1. 语法糖
2. 概述

语法糖是计算机语言中添加的某种语法，这种语法对于语言的编译结果和功能没有实际影响，但是却能够方便开发者的开发。Java中的语法糖有泛型，变长参数，自动装箱拆箱等等。

1. 泛型
2. 概述

泛型的本质是参数化类型，即可以将操作的数据类型指定为方法签名中的一种特殊参数，这种参数类型可以用在类，接口和方法的创建中，分别构成泛型类，泛型接口和泛型方法。泛型让程序员能够针对泛化的数据类型编写相同的算法，极大地增强了抽象能力。

1. Java的泛型实现方式

Java的泛型实现方式为类型擦除式泛型，而C#选择的是具现化式泛型，即C#的泛型无论在程序源码，编译后的中间语言表示中，或是运行期都是真实存在的；而Java的泛型只在程序源码中存在，在编译后的字节码文件中，全部泛型都被替换为原来的裸类型，并且在相应的地方插入了强制转型代码，因此对于运行期的Java语言来说，ArrayList<int>和ArrayList<String>是同一个类型。

1. 类型擦除

为了让之前编译的Class文件依然能够在引入泛型后继续运行，要么添加一套泛型化的新类型，要么将已有的类型泛型化，Java选择了后者，如ArrayList，原地泛型化为ArrayList<T>，且要保证以前直接用ArrayList的代码在泛型新版本里面还能继续使用这个容器，就必须让所有泛型化的实例类型如ArrayList<Integer>自动成为ArrayList的子类型。否则类型转换就是不安全的，这样就引入了裸类型的概念，裸类型应被视为所有该类型泛型化实例的共同父类型，这有这样，子类到父类的转换才是安全的。

裸类型是通过在编译时直接将ArrayList<Integer>还原为ArrayList，只在元素访问，修改时自动插入一些强制类型转换和检查指令实现的，但是这样又会出现类型转换的问题，如int,long类型和Object之间的强制转换是不支持的，但是的Java给出的解决方法是不支持原生类型的泛型，直接使用ArrayList<Integer>, ArrayList<Long>，遇到原生类型时自动装箱，拆箱。这种解决方式导致了无数的拆箱装箱，使得Java泛型性能很差。

同时，运行期无法获取到泛型类型信息，也让代码变得相当啰嗦，当泛型被擦除后，拥有相同方法签名的方法是无法在一起编译的，因为无法共存于同一个Class文件中，如果我们为这些方法添加了不同的返回值后，偏移可以通过了。因为返回值并不包含在方法的特征签名中，所以返回值不参与重载选择，但是在Class文件格式中，只要描述符不是完全一致的两个方法就可以共存，而描述符包含了返回值，因此不同的返回值可以让编译通过。

Signature属性的作用就是存储一个方法在字节码层面的特征签名，这个属性中保存的参数类型并不是原生类型，而是包括了参数化类型的信息，为了解决伴随泛型而来的参数类型的识别问题。从Signature属性可以看出，擦除法所谓的擦除仅仅是将方法的Code属性中的字节码进行擦除，实际上元数据还是保留了泛型信息，编码时可以通过反射取得参数化类型。

1. 自动装箱，拆箱与遍历循环

自动装箱，拆箱在编译之后被转换成了对应的包装和还原方法，如

List<Integer> list = Arrays.asList(1, 2);

转换为了：

List list = Arrays.asList(new Integer[]{Integer.valueOf(1), Integer.valueOf(2)});

而遍历循环则是将代码还原为了迭代器的实现，这也是遍历循环需要被遍历的类实现Iterable接口的原因：

for(int I : list){ sum += I;}

转换为了：

for(Iterator localIterator = list.iterator(); localIterator.hashNext(); ){

int I = ((Integer)localIterator.next()).intValue();

sum += I;

}

1. 条件编译

C，C++使用预处理器来完成条件编译，目的是解决编译时的代码依赖关系，Java中没有使用预处理器，因为Java的编译方式是将所有编译单元的语法树顶级节点输入到待处理列表后再进行编译，因此每一个文件之间都能够提供符号信息，就无须用到预处理器。

Java中当然也可以进行条件编译，方法就是使用条件为常量的if语句，它在编译阶段就会被运行，生成的字节码也会包含该语句，如下所示：

if(true){

System.out.println(“123”);

}

根据boolean值，编译器会将分支中不成立的代码块消除掉，这一工作将会在编译器解除语法糖阶段完成。由于条件编译只能在if语句中实现，因此只能实现语句块级别的条件编译，没办法实现根据条件调整整个Java类的结构。