Dt : 21/11/2022

***imp**

Thread Synchronization:

=>The process of odering the threads for execution is known as Thread Synchronization

=>Thread synchronization process can be performed in two ways:

1.Mutual Exclusion process

2.Thread Communication process

1.Mutual Exclusion process:

=>The process of locking the programming resources and ordering the threads for execution is known as Mutual Exclusion process.

(Programming Resources : Class,Object,Method)

=>This Mutual Exclusion process can be performed in three ways:

(a)synchronized block - Object Locking process

(b)synchronized method - Instance method Locking process

(c)static synchronization - Class Locking process

(a)synchronized block:

=>The process of declaring some statements using "synchronized" keyword is known as synchronized block.

=>we use synchronized block to lock the objects.

syntax:

synchronized(object_ref)

{

//statements

}

Ex-program :

Printer.java

```
package test;
public class Printer {
    public void print(int n,String uname) {
     for(int i=1;i<=n;i++) {
          System.out.println("Print out for User : "+uname);
          try {
               Thread.sleep(2000);
          }catch(Exception e) {e.printStackTrace();}
     }
    }
}
```

UserOne.java

```
package test;
public class UserOne implements Runnable{
     public Printer p=null;
     public UserOne(Printer p) {
          this.p=p;
     }
     @Override
     public void run() {
          synchronized(p)
          {
     p.print(5, "RAM");
          }
     }
}
```

UserTwo.java

```java
package test;
public class UserTwo implements Runnable{
     public Printer p=null;
     public UserTwo(Printer p) {
          this.p=p;
     }
     @Override
    public void run() {
          synchronized(p)
          {
          p.print(5, "RAJ");
          }
    }
}
```

DemoThread3.java(MainClass)

```java
package maccess;
import test.*;
public class DemoThread3 {
     public static void main(String[] args) {
        Printer p = new Printer();

        UserOne ob1 = new UserOne(p);
        UserTwo ob2 = new UserTwo(p);

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

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

o/p:

```
Print out for User : RAM

Print out for User : RAM

Print out for User : RAM

Print out for User : RAM

Print out for User : RAM
```

Print out for User : RAJ

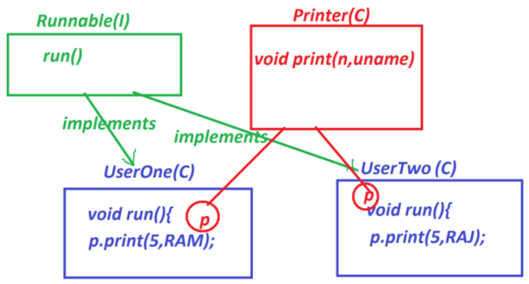
Print out for User : RAJ

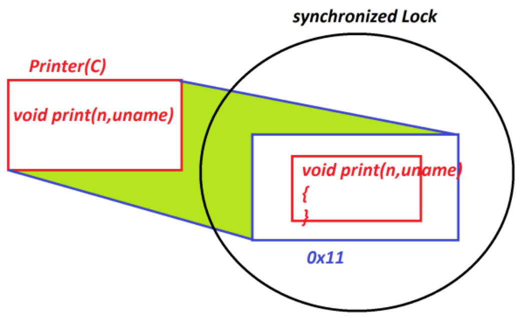
Print out for User : RAJ

Print out for User : RAJ

Print out for User : RAJ

Diagram:





==

Limitation of Object Locking process:

=>In Object Locking process the total instance members available within the Object,will be under the lock.

==

(b)synchronized method:

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

=>In this process,the Instance method will be under the lock and the method can be used by one user at-a-time

syntax:

synchronized return_type method_name(para_list)

{

//method_body

}

Ex-program:

Printer.java

```
package test;
public class Printer {
    public synchronized void print(int n,String uname) {
     for(int i=1;i<=n;i++) {
          System.out.println("Print out for User : "+uname);
          try {
               Thread.sleep(2000);
          }catch(Exception e) {e.printStackTrace();}
     }
    }
}
```

UserOne.java

```
package test;
public class UserOne implements Runnable{
     public Printer p=null;
     public UserOne(Printer p) {
          this.p=p;
     }
     @Override
     public void run() {
     p.print(5, "RAM");
     }
}
```

UserTwo.java

```
package test;
public class UserTwo implements Runnable{
     public Printer p=null;
     public UserTwo(Printer p) {
           this.p=p;
     }
     @Override
    public void run() {
          p.print(5, "RAJ");
    }
}
```

DemoThread3.java(MainClass)

```
package maccess;
import test.*;
public class DemoThread4 {
     public static void main(String[] args) {
        Printer p = new Printer();

        UserOne ob1 = new UserOne(p);
        UserTwo ob2 = new UserTwo(p);

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

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

o/p:

```
Print out for User : RAM

Print out for User : RAM

Print out for User : RAM

Print out for User : RAM

Print out for User : RAM

Print out for User : RAJ

Print out for User : RAJ

Print out for User : RAJ
```

Print out for User : RAJ

Print out for User : RAJ

==

(c)static synchronization:

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

syntax:

synchronized static return_type method_name(para_list)

{

//method_body

}

=>In static synchronization process the lock is applied on class and all static members of class will be synchronized.(Class Locking process)

Ex:

Printer.java

```
package test;
public class Printer {
    public synchronized static void print(int n,String uname) {
     for(int i=1;i<=n;i++) {
          System.out.println("Print out for User : "+uname);
          try {
               Thread.sleep(2000);
          }catch(Exception e) {e.printStackTrace();}
     }
    }
}
```

UserOne.java

```
package test;
public class UserOne implements Runnable{
     @Override
     public void run() {

     Printer.print(5, "RAM");

  }
}
```

UserTwo.java

```
package test;
public class UserTwo implements Runnable{
     @Override
    public void run() {
          Printer.print(5, "RAJ");
    }
}
```

DemoThread3.java(MainClass)

```
package maccess;
import test.*;
public class DemoThread5 {
     public static void main(String[] args) {


        UserOne ob1 = new UserOne();
        UserTwo ob2 = new UserTwo();

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

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

o/p:

```
Print out for User : RAM

Print out for User : RAM
```

Print out for User : RAM

Print out for User : RAM

Print out for User : RAM

Print out for User : RAJ

Print out for User : RAJ

Print out for User : RAJ

Print out for User : RAJ

Print out for User : RAJ

===

**imp*

2.Thread Communication process:

=>The process of establishing Communication b/w threads using the following methods from java.lang.Object class is known as "Thread Communication process".

(a)wait()

(b)notify()

(c)notifyAll()

(a)wait():

=>wait() method is used to stop the thread execution temporarly until it receives msg in the form of notify() or notifyAll()

Method Signature:

public final void wait() throws java.lang.InterruptedException;

(b)notify():

=>notify() method will execute the locked resource completedly and unlock the resource,and send the msg to the next waiting thread.

Method Signature:

public final native void notify();

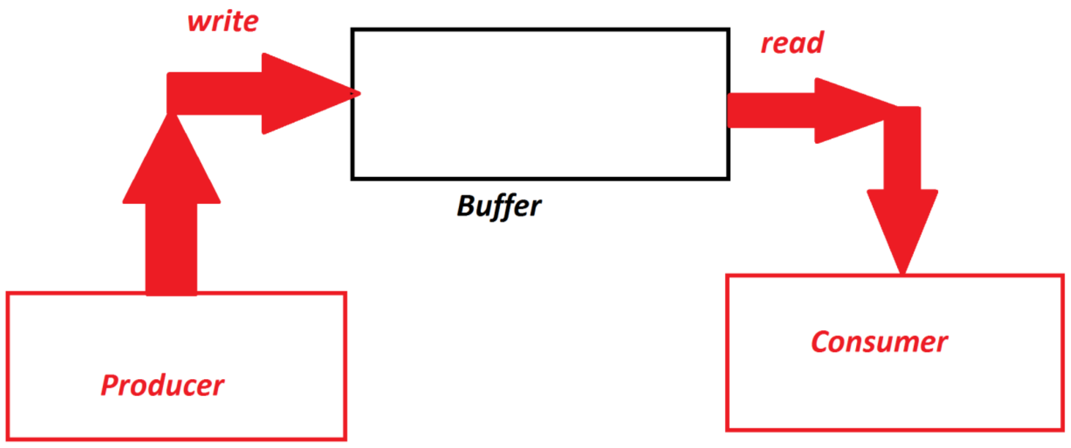
(c)notifyAll():

=>notifyAll() method will execute the locked resource completedly and unlock the resource,and send the msg to the next waiting multiple threads.

Method Signature:

public final native void notifyAll();

Ex:(Program to demonstrate Producer-Consumer problem)



rule : Consumer must wait until Producer writes the data