1. enum实例对象大小写严格区分,不能继承(默认继承Enum类),可以实现接口,没有实例化对象也需要在行首加上分号

枚举实现接口，枚举中的对象自动向上转型变为接口对象

1. switch支持枚举
2. 可变参数

类型...参数名称

一个方法有且只能有一个可变参数，并且放在方法的最后一个参数。可变参数的本质还是数组

1. foreach循环：用于数组与类集的简单输出，如需对数组有修改，还是需要用for循环。
2. 静态导入：import static 包名.类名.\*;//导入包内所有的静态方法
3. 泛型：只允许接受类，所有基本类型必须使用包装类。

①泛型类：指在类定义时不会设置类中的属性或方法中参数的具体类型（Object），而是在类使用的时候再定义。

优点：不需强转，需要什么类型，使用时定义什么类型即可。若类型不匹配，编译时就会报错，避免了运行时异常。（因为Object类需要强转或向下转型，若没有做到，只能在运行时报出异常，非常不安全）

class 类名<T>{

private T num;}

T被称为类型参数，用于指代任何类型。

T：代表一般类

E：代表element 常用于泛型类中的属性

K、V：键值对，用于Map集合:

S:代表subtype，用于表示子类

若类中的成员变量需要不同的类型，则在类定义泛型时定义多个类型参数，用逗号分隔

使用时前写后省Person<String> person=new<> Person();

②泛型方法：

public <T> void method(T t){

}

<T>是个占位符，表示其是泛型方法，与返回值无关

当泛型类与泛型方法共存时，泛型类中的类型参数与泛型方法中的类型参数没有关系，泛型方法始终以自己定义的类型参数为准。

规范起见：泛型方法类型参数与泛型类的类型参数不能同名。

通配符:含有泛型的类作为参数时，需指定其类型，因此就可能会出现多个方法重载，因此

（1）采用通配符 <?> 在方法参数中使用，表示参数可以接收任意类型的参数。

使用通配符的类参数，只能取得类中数据，不能修改数据，因为类型不确定，无法设置确定值。

（2）采用通配符<? extends 类名>，表示设置泛型的类上限，即只能接收此类及其子类，如 ？ extends Number 表示泛型必须是Number及其子类

只能取得类中属性，不能修改值（发生父类到子类的向下转型，需强转，由于子类不确定，因此无法转型）

（3）<？super 类名>表示设置泛型的类下限，只能接收此类及其父类，如<？ super String>表示只能接收String 以及其父类Object.

可以设置属性值，因为子类到父类是会自动向上转型

③泛型接口

interface 接口名<T>

子类实现接口可以保留泛型，继续成为泛型类

class 子类 <T> implements 接口名<T>{}

子类实现接口确定好类型

class 子类 implements 接口名<T>{}

1. （面）类型擦除（语法糖）：仅存在源码阶段，编译后就消失不见。如泛型，自动拆装箱、String类中的+

泛型信息仅存在于代码编译阶段，进入JVM之前，与泛型相关信息会被擦除掉，专业术语：类型擦除

换句话说：泛型与普通类在Java虚拟机中没有任何区别

泛型类进入JVM之前会进行类型擦除，之前泛型类的类型参数若没有指定上限，会被擦除为Object类型。如果指定上限，则类型参数会被擦除为相应类型上限。

JDK1.8接口的新特性

1.使用default定义普通方法，需要通过接口对象调用

2.使用static定义的静态方法，直接使用接口名调用