Threading sample:

1.

// Create a second thread.

class SampleThread implements Runnable

{

Thread t;

SampleThread()

{

// Create a new, second thread

t = new Thread(this, "Demo Thread");

System.out.println("Child thread: " + t);

t.start();

// Start the thread

}

// This is the entry point for the second thread.

public void run() {

try {

for(int i = 5; i > 0; i--) {

System.out.println("Child Thread" +i);

Thread.sleep(500);

}

} catch (InterruptedException e) {

System.out.println("Child interrupted.");

}

System.out.println("Exiting child thread.");

}

}

class ThreadDemo

{

public static void main(String args[])

{

Thread t1=Thread.currentThread();

System.out.println("Main Thread" +t1);

new SampleThread(); // create a new thread

try {

for(int i = 5; i > 0; i--) {

System.out.println("Main Thread: " + i);

Thread.sleep(1000);

}

} catch (InterruptedException e)

{

System.out.println("Main thread interrupted.");

}

System.out.println("Main thread exiting.");

}

}

}

Thread Priority:

1.

class TestMultiPriority1 extends Thread

{

 public void run()

{

   System.out.println("running thread name is:"+Thread.currentThread().getName());

   System.out.println("running thread priority is:"+Thread.currentThread().getPriority());

    }

 public static void main(String args[]){

  TestMultiPriority1 m1=new TestMultiPriority1();

  TestMultiPriority1 m2=new TestMultiPriority1();

  m1.setPriority(Thread.MIN\_PRIORITY);

  m2.setPriority(Thread.MAX\_PRIORITY);

  m1.start();

  m2.start();

  }

}

Multi-Thread:

1.

// Create multiple threads.

class NewThread implements Runnable {

String name; // name of thread

Thread t;

NewThread(String threadname) {

name = threadname;

t = new Thread(this, name);

System.out.println("New thread: " + t);

t.start(); // Start the thread

}

// This is the entry point for thread.

public void run()

{

try

{

for(int i = 5; i > 0; i--) {

System.out.println(name + ": " + i);

Thread.sleep(1000); }

}

catch (InterruptedException e)

{

System.out.println(name + "Interrupted");

}

System.out.println(name + " exiting.");

}

}

class MultiThreadDemo {

public static void main(String args[]) {

new NewThread("One"); // start threads

new NewThread("Two");

new NewThread("Three");

try

{

Thread.sleep(10000);

}

catch (InterruptedException e)

{

System.out.println("Main thread Interrupted"); }

System.out.println("Main thread exiting.");

}

}

Inter-Thread Communication

1.

class Q {  
int n;  
boolean valueSet = false;  
synchronized int get() {  
while(!valueSet)  
try

{  
wait();  
}

catch(InterruptedException e)

{  
System.out.println("InterruptedException caught");  
}  
System.out.println("Got: " + n);  
valueSet = false;  
notify();  
return n;  
}  
synchronized void put(int n) {  
while(valueSet)  
try

{  
wait();  
}

catch(InterruptedException e)

{  
System.out.println("InterruptedException caught");  
}  
this.n = n;  
valueSet = true;  
System.out.println("Put: " + n);  
notify();  
}  
}  
class Producer implements Runnable {  
Q q;  
Producer(Q q) {  
this.q = q;  
new Thread(this, "Producer").start();  
}  
public void run() {  
int i = 0;  
while(true) {  
q.put(i++);  
}  
}  
}  
class Consumer implements Runnable {  
Q q;  
Consumer(Q q) {  
this.q = q;  
new Thread(this, "Consumer").start();  
}  
public void run() {  
while(true) {  
q.get();  
}  
}  
}  
class PCFixed {  
public static void main(String args[]) {  
Q q = new Q();  
new Producer(q);  
new Consumer(q);  
System.out.println("Press Control-C to stop.");  
}  
}

sample op:

Put: 1  
Got: 1  
Put: 2  
Got: 2  
Put: 3  
Got: 3  
Put: 4  
Got: 4  
Put: 5  
Got: 5

Synchronisation:

|  |  |
| --- | --- |
| // This program is not synchronized.  class Callme {  void call(String msg) {  System.out.print("[" + msg);  try {  Thread.sleep(1000);  } catch(InterruptedException e) {  System.out.println("Interrupted");  }  System.out.println("]");  }  }  class Caller implements Runnable {  String msg;  Callme target;  Thread t;  public Caller(Callme targ, String s) {  target = targ;  msg = s;  t = new Thread(this);  t.start();  }  public void run() {  target.call(msg);  }  }  class Synch {  public static void main(String args[]) {  Callme target = new Callme();  Caller ob1 = new Caller(target, "Hello");  Caller ob2 = new Caller(target, "Synchronized");  Caller ob3 = new Caller(target, "World");  // wait for threads to end  try {  ob1.t.join();  ob2.t.join();  ob3.t.join();  } catch(InterruptedException e) {  System.out.println("Interrupted");  }  }  }  Here is the output produced by this program:  Hello[Synchronized[World]  ]  ] | // This program uses a synchronized block.  class Callme {  void call(String msg) {  System.out.print("[" + msg);  try {  Thread.sleep(1000);  } catch (InterruptedException e) {  System.out.println("Interrupted");  }  System.out.println("]");  }  }  class Caller implements Runnable {  String msg;  Callme target;  Thread t;  public Caller(Callme targ, String s) {  target = targ;  msg = s;  t = new Thread(this);  t.start();  }  // synchronize calls to call()  public void run() {  synchronized(target) { // synchronized block  target.call(msg);  }  }  }  class Synch1 {  public static void main(String args[]) {  Callme target = new Callme();  Caller ob1 = new Caller(target, "Hello");  Caller ob2 = new Caller(target, "Synchronized");  Caller ob3 = new Caller(target, "World");  // wait for threads to end  try {  ob1.t.join();  ob2.t.join();  ob3.t.join();  } catch(InterruptedException e) {  System.out.println("Interrupted");  }  }  } |