java下Class.forName的作用是什么，为什么要使用它

Class.forName(xxx.xx.xx) 返回的是一个类

首先你要明白在java里面任何class都要装载在虚拟机上才能运行。这句话就是装载类用的(和new 不一样，要分清楚)。

至于什么时候用，你可以考虑一下这个问题，给你一个字符串变量，它代表一个类的包名和类名，你怎么实例化它？只有你提到的这个方法了，不过要再加一点。

A a = (A)Class.forName("pacage.A").newInstance();

这和你

A a = new A()；

是一样的效果。

关于补充的问题

答案是肯定的，jvm会执行静态代码段，你要记住一个概念，静态代码是和class绑定的，class装载成功就表示执行了你的静态代码了。而且以后不会再走这段静态代码了。

Class.forName(xxx.xx.xx) 返回的是一个类

Class.forName(xxx.xx.xx);的作用是要求JVM查找并加载指定的类，也就是说JVM会执行该类的静态代码段

动态加载和创建Class 对象，比如想根据用户输入的字符串来创建对象

String str = 用户输入的字符串

Class t = Class.forName(str);

t.newInstance();

在初始化一个类，生成一个实例的时候，newInstance()方法和new关键字除了一个是方法，一个是关键字外，最主要有什么区别？它们的区别在于创建对象的方式不一样，前者是使用类加载机制，后者是创建一个新类。那么为什么会有两种创建对象方式？这主要考虑到软件的可伸缩、可扩展和可重用等软件设计思想。

Java中工厂模式经常使用newInstance()方法来创建对象，因此从为什么要使用工厂模式上可以找到具体答案。 例如：

class c = Class.forName(“Example”);

factory = (ExampleInterface)c.newInstance();

其中ExampleInterface是Example的接口，可以写成如下形式：

String className = "Example";

class c = Class.forName(className);

factory = (ExampleInterface)c.newInstance();

进一步可以写成如下形式：

String className = readfromXMlConfig;//从xml 配置文件中获得字符串

class c = Class.forName(className);

factory = (ExampleInterface)c.newInstance();

上面代码已经不存在Example的类名称，它的优点是，无论Example类怎么变化，上述代码不变，甚至可以更换Example的兄弟类Example2 , Example3 , Example4……，只要他们继承ExampleInterface就可以。

从JVM的角度看，我们使用关键字new创建一个类的时候，这个类可以没有被加载。但是使用newInstance()方法的时候，就必须保证：1、这个类已经加载；2、这个类已经连接了。而完成上面两个步骤的正是Class的静态方法forName()所完成的，这个静态方法调用了启动类加载器，即加载java API的那个加载器。

现在可以看出，newInstance()实际上是把new这个方式分解为两步，即首先调用Class加载方法加载某个类，然后实例化。 这样分步的好处是显而易见的。我们可以在调用class的静态加载方法forName时获得更好的灵活性，提供给了一种降耦的手段。

最后用最简单的描述来区分new关键字和newInstance()方法的区别：

newInstance: 弱类型。低效率。只能调用无参构造。

new: 强类型。相对高效。能调用任何public构造。