Java面向对象(初级)

Java语言基础
#一、类与对象

类class

类是对象的模板，包含对象的属性(成员变量)和方法(函数)，但是并没有实际的值
属性（也称为成员变量）：用来描述对象的特征。
方法（也称为成员方法）：用来描述对象的行为。
类属性的定义
类的访问修饰符

1. public: 表示可以被任何类中访问
2. private: 表示只能在当前类中访问
3. protected: 表示可以再同一个包或者子类中使用
4. 无修饰符：默认表示可以在同一个包中访问

类的基本结构

```
访问权限 class 类名 {//类声明
访问权限 数据类型 变量名;//成员变量声明
访问权限 构造方法名(形参){初始化参数;}//构造方法 该方法名必须与类名一致
访问权限 返回类型 方法名(形参){对象行为;}//此处的方法是对对象的行为进行描述
}
}
```

对象object

对象是类的实例，可以通过类来创建多个对象，每个对象都有自己的属性值和方法
对象的实例化创建使用new来实现(快捷键：new 类名().var+enter)
对象的定义：类名称 对象名称=new 类名称();//new 类名称()所创建的对象空间才是真正的对象，对象名称是对其的引用
方法的调用：对象.方法(实参列表)
对象在内存中存在形式：

1. 栈：一般存放基本数据类型
2. 堆：存放对象
3. 方法区：常量池(常量：比如字符串)，类加载信息
   创建对象流程
4. 加载类信息，只加载一次

5. 在堆中分配空间
6. 完成对象初始化首先进行默认初始化，再进行显式初始化，最后进行构造器初始化
7. 将堆中的地址返回给对象名，对象名是对对象的引用

方法method

构造方法的时候方法的名称必须与类名相同，并且没有返回值
类的方法的定义

```
返回类型 方法名(参数列表) {
    // 方法体
    // 执行任务
    return 返回值; // 如果返回类型不是 void
}
```

在Java中，所有的操作都在类中定义，这些操作都是方法，因此方法就充当其他语言中函数的作用

1. 一个方法可以有任意个参数，参数类型可以是任意类型，可以是基本类型或者引用类型；
2. 用带参数的列表时候传参必须使用相同类型或者兼容类型的参数；
3. 方法定义的参数称为形式参数，调用时候的参数是实际参数，形参与实参必须类型一致或兼容，个数顺序一致；
4. 方法不能嵌套定义。
   方法的调用

```
对象名.方法名(参数列表);
```

1. 如果需要返回数据则需要使用return语句返回需要的值；使用方法返回值的时候注意方法返回的类型与接收的类型要一致
2. 同一个类的方法中调用其他方法可以直接使用
3. 跨类调用需要通过对象名调用(这里与访问修饰符有关)
4. 方法中基本数据类型传递的是值，形参的改变并不影响实际参数的值的变化
5. 引用类型例如数组、对象，传递的是地址，形参与实际参数共同存一个地址，因此形参指向数组的改变会引起原来实参的改变；形式参数中置空会使指针为空，形式参数中将其new重新定义一个新的对象，其开辟新的空间，指向新的对象并且不返回
6. 在方法中新定义的变量为局部变量，使用后就会立马释放

方法递归调用recursion

递归的重要规则：

1. 执行一个方法的时候就会创建一个独立的栈空间
2. 方法的局部变量是独立的，不会相互影响

3. 方法中使用的是引用类型的变量则不会相互影响
4. 递归必须明确递归退出的边界，否则会出现无限递归
5. 递归执行完毕后或者执行return后就会返回，谁调用就返回给谁

```
public class method05{
    public static void main(String[] args){
    T t1=new T();
    int num=t1.factorial(6);
    System.out.println(num);
    }
}
class T{
public int factorial(int n){
    if(n==1){
    return 1;
    }
    else{
    return factorial(n-1)*n;
    }
}
}
```

方法重载overload

java中允许同一个类中，多个同名方法的存在，但要求形参列表不一致。方法的重载减轻了起名和记名字的麻烦。
例如System类中的PrintSteam字段
注意事项

1. 方法名必须相同
2. 参数列表必须不同，参数个数，参数类型，参数顺序至少有一个不同，参数名字无要求，只有参数名字不一样不构成重载
3. 返回类型无要求，并不是构成方法重载的条件

```
public class overlord02{
    public static void main(String[] args){
    int m=1000;
    int n=9999;
    double M=202020;
    double N=2929929;
    double k=10010;
    M Method=new M();
    System.out.println(Method.max(m,n));
```

```
    System.out.println(Method.max(M,N));
    System.out.println(Method.max(M,N,k));
    }
}
class M{
    int max(int m,int n){
    return m>n?m:n;
    }
    double max(double m,double n){
    return m>n?m:n;
    }
    double max(double m,double n,double k){
    return m>n?m:n;
    }
}
```

java允许将同一个类中同名同功能但是参数个数不同的参数，封装成一个方法，可以通过可变参数实现。

```
//可变参数格式
数据类型...数组名
```

1. 可变参数最后得到的是一个数组，因此可变参数可以直接接收一个数组。
2. 同时可变参数与普通参数放在一起形成形参列表的时候必须保证可变参数位于最后
3. 一个形参列表最多只有一个可变参数

作用域

1. 全局变量可以不赋值直接使用，因为有默认值，全局变量可以被本类或者其他类使用
2. 局部变量必须赋值才能使用，因为没有默认值，局部变量只能在本类中使用
3. 属性和局部变量可以重名，使用时候遵循就近原则
4. 在同一个作用域或者成员方法中，两个局部变量不能重名，否则会重复定义
5. 属性的生命周期比较长，伴随的对象创建与消失，局部变量生命周期比较短，伴随代码块的执行而创建，伴随代码块的结束而消失
6. 全局变量/属性可以加修饰符，局部变量不可以加修饰符

构造方法/构造器constructor

一个对象创建好之后，再给它的属性进行赋值，这时候就可以使用构造器。对新对象的初始化(alt+insert)

```
修饰符 方法名(形参列表){
属性名=形参名;
}
```

构造器的方法名必须与类名字一样
构造器没有返回值，不需要写void
形参赋值给属性
new一个对象的时候直接通过构造器指定名字和年龄，完成对象的属性初始化，并不是创建对象
一个类可以多个构造器，即使用重载

关键字this

this实际是对象的隐藏属性，指向对象本身，this就代表当前对象。哪个对象使用就是代表哪个对象

```
this.属性//为当前所在对象的属性
```

1. this可以访问本类的属性，方法，构造器
2. 访问成员方法this.方法名(参数列表)
3. 在构造器中只能在构造器中使用，在构造器中访问另外一个构造器。对this的调用必须在构造器的第一个语句
4. this只能在类定义的方法使用

匿名对象

匿名对象创建使用后直接销毁，不能继续使用
匿名对象的作用是为了调用需要使用的方法，临时建立一个对象，因此匿名对象只能使用一次

```
new 类名().方法名();
```