Java中的构造方法

1. 构造方法（Constructor）的作用：**完成对象属性的初始化工作**。
2. 构造方法必须与类名一致；
3. 没有具体的返回值，所以不需要指定返回值类型；
4. 在定义一个类时，如果没有声明一个构造方法，那么Java编译器会生成一个默认的**空参数的构造方法**；如果声明了一个构造方法，无论有参数或者无参数，则Java编译器不会再生成构造方法了，空参数的也不会生成；
5. 默认的构造方法是一个没有参数的且方法体为空的构造方法；
6. new后面的小括号表示的意义：如果构造方法不接收参数，则直接就是空小括号；如果构造方法接收参数，则小括号里面的参数应该与构造方法的参数列表的数据类型、参数个数、参数顺序保持一致。
7. 构造方法是创建对象时需要执行的方法。构造方法的作用是：可以给对象初始化（赋初值）。而且创建对象必须通过调用相应的构造方法进行初始化。
8. 一般成员方法和构造方法的区别：
9. 构造方法在对象创建时就会执行，对对象进行初始化；而一般成员方法是在创建对象后，需要此方法功能时才进行调用；
10. 构造方法只是在对象创建时调用一次，而一般成员方法可以调用多次；
11. 构造方法没有返回值，而一般成员方法有返回值。
12. 什么时候需要定义构造方法？

在描述事物时，该事物一旦存在就应当具有一些内容，这些内容就应当定义在构造方法体中。

1. 构造方法的重载（overload）：与一般成员方法的重载相似。
2. 构造方法的内存图解：构造方法同一般成员方法一样，在需要执行时，就会进栈，执行完后就出栈。
3. 构造方法中可以调用此生成对象的成员方法或者成员变量。
4. **在构造方法中可以调用一般的成员方法。**
5. **构造方法不会发生重写**，因为构造方法的方法名始终与所在类的类名一致，所以不会出现子父类的构造方法重名现象，更不可能发生重写了。
6. **构造方法也不会被继承**，而是利用super（\*\*\*\*\*）；进行调用父类的构造方法。
7. 子类的实例化过程：子类中的所有的构造方法在默认情况下，都会访问父类中的空参数的构造方法。
8. **在构造方法中利用this（\*\*\*）；调用自己本类的其他的构造方法，但是必须在构造方法中的第一行；同时利用super（\*\*\*）；调用父类的构造方法时，也必须放在第一行。因此在子类的构造方法中要么选择this（）；要么选择super（\*\*\*）；子类构造方法中没有默认的this（）；要想调用本类的其他构造方法，必须显式地调用；但是super（）；会隐式的添加进来。一旦子类构造方法中出现this（\*\*\*）；语句，编译器便不会再隐式的提供super（）；了。可以保证的是，子类中其他的构造方法中肯定会执行到相应的super（\*\*\*）；语句；因为此构造方法不调用父类构造方法，this（\*\*\*）相应的构造方法里可以存在super（\*\*\*）；从而最终肯定会调用到的父类的构造方法。**
9. **在子类的构造方法中（若无this（\*\*\*）；语句），则有两个隐式语句：super（）；和 return（）；若有this（\*\*\*）；语句，则就只有一个隐式语句return（）；。**
10. **构造方法的权限一般要比此类的权限要相同或者小，因为类是可以访问的，其构造方法才有意义。在单例化模式中，把构造方法私有化（private），就可以禁止外界利用new进行对此类的实例化了。**
11. **Java中的任何一个类都是Object类的子类，要么直接要么间接的继承Object类，所以创建任何一个类，其实都是子类，任何一个类中的构造方法中都有一个隐式语句super（）语句，除了第一句是this（\*\*\*）；之外。**