**基础题**

### 练习一：多线程开启

问题：

请描述Thread类中的start()方法与run()方法的区别。

答：

线程对象调用run()方法不开启线程，仅是对象调用方法。线程对象调用start()方法开启线程，并让jvm调用run()方法在开启的线程中执行。

### 练习二：创建多线程

问题：

请描述创建线程的两种方式。

答：

* **第一种方式是将类声明为 Thread 的子类。**

1. 定义Thread类的子类，并重写该类的run()方法，该run()方法的方法体就代表了线程需要完成的任务,因此把run()方法称为线程执行体。
2. 创建Thread子类的实例，即创建了线程对象。
3. 调用线程对象的start()方法来启动该线程。

* **第二种方式是声明一个类实现Runnable 接口。**

1. 定义Runnable接口的实现类，并重写该接口的run()方法，该run()方法的方法体同样是该线程的线程执行体。

2. 创建Runnable实现类的实例，并以此实例作为Thread的target来创建Thread对象，Thread对象才是真正的线程对象。

3. 调用线程对象的start()方法来启动线程。

### 练习三：多线程练习

问题：

请编写程序，分别打印主线程的名称和子线程的名称。

要求使用两种方式实现：

第一种方式：继承Thread类。

第二种方法：实现Runnable接口。

答：

**操作步骤描述**

* **第一种方式：继承Thread类**

1.定义一个子线程的类，继承Thread类；

2.在子线程类中重写run方法，在run方法中打印子线程的名称；

3.定义一个测试类；

4.在main方法中打印主线程的名称；

5.在main方法中创建子线程对象；

6.调用子线程对象的start方法，开启子线程；

* **第二种方式：实现Runnable接口**

1.定义一个子任务类，实现Runnable接口；

2.在子任务类中重写run方法，在run方法中打印子线程的名称；

3.定义一个测试类；

4.在main方法中打印主线程的名称；

5.在main方法中创建一个子任务对象；

6.在main方法中创建一个Thread类的对象，并把子任务对象传递给Thread类的构造方法；

7.调用Thread类对象的start方法开启子线程；

**操作步骤答案**

* **第一种方式：继承Thread类**

*/\*  
 \* 1.定义一个子线程的类，继承Thread类；  
 \*/*

|  |
| --- |
| **public class** SubThread **extends** Thread {  */\*  \*2.在子线程类中重写run方法，在run方法中打印子线程的名称；  \*/* **public void** run() {  *// 打印子线程的名称* System.***out***.println(**"subThread:"** + Thread.*currentThread*().getName());  } } |

*/\*  
 \* 3.定义一个测试类  
 \*/*

|  |
| --- |
| **public class** ThreadDemo {  **public static void** main(String[] args) {  *// 4.在main方法中打印主线程的名称；* System.***out***.println(**"main:"** + Thread.*currentThread*().getName());  *// 5.在main方法中创建子线程对象；* SubThread st = **new** SubThread();  *// 6.调用子线程对象的start方法，开启子线程。* st.start();  } } |

* **第二种方式：实现Runnable接口**

*/\*  
 \* 1.定义一个子任务类，实现Runnable接口。  
 \*/*

|  |
| --- |
| **public class** SubRunnable **implements** Runnable {  @Override  **public void** run() {  *// 2.在子任务类中重写run方法，在run方法中打印子线程的名称。* System.***out***.println(**"SubRunnable:"**+ Thread.*currentThread*().getName());  } } |

*/\*  
 \* 3.定义一个测试类。  
 \*/*

|  |
| --- |
| **public class** RunnableDemo {  **public static void** main(String[] args) {  *// 4.在main方法中打印主线程的名称。* System.***out***.println(**"RunnableDemo:"**+ Thread.*currentThread*().getName());  *// 5.在main方法中创建一个子任务对象。* SubRunnable r = **new** SubRunnable();  *// 6.在main方法中创建一个Thread类的对象，并把子任务对象传递给Thread类的 构造方法。* Thread t = **new** Thread(r);  *// 7.调用Thread类对象的start方法开启子线程。* t.start();  } } |

### 练习四：实现Runnable接口的优势

问题：

请描述实现Runnable接口比继承Thread类所具有的优势：

答：

1. 适合多个相同的程序代码的线程去共享同一个资源。

2. 可以避免java中的单继承的局限性。

3. 增加程序的健壮性，实现解耦操作，代码可以被多个线程共享，代码和数据独立。

4. 线程池只能放入实现Runable或callable类线程，不能直接放入继承Thread的类。

### 练习五：线程状态

问题：请描述在线程的生命周期中， 有几种状态呢 ?

答：

1. NEW(新建) 线程刚被创建，但是并未启动。
2. Runnable(可运行)

线程可以在java虚拟机中运行的状态，可能正在运行自己代码，也可能没有，这取决于操作系统处理器。

1. Blocked(锁阻塞)

当一个线程试图获取一个对象锁，而该对象锁被其他的线程持有，则该线程进入Blocked状态；当该线程持有锁时，该线程将变成Runnable状态。

1. Waiting(无限等待)

一个线程在等待另一个线程执行一个（唤醒）动作时，该线程进入Waiting状态。进入这个状态后是不能自动唤醒的，必须等待另一个线程调用notify或者notifyAll方法才能够唤醒。

1. Timed Waiting(计时等待)

同waiting状态，有几个方法有超时参数，调用他们将进入Timed Waiting状态。这一状态将一直保持到超时期满或者接收到唤醒通知。带有超时参数的常用方法有Thread.sleep 、Object.wait。

1. Teminated(被终止)

因为run方法正常退出而死亡，或者因为没有捕获的异常终止了run方法而死亡。