

**import** java.util.concurrent.ExecutorService;

**import** java.util.concurrent.Executors;

**import** java.util.concurrent.Future;

**public** **class** DemoExecutorUsage {

**private** **static** ExecutorService *executor* = **null**;

**private** **static** **volatile** Future *taskOneResults* = **null**;

**private** **static** **volatile** Future *taskTwoResults* = **null**;

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

*executor* = Executors.*newFixedThreadPool*(2);

**while** (**true**)

{

**try**

{

*checkTasks*();

Thread.*sleep*(1000);

} **catch** (Exception e) {

System.***err***.println("Caught exception: " + e.getMessage());

}

}

}

**private** **static** **void** checkTasks() **throws** Exception {

**if** (*taskOneResults* == **null**

|| *taskOneResults*.isDone()

|| *taskOneResults*.isCancelled())

{

*taskOneResults* = *executor*.submit(**new** TestOne());

}

**if** (*taskTwoResults* == **null**

|| *taskTwoResults*.isDone()

|| *taskTwoResults*.isCancelled())

{

*taskTwoResults* = *executor*.submit(**new** TestTwo());

}

}

}

**class** TestOne **implements** Runnable {

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

**while** (**true**)

{

System.***out***.println("Executing task one");

**try**

{

Thread.*sleep*(1000);

} **catch** (Throwable e) {

e.printStackTrace();

}

}

}

}

**class** TestTwo **implements** Runnable {

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

**while** (**true**)

{

System.***out***.println("Executing task two");

**try**

{

Thread.*sleep*(1000);

} **catch** (Throwable e) {

e.printStackTrace();

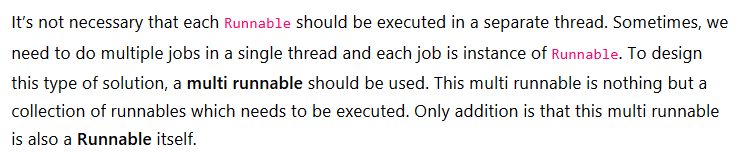
}

}

}

}





**public** **class** TaskThree **implements** Runnable {

@Override

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

System.***out***.println("Executing Task Three");

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}



**import** java.util.List;

**public** **class** MultiRunnable **implements** Runnable {

**private** **final** List<Runnable> runnables;

**public** MultiRunnable(List<Runnable> runnables) {

**this**.runnables = runnables;

}

@Override

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

**for** (Runnable runnable : runnables) {

**new** Thread(runnable).start();

}

}

}



**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.concurrent.ArrayBlockingQueue;

**import** java.util.concurrent.BlockingQueue;

**import** java.util.concurrent.RejectedExecutionHandler;

**import** java.util.concurrent.ThreadPoolExecutor;

**import** java.util.concurrent.TimeUnit;

**public** **class** MultiTaskExecutor {

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

BlockingQueue<Runnable> worksQueue = **new** ArrayBlockingQueue<Runnable>(10);

RejectedExecutionHandler rejectionHandler = **new** RejectedExecutionHandelerImpl();

ThreadPoolExecutor executor = **new** ThreadPoolExecutor(3, 3, 10, TimeUnit.***SECONDS***, worksQueue, rejectionHandler);

executor.prestartAllCoreThreads();

List<Runnable> taskGroup = **new** ArrayList<Runnable>();

taskGroup.add(**new** TestOne());

taskGroup.add(**new** TestTwo());

taskGroup.add(**new** TestThree());

worksQueue.add(**new** MultiRunnable(taskGroup));

}

}

**import** java.util.concurrent.RejectedExecutionHandler;

**import** java.util.concurrent.ThreadPoolExecutor;

**class** RejectedExecutionHandelerImpl **implements** RejectedExecutionHandler

{

@Override

**public** **void** rejectedExecution(Runnable runnable,

ThreadPoolExecutor executor)

{

System.***out***.println(runnable.toString() + " : I've been rejected ! ");

}

}



