How to perform single task by multiple threads in Java?

If you have to perform a single task by many threads, have only one run() method. For example:

Program of performing single task by multiple threads

FileName: TestMultitasking1.java

```
class TestMultitasking1 extends Thread{
 public void run(){
  System.out.println("task one");
 }
 public static void main(String args[]){
  TestMultitasking1 t1=new TestMultitasking1();
  TestMultitasking1 t2=new TestMultitasking1();
  TestMultitasking1 t3=new TestMultitasking1();

  t1.start();
  t2.start();
  t3.start();
 }
}
```

Test it Now

Output:

```
task one
task one
task one
```

Program of performing single task by multiple threads

FileName: TestMultitasking2.java

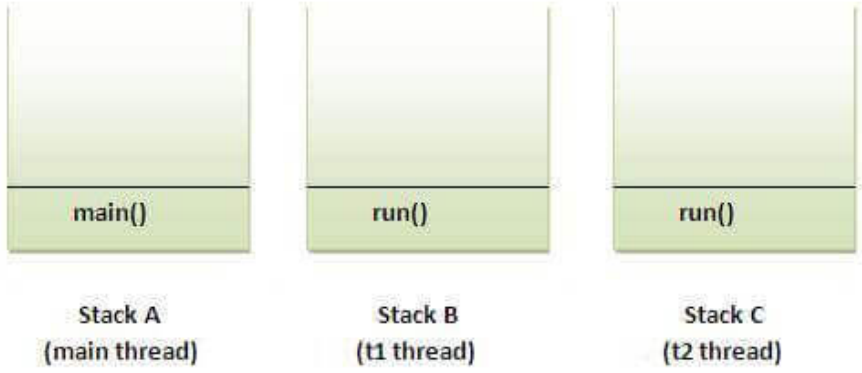
```
class TestMultitasking2 implements Runnable{
public void run(){
System.out.println("task one");
}

public static void main(String args[]){
Thread t1 =new Thread(new TestMultitasking2());//passing annonymous object of TestMultitasking2 cla
Thread t2 =new Thread(new TestMultitasking2());

t1.start();
t2.start();

 }
}
```

Test it Now

Output:

```
task one
task one
```

Note: Each thread run in a separate callstack.



How to perform multiple tasks by multiple threads (multitasking in multithreading)?

If you have to perform multiple tasks by multiple threads,have multiple run() methods.For example:

Program of performing two tasks by two threads

FileName: TestMultitasking3.java

```
class Simple1 extends Thread{
 public void run(){
   System.out.println("task one");
 }
}

class Simple2 extends Thread{
 public void run(){
   System.out.println("task two");
 }
}

 class TestMultitasking3{
 public static void main(String args[]){
  Simple1 t1=new Simple1();
  Simple2 t2=new Simple2();
```

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

Test it Now

Output:

```
task one
task two
```

Same example as above by anonymous class that extends Thread class:

Program of performing two tasks by two threads

FileName: TestMultitasking4.java

```
class TestMultitasking4{
 public static void main(String args[]){
  Thread t1=new Thread(){
    public void run(){
      System.out.println("task one");
    }
  };
  Thread t2=new Thread(){
    public void run(){
      System.out.println("task two");
    }
  };
```

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

Test it Now

Output:

```
task one
task two
```

Same example as above by anonymous class that implements Runnable interface:

Program of performing two tasks by two threads

FileName: TestMultitasking5.java

```
class TestMultitasking5{
 public static void main(String args[]){
  Runnable r1=new Runnable(){
    public void run(){
      System.out.println("task one");
    }
  };

  Runnable r2=new Runnable(){
    public void run(){
      System.out.println("task two");
    }
  };

  Thread t1=new Thread(r1);
  Thread t2=new Thread(r2);

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

```
 }
}
```

Test it Now

Output:

```
task one
task two
```

Printing even and odd numbers using two threads

To print the even and odd numbers using the two threads, we will use the synchronized block and the notify() method. Observe the following program.

FileName: OddEvenExample.java

```
// Java program that prints the odd and even numbers using two threads.
// the time complexity of the program is O(N), where N is the number up to which we
// are displaying the numbers
public class OddEvenExample
{
// Starting the counter
int contr = 1;
static int NUM;
// Method for printing the odd numbers
public void displayOddNumber()
{
// note that synchronized blocks are necessary for the code for getting the desired
```

```
// output. If we remove the synchronized blocks, we will get an exception.
synchronized (this)
{
// Printing the numbers till NUM
while (contr < NUM)
{
// If the contr is even then display
while (contr % 2 == 0)
{
// handling the exception handle
try
{
wait();
}
catch (InterruptedException ex)
{
ex.printStackTrace();
}
}
// Printing the number
System.out.print(contr + " ");
// Incrementing the contr
contr = contr + 1;
// notifying the thread which is waiting for this lock
notify();
}
}
}
// Method for printing the even numbers
public void displayEvenNumber()
{
synchronized (this)
{
// Printing the number till NUM
while (contr < NUM)
{
// If the count is odd then display
while (contr % 2 == 1)
{
```

```
// handling the exception
try
{
wait();
}
catch (InterruptedException ex)
{
ex.printStackTrace();
}
}
// Printing the number
System.out.print(contr + " ");
// Incrementing the contr
contr = contr +1;
// Notifying to the 2nd thread
notify();
}
}
}
// main method
public static void main(String[] argvs)
{
// The NUM is given
NUM = 20;
// creating an object of the class OddEvenExample
OddEvenExample oe = new OddEvenExample();
// creating a thread th1
Thread th1 = new Thread(new Runnable()
{
public void run()
{
// invoking the method displayEvenNumber() using the thread th1
oe.displayEvenNumber();
}
});
// creating a thread th2
Thread th2 = new Thread(new Runnable()
{
public void run()
```

```
{
// invoking the method displayOddNumber() using the thread th2
oe.displayOddNumber();
}
});
// starting both of the threads
th1.start();
th2.start();
}
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

← Prev

Next →

Help Others, Please Share

Learn Latest Tutorials

Splunk tutorial
Splunk

SPSS tutorial
SPSS

Swagger tutorial
Swagger

T-SQL tutorial
Transact-SQL

Tumblr tutorial
Tumblr

React tutorial
ReactJS

Regex tutorial
Regex

Reinforcement learning tutorial
Reinforcement Learning

R Programming tutorial
R Programming

RxJS tutorial
RxJS

React Native tutorial
React Native

Python Design Patterns
Python Design Patterns

Python Pillow tutorial
Python Pillow

Python Turtle tutorial
Python Turtle

Keras tutorial
Keras

Preparation

Aptitude
Aptitude

Logical Reasoning
Reasoning

Verbal Ability
Verbal Ability

Interview Questions
Interview Questions

Company Interview Questions
Company Questions

Trending Technologies

B.Tech / MCA