**package** SWINGP;

**class** A{

**public** **void** show()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hi");

}

}

}

**class** B{

**public** **void** show()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hello");

}

}

}

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

o1.show();

o2.show();

}

}

Hi

Hi

Hi

Hi

Hi

Hello

Hello

Hello

Hello

Hello

When u will run above prog u will get op like this

But if u want to run op after some time like after 500 miliseconds then u hv to use thread.sleep(500)

So u will get output like this

**package** SWINGP;

**class** A{

**public** **void** show()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hi");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**class** B{

**public** **void** show()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hello");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

o1.show();

o2.show();

}

}

Now using threads

**package** SWINGP;

**class** A **extends** Thread{

**public** **void** show()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hi");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**class** B **extends** Thread{

**public** **void** show()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hello");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

o1.show();

o2.show();

}

}

But still will get same op we want output like hi

Hello

Hi

Hello then …..instead of public void show () use run() method which run automatically by calling to start() method.

**package** SWINGP;

**class** A **extends** Thread{

**public** **void** run()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hi");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**class** B **extends** Thread{

**public** **void** run()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hello");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

o1.start();

o2.start();

}

}

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hello

Hello

Hi

hi

Now op may be sometime hi hello hello hi also because it is the scheduler that decided which thread to run it may choose any of these two ……so op may vary…

We cannot use multiple inheritance in java so we can achieve using interface

**package** SWINGP;

**class** A **implements** Runnable{

**public** **void** run()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hi");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**class** B **implements** Runnable{

**public** **void** run()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hello");

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

o1.start();

o2.start(); // extends \*

}

}

But while doing this will get error as we don’t have start() here with interface/implements…..

So what is the solution

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(); //create a thread class obj t1

Thread t2 =**new** Thread();

t1.start();

t2.start();

}

But still we r not getting any output so why as we have

created a thread class obj t1 but not linked with the object of class …

so we need to do it

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

**Thread t1=new Thread(o1);**

**//pass the class obj to thread(o1)**

**Thread t2 =new Thread(o2);**

t1.start();

t2.start();

}

**or**

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

}

}

//To understand join()

**public** **class** Mythread

{

**public** **static** **void** main(String[] args)

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println("bye");

}

}

Output

Hi

**bye**

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

T1 hi

T2 hello

Bye :: main() default thread

Why because hi then pause and main get chance to execute bye then hello….etc.

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

t1.join();

t2.join();

System.***out***.println("bye");

}

}

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

bye

**what join does**

[**join() :**](https://www.geeksforgeeks.org/joining-threads-in-java/) When the join() method is called, the current thread will simply wait until the thread it is joining with is no longer alive.  
Or we can say the method that you will more commonly use to wait for a thread to finish is called join( ). This method waits until the thread on which it is called terminates. Its name comes from the concept of the calling thread waiting until the specified thread joins it.

**Additional forms of join( ) allow you to specify a maximum amount of time that you want to wait for the specified thread to terminate.**

so we will get above output……

**isalive() :**

**as to chk whether a thread is active or not**

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println(t1.isAlive()); // output true here

t1.join();

t2.join();

System.***out***.println("bye");

}

}

Hi

**true**

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

bye

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println(t1.isAlive());

t1.join();

t2.join();

System.***out***.println(t1.isAlive());

System.***out***.println("bye");

}

}

Output

Hi

**true**

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

**Hello**

**False // as terminated**

bye

**Thread priority**

**Default priority set by java ….**

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

System.***out***.println(t1.getPriority());

System.***out***.println(t2.getPriority());

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println(t1.isAlive());

t1.join();

t2.join();

System.***out***.println(t1.isAlive());

System.***out***.println("bye");

}

}

**Output**

**5**

**5**

Hi

true

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

false

bye

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.setPriority(1);

t2.setPriority(10);

System.***out***.println(t1.getPriority());

System.***out***.println(t2.getPriority());

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println(t1.isAlive());

t1.join();

t2.join();

System.***out***.println(t1.isAlive());

System.***out***.println("bye");

}

}

As we set priorities but difficult to remember it ………………

**1**

**10**

Hi

true

Hello

Hi

Hello

Hi

Hello

Hi

Hello

Hi

Hello

false

bye

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

t1.setPriority(Thread.***MIN\_PRIORITY***);

t2.setPriority(Thread.***MAX\_PRIORITY***);

System.***out***.println(t1.getPriority());

System.***out***.println(t2.getPriority());

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println(t1.isAlive());

t1.join();

t2.join();

System.***out***.println(t1.isAlive());

System.***out***.println("bye");

}

}

Default values :

t1.setPriority(Thread.***MIN\_PRIORITY***); 1

t2.setPriority(Thread.***MAX\_PRIORITY***); 10

t3.setPriority(Thread.***NORM\_PRIORITY***); 5

**package** SWINGP;

**class** A **implements** Runnable{

**public** **void** run()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hi " + Thread.*currentThread*().getPriority());

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**class** B **implements** Runnable{

**public** **void** run()

{

**for**(**int** i=0;i<5;i++)

{

System.***out***.println("Hello " + Thread.*currentThread*().getPriority());

**try** {

Thread.*sleep*(500);

}**catch**(Exception e) {}

}

}

}

**public** **class** Mythread

{

**public** **static** **void** main(String[] args) **throws** Exception

{

A o1=**new** A();

B o2=**new** B();

Thread t1=**new** Thread(o1);

Thread t2 =**new** Thread(o2);

Thread t3 =**new** Thread(o2);

t1.setPriority(Thread.***MIN\_PRIORITY***);

t2.setPriority(Thread.***MAX\_PRIORITY***);

System.***out***.println(t1.getPriority());

System.***out***.println(t2.getPriority());

t1.start();

**try** {

Thread.*sleep*(10);

}**catch**(Exception e) {}

t2.start();

System.***out***.println(t1.isAlive());

t1.join();

t2.join();

System.***out***.println(t1.isAlive());

System.***out***.println("bye");

}

}

1

10

Hi 1

true

Hello 10

Hi 1

Hello 10

Hi 1

Hello 10

Hi 1

Hello 10

Hi 1

Hello 10

false

bye

if u want to chk for current thread priority values then…..u can use above code….

**package** SWINGP;

**class** counter {

**int** count;

**public** **void** increment()

{

count++;

}

}

**public** **class** syncdemo {

**public** **static** **void** main(String[] args) **throws** Exception

{

counter c =**new** counter();

c.increment();

System.***out***.println("Count " + c.count);

}

}

Now in above prog we have a class counter by which we count how many times we called counter class using c.count =1. As output

If I want c.count =2 then

**package** SWINGP;

**class** counter {

**int** count;

**public** **void** increment()

{

count++;

}

}

**public** **class** syncdemo {

**public** **static** **void** main(String[] args) **throws** Exception

{

counter c =**new** counter();

**c.increment();**

**c.increment();**

System.***out***.println("Count " + c.count);

}

}

Then output

Count 2

Now I want it using threads then ….

**package** SWINGP;

**class** counter {

**int** count;

**public** **void** increment()

{

count++;

}

}

**public** **class** syncdemo {

**public** **static** **void** main(String[] args) **throws** Exception

{

counter c =**new** counter();

Thread t1= **new** Thread(**new** Runnable()

{

**public** **void** run()

{

**for**(**int** i=1;i<=1000;i++)

{

c.increment();

}

}

});

t1.start();

t1.join();

System.***out***.println("Count " + c.count);

}

}

Here output will be count 1000 ….. now I want to perform these actions using 2 threads then…

**package** SWINGP;

**class** counter{

**int** count;

**public** **void** increment()

{

count++;

}

}

**public** **class** syncdemo {

**public** **static** **void** main(String[] args) **throws** Exception

{

counter c =**new** counter();

Thread t1= **new** Thread(**new** Runnable()

{

**public** **void** run()

{

**for**(**int** i=1;i<=1000;i++)

{

c.increment();

// System.out.println("Count " + c.count);

}

}

});

Thread t2= **new** Thread(**new** Runnable()

{

**public** **void** run()

{

**for**(**int** i=1;i<=1000;i++)

{

c.increment();

}

}

});

t1.start();

t2.start();

t1.join();

System.***out***.println("Count " + c.count);

}

}

Now every time when u run ur window output u will get a different output

Like

Count 1835

Count 1531

Count 1931

Count 1775

Because when 2 threads r running …

When one thread called counter class let it get value 0 and increment counter = counter +1 so gets printed count = 1 then 2nd thread come and reads counter = 1 and increments by 2 and so on but at some time it happens that both will get same value but instead of thread 1 thread 2 will get chance for execution and it increments its value so these happens many time and we r not getting proper output so to do it in proper manner we have **synchronized threads …**

**class** counter{

**int** count;

**public** **synchronized** **void** increment()

{

count++;

}

}

**public** **class** syncdemo {

**public** **static** **void** main(String[] args) **throws** Exception

{

counter c =**new** counter();

Thread t1= **new** Thread(**new** Runnable()

{

**public** **void** run()

{

**for**(**int** i=1;i<=1000;i++)

{

c.increment();

// System.out.println("Count " + c.count);

}

}

});

Thread t2= **new** Thread(**new** Runnable()

{

**public** **void** run()

{

**for**(**int** i=1;i<=1000;i++)

{

c.increment();

// System.out.println("Count " + c.count);

}

}

});

t1.start();

t2.start();

t1.join();

System.***out***.println("Count " + c.count);

}

}

Output

Count 2000

We have a class q with two methods one is put and another as get

**package** SWINGP;

**class** Q

{

**int** num;

**public** **void** put(**int** num)

{

**this**.num=num;

}

**public** **int** get()

{

**return** num;

}

}

**public** **class** interthread {

**public** **static** **void** main(String[] args) **throws** Exception

{

}