二、子类与继承

子类和父类

当我们编写一个类的时候发现某一个类有我们想要的成员变量和方法，我们可以将编写的类声明为这个类的子类，就可以达到复用这类的成员变量和方法的效果。继承是由一个已经存在的类，创建一个新的类的一种机制，继承可以让我们的工作不必从头做起。可以使用关键字extends来定义一个类的子类.

```
class Student extends People {
}
```

注意：Object 类是所有类的祖先类，所有的类都是Object类的子孙类，一个类只能有一个父类，但是可以有多个子类。

子类的继承

继承是指：子类继承父类的成员变量作为自己的成员变量，子类可以在自己定义的实例方法中操作这个成员变量，子类继承父类的方法作为自己的方法，子类可以在自己定义的实例方法中调用这个方法。继承是完全意义上的继承，即子类完全拥有这个变量或者方法，就好像在子类中自己定义的一样。如果在子类自己定义的实例方法中不能操作这个变量，或者调用这个方法，那么变量或者方法就没有被子类继承。

1、在同一个包中的继承

子类如果跟父类在同一个包中，那么子类就继承了父类中不是private的成员变量和方法，继承的成员变量和方法的访问权限保持不变。(这个访问权限是针对对象的，使用类创建一个对象时，这个对象能否通过点运算符访问自己的成员变量和方法)

2、不在同一个包中的继承

如果子类跟父类不在同一个包中，那么子类只继承了父类中是public、protected的成员变量和方法，父类中友好的、private的成员变量和方法不会被继承。跟在同一个包的时候相比，有好的变量和方法不被继承。（但是如果对象和类不在同一个包时，对象不能访问protected的成员变量和方法）

注意：假设有一个类B，它继承于类A，同时在另一个类C中使用B创建了一个对象，如果类C和类B在同一个包中，那么这个对象可以访问类B自己定义的成员变量、调用类B自己定义的方法。对于类B从类A继承的成员变量和方法，如果类C和类A在一个包中，这个对象就可以访问类B继承得成员变量、调用类B继承的方法，如果不在一个包中就不能访问，也不能调用。

子类与对象

当使用子类的构造方法创建一个子类对象时，不仅子类中声明的成员变量被分配了内存空间，父类中的成员变量，也被分配了内存空间，但是只将子类继承的那部分成员变量作为分配给子类对象的变量，子类未继承的成员变量，如父类中Private的成员变量不会被分配给子类对象，但是Private成员变量依然被分配了内存空间，不在同一个包中的友好变量也是如此，但是子类继承的方法，却可以操作子类未继承的变量。

```
public class People {
    private int a = 100;                          //私有成员变量，不会被继承
    public int getA() {
        return a;
    }
}
public class Student extends People {          //类Student继承于People
    int sum = 0;
    public add(int ... x) {
        for(int i:x) {
            sum = sum + i;
        }
    }
}
public class Ex4 {
    public static void main(String [] args) {
        Student student = new Student();
        System.out.println(student.getA);        //对象student可以使用继承的方法来操作未
    }                                                    //被子类继承的变量
}
```

instanceof运算符

instanceof运算符是一个双目运算符，左边的操作元时一个对象，右边的操作元时一个类，如果左边的对象是右边的类或者子类所创建的对象，那么运算的结果是true，否则返回false。

成员变量的隐藏

当我们在编写一个子类时，我们可以声明子类的成员变量，当我们声明的成员变量和子类从父类继承的成员变量名字相同时，子类就会隐藏所继承的变量，只要名字相同，类型不

同也会被隐藏。需要注意的一些地方：

1、子类对象或者子类自己定义的方法操作与父类同名的成员变量指的是子类自己重新声明的成员变量，不是子类继承的成员变量。

2、子类对象可以调用子类继承的方法操作子类继承的成员变量或者隐藏的成员变量，但不可以操作子类重新声明的变量，即子类继承的方法操作的成员变量一定是子类继承的成员变量或者隐藏的成员变量。

注意：子类继承的方法只能操作子类继承的变量或者子类隐藏的变量，子类新增的方法可以操作子类继承的变量、子类自己声明的变量，子类隐藏的变量（使用super关键字）。

方法的重写

当我们在编写一个子类时，我们可以在子类中定义方法，如果这个方法的名字、类型、参数个数、参数类型与子类继承的某个方法一致，（类型也可以父类型的子类型，如父类方法返回类型是People，子类是Student），那么这方法称为子类重写的方法。

通过方法的重写子类可以隐藏继承的方法，体现子类独特的行为，子类对象调用这个方法，一定是子类重写的方法，而不是继承的方法。子类重写的方法可以操作子类继承的变量，子类声明的变量，子类继承的方法，子类定义的方法。也可以通过super关键字调用子类隐藏的变量和方法。

注意：重写时不能降低方法的访问权限，但是可以提高方法的范文权限

super关键字

子类一但隐藏了继承的成员变量和方法，子类对象就无法直接使用隐藏的成员变量和方法，子类隐藏的成员变量和方法由super关键字来负责管理操作。

```
public class People {
    int a = 10;
    int add(int x, int y) {
        return x + y;
    }
}
public class Student extends People {
    int a = 5;
    int add(int x, int y) {
        int p = super.add(x,y);
        return p + super.a;
    }
}
```

```
public class Ex5 {
    public static void main(String [] args) {
        Student student = new Student();
        System.out.println(student.a);
        System.out.println(student.add(1,2));
    }
}
```

子类不继承父类的构造方法，当使用子类的构造方法创建一个对象时，子类的构造方法总是会先调用的父类的某个构造方法，如果没有显示的声明调用哪一个父类的构造方法，就默认调用父类的无参数的构造方法，（但是父类如果没有提供无参数的构造方法，就会产生错误）。

在子类的构造方法中，要使用super关键字来调用父类的构造方法，而且必须是子类构造方法中的第一条语句。如果没有显示地写出，就默认有 super();，调用父类无参数的构造方法。

注意：当中父类中定义了多个构造方法时，应当包括一个无参数的构造方法，防止继承时出现错误。

final关键字

final关键字可以修饰类，方法，成员变量，局部变量。

如果一个类使用final关键字来修饰，那么这个类不允许被继承，即不能有子类；如果一个方法使用final来修饰，那么这个类不允许被重写，子类只能继承这个方法，不能重写这个方法；如果使用final来修饰一个变量，那它就是一个常量，常量在声明时必须要赋值，系统不会给常量赋默认值。常量在程序运行期间不允许再做修改。

对象的上转型对象

当把一个子类的对象的引用赋值给一个父类对象时，那么就称这个父类对象是这个子类对象的上转型对象。

```
Father f;
Son s = new Son();
f = s;
```

或者

```
Father f = new Son();
```

对象的上转型对象的实体是由子类负责创建的，上转型对象会失去原对象的一些属性和方法，相当于时子类对象的一个简化对象。具有以下特点：

1、上转型对象不能操纵子类新增的变量，不能操纵子类新增的方法。

2、上转型对象可以操作子类继承和隐藏的变量，可以操作子类继承的方法和子类重写的方法。

3、如果某一个子类重写了父类的某一个方法，那么子类的上转型对象调用这个方法，一定是子类重写的这个方法。（涉及到方法的动态绑定）

4、如果子类中有跟父类重名的static方法，方法的类型参数都一致，子类并未重写这个方法，static方法是属于类本身的。上转型对象调用这个方法，调用的是父类的static方法，而不是子类的static方法。static方法是静态绑定的。

继承与多态

如果一个类有许多的子类，并且这些子类都重写了父类的某一个方法，当把一个子类对象的引用赋值给一个父类对象时，这个上转型对象调用这个方法时就会产生许多不同的结果，也就是多态。

```
class Animal {
    public void speak() {

    }
}
class Dog extends Animal {
    public void speak() {
        System.out.println("汪");
    }
}
class Cat extends Animal {
    public void speak() {
        System.out.println("喵");
    }
}
public class Ex8 {
    public static void main(String[] args) {
        Dog d = new Dpg();
        Cat c = new Cat();
        d.speak;
        c.speak;
    }
```

```
}
```

abstract类和abstract方法

使用abstract修饰的类称为abstract类，使用abstract修饰的方法称为abstract方法：

```
abstract class A {
    abstract void f ();
}
```

1、abstract方法不能有方法体，只能声明不能实现。同时不允许使用abstract和final同时修饰一个方法，也不允许使用static修饰abstract方法，abstract方法只能是实例方法，不能时静态方法。

2、abstract类中可以有abstract方法也可以有非abstract方法，也可以没有abstract方法，普通方法中不能有abstract方法。abstract类不能使用new运算符创建类的实例，但是可以把子类对象的引用赋值给abstract类变量。

3、当一个普通类继承于一个abstract类时，子类必须重写父类中的所有abstract方法，并给出方法体。当一个abstract类继承于另一个abstract类时，子类可以重写父类的abstract方法或者继承父类的abstract方法。

注意：abstract类可以抽象出重要的行为标准，该行为标准用abstract方法来表示，抽象类封装了子类必须要有的行为，但是具体的行为有子类自己给出。即根据父类抽象方法给出具体的行为。