一、类与对象

面向对象语言

面向对象语言具有面向过程语言缺少的一个最本质的概念，那就是“对象”，面向对象的语言的最核心的内容就是“对象”，在现实生活中，“行为”往往由某个个体产生，既被某个对象所拥有。Java就是这样一门语言，java中所有的”行为“被Java中某一个对象所拥有，对象调用方法产生”行为“，这种模式更加符合人的思维模式，程序员通过Java语言能够编出易维护、拓展、和复用的代码。面向对象变成主要有以下三个特性：

1.封装性

面向对象的核心思想之一就是将数据和对数据的操作封装在一起。通过抽象，从具体的实例中抽取出某一类事物所具有的一般的”性质“和“行为”，形成Java中类的概念。通过类这个模板，可以创建出许多对象，这些对象的变量体现出对象拥有的数据(属性)，对象调用方法体现对象的行为（对数据的操作）。

2.继承

继承是面向对象语言的一种编程模式，Java中的子类可以继承父类的属性（变量）和方法，即父类的数据和对数据的操作。同时子类也可以新增自己的属性和方法，也可以通过重写（overwrite）隐藏从父类继承的方法和属性。

3.多态

Java中的多态可以分为两种，一种发生在同一个类中：同一个类中的方法可以具有相同的名字，但方法中的参数要不同，可以根据传入参数的不同调用不同的方法，称为重载（overload）。另一种是发生在继承的时候：子类继承父类的某个方法之后，可以重写这个方法，不同的子类可以进行不同的重写。

类(class)

类是Java中的一种”数据类型“，由一个类所声明的变量叫做对象变量。类的目的是抽象出一类事物所共有的属性和行为，并用一定的语法格式来描述所抽象的出的属性和行为。抽象的关键是抓住事物的属性和行为，既数据和对数据所进行的操作，这些在类中通过类的变量和类的方法来实现。在类体中包含两部分：

1.变量的声明：体现了对象的属性

2.方法的定义：体现了对象的行为

注意：在一个类中也可以创建一个内部类，但是不可以在方法体之外出现Java语句，只能有变量的声明，声明变量时可以赋初值。变量可以是Java中的基础数据类型和引用数据类型（数组，对象，接口）。下面这样的定义是错误的：

```
public class People {
    int a;
```

```
    int b;
    a = 0;  // 这是Java语句，对成员变量的操作不能出现在方法体和内部类之外
    b = 1;//但是可以 int a = 0;声明时赋初值
    int add(a,b) {
        return a+b;
    }
}
```

类中的变量

在类体中声明的变量叫做成员变量，在方法体中声明的变量叫做局部变量。下面是局部变量与成员变量的区别：

1. 成员变量：成员变量的在整个类中都有效，与变量的声明的位置没有关系，但是提倡在定义一个类时，先定义成员变量再定义方法。成语变量如果在声明时没有赋予初值，Java会自动赋予一个默认值。
2. 局部变量：局部变量的的有效范围就是在声明它的位置之后才有效，在它之前的Java语句都不能使用这个变量,也就是说局部变量的有效范围与其声明的位置有关。但是方法中的参数在整个方法中都有效。局部变量Java不会赋予默认值，所以在使用局部变量之前要赋予一个值。

方法

方法的返回值类型可以是任何一种数据类型，当方法的返回类型是void时，方法就不需要返回值，方法中的参数也可以时任何一种数据类型，在方法的内部就可以对成员变量、局部变量、传入的参数进行操作。

注意：如果局部变量与成员变量具有相同的名字，那么在这个方法中，成员变量被暂时隐藏。如果想使用成员变量就需要使用this关键字。

```
public class People {
    int x = 1;
    int y = 2;
    int add() {
        int x = 10;
         return(x + y);//结果是12
    }
    int add1() {
        return (this.x+y);//结果是3
```

```
    }
}
```

类的构造方法

构造方法是类中的一种特殊的方法，在使用类来创建一个对象时需要使用它的构造方法，构造方法没有返回值类型（void也是一种类型），同时方法名必须要与类名一致，Java允许一个类中存在多个构造方法但这些构造方法的参数必须要不同，既参数个数不同，或者参数个数相同参数类型不同。如果一个类中没有编写构造方法，Java会提供一个默认的构造方法，既　类名() {}，没有参数。，方法体中没有Java语句。如果编写了构造方法就不提供默认的构造方法，这在有些时候会引起一些错误（在用子类构造方法创建一个子类对象时，总会先调用父类的某个构造方法，如果子类没有指定调用父类的哪一个构造方法就会调用父类默认的构造方法，但是父类如果编写了构造方法父类就没有默认的构造方法，此时就会出现错误）。

```
public class People {
    int a = 1;
    int b = 2;
    int c;
    People(int x , int y) {            //这个一个构造方法
        c = x + y;
    }
    People(int x; ) {                  //这也是一个构造方法
        c = a + b+ x;
    }
    int People() {}                    //这不是构造方法
}
```

创建对象的步骤

创建一个对象分为两个步骤：

1.使用一个类声明一个对象变量：People people;

2.为这个对象分配变量：　people =　new People(1,2); //这里要使用构造方法

声明一个对象变量后，这个对象中没有存储任何数据，此时的对象是一个空对象，它没有得到任何“实体”，也就是该对象没有拥有任何变量，必须为该对象分配变量后才能使用对象。对象的内存模型：

people

如何为一个对象分配变量？当使用 new 运算符和构造方法时，new People(1，2);系统会执行以下步骤为对象分配变量：

1. 首先为类中的成员变量分配内存空间，People类中的成员变量a和b都会被分配内存空间。
2. 执行指定构造方法中的语句，如果成员变量在声明时没有指定具体的值，构造方法中也没右赋值，那系统会为成员变量分配一个默认值，整型变量的默认值是0，浮点型是0.0，boolean型是false，引用型是null。
3. 在为变量分配内存之后，系统将会计算出一个引用值，是一个十六进制的数，这个值包含着代表这些成员变量内存位置以及相关的重要信息。既new People(1，2); 是一个引用值，如果把这个引用值赋值给people那么，people对象就拥有了之前分配了内存的成员变量，这些变量将由people对象管理。这时候pepleo对象才拥有了实体。

注意：当使用类的构造方法创建了许多对象时，这些对象所拥有的实体，既变量是不同的，系统会为不同的对象在不同的内存空间分配变量，（类变量除外）在引用为计算出之前，不能说一个对象已经诞生，对象中存放的是计算出来的引用。如果两个对象的引用完全一致，那么这两个对象拥有相同的实体（变量）。

```
People people1 = new People(1,2);
People people2;
people 2 = people1;                    //people1和people2拥有一样的实体
```

Java的垃圾收集机制

Java的垃圾收集机制会周期性的检查某一个实体是否不再被任何对象所拥有，如果是，Java会释放这个实体所占用的内存空间。执行System类的gc()方法可以立即进行垃圾收集操作。

```
People people1 = new People(1,2);
People people2 = new People(2);
people2 = people1;                    //系统会释放原来分配给people2的实体所占用的内存资源
```

参数传值

参数属于局部变量，当对象调用方法时，参数被分配内存空间，同时要求向参数传递一个具体的值。在Java中参数的值，都是传递的值的拷贝，也就是说，参数的具体值是向参数传递值时那个值的一个“副本”，改变参数的值，不会影响向参数传值的值，改变传值的值，也不会影响参数的值。

对于基本数据类型的参数，不可以向该参数传递级别高于该参数的的值，不可以向int型参数传递一个double值。但可以传递低于参数级别的值，可以向一个double型参数，传递一个float值。

对与引用型的参数，向该参数“传值”传递的是对象变量中存放的引用值而不是对象变量所引用的实体。如果改变参数变量所引用的实体会导致原对象的实体也发生改变，因为这两个对象所拥有一样的实体，反过来也一样。

```
public class Battery {
    int electricityAmount;
    Battery(int amount) {
        electricityAmount = amount;
    }
}
public class Radio {
    void openRadio(Battery battery) {              //向参数传递一个Battery
        //battery对象所拥有的实体与nanfu实体一致
        battery.electricityAmount -= 10;
    }
}
public class Ex1 {
    public static void main(String [] args) {
        Battery nanfu = new Battery(100);
        Radio radio = new Radio();
        //改变battery对象的实体，nanfu的实体也会发生改变
        radio.openRadio(nanfu);
        System.out.println(nanfu.electricityAmount);
    }
}
```

可变参数

可变参数是指在声明方法时不给出参数列表的某一项到最后一项的参数名字和个数，使用一个参数代表来表示，但是这些参数的类型必须一致，同时，这些参数的最后一个参数必须是整个参数列表的最后一个参数，如：

```
public int add(int ... x) {                //x是参数代表，x.length表示参数的个数，x[i]可以代表第几个
    for (int i:x) {                        //参数，使用for循环遍历参数列表
        sum = sum +i;                      //但是注意不能这样使用 add(int ...x, int y)
    }                                      //y是最后一个参数，不能这样使用
}
```

对象的组合

一个类中的成员变量可以是某一个对象，如果使用该类创建对象，那么整个对象就会组合了其他的对象，同时，一个类的方法的参数，也可以是某一个对象。

注意：如果一个对象a中的成员变量包含另一个对象b，那么这个a可以委托该b调用它的方法，a也可以随时更换所包含的对象,具体来说是更换b的实体。

实例成员变量与类成员变量

成员变量可以分为实例变量和类变量（静态变量），使用static关键字修饰的是类变量，不使用static修饰的是实例变量。

```
class People {
    static int x;              //x是变量，当People的字节码被加载到内存时，x就被分配了内存
    int y;                     //在别的对象中使用People.x访问x变量，y是实例变量
}
```

实例变量和类变量的区别：

1.当使用类创建多个对象时，这些对象的实例变量时不同的，即为不同的对象分配的实例变量占用不同的内存空间。改变一个对象的实例变量的值，不会影响其他对象的实例变量的值。

2.当使用类创建多个对象时，他们的类变量时相同的，即为不同的对象分配的类变量占用同一处内存，只要一个对象改变了类变量的值，其他的对象的类变量的值都会跟着改变。

3.当Java程序执行时，类的字节码被加载到内存，当未使用类创建对象时，实例变量不会被分配内存，但是类中的类变量会被分配内存，当使用类创建对象时，实例变量被分配内存，类变量不再分配内存，所有的对象共享同一个类变量。

4.实例变量只能通过类访问，类变量还可以通过类名直接访问。

实例方法和类方法

类中的方法可以分为类方法和实例方法，使用static修饰的是类方法，不用static修饰的时实例方法。

```
class People {
    static int add(int x, int y) {
        return x+y;
    }
    int add1(int x, int y) {
        return x+y;
    }
}
public class Ex2 {
    public static void main(String[] args) {
        System.out.println(People.add(1+2));    //可以调用add方法，但是不可以调用add1方法
    }
}
```

实例方法和类方法的区别：

1.当类的字节码被加载到内存时，类的实例方法不会被分配的方法的入口地址，只有当使用类创建对象之后，实例方法才被分配入口地址，而且方法的入口地址只会在创建第一个对象时分配一次，所有的对象共享这个入口地址，当所有对象消失时，方法入口地址被取消。实例方法中可以操作实例变量和类变量。

2.当类的字节码文件被加载到内存时，类的类方法会立即被分配入口地址，类方法可以通过类名调用，也可以通过对象调用。类方法中只能操作类变量，因为当未创建对象时，内存中没有实例变量，但是存在类变量。

注意：当一个方法不需要使用或者操作任何实例变量时，可以考虑将这个方法设计为类方法。

方法的重载（overload）

方法的重载体现了行为的多态性，方法重载指的是在一个类中，可以具有多个名字相同的方法，方法的返回值和名字不参与比较，但是这些方法要满足下列条件：

1.方法的参数类型不同

2.方法的参数个数不同

```
public class People {
    int add(int a, int b) {
```

```
        return a+b;
    }
    int add(double a,double b) {
        return a+b;
    }
}
```

可以根据传入方法的参数的不同调用不同的方法，体现行为的多态性。

this关键字

this关键字表示某个对象，this关键字只能在构造方法或者实例方法中使用，不能在类方法中使用。类方法被分配入口地址时，还未使用类创建任何对象。

在构造方法中使用this关键字时，表示，使用该构造方法创建的对象，this关键字在实例方法中使用时代表正在调用该方法的对象。

访问权限

访问权限针对的是对象，是指一个对象能否通过"."点运算符来访问自己的变量或者调用类中的方法，访问修饰符有public,protected,private,访问修饰符用来修饰类中的成员变量或者方法。

私有变量和私有方法：

在一个类中使用private修饰的变量和方法称为私有变量和私有方法，在其他类中，使用这个类创建的对象不能访问私有变量，也不能调用类中的私有方法。

```
class People {
    private int a;
    private int add(int x, int y) {
        return x+y;
    }
}
public class Ex3 {
    public static void main(String [] args) {
        People people = new People();
        people.add(1,2);                //产生错误
        people.a;                       //产生错误
    }
}
```

共有变量和共有方法:

在一个类中使用public修饰的变量和方法成为共有变量和方法，在其他类中，使用这个类创建的对象，能够访问自己的public变量调用public方法。

友好变量和友好方法：

在一个类a中不使用任何访问修饰符的变量和方法称为友好变量和方法，在其他类中，使用该类创建一个对象，如果a和其他类在同一个包中，那么这个对象能够访问自己的友好变量，调用友好方法。如果不在一个包中，使用import语句导入a类，那么这个对象不能访问自己的友好变量和调用友好方法。

受保护变量和受保护的方法：

在一个类a中使用protected修饰的变量和方法受保护变量和受保护的方法，在其他类中，使用该类创建一个对象，如果a和其他类在同一个包中，那么这个对象能够访问自己的受保护变量，受保护的方法。如果不在一个包中，使用import语句导入a类，那么这个对象不能访问自己的受保护变量和受保护的方法。

注意：受保护与友好之间的区别体现在子类继承父类的时候，并不是没有区别

public类和友好类：

可以使用关键字public修饰一个类，这个类成为public类，不使用public修饰则称为一个友好类，可以在任何一个类中使用public类创建一个对象，但是使用友好类创建对象时，要保证这两个类在一个包中。不能使用private和protected修饰类。

基本类型的封装类

Java的基本数据类型包括boolean、byte、short、char、int、long、float、double，Java同时也提供了相应的包装类，这些类在Java.lang包中，分别为，Byte、Integer、Short、Long、Float、Double、Character。Double类使用构造方法Double(double num)创建一个Double类型的对象，使用doubleValue()方法返回含有的float型数据，其他数据类型与Double类型相似。

对象数组：

```
People [] people = new People[10];
```

以上代码创建了一个数组people，该数组包含了10个People对象，但这些对象并未分配实体，

```
for(int i=0;i<10;i++) {
    people[i] = new people();          //创建实体
}
```