1. Java继承的特点
2. Java只支持单继承，不支持多继承。即一个类只能有一个父类，不能有多个父类。
3. Java支持多层继承。

二．继承的格式：  
通过extends关键字，可以声明一个子类继承另外一个父类。格式如下：  
class 父类 {

...

}

class 子类 extends 父类 {

...

}

三．继承中成员变量的访问

（一）不重名成员变量的访问

如果父类和子类中的成员变量是不重名的，那么对于它们的访问是没有二义性的，直接通过对象调用即可。

（二）重名成员变量的访问

若父类与子类有重名的成员变量，那么在访问时就需要进行区分。在子类中访问父类中的非私有成员变量时，需要使用super关键字，修饰父类成员变量。而访问子类本身的重名成员变量时，可以用this修饰。

注意：若在子类的成员方法中又定义了同名的局部变量时，直接用变量名调用，而子类的重名成员变量用this修饰，父类的重名成员变量用super修饰。

如下所示：

class Fu {

// Fu中的成员变量。

int num = 5;

}

class Zi extends Fu {

// Zi中的成员变量

int num = 6;

public void show() {

//访问父类中的num

System.out.println("Fu num=" + super.num);

//访问子类中的num

System.out.println("Zi num=" + this.num);

}

}

class ExtendsDemo03 {

public static void main(String[] args) {

       // 创建子类对象

Zi z = new Zi();

       // 调用子类中的show方法

z.show();

}

}

四．继承中成员方法的访问

（一）不重名成员方法的访问

父类和子类的成员方法没有重名，那么可以用对象直接调用成员方法。如在子类对象中调用成员方法，会先在子类中查找有没有对应的方法，若存在就直接执行；若没有，就向上找父类中的方法。

（二）重名成员方法的访问

父类和子类中的成员方法重名，是通过方法重写实现的，

方法重写：子类中重新写一个父类中的某一方法（方法名和参数列表都相同），会出现覆盖效果，也称为重写或者复写。

从名字就可以看出，子类的方法重写对于父类中的重名方法有覆盖效果，若子类对象调用该重写方法，则直接用子类中重写的方法，而不会去调用父类中的重名方法。

class Fu {

public void show() {

System.out.println("Fu show");

}

}

class Zi extends Fu {

//子类重写了父类的show方法

public void show() {

System.out.println("Zi show");

}

}

public class ExtendsDemo05{

public static void main(String[] args) {

Zi z = new Zi();

      // 子类中有show方法，只执行重写后的show方法

z.show();  // Zi show

}

}

（三）重写的应用

许多业务是不断更新迭代的，但是我们不能随意更改之前已经上线的业务代码，可以通过定义一个新的类，利用原有业务中的共性内容，并通过重写为新的子类添加新的功能。

五．继承中的构造方法

子类能够继承父类的成员方法，但是无法继承父类的构造方法，因为构造方法的名字与类名一致。但是我们能够通过子类调用父类的构造方法。

（一）super()

子类构造方法中默认隐含了一个super()调用，调用父类的构造方法，并且仅仅是默认的无参构造函数。即子类在构造时，会先调用父类的无参构造函数，再调用自己的构造函数。

（二）有参构造函数

当父类中定义了有参构造函数，而没有定义无参构造函数时，子类的构造方法就会出错，因为子类只是默认调用了父类的无参构造函数，此时的父类中没有无参构造函数。因此，我们需要手动定义父类的无参构造函数或者为子类构造函数调用父类的有参构造函数。