1.字符串常量池(String Constant Pool):

1.**字面值**形式： JVM会自动根据字符串常量池中字符串的实际情况来决定是否创建新对象 。当以字面值形式创建一个字符串时，总是先检查字符串池是否含存在该对象。若存在，则直接返回池中的实力引用；若不存在，实例化字符串并放入池中。但通过 new 操作符创建的字符串对象不指向字符串池中的任何对象，他存在堆中。（JDK7 字符串常量池也可存在字符串对象的引用，不存在字符串对象）

2.**手动入池**

一个初始为空的字符串池，它由类 String 私有地维护。 intern()函数：

在1.6中，intern的处理是 先判断字符串常量是否在字符串常量池中，如果存在直接返回该常量，如果没有找到，则将该字符串常量加入到字符串常量区，也就是在字符串常量区建立该常量；

在1.7中，intern的处理是 先判断字符串常量是否在字符串常量池中，如果存在直接返回该常量，如果没有找到，说明该字符串常量在堆中，则处理是把堆区该对象的引用加入到字符串常量池中，以后别人拿到的是该字符串常量的引用，实际存在堆中；

```
public class TestString{
    public static void main(String args[]){
        String str1 = "abc";
        String str2 = new String("abc");
        String str3 = s2.intern();

        System.out.println( str1 == str2 );   //false
        System.out.println( str1 == str3 );   //true
    }
```

}

str1引用的是常量池（方法区）的对象；str2引用的是堆中的对象。但是由于“abc”已存在于常量池中，str2.intern()直接返回字符串常量池中的“abc”给str3引用，所以 str1 和 str3 指向同一对象。

1.1:字符串常量池在Java内存区域的哪个位置？

- 在JDK6.0及之前版本，字符串常量池是放在Perm Gen区(也就是方法区的运行时常量池)中；
- 在JDK7.0版本，字符串常量池被移到了堆中了。可能是由于方法区的内存空间太小了。

1.2:字符串常量池是什么？

- 在HotSpot VM里实现的string pool功能的是一个StringTable类，它是一个Hash表，默认值大小长度是1009；这个StringTable在每个HotSpot VM的实例只有一份，被所有的类共享。字符串常量由一个一个字符组成，放在了StringTable上。
- 在JDK6.0中，StringTable的长度是固定的，长度就是1009，因此如果放入String Pool中的String非常多，就会造成hash冲突，导致链表过长，当调用String#intern()时会需要到链表上一个一个找，从而导致性能大幅度下降；
- 在JDK7.0中，StringTable的长度可以通过参数指定：

```
-XX:StringTableSize=66666
```

1.3:字符串常量池里放的是什么？

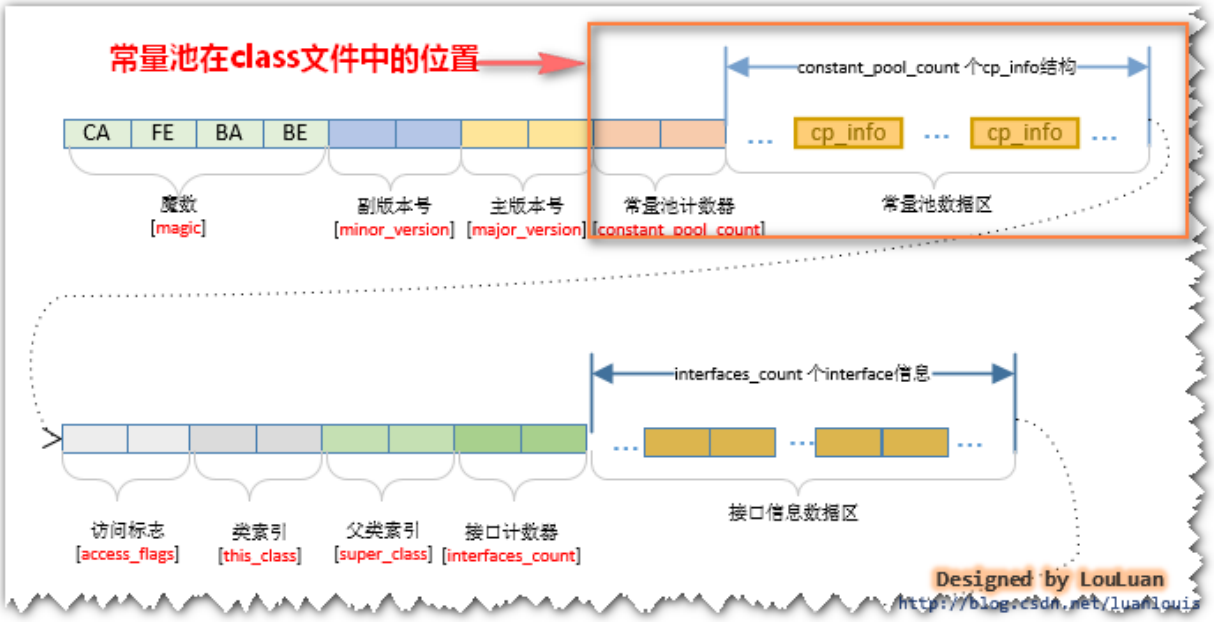
- 在JDK6.0及之前版本中，String Pool里放的都是字符串常量；

- 在JDK7.0中，由于String.intern()发生了改变，因此String Pool中也可 以存放放于堆内的字符串对象的引用。

2.class常量池(Class Constant Pool)：

2.1:class常量池简介：

- 每一个Java类被编译后形成一份class文件存在方法区中；class文件中除了包含类的版本、字段、方法、接口等描述信息外，还有一项信息就是常量池(constant pool table)【此时还没有加载在内存中，在文件中。又称静态常量池】，用于存放源码编译器生成的各种字面量(Literal)和符号引用(Symbolic References)；可以将类的信息的class文件内容的一个框架，里面具体的内容通过常量池来存储。（JDK1.7 移到堆中）



2.2:字面量和符号引用：

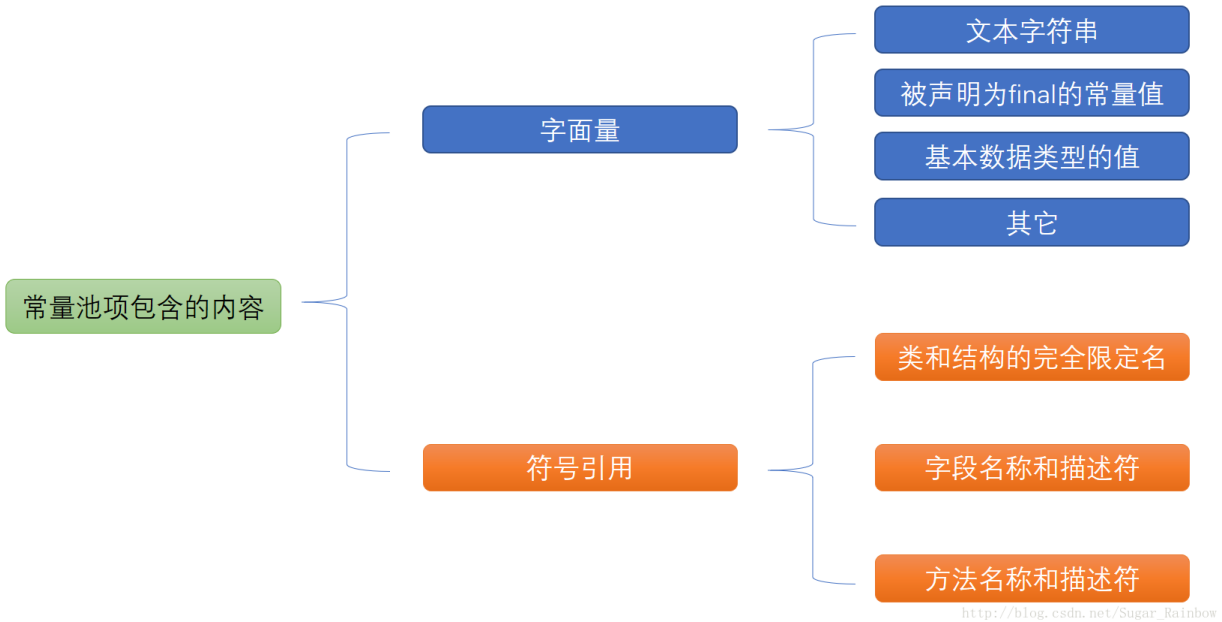
- 字面量包括：

  1. 字符串字面量
  2. 八种基本类型的值
  3. 被声明为final的常量等;

- 符号引用（编译原理概念。）：

  1. 类和方法的全限定名。eg. java/lang/String
  2. 字段的名称和描述符。字段就是类或者接口中声明的变量
  3. 方法的名称和描述符。描述符相当‘参数类型+返回值’

（符号引用其实引用的就是常量池里面的字符串，不是直接存储字符串，而是存储字符串在常量池里的索引。）



3. 运行时常量池(Runtime Constant Pool)：

当编译形成的class文件被加载（类装载到内存，在类的准备阶段）后，jvm会将静态常量池里的内容转移到动态常量池里，在静态常量池的符号引用有一部分是会被转变为直接引用的（类加载的解析阶段）。eg.类的静态static方法或私有private方法，实例构造方法，父类方法（因为这些方法不能被重写其他版本，所以能在加载的时候就可以将符号引用转变为直接引用，而其他的一些方法是在这个方法被第一次调用的时候才会将符号引用转变为直接引用的。）（JDK1.7 移到堆中）

而且动态常量池里的内容是能动态添加的。例如调用String的intern方法就能将string的值添加到String常量池中，这里String常量池是包含在动态常量池里的，但在jdk1.8后，将String常量池放到了堆中。

运行时常量池是全局共享，多个类共用一个运行时常量池。并且class文件中常量池多个相同的字符串在运行时常量池只会存在一份。