**Assign same task to multiple threads**

class MyThread extends Thread{

public void run(){

System.out.println("task1...");

}

}

class Main{

public static void main(String args[]){

MyThread t1=new MyThread();

MyThread t2=new MyThread();

MyThread t3=new MyThread();

t1.start();

t2.start();

t3.start();

}

}

**Assign different task to multiple threads**

class MyThread1 extends Thread{

public void run(){

System.out.println("task1...");

}

}

class MyThread2 extends Thread{

public void run(){

System.out.println("task2...");

}

}

class Main{

public static void main(String args[]){

MyThread1 t1=new MyThread1();

MyThread2 t2=new MyThread2();

t1.start();

t2.start();

}

}

**Deadlock**

When a thread holds a resource and waits for another resource to be released by second thread, the second thread holding a resource and waiting for a resource to be released by first thread, then in such case both the thread will be waiting and they never execute. This is called deadlock.

class Test{

public synchronized void show1(Best b){

System.out.println("Thraed1 start execution of show1()..");

try{

Thread.sleep(6000);

}

catch(Exception e){

}

System.out.println("Thraed1 is trying to call display method of Best class");

b.display();

}

public synchronized void display(){

System.out.println("display() method of Best class");

}

}

class Best{

public synchronized void show2(Test t){

System.out.println("Thraed2 start execution of show2()..");

try{

Thread.sleep(6000);

}

catch(Exception e){

}

System.out.println("Thraed2 is trying to call display method of Test class");

t.display();

}

public synchronized void display(){

System.out.println("display() method of Test class");

}

}

class Deadlock extends Thread{

Test t = new Test();

Best b = new Best();

public void m1(){

this.start();

t.show1(b);

}

public void run(){

b.show2(t);

}

public static void main(String args[]){

Deadlock d = new Deadlock();

d.m1();

}

}