新增方式一：實現Callable接口。 --- JDK 5.0新增

//1.創建一個實現Callable的實現類

**class** NumThread **implements** Callable{

//2.實現call方法，將此線程需要執行的操作聲明在call()中

@Override

**public** Object call() **throws** Exception {

**int** sum = 0;

**for** (**int** i = 1; i <= 100; i++) {

**if**(i % 2 == 0){

System.***out***.println(i);

sum += i;

}

}

**return** sum;//int自動裝箱成Integer類,回傳類型才是Object類型

}

}

**public** **class** ThreadNew {

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

//3.創建Callable接口實現類的對象

NumThread numThread = **new** NumThread();

//4.將此Callable接口實現類的對象作為傳遞到FutureTask構造器中，創建FutureTask的對象

FutureTask futureTask = **new** FutureTask(numThread);

//5.將FutureTask的對象作為參數傳遞到Thread類的構造器中，創建Thread對象，並調用start()

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

**try** {

//6.獲取Callable中call方法的返回值

//get()返回值即為FutureTask構造器參數Callable實現類重寫的call()的返回值。

Object sum = futureTask.get();

System.***out***.println("總和為：" + sum);

} **catch** (InterruptedException e) {

e.printStackTrace();

} **catch** (ExecutionException e) {

e.printStackTrace();

}

}

}

說明：

\* 如何理解實現Callable接口的方式創建多線程比實現Runnable接口創建多線程方式強大？

\* 1. call()可以有返回值的。

\* 2. call()可以拋出異常，被外面的操作捕獲，獲取異常的信息

\* 3. Callable是支持泛型的

新增方式二：使用線程池

**class** NumberThread **implements** Runnable{

@Override

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

**for**(**int** i = 0;i <= 100;i++){

**if**(i % 2 == 0){

System.***out***.println(Thread.*currentThread*().getName() + ": " + i);

}

}

}

}

**class** NumberThread1 **implements** Runnable{

@Override

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

**for**(**int** i = 0;i <= 100;i++){

**if**(i % 2 != 0){

System.***out***.println(Thread.*currentThread*().getName() + ": " + i);

}

}

}

}

**public** **class** ThreadPool {

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

//1. 提供指定線程數量的線程池

ExecutorService service = Executors.*newFixedThreadPool*(10);

//返回的對象賦值的不是接口對象,接口不能造對象,就一定是接口實現類的對

象:ThreadPoolExecutor

ThreadPoolExecutor service1 = (ThreadPoolExecutor) service;

//設置線程池的屬性

// System.out.println(service.getClass());//顯示當前類的類型

// service1.setCorePoolSize(15);

// service1.setKeepAliveTime();

//2.執行指定的線程的操作。需要提供實現Runnable接口或Callable接口實現類的對象

service.execute(**new** NumberThread());//適合適用於Runnable

service.execute(**new** NumberThread1());//適合適用於Runnable

// service.submit(Callable callable);//適合使用於Callable

//3.關閉連接池

service.shutdown();

}

}

說明：

好處：

\* 1.提高響應速度（減少了創建新線程的時間）

\* 2.降低資源消耗（重覆利用線程池中線程，不需要每次都創建）

\* 3.便於線程管理

\* corePoolSize：核心池的大小

\* maximumPoolSize：最大線程數

\* keepAliveTime：線程沒有任務時最多保持多長時間後會終止

面試題：Java中多線程的創建有幾種方式?