1. **基本语法：**

1.标识符（数字，下划线，字母，￥），类和接口大驼峰，变量和方法小驼峰，常量大写和下划线，包小写

变量：分为属性（也叫成员变量），和局部变量（包括形参）

局部变量：放在方法，构造器，代码块

2.数据类型与堆栈

基本数据类型：

整数：byte，short，int，long 分别1248字节，long的结尾+L

小数：float，double，4,8字节，float结尾+F

字符：char，2个字节

布尔类型：boolean，值为True和Flase

引用数据类型：

数组：int[] a=new int[5]; 或者 int[] a=new int[]{1,2,3};，多维int[][] a

对象：String a=new String();

对象数组：Sting[] a=new String[5];

堆栈方法区：

栈放置局部变量及引用数据类型的变量名，堆放置引用数据类型new的数据

方法区放类的加载信息（类的缓存），常量，静态域

特别注意：当对象没有new在堆空间，而被已new的对象赋值时，赋值的是地址

如，

Student a=new Student();

Student b=a;

Student c=new Student();

c=a;

其中b赋了a的地址，所以b与a指向同一个对象，对b修改会影响a

c因为new了，所以c赋的是a的值，所以c和a是相互独立的，互不影响

3.类型转换与运算符

类型转换：自动类型提升和强制类型转换。

强制类型转换：类要强转类必须是继承关系

自动类型提升，小转大，比int小转int

运算符：

与C语言不同，&和|在连接数值是按位运算，连接布尔值是逻辑运算

由于自动类型提升，建议使用a++而不使用a=a+1

三目运算符： 表达式0 ? 表达式1 : 表达式2 表达式0的结果为True执行表达式1 Flase执行表达式2，表达式1和表达式2类型要一样，不一样也会自动类型提升

4.流程控制：顺序，分支，循环

**二、面向对象**

0.类的生存周期：

类的加载（编译完成就加载）：类加载的方法区，静态的成员放到静态域

创建对象：将非静态的成员放在堆中

对象销毁：非静态成员销毁

类的销毁（关闭JVM）：静态成员销毁

1.类和类的成员，关键字：

类：匿名对象，javabean，UML类图，java设计模式（23种）

UML类图：实线下划线是构造器，+public，-private，实线箭头指向父类

匿名对象（new String()，由于没有名字，只能用一次），可以new 类名().方法名()调用方法

Javabean（类本身和构造器是public，属性private，同时提供公开的set，get方法）

类的成员：属性，方法，构造器，代码块，内部类

方法：

重载（方法名相同，形参列表不同就构成重载，包括顺序不同，与权限和返回类型无关）

可变形参（如String … a 只能有一个且放在最后 类似于传递数组String[] a）

参数传递：只有值传递

构造器：

权限与类同步，可构成重载，作用是创建对象是做初始化操作

若不写任何构造器则会默认提供一个空参构造器，若写了重载的构造器则不提供，需要自己定义一个空参构造器（尽量定义一个）

代码块：{},static{}

也叫初始化块，用来初始化

静态代码块在加载类时就执行，非静态的代码块在创建对象时就执行

可以有多个代码块，但一般只写一个

内部类：要与作为属性的类区分开来，相对的，包含内部类的类叫外部类，内部类与类一样，可以被final（不能被继承），abstract修饰，与类不同。内部类还可以被static修饰，权限修饰符全部能使用，类有的成员内部类都能有

成员内部类：分为静态和非静态

1. 创建对象：

静态：外部类.内部类 a=new 外部类.内部类();

非静态：

外部类 a=new 外部类();

外部类.内部类 b=a.new 内部类();

1. 调用外部类成员：省略了 外部类.this.因为内部类是外部类的成员，所以能调用外部类private的成员

局部内部类：放在方法，构造器，代码块

1. 最常用的就是放在方法中，方法中创建实现类实现接口，返回实现类的对象
2. 局部内部类的方法中的局部变量必须是final的，final可省略但还是常量

关键字：import，packet，this，super，instanceof，static，final

this.属性/方法，调用本类属性方法

this(参数)，调用本类构造器，可以传参，每个构造器中只能有一个this(参数)且放在首行

super.属性/方法，子类调用父类的属性方法，子类中的属性可以与父类的属性同名，此时用suoer区分，调用方法则可以调用未被重写的父类方法

super(参数) 调用父类构造器，可以传参，每个子类构造器中只能有一个this(参数)或super(参数)且放在首行，若this和super都没有或没有定义构造器用默认的空参构造器，则会默认有一个空参的super()，若此时父类没有空参的构造器则会报错

（当父类定义了其他构造器，默认的空参构造器就不会提供，开发时尽量写一个空参构造器）

static:可以修饰变量，方法，代码块，内部类，被static修饰的成员不再是每个对象各有一份，而是一个类的所有对象共用一个，也就是说，修改其中一个静态成员会影响所有对象（因为在内崔空间中，只有一个这个成员在方法区的静态域中）

静态的属性/方法也叫类变量/类方法，非静态属性/方法也叫实例变量/实例方法

静态成员在类加载时就在内存中，早于对象创建，因此，可以不创建对象直接用 类名.成员调用 ，静态的成员内部也只能帆静态的成员

Static应用：

属性：

1. ：若一个类的所有对象有共同不变的属性，就可以用static修饰该属性，节省内存空间，也方便修改，常量有这种特性，一般推荐用static
2. ：自动设置ID，只需要有一个静态的计数器变量，在构造器中++并赋值，就可每次创建对象时设置连续的ID

方法：

1. ：对于封装性，私有的属性的get，set方法一般也是static的
2. ：工具类的方法一般是static，方便调用，不需要创建对象

main方法是static，只能调用主类中main外的静态属性/方法，非静态的只能创建对象再调用

（main的形参是跟控制台交互的，功能类似于scaner，main函数只能有一个但可以放在任何类中）

final：修饰类（不能被继承，但是属性还是可以修改），修饰方法（不能被重写），

修饰变量（不能修改，未赋值的final属性可以用构造器或代码块赋值，赋值后也不能修改）

修饰内部类与类一样，要与作为属性的类区分

还可以修饰类、变量形参

注意：类也可以成为属性，此时作为属性的类不能修改

static final属性也叫全局常量

2.面向对象三大特性，封装，继承，多态

封装：提供四种权限修饰（private，缺省，protected，public）

类的成员都可以用四种权限修饰，类本身只能缺省或public

private：只能在本类调用，就算子类继承了，也不能调用

缺省：不加任何关键字，只能在本包中调用，就算不同包的子类继承了，也不能调用，方法也不能重写

protected：同一个项目不同包的子类可调用

public：同一个项目均可调用

封装性的体现：

1. 对属性私有化，同时提供公开的set，get方法给外部调用，用于设置和访问属性
2. 私有化方法，只在本类调用

继承：class 子类（派生类） extends 父类（超类/基类）{}，java中所有类都继承于object类

减少代码冗余；扩展性好；为多态提供前提

子类继承父类所有属性和方法（与权限无关，但权限会影响子类访问），构造器用super()调用父类的

子类继承父类属性和方法的原理是调用了父类的空参构造器，虽然调用了多个构造器，但也只是在堆中new了一个子类对象

一个类可以有多个子类，但一个类只能有一个父类（单继承性，不像接口和C++的类可以多重继承）；父类称为直接父类，父类的父类称为间接父类；

重写：子类重写方法覆盖继承过来的方法

1. 方法名和形参列表与被重写方法相同
2. Void或者基本数据类型不变，返回类型是对象，则重写的方法返回类型可以是他的子类
3. Private的方法不能重写，重写权限不小于被重写权限
4. 重写的异常类型不大于被重写异常类型
5. 重写和被重写要么都是static，要么都是非static的，staic的不构成重写

多态：父类 对象名=new 子类 （父类引用指向子类对象/子类对象赋值给父类引用）

需要继承作为前提，也要有重写的方法否则多态就无意义了

多态性是运行时行为，多态性只适用于方法不适用于属性，调用的是父类的属性。

虚拟方法调用（虚方法）：①编译时寻找父类的方法；②之后运行才有在子类找重写的方法

（也就是说调用的方法父类必须有否则报错）

用途：方法中的形参是父类，要调用子类时，无需重载各个子类形参的方法，只需用相应子类的匿名对象赋值给形参即可（如Person与Student是父子类关系，有一方法

void func(Person p) 但想传入Student对象，此时无需重载方法，只需利用多态传入new Student()即可 ）

若要调用子类属性：

向下转型：用强制类型转换 （子类）对象名，注意强转括号的类型必须是相应的子类（多态性相当于向上转型）

如：

Person p1=new Man();

Perspn p2=new Women();

//注意p1只能用Man

Man m1=(Man)p1;

instanceof关键字：判断是否是相应的子类，防止错误，在向下转型是一般要加上判断

if( p1 instanceof Man) //返回True，如果是 p1 instanceof Women 则返回Flase

instanceof后面可跟直接父类或间接父类甚至哦object都返回True

3.三种类（Object类，包装类，抽象类）与接口

Object类：是所有java类的根父类，没有属性，有一个空参无方法体的构造器，有所有类通用的方法，它的子类一般会重写它的方法

包装类：将基本数据类型包装成类称为包装类，使其具有面向对象的特征

类名与基本数据类型一样只是首字母大写，如Float类，Double类，int和char特殊，分别是Integer和character，所有数值型包装类都有共同的父类Number类

基本数据类型、包装类之间的转换（其实就是设置和调用属性）

1. 基本数据类型转换为包装类：（设置属性值）

调用包装类的构造器，传参即可，可以传字符串（由重载的构造器处理）

Integer a=new Integer(123); //也可传递”123”

Boolean比较特殊，只有传递True，true，”True”，”true”，才传入了True，否则都是Flase

1. 装类转换为基本数据类型（就是调用属性）调用包装类的方法

int b=a.intValue(); // .xxxValue xxx是相应的类型

1. 自动装箱与自动拆箱

编译器会自动识别将基本数据类型转换成包装类（装箱），反之也是（拆箱）

如：func(Object obj)可以直接传入基本数据类型 func(12.3)，先将12.3自动装箱成Double类，又因为Double是Object的子类，利用多态性即可传入

如：cla是一个包装类，da是一个基本数据类型

Int a=cla; //自动拆箱

Integer b=da; //自动装箱

都是可以赋值的

有了自动装箱和自动拆箱之后，基本数据类型/包装类与String类之间的转换就是一样的了

转换成String类：

1. 连接法：String str=123+””; 即可
2. 调用String类的方法：String.valueOf(参数)

String类转换成基本数据类型/包装类：

调用包装类的方法：parseXxx(String str) Xxx是相应的类型

如：

int i=Integer.parseInt(“123”); //转成基本数据类型

Double d=Double.parseDouble(“12.3”); //转成包装类

转成boolean只有”True”,”true”才是True否则都是Flase

抽象类：abstract修饰，abstract能修饰类和方法，内部类

抽象类不能创建对象，抽象方法不能被调用也没有方法体并且以分号结尾

有抽象方法的类一定是抽象类，反之不然

只有重写了所有抽象方法的子类才能实例化，否则加上abstract变成抽象类

应用：当父类不知道要定义具体的什么方法，就可以定义抽象类，抽象方法由子类根据需要重写，根据这种特性抽象类不能用final，抽象方法不能用final，static，private

抽象类的匿名子类：

抽象类 className=new 抽象类()

{

重写的抽象方法1

……………..

重写的抽象方法n

};

匿名子类+匿名对象：

Func(new 抽象类()

{

重写的抽象方法1

……………..

重写的抽象方法n

})

接口：interface 接口名{}，接口是public的

JDK7之前：

接口的成员只有全局常量（static final）和抽象方法，都是pubilic的

接口的public，public static final和public abstract可以省略，但还是public的接口，全局常量和抽象方法

类实现接口：

class 类名 implements 接口名{}

该类必须覆盖（就是重写，也叫实现）所有抽象方法才能创建对象，此时称为实现类，否则是抽象类加上abstrct

类多实现接口（相当于多继承）：

class 类名 implements 接口1,接口2,接口3{}

class 子类 extends 父类 implements 接口1,接口2,接口3{}

类与类之间继承，类实现接口，接口与接口之间也叫继承且支持多继承：

interface 子接口名 extends 父接口名{}

interface 子接口名 extends 父接口1,父接口2{}

多实现和多继承时，若有相同的抽象方法，则只视为一个，实现一个即可

类与接口之间实现、继承时，若有同名的属性，则必须用this super 接口名.属性 区分

抽象类和接口区别：都不能创建对象。但抽象类有构造器

接口支持多态，类的多态是子类，而接口是实现类

匿名对象，匿名实现类，以及连着一起用都和抽象类一样

Java8接口新特性：

接口还可以定义静态方法和默认方法，都是public的，且public可省略，省略还是共有的

静态方法实现类无法调用，只能接口调用，逐渐替代工具类；

默认方法用default修饰，是普通的非静态方法，实现类可以调用可以重写，以下两种情况，重写之后调用的都是重写的方法，重写之后，也能用 super. 接口名.super. 调用父类/接口的方法:

1. 类继承的，与实现的接口有同名同参的方法，则该类调用时，是父类的方法（方法的类优先原则）
2. 类多实现接口时，不同接口有同名同参的方法，则该类不能调用，只能用 接口名.super.方法名() 调用

当继承与多实现同时存在时，也是调用父类的

**三：异常处理**