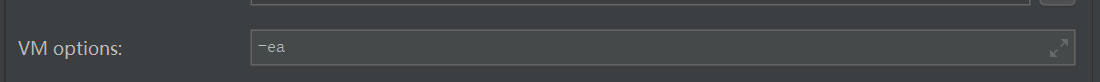
一.assert（java中的断言）

语法：

在Java中，assert关键字是从JAVA SE 1.4 引入的，为了避免和老版本的Java代码中使用了assert关键字导致错误，Java在执行的时候默认是不启动断言检查的，如果要开启断言检查，则需要用开关-enableassertions或-ea来开启。

idea中开启断言：



assert关键字语法很简单，有两种用法：

>>1、assert <boolean表达式>

如果<boolean表达式>为true，则程序继续执行。

如果为false，则程序抛出AssertionError，并终止执行。

>>2、assert <boolean表达式> : <错误信息表达式>

如果<boolean表达式>为true，则程序继续执行。

如果为false，则程序抛出java.lang.AssertionError，并输入<错误信息表达式>。

2.assert总结

>>1、assert关键字需要在运行时候显式开启才能生效，否则你的断言就没有任何意义。而现在主流的Java IDE工具默认都没有开启-ea断言检查功能。这就意味着你如果使用IDE工具编码，调试运行时候会有一定的麻烦。并且，对于Java Web应用，程序代码都是部署在容器里面，你没法直接去控制程序的运行，如果一定要开启-ea的开关，则需要更改Web容器的运行配置参数。这对程序的移 植和部署都带来很大的不便。

>>2、用assert代替if是陷阱之二。assert的判断和if语句差不多，但两者的作用有着本质的区别：assert关键字本意上是为测试 调试程序时使用的，但如果不小心用assert来控制了程序的业务流程，那在测试调试结束后去掉assert关键字就意味着修改了程序的正常的逻辑。

>>3、assert断言失败将面临程序的退出。这在一个生产环境下的应用是绝不能容忍的。一般都是通过异常处理来解决程序中潜在的错误。但是使用断言就很危险，一旦失败系统就挂了。

3.断言的使用场景

它主要使用在代码开发和测试时期，用于对某些关键数据的判断，如果这个关键数据不是你程序所预期的数据，程序就提出警告或退出。

二.synchronized

1.synchronied使用场景

在并发编程中存在线程安全问题，主要原因有：1.存在共享数据 2.多线程共同操作共享数据。关键字synchronized可以保证在同一时刻，只有一个线程可以执行某个方法或某个代码块，同时synchronized可以保证一个线程的变化可见（可见性）。

最大的特征就是在同一时刻只有一个线程能够获得对象的监视器（monitor），从而进入到同步代码块或者同步方法之中，即表现为**互斥性（排它性）**。每次只能通过一个线程。

------------------------继承Thread类创建线程---------------------

通过继承Thread类来创建并启动多线程的一般步骤如下

1】d定义Thread类的子类，并重写该类的run()方法，该方法的方法体就是线程需要完成的任务，run()方法也称为线程执行体。

2】创建Thread子类的实例，也就是创建了线程对象

3】启动线程，即调用线程的start()方法

代码实例

public class MyThread extends Thread{//继承Thread类

　　public void run(){

　　//重写run方法

　　}

}

public class Main {

　　public static void main(String[] args){

　　　　new MyThread().start();//创建并启动线程

　　}

}

------------------------实现Runnable接口创建线程---------------------

通过实现Runnable接口创建并启动线程一般步骤如下：

1】定义Runnable接口的实现类，一样要重写run()方法，这个run（）方法和Thread中的run()方法一样是线程的执行体

2】创建Runnable实现类的实例，并用这个实例作为Thread的target来创建Thread对象，这个Thread对象才是真正的线程对象

3】第三部依然是通过调用线程对象的start()方法来启动线程

代码实例：

public class MyThread2 implements Runnable {//实现Runnable接口

　　public void run(){

　　//重写run方法

　　}

}

public class Main {

　　public static void main(String[] args){

　　　　//创建并启动线程

　　　　MyThread2 myThread=new MyThread2();

　　　　Thread thread=new Thread(myThread);

　　　　thread().start();

　　　　//或者    new Thread(new MyThread2()).start();

　　}

}