面向对象第二天：

回顾：

1. 什么是类？什么是对象？
2. 如何创建类？如何创建对象？如何访问成员？
3. this：指代当前对象，哪个对象调用方法它指的就是哪个对象
   - this.成员变量名----------访问成员变量

     > 当成员变量与局部变量同名时，若想访问成员变量，则this不能省略
4. 构造方法：
   - 给成员变量赋初值、与类同名、没有返回值类型(连void都没有)，创建(new)对象时被自动调用。若自己不写构造方法，则默认一个无参构造，若自己写了构造方法，则不再默认提供。

精华笔记：

1. 继承：
   - 作用：代码复用
   - 通过extends实现继承
   - 超类/基类/父类：共有的属性和行为

     派生类/子类：特有的属性和行为
   - 派生类可以访问：超类的+派生类的，超类不能访问派生类的
   - 一个超类可以有多个派生类，一个派生类只能有一个超类---------单一继承
   - 具有传递性
   - java规定：构造派生类之前必须先构造超类
     - 在派生类的构造方法中若没有调用超类的构造方法，则默认super()调用超类的无参构造方法
     - 在派生类的构造方法中若自己调用了超类的构造方法，则不再默认提供

       > 注意：super()调用超类构造方法，必须位于派生类构造方法中的第1行
2. super：指代当前对象的超类对象
   - super.成员变量名-------------------------访问超类的成员变量

     > 当超类成员变量和派生类成员变量同名时，super指超类的，this指派生类的
     >
     > 若没有同名现象，则写super和this是一样的
   - super.方法名()-----------------------------调用超类的方法
   - super()---调用超类的构造方法
3. 方法的重写(override/overriding)：重新写、覆盖
   - 发生在父子类中，方法名相同，参数列表相同

- 重写方法被调用时，看对象的类型----------new谁就调谁的，这是规定

```
class 餐馆{
    void 做餐(){ 做中餐 }
}
//1)我还是想做中餐----------------------不需要重写
class Aoo extends 餐馆{
}
//2)我想改做西餐------------------------需要重写
class Boo extends 餐馆{
    void 做餐(){ 做西餐 }
}
//3)我想在中餐基础之上加入西餐------------需要重写(先super中餐，再加入西餐)
class Coo extends 餐馆{
    void 做餐(){
        super.做餐();
        做西餐
    }
}
```

笔记：

1. 继承：
   - 作用：代码复用
   - 通过extends实现继承
   - 超类/基类/父类：共有的属性和行为

     派生类/子类：特有的属性和行为
   - 派生类可以访问：超类的+派生类的，超类不能访问派生类的
   - 一个超类可以有多个派生类，一个派生类只能有一个超类---------单一继承
   - 具有传递性
   - java规定：构造派生类之前必须先构造超类
     - 在派生类的构造方法中若没有调用超类的构造方法，则默认super()调用超类的无参构造方法
     - 在派生类的构造方法中若自己调用了超类的构造方法，则不再默认提供

       > 注意：super()调用超类构造方法，必须位于派生类构造方法中的第1行

       ```
       public class Person {
           String name;
           int age;
           String address;
           Person(String name,int age,String address){
               this.name = name;
               this.age = age;
               this.address = address;
           }

           void eat(){
       ```

```
        System.out.println(name+"正在吃饭...");
    }
    void sleep(){
        System.out.println(name+"正在睡觉...");
    }
    void sayHi(){
        System.out.println("大家好，我叫"+name+"，今年"+age+"岁了，家
住"+address);
    }
}

public class Student extends Person{
    String className;
    String stuId;
    Student(String name,int age,String address,String
className,String stuId){
        super(name,age,address); //传递的是name/age/address的值
        this.className = className;
        this.stuId = stuId;
    }

    void study(){
        System.out.println(name+"正在学习...");
    }
}

public class Teacher extends Person{
    double salary;
    Teacher(String name,int age,String address,double salary){
        super(name,age,address);
        this.salary = salary;
    }

    void teach(){
        System.out.println(name+"正在讲课...");
    }
}

public class Doctor extends Person {
    String title;
    Doctor(String name,int age,String address,String title){
        super(name,age,address);
        this.title = title;
    }
    void cut(){
        System.out.println(name+"正在做手术...");
    }
}

public class ExtendsTest {
    public static void main(String[] args) {
        Student zs = new Student("张三",25,"廊坊","jsd2302","001");
        zs.eat();
        zs.sleep();
        zs.sayHi();
```

```
        zs.study();

        Teacher ls = new Teacher("李四",35,"佳木斯",6000.0);
        ls.eat();
        ls.sleep();
        ls.sayHi();
        ls.teach();

        Doctor ww = new Doctor("王五",46,"山东","主任医师");
        ww.eat();
        ww.sleep();
        ww.sayHi();
        ww.cut();
    }
}
```

2. super：指代当前对象的超类对象
   - super.成员变量名----------访问超类的成员变量

     > 当超类成员变量和派生类成员变量同名时，super指超类的，this指派生类的
     >
     > 若没有同名现象，则写super和this是一样的

   - super.方法名()----------调用超类的方法
   - super()----------调用超类的构造方法

```
public class SuperDemo {
    public static void main(String[] args) {
        Boo o = new Boo();
    }
}


//在派生类的构造方法中若自己调用了超类的构造方法，则不再默认提供
class Coo{
    Coo(int a){
    }
}
class Doo extends Coo{
    Doo(){
        super(5);
    }
    /*
    //如下代码为默认的
    Doo(){
        super();
    }
     */
}

class Aoo{
    Aoo(){
        System.out.println("超类构造方法");
    }
}
```

```
class Boo extends Aoo{
    Boo(){
        super(); //默认的，调用超类的无参构造方法
        System.out.println("派生类构造方法");
    }
}
```

3. 方法的重写(override/overriding)：重新写、覆盖
    - 发生在父子类中，方法名相同，参数列表相同
    - 重写方法被调用时，看对象的类型----------new谁就调谁的，这是规定

```
class 餐馆{
    void 做餐(){ 做中餐 }
}
//1)我还是想做中餐----------不需要重写
class Aoo extends 餐馆{
}
//2)我想改做西餐----------需要重写
class Boo extends 餐馆{
    void 做餐(){ 做西餐 }
}
//3)我想在中餐基础之上加入西餐----------需要重写(先super中餐，再加入西餐)
class Coo extends 餐馆{
    void 做餐(){
        super.做餐();
        做西餐
    }
}
```

```
public class Person {
    String name;
    int age;
    String address;
    Person(String name,int age,String address){
        this.name = name;
        this.age = age;
        this.address = address;
    }

    void eat(){
        System.out.println(name+"正在吃饭...");
    }
    void sleep(){
        System.out.println(name+"正在睡觉...");
    }
    void sayHi(){
        System.out.println("大家好，我叫"+name+"，今年"+age+"岁了，家
住"+address);
    }
}

public class Student extends Person{
    String className;
```

```
    String stuId;
    Student(String name,int age,String address,String className,String
stuId){
        super(name,age,address); //传递的是name/age/address的值
        this.className = className;
        this.stuId = stuId;
    }

    void study(){
        System.out.println(name+"正在学习...");
    }

    void sayHi(){
        System.out.println("大家好，我叫"+name+"，今年"+age+"岁了，家
住"+address+"，所在班级为:"+className+"，学号为："+stuId);
    }
}

public class Teacher extends Person{
    double salary;
    Teacher(String name,int age,String address,double salary){
        super(name,age,address);
        this.salary = salary;
    }

    void teach(){
        System.out.println(name+"正在讲课...");
    }

    void sayHi(){
        System.out.println("大家好，我叫"+name+"，今年"+age+"岁了，家
住"+address+"，工资为:"+salary);
    }
}

public class Doctor extends Person {
    String title;
    Doctor(String name,int age,String address,String title){
        super(name,age,address);
        this.title = title;
    }
    void cut(){
        System.out.println(name+"正在做手术...");
    }
}

public class ExtendsTest {
    public static void main(String[] args) {
        Student zs = new Student("张三",25,"廊坊","jsd2302","001");
        zs.eat();
        zs.sleep();
        zs.sayHi();
        zs.study();

        Teacher ls = new Teacher("李四",35,"佳木斯",6000.0);
```

```
        ls.eat();
        ls.sleep();
        ls.sayHi();
        ls.teach();

        Doctor ww = new Doctor("王五",46,"山东","主任医师");
        ww.eat();
        ww.sleep();
        ww.sayHi();
        ww.cut();
    }
}
```

4. 综合练习：

```
public class Animal {
    String name;
    int age;
    String color;
    Animal(String name,int age,String color){
        this.name = name;
        this.age = age;
        this.color = color;
    }
    void drink(){
        System.out.println(color+"色的"+age+"岁的"+name+"正在喝水...");
    }
    void eat(){ //---------------明天继续改造
        System.out.println(color+"色的"+age+"岁的"+name+"正在吃饭...");
    }
}

public class Dog extends Animal {
    Dog(String name,int age,String color){
        super(name,age,color);
    }
    void lookHome(){
        System.out.println(color+"色的"+age+"岁的狗狗"+name+"正在看家...");
    }
    void eat(){
        System.out.println(color+"色的"+age+"岁的狗狗"+name+"正在吃肯头...");
    }
}

public class Chick extends Animal {
    Chick(String name,int age,String color){
        super(name,age,color);
    }
    void layEggs(){
        System.out.println(color+"色的"+age+"岁的小鸡"+name+"正在下蛋...");
    }
    void eat(){
        System.out.println(color+"色的"+age+"岁的小鸡"+name+"正在吃小米...");
    }
}
```

```
public class Fish extends Animal {
    Fish(String name,int age,String color){
        super(name,age,color);
    }
    void eat(){
        System.out.println(color+"色的"+age+"岁的小鱼"+name+"正在吃小虾...");
    }
}

public class Test {
    public static void main(String[] args) {
        Dog dog1 = new Dog("小黑",2,"黑");
        dog1.eat();
        dog1.drink();
        dog1.lookHome();

        Dog dog2 = new Dog("小白",1,"白");
        dog2.eat();
        dog2.drink();
        dog2.lookHome();

        Dog dog3 = new Dog("小强",2,"黑白");
        dog3.eat();
        dog3.drink();
        dog3.lookHome();

        Chick chick = new Chick("花花",1,"棕");
        chick.eat();
        chick.drink();
        chick.layEggs();

        Fish fish = new Fish("金金",2,"金");
        fish.eat();
        fish.drink();
    }
}
```

补充：

1. 泛化：从程序设计角度而言叫泛化，从代码实现角度而言叫继承，泛化就是继承
2. 继承要符合is(是)的关系
3. 继承的是超类的成员变量和普通方法，不包括超类的构造方法，超类的构造方法是被派生类通过super()来调用的，而不是继承的

```
class Aoo{
    int a;
    Aoo(){
    }
    void show(){
    }
}
class Boo extends Aoo{
    继承了Aoo类的a+show()，并没有继承Aoo类的构造方法
}
```

4. 明日单词：

```
1)swim:游泳
2)abstract:抽象的
3)interface:接口
4)implements:实现
```