**package** com.captain.demo.crazy;

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

**import** java.util.concurrent.Callable;

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

/\*\*

\* 注水线程

\* Created by captain on 2017/7/27.

\*/

**public** **class** InputThread **implements** Callable<Object>{

**private** BlockingQueue<String> queue;

**public** InputThread(BlockingQueue queue){

**this**.queue = queue;

}

@Override

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

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

**boolean** flag = **true**;

**for**(**int** cnt=0;cnt<5;cnt++){

flag = queue.offer("一立方米");

**if**(!flag){

**break**;

}

}

System.***out***.println("注水5立方米，当前水量："+queue.size());

**if**(!flag){

**break**;

}

TimeUnit.***SECONDS***.sleep(1);

}

**return** "success";

}

}

package com.captain.demo.crazy;

import java.util.concurrent.BlockingQueue;

import java.util.concurrent.Callable;

import java.util.concurrent.TimeUnit;

/\*\*

\* 放水线程

\* Created by captain on 2017/7/27.

\*/

public class OutputThread implements Callable<Object>{

private BlockingQueue<String> queue;

public OutputThread(BlockingQueue queue){

this.queue = queue;

}

@Override

public Object call() throws InterruptedException {

while(true){

if(queue.size()==10000){

break;

}

for(int cnt=0;cnt<3;cnt++){

queue.poll(10,TimeUnit.SECONDS);

}

System.out.println("放水3立方米，当前水量："+queue.size());

TimeUnit.SECONDS.sleep(1);

}

return "success";

}

}

package com.captain.demo.crazy;

import java.util.ArrayList;

import java.util.List;

import java.util.TimerTask;

import java.util.concurrent.ArrayBlockingQueue;

import java.util.concurrent.BlockingQueue;

import java.util.concurrent.Callable;

import java.util.concurrent.ExecutionException;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.FutureTask;

import java.util.concurrent.TimeUnit;

import java.util.concurrent.TimeoutException;

/\*\*

\* 定期放水类

\* Created by captain on 2017/7/27.

\*/

public class WaterTimeTask extends TimerTask {

public static List<Object> runCheckCallable(List<Callable<Object>> l, Boolean b)

{

FutureTask<Object> futureTask1 = new FutureTask<Object>(l.get(0));

// 将Callable写的任务封装到一个由执行者调度的FutureTask对象

FutureTask<Object> futureTask2 = new FutureTask<Object>(l.get(1));

// 创建线程池并返回ExecutorService实例

ExecutorService executor = Executors.newFixedThreadPool(2);

// 执行任务

executor.execute(futureTask1);

executor.execute(futureTask2);

while (true)

{

try {

if(futureTask1.isDone() && futureTask2.isDone()){// 两个任务都完成

System.out.println("Done");

// 关闭线程池和服务

executor.shutdown();

List<Object> Lobject = null;

Lobject.add(l.get(0));

Lobject.add(l.get(1));

return Lobject;

}

if(!futureTask1.isDone()){ // 任务1没有完成，会等待，直到任务完成

System.out.println("FutureTask1 output="+futureTask1.get());

}

System.out.println("Waiting for FutureTask2 to complete");

String s = (String) futureTask2.get(200L, TimeUnit.MILLISECONDS);

if(s !=null){

System.out.println("FutureTask2 output="+s);

}

}

catch (InterruptedException | ExecutionException e)

{

e.printStackTrace();

}

catch(TimeoutException e)

{

//do nothing

}

}

}

@Override

public void run() {

BlockingQueue<String> queue = new ArrayBlockingQueue<String>(10000);

InputThread thread1 = new InputThread(queue);

InputThread thread3 = new InputThread(queue);

OutputThread thread2 = new OutputThread(queue);

List<Callable<Object>> threadList = new ArrayList<>(2);

threadList.add(thread1);

threadList.add(thread2);

threadList.add(thread3);

List<Object> returnValue = null;

// try {

// returnValue = ThreadUtil.runCheckCallable(threadList,true);

//} catch (InterruptedException e) {

// e.printStackTrace();

// } catch (ExecutionException e) {

// e.printStackTrace();

// }

returnValue = runCheckCallable(threadList,true);

System.out.println("泳池已经注满");

System.out.println(returnValue.get(0));

System.out.println(returnValue.get(1));

}

}

package com.captain.demo.crazy;

import java.util.Calendar;

import java.util.Timer;

/\*\*

\* Created by captain on 2017/7/27.

\*/

public class MainDrive {

public static void main(String[] args) {

Calendar calendar = Calendar.getInstance();

calendar.set(

calendar.get(Calendar.YEAR),

calendar.get(Calendar.MONTH),

calendar.get(Calendar.DATE),

20,0,0

);

WaterTimeTask task = new WaterTimeTask();

Timer timer = new Timer();

timer.schedule(task,0,1000);

}

}