类加载过程（第一次使用该类）

Java是使用 双亲委派模型 来进行类的加载的，所以在描述类加载过程前，我们先看一下它的工作过程：

双亲委托模型的工作过程是：

　　如果一个类加载器（ClassLoader）收到了类加载的请求，它首先不会自己去尝试加载这个类，

　　而是把这个请求委托给父类加载器去完成，每一个层次的类加载器都是如此，因此所有的加载请求最终都应该传送到顶层的

　　启动类加载器中，只有当父类加载器反馈自己无法完成这个加载请求（它的搜索范围中没有找到所需要加载的类）时，

　　子加载器才会尝试自己去加载。

使用双亲委托机制的好处是：

　　能够有效确保一个类的全局唯一性，当程序中出现多个限定名相同的类时，类加载器在执行加载时，始终只会加载其中的某一个类。

1、加载

　　　　 由类加载器负责根据一个类的全限定名来读取此类的二进制字节流到JVM内部，并存储在运行时内存区的方法区，然后将其转换为一个与目标类型对应的java.lang.Class对象实例

2、验证

格式验证：验证是否符合class文件规范

语义验证：检查一个被标记为final的类型是否包含子类；检查一个类中的final方法是否被子类进行重写；确保父类和子类之间没有不兼容的一些方法声明（比如方法签名相同，但方法的返回值不同）

操作验证：在操作数栈中的数据必须进行正确的操作，对常量池中的各种符号引用执行验证（通常在解析阶段执行，检查是否可以通过符号引用中描述的全限定名定位到指定类型上，以及类成员信息的访问修饰符是否允许访问等）

3、准备

为类中的所有静态变量分配内存空间，并为其设置一个初始值（由于还没有产生对象，实例变量不在此操作范围内）

被final修饰的static变量（常量），会直接赋值；

4、解析

将常量池中的符号引用转为直接引用（得到类或者字段、方法在内存中的指针或者偏移量，以便直接调用该方法），这个可以在初始化之后再执行。

解析需要静态绑定的内容。 // 所有不会被重写的方法和域都会被静态绑定

　　以上2、3、4三个阶段又合称为链接阶段，链接阶段要做的是将加载到JVM中的二进制字节流的类数据信息合并到JVM的运行时状态中。

5、初始化（先父后子）

4.1 为静态变量赋值

4.2 执行static代码块

注意：static代码块只有jvm能够调用

　　　如果是多线程需要同时初始化一个类，仅仅只能允许其中一个线程对其执行初始化操作，其余线程必须等待，只有在活动线程执行完对类的初始化操作之后，才会通知正在等待的其他线程。

因为子类存在对父类的依赖，所以类的加载顺序是先加载父类后加载子类，初始化也一样。不过，父类初始化时，子类静态变量的值也有有的，是默认值。

最终，方法区会存储当前类类信息，包括类的静态变量、类初始化代码（定义静态变量时的赋值语句 和 静态初始化代码块）、实例变量定义、实例初始化代码（定义实例变量时的赋值语句实例代码块和构造方法）和实例方法，还有父类的类信息引用。

Java中new一个对象的步骤：

1. 当虚拟机遇到一条new指令时候，首先去检查这个指令的参数是否能 在常量池中能否定位到一个类的符号引用 （即类的带路径全名），并且检查这个符号引用代表的类是否已被加载、解析和初始化过，即验证是否是第一次使用该类。如果没有（不是第一次使用），那必须先执行相应的类加载过程（class.forname()）。

2. 在类加载检查通过后，接下来虚拟机将 为新生的对象分配内存 。对象所需的内存的大小在类加载完成后便可以完全确定，为对象分配空间的任务等同于把一块确定大小的内存从Java堆中划分出来，目前常用的有两种方式，根据使用的垃圾收集器的不同使用不同的分配机制：

　　2.1. 指针碰撞（Bump the Pointer）：假设Java堆的内存是绝对规整的，所有用过的内存都放一边，空闲的内存放在另一边，中间放着一个指针作为分界点的指示器，那所分配内存就仅仅把那个指针向空闲空间那边挪动一段与对象大小相等的距离。

　　2.2. 空闲列表（Free List）：如果Java堆中的内存并不是规整的，已使用的内存和空间的内存是相互交错的，虚拟机必须维护一个空闲列表，记录上哪些内存块是可用的，在分配时候从列表中找到一块足够大的空间划分给对象使用。

3. 内存分配完后，虚拟机需要将分配到的内存空间中的数据类型都 初始化为零值（不包括对象头）；

4. 虚拟机要 对对象头进行必要的设置 ，例如这个对象是哪个类的实例（即所属类）、如何才能找到类的元数据信息、对象的哈希码、对象的GC分代年龄等信息，这些信息都存放在对象的对象头中。

至此，从虚拟机视角来看，一个新的对象已经产生了。但是在Java程序视角来看，执行new操作后会接着执行如下步骤：

5. 调用对象的init()方法 ,根据传入的属性值给对象属性赋值。

6. 在线程 栈中新建对象引用 ，并指向堆中刚刚新建的对象实例。