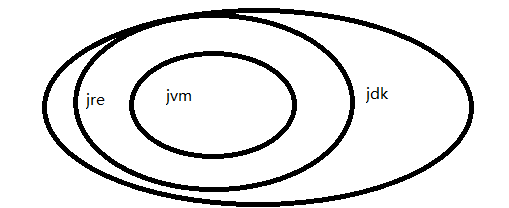
1. Jvm,jre,jdk的区别和联系？？？



1. Java迄今为止，发布了多少个版本，哪些版本具有里程碑式的重大意义，具体是什么重大的改变和新特性？？

目前到 12版本

1.2 版本：J2SE，J2EE，J2ME。**JIT（Just In Time）编译器**。

1.5版本：java.util.concurrent，注解

1.7版本： 提供了**G1收集器**

**1.8版本**：接口扩展，lambda ，并行流集合 Stream

1.9版本：HTTP 2 客户端，响应式流（Reactive Streams)

**1.10版本**： var，**并行Full GC的G1**

**1.12 ……………………………..**

**有个重大事件： oracle放弃java，**

在2019年1月底，Oracle正式结束了针对非Oracle客户商业用户的Oracle JDK免费公开更新。没有Oracle支持合同，这些用户将无法再获得更新。此外，Oracle已更改了Oracle JDK许可证（BCPL），因此JDK 11及更高版本的商业用途将需要Oracle订阅。

3. java语言规范，jvm规范，jmm区别和联系？？？

Jvm规范是一个接口，一个协议：具体实现



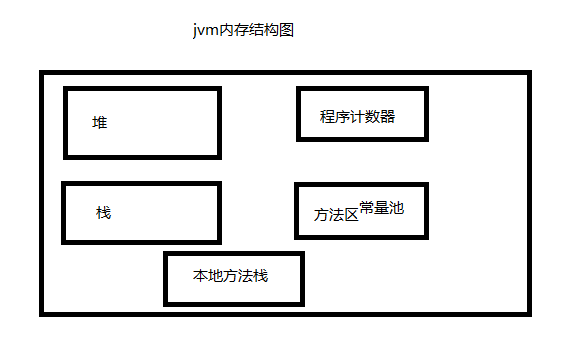
* **HotSpot VM**
* **J9 VM**
* **Zing VM**

JRockit

Java 语言规范：语言级别的协议。

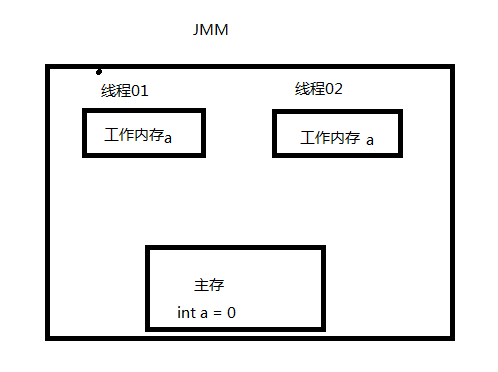
JMM:JAVA MEMORY MODEL，java内存模型，也是一个协议，是为了简化java多线程并发程序的开发，java组织抽象了一个跨系统，跨硬件的层。这样我们写的代码就具有统一性。

4 . JVM的内存运行时数据区结构？？



**哪些里面的数据是线程安全的？？堆里面的数据，部分方法区的数据。**

5. JMM？？？？



三大特征：  
**原子性 :**

**可见性 :**

**有序性 :**

原则：

Happens---before

实现方法：用**synchronized（三个特性都满足）**

**Volatile:(可见性，有序性)**

**Java.util.concurrent:工具类**

**6. 单例模式**

**饿汉式：**

|  |
| --- |
| **package** mianshi.dm;  **public** **class** Single {  **private** **static** Single *single* = **new** Single();  **private** Single() {}  **public** **static** Single getInstance() {  **return** *single*;  }  } |

**线程安全的！！！**

**懒汉式：**

|  |
| --- |
| **package** mianshi.dm;  **public** **class** Single {  **private** **static** Single *single*;  **private** Single() {}  **public** **static** Single getInstance() {  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  *single* = **new** Single();  **return** *single*;  }  }  } |

**问题：线程安全吗？ 不安全**

**解决：加锁**

|  |
| --- |
| **public** **synchronized** **static** Single getInstance() {  //get lock  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  *single* = **new** Single();  **return** *single*;  }  //unlock  } |

**缺点：这种加锁方法，范围越大并发性越低。**

**2 缩小锁的范围**

|  |
| --- |
| **public** **static** Single getInstance() {  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  **synchronized** (Single.**class**) {  *single* = **new** Single();  **return** *single*;  }  }  } |

**优点：提高性能**

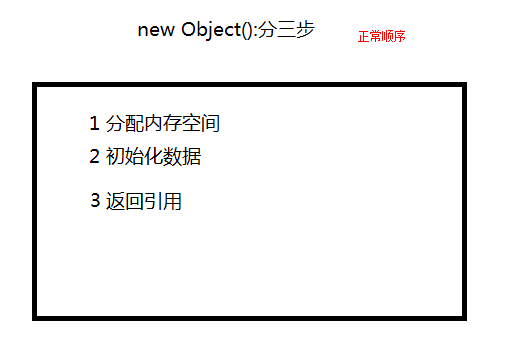
**缺点：还是有安全问题**

**3 双重检查锁**

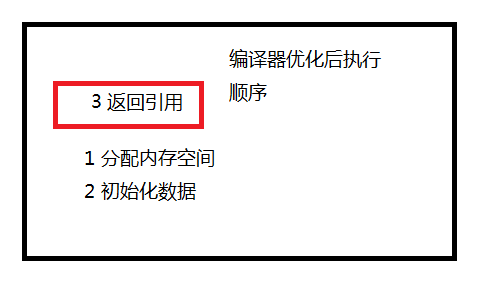
|  |
| --- |
| **public** **static** Single getInstance() {  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  **synchronized** (Single.**class**) {  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  *single* = **new** Single();  **return** *single*;  }  }  }  } |

**优点：性能很好，看似也安全**

**缺点：其实并不安全,有个乱排序问题**

****

**优化后结果：**

****

**会出现返回的对象没有初始化完全！！**

**4 volatile**

**加了这个关键字，就不乱排序了。禁止重排序！！！**

**注意点：jdk5.0版本以后才可以！！**

|  |
| --- |
| **private** **volatile** **static** Single *single*;  **private** **int** number;  **private** Single() {}  **public** **static** Single getInstance() {  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  **synchronized** (Single.**class**) {  **if**(*single*!=**null**) {  **return** *single*;  }**else** {  *single* = **new** Single();  **return** *single*;  }  }  }  } |

**5 静态内部类**

|  |
| --- |
| public class Singleton {  private Singleton() {}  private static class SingletonInstance {  private static final Singleton INSTANCE = new Singleton();  }  public static Singleton getInstance() {  return SingletonInstance.INSTANCE;  }  } |

**6 枚举**

|  |
| --- |
| **public enum Singleton {**  **INSTANCE;**  **public void whateverMethod() {**  **}**  **}** |