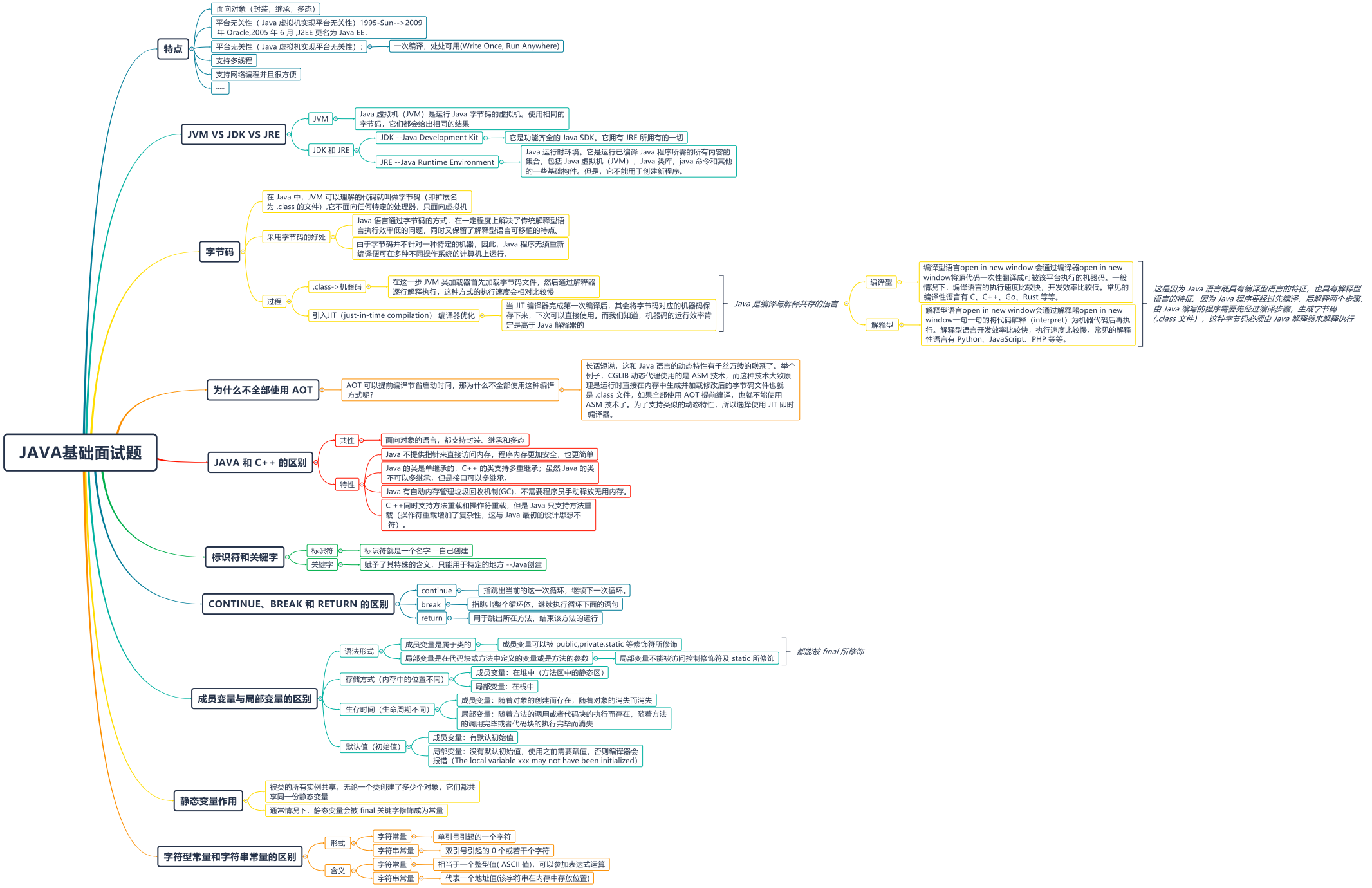
JAVA基础面试题
特点
面向对象（封装，继承，多态）
平台无关性（ Java 虚拟机实现平台无关性）1995-Sun-->2009 年 Oracle,2005 年 6 月 ,J2EE 更名为 Java EE,
平台无关性（ Java 虚拟机实现平台无关性）；
一次编译，处处可用(Write Once, Run Anywhere)
支持多线程
支持网络编程并且很方便
.....
JVM VS JDK VS JRE
JVM
Java 虚拟机（JVM）是运行 Java 字节码的虚拟机。使用相同的字节码，它们都会给出相同的结果
JDK 和 JRE
JDK --Java Development Kit
它是功能齐全的 Java SDK。它拥有 JRE 所拥有的一切
JRE --Java Runtime Environment
Java 运行时环境。它是运行已编译 Java 程序所需的所有内容的集合，包括 Java 虚拟机（JVM），Java 类库，java 命令和其他的一些基础构件。但是，它不能用于创建新程序。
字节码
在 Java 中，JVM 可以理解的代码就叫做字节码（即扩展名为 .class 的文件），它不面向任何特定的处理器，只面向虚拟机
采用字节码的好处
Java 语言通过字节码的方式，在一定程度上解决了传统解释型语言执行效率低的问题，同时又保留了解释型语言可移植的特点。
由于字节码并不针对一种特定的机器，因此，Java 程序无须重新编译便可在多种不同操作系统的计算机上运行。
过程
.class->机器码
在这一步 JVM 类加载器首先加载字节码文件，然后通过解释器逐行解释执行，这种方式的执行速度会相对比较慢
引入JIT（just-in-time compilation） 编译器优化
当 JIT 编译器完成第一次编译后，其会将字节码对应的机器码保存下来，下次可以直接使用。而我们知道，机器码的运行效率肯定是高于 Java 解释器的
Java 是编译与解释共存的语言
编译型
编译型语言open in new window 会通过编译器open in new window将源代码一次性翻译成可被该平台执行的机器码。一般情况下，编译语言的执行速度比较快，开发效率比较低。常见的编译性语言有 C、C++、Go、Rust 等等。
解释型
解释型语言open in new window会通过解释器open in new window一句一句的将代码解释（interpret）为机器代码后再执行。解释型语言开发效率比较快，执行速度比较慢。常见的解释性语言有 Python、JavaScript、PHP 等等。
这是因为 Java 语言既具有编译型语言的特征，也具有解释型语言的特征。因为 Java 程序要经过先编译，后解释两个步骤，由 Java 编写的程序需要先经过编译步骤，生成字节码（.class 文件），这种字节码必须由 Java 解释器来解释执行
为什么不全部使用 AOT
AOT 可以提前编译节省启动时间，那为什么不全部使用这种编译方式呢?
长话短说，这和 Java 语言的动态特性有千丝万缕的联系了。举个例子，CGLIB 动态代理使用的是 ASM 技术，而这种技术大致原理是运行时直接在内存中生成并加载修改后的字节码文件也就是 .class 文件，如果全部使用 AOT 提前编译，也就不能使用 ASM 技术了。为了支持类似的动态特性，所以选择使用 JIT 即时编译器。
JAVA 和 C++ 的区别
共性
面向对象的语言，都支持封装、继承和多态
特性
Java 不提供指针来直接访问内存，程序内存更加安全，也更简单
Java 的类是单继承的，C++ 的类支持多重继承；虽然 Java 的类不可以多继承，但是接口可以多继承。
Java 有自动内存管理垃圾回收机制(GC)，不需要程序员手动释放无用内存。
C ++同时支持方法重载和操作符重载，但是 Java 只支持方法重载（操作符重载增加了复杂性，这与 Java 最初的设计思想不符）。
标识符和关键字
标识符
标识符就是一个名字 --自己创建
关键字
赋予了其特殊的含义，只能用于特定的地方 --Java创建
CONTINUE、BREAK 和 RETURN 的区别
continue
指跳出当前的这一次循环，继续下一次循环。
break
指跳出整个循环体，继续执行循环下面的语句
return
用于跳出所在方法，结束该方法的运行
成员变量与局部变量的区别
语法形式
成员变量是属于类的
成员变量可以被 public,private,static 等修饰符所修饰
局部变量是在代码块或方法中定义的变量或是方法的参数
局部变量不能被访问控制修饰符及 static 所修饰
都能被 final 所修饰
存储方式（内存中的位置不同）
成员变量：在堆中（方法区中的静态区）
局部变量：在栈中
生存时间（生命周期不同）
成员变量：随着对象的创建而存在，随着对象的消失而消失
局部变量：随着方法的调用或者代码块的执行而存在，随着方法的调用完毕或者代码块的执行完毕而消失
默认值（初始值）
成员变量：有默认初始值
局部变量：没有默认初始值，使用之前需要赋值，否则编译器会报错（The local variable xxx may not have been initialized）
静态变量作用
被类的所有实例共享。无论一个类创建了多少个对象，它们都共享同一份静态变量
通常情况下，静态变量会被 final 关键字修饰成为常量
字符型常量和字符串常量的区别
形式
字符常量
单引号引起的一个字符
字符串常量
双引号引起的 0 个或若干个字符
含义
字符常量
相当于一个整型值(ASCII 值)，可以参加表达式运算
字符串常量
代表一个地址值(该字符串在内存中存放位置)