Dt : 31/10/2023

Limitation of Object Locking process(Synchronized block):

=>In Object Locking process,all instance methods within the Object are under the lock and all the methods are available to one user at-a-time, which is not preferable in all situations.

==========

(b)synchronized method - Instance method locking process

=>The process of declaring instance method with synchronized keyword is known as synchronized method.

=>In synchronized method process,only one method of Object will be under the lock and the method is available to one user at-a-time.

structure synchronized method:

```
synchronized return_type method_name(para_list)
{
 //method_body
}
```

Ex-program:

p1 : Available.java

```
package p1;
public class Available {
   public static int available=1;
```

Venkatesh Maipathii

```
}
```

p1 : Selection.java

```
package p1;
import java.util.*;
public class Selection
{
    public synchronized void select(int n,String name)
    {
    if(n<=Available.available) {
         System.out.println(n+" Tickets booked for
"+name+" Time: "+new Date());
         try {
              Thread.sleep(2000);
         }catch(Exception e) {e.printStackTrace();}
         Available.available=Available.available-n;

    }else {
         System.out.println("Tickets not available for
"+name);
    }
    }
}
```

p1 : User1.java

```
package p1;
public class User1 implements Runnable
{
   public Selection ob=null;
   public User1(Selection ob)
   {
        this.ob=ob;
   }
     @Override
   public void run()
   {
      ob.select(1, "Alex");
```

Venkatesh Maipathi

```
    }
}
```

p1 : User2.java

```
package p1;
public class User2 implements Runnable
{
        public Selection ob=null;
        public User2(Selection ob)
        {
             this.ob=ob;
        }
          @Override
        public void run()
        {
            ob.select(1, "Ram");
        }
}
```

p2 : DemoThread6.java(MainClass)

```
package p2;
import p1.*;
public class DemoThread6
{
     public static void main(String[] args)
     {
         Selection ob = new Selection();
        User1 ob1 = new User1(ob);//Con_call
        User2 ob2 = new User2(ob);//Con_call

        Thread t1 = new Thread(ob1);
        Thread t2 = new Thread(ob2);

        t1.start();
        t2.start();
     }
}
```

Venkatesh Maipathii

==========

(c)static synchronization - Class Locking process

=>The process of declaring static method with synchronized keyword is known as static synchronization.

=>In Static synchronization the lock is applied on the class and which is known as Class Locking process

=>In Static synchronization all the static members of class are available to one user at-a-time

structure static synchronized method:

```
synchronized static return_type method_name(para_list)
{
 //method_body
}
```

Ex-program:

p1 : Available.java

```
package p1;
public class Available {
    public static int available=1;
}
```

Venkatesh Maipathi

p1 : Selection.java

```
package p1;
import java.util.*;
public class Selection
{
    public synchronized static void select(int n,String
name)
    {
    if(n<=Available.available) {
        System.out.println(n+" Tickets booked for
"+name+" Time: "+new Date());
        try {
            Thread.sleep(2000);
        }catch(Exception e) {e.printStackTrace();}
        Available.available=Available.available-n;

    }else {
        System.out.println("Tickets not available for
"+name);
    }
    }
}
```

p1 : User1.java

```
package p1;
public class User1 implements Runnable
{
    @Override
   public void run()
   {
       Selection.select(1, "Alex");
   }
}
```

p1 : User2.java

Venkatesh Maipathi

```
package p1;
public class User2 implements Runnable
{
        @Override
        public void run()
        {
            Selection.select(1, "Ram");
        }
}
```

p2 : DemoThread7.java(MainClass)

```
package p2;
import p1.*;
public class DemoThread7
{
     public static void main(String[] args)
     {
         User1 ob1 = new User1();//Con_call
         User2 ob2 = new User2();//Con_call

         Thread t1 = new Thread(ob1);
         Thread t2 = new Thread(ob2);

         t1.start();
         t2.start();
     }
}
```

==========
=

**imp*

2.Thread Communication process:

=>The process of establishing communication b/w threads is known as Thread Communication process,and which is also known as Inter Thread

Venkatesh Maipathii

Communication process(ITC).

=>we use the following methods from java.lang.Object class to perform Thread Communication process:

(a)wait()

(b)notify()

(c)notifyAll()

(a)wait():

=>wait() method will make the thread wait(stop the thread) until it receives msg in the form of notify() or notifyAll()

(b)notify():

=>notify() method will unlock the resource and send the msg to the waiting thread.

(c)notifyAll():

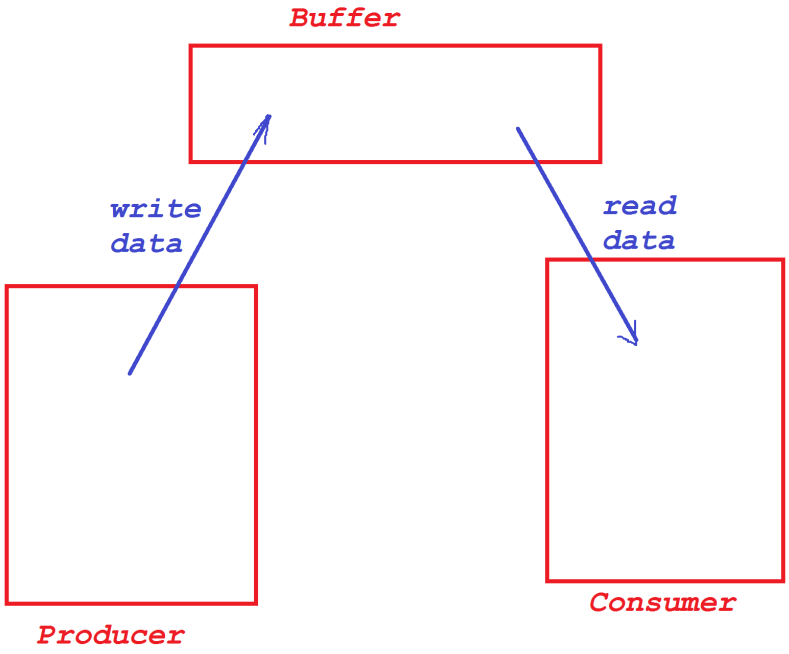
=>notifyAll() method also will unlock the resource and send the msg to multiple waiting threads.

Ex-program:

Demonstarting Thread-Communication-process using Producer-Consumer problem

Venkatesh Maipathii

Layout:



Producer will write the data to buffer
Consumer will read the data from the buffer

> *Consumer must wait until Producer writes the data to Buffer,else leads to DeadLock or generate Wrong results*

p1 : Producer.java

```
package p1;
public class Producer implements Runnable
{
    public StringBuffer sb;
    public Producer()
    {
        sb = new StringBuffer();
```

```
    }
    @Override
    public void run()
    {
       try {
           synchronized(sb) {
                for(int i=1;i<=10;i++){
                    sb.append(i+":");
                    System.out.println("Producer writing
data to buffer...");
                    Thread.sleep(2000);
                }//end of loop
                sb.notify();//send message to waiting
thread
           }//end of lock
       }catch(Exception e) {e.printStackTrace();}
    }
}
```

p1 : Consumer.java

```
package p1;
public class Consumer implements Runnable
{
    public Producer prod;
    public Consumer(Producer prod)
    {
        this.prod=prod;
    }
    @Override
    public void run()
    {
        try {
            synchronized(prod.sb)
            {
                System.out.println("Consumer
started...but blocked on wait...");
                prod.sb.wait();
                System.out.println("====Display using
Consumer===");
```

Venkatesh Maipathi

```
                System.out.println(prod.sb.toString());
            }//end of lock
        }catch(Exception e) {e.printStackTrace();}
    }
}
```

p2 : DemoThread8.java(MainClass)

```
package p2;
import p1.*;
public class DemoThread8 {
    public static void main(String[] args) {
        Producer p = new Producer();
        Consumer c = new Consumer(p);

        Thread t1 = new Thread(p);
        Thread t2 = new Thread(c);

        t2.setPriority(Thread.MAX_PRIORITY);
        t1.setPriority(Thread.MIN_PRIORITY);

        t2.start();
        t1.start();
    }
}
```

o/p:

Consumer started...but blocked on wait...

Producer writing data to buffer...

Producer writing data to buffer...

Producer writing data to buffer...

Producer writing data to buffer...

Producer writing data to buffer...

Venkatesh Maipathii

Producer writing data to buffer...

Producer writing data to buffer...

Producer writing data to buffer...

Producer writing data to buffer...

Producer writing data to buffer...

====Display using Consumer===

1:2:3:4:5:6:7:8:9:10:

==========

Venkatesh Maipathii