# Java面试突击

1. 基础语法
   1. 基本数据结构
      1. 哪一种基本数据类型不能和其他基本数据类型进行类型转换？
         1. 布尔类型boolean
   2. 方法
      1. Java中，参数的传递使用值传递还是引用传递？
         1. Java中只有值传递。值传递的含义是将实参的值传递给形参，当参数类型是基本数据类型时，传递的是实参的值，当参数类型是对象时，传递的是对象的引用，但是不能让实参引用新的对象。
      2. 使用方法的重载时，什么情况下会出现编译错误？
         1. 出现歧义调用的时候会出现编译错误。如果一个方法调用有多个可能的匹配，且编译器无法判断哪个方法最匹配，则称为歧义调用。
   3. 递归
      1. 递归的优点和缺点分别有哪些？
         1. 递归的优点是代码简洁且易于理解，缺点是时间和空间的消耗比较大、可能产生重复计算以及可能导致调用栈溢出。
      2. 可以通过什么方式解决递归的缺点？
         1. 解决递归可以使用尾递归，通过加记忆化的方式避免重复计算从而提高效率，以及改用迭代实现。
2. 面向对象
   1. 静态和实例
      1. 使用什么关键字可以将类成员声明为静态？
         1. 关键字static
      2. 从外部调用类成员时，对于静态的类成员和实例的类成员，分别通过什么方式调用？
         1. 从外部调用静态的类成员时，可以通过类名调用，也可以通过方法名调用。从外部调用实例的类成员，则只能通过对象名调用
      3. 从内部访问类成员时，静态方法是否可以访问静态的类成员和实例的类成员？实例方法是否可以访问静态的类成员和实例的类成员？
         1. 静态方法只能访问静态的类成员，不能访问实例的类成员。实例方法既可以访问实例的类成员，也可以访问静态的类成员。
      4. 为什么静态方法不能访问实例的类成员呢？
         1. 实例的类成员依赖于具体对象（实例），而静态方法不依赖于任何实例，因此不存在静态方法直接或间接地访问实例或实例的类成员的情况。
   2. 初始化块
      1. 静态初始化块和非静态初始化块的执行顺序是什么，以及分别会执行多少次？
         1. 静态初始化块的优先级最高，会最先执行，会在非静态初始化块之前执行。静态初始化块只在类第一次被加载时执行，非静态初始化块在每次创建对象时执行一次，因此创建了多少个对象，就会执行多少次非静态初始化块。
      2. 静态初始化块和main方法哪个先被执行？
         1. 静态初始化块会在类第一次被加载时最先执行，因此静态初始化块的执行在main方法之前。
   3. this关键字
      1. 关键字this的含义是什么？
         1. 关键字this代表当前对象的引用。当前对象指的是调用类中的属性或方法的对象
      2. 关键字this在方法中以及在构造方法中分别表示什么？
         1. 关键字this可以用于引用对象的属性，在方法和构造方法中都可以通过关键字this调用其他构造方法。
      3. 关键字this是否可以在静态方法中使用？说明理由
         1. 关键字this不可以在静态方法中使用。关键字this代表的是对象的引用，而静态方法不依赖于类的具体对象。
   4. 可见修饰符和数据域封装
      1. 如果类成员不加任何可见性修饰符，则可以从哪里访问该类成员？
         1. 不加任何可见性修饰符称为默认default修饰符，此时类成员可以在同一个包里的任何类中访问。
      2. 可见性修饰符public和private分别表示可以从哪里访问类成员？
         1. 可见性修饰符public表示类成员可以在任何类中访问，可见性修饰符private表示类成员只能从自身所在的类中访问。
      3. 什么是数据域封装？为什么要使用数据域封装？
         1. 数据域封装指的是对数据域使用private修饰符，将数据域声明为私有域。使用数据域封装的目的是避免从类的外部直接修改数据域的值。
      4. 在使用数据域封装之后，应如何读取和修改私有数据域的值？
         1. 可以编写get方法读取数据域的值，编写set方法修改数据域的值
   5. 字符串
      1. String类型的值是否可变？对String类型的值进行修改操作是否修改了String的内容？
         1. 由于String的源码中，存储字符串内容的数组使用关键词final修饰，因此String类型变量不能再引用其他的对象。对String类型的值进行修改，没有修改String的内容，而是创建了新的String对象。
      2. StringBuilder和StringBuilder哪一种类型的效率更高？两种类型分别适用于什么样的场合？
         1. StringBuilder的效率更高，因为StringBuffer对方法加了同步锁，而StringBuilder没有对方法加同步锁。虽然StringBuilder的效率更高，但是因为没有对方法加同步锁，因此在多线程环境下不保证线程安全，只适合单线程环境，而StringBuffer可以用于多线程环境。