case study:

Case 1:

```
Display d1=new Display();
Display d2=new Display();
MyThread t1=new MyThread(d1,"dhoni");
MyThread t2=new MyThread(d2,"yuvaraj");
t1.start();
t2.start();
```

conclusion

Diagram:



- Even though we declared wish() method as synchronized but we will get irregular output in this case, because both Threads are operating on different objects.

Class level lock:

- Every class in java has a unique lock. If a Thread wants to execute a static synchronized method then it required class level lock.
- Once a Thread got class level lock then it is allow to execute any static synchronized method of that class.

Which Method is synchronized or not

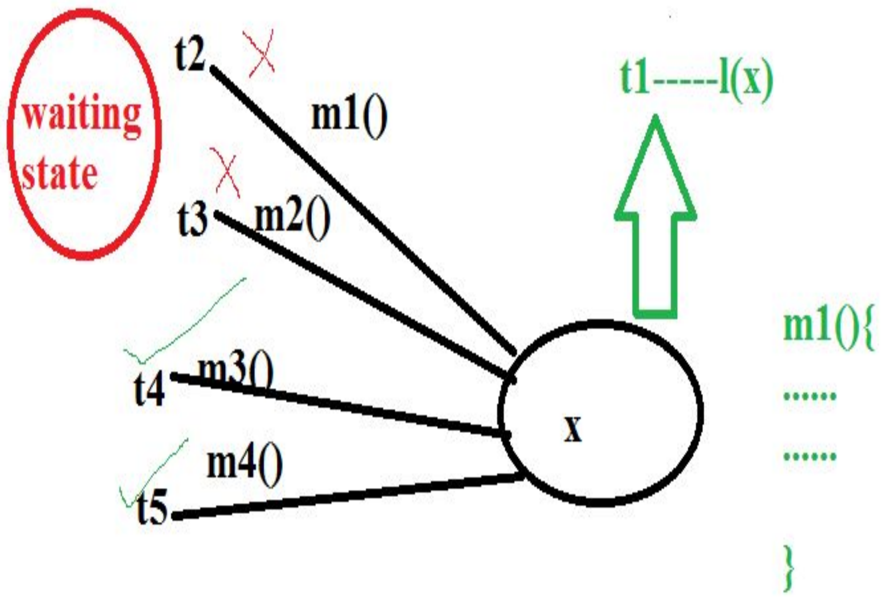
```
class X{
Synchronization method(){
when we performing update operation
(add/remove/update)
where state of changing
}

non-Synchronization method(){
wherever object state won't be change
like read Operation
}
}
```

Example

```
class Reservation{
non synchronized checkAvialability(){
//Just Read Operation
}

synchronized book_ticket(){
//update
}
}
```

Example

```
class X{

public static sync void  m1()
public static sync void  m2()

public static void   m3()

public void m4()


}
```



Class level lock

- While a Thread executing any static synchronized method the remaining Threads are not allow to execute any static synchronized method of that class simultaneously.
- But remaining Threads are allowed to execute normal static methods, and normal instance methods simultaneously.