目录

[Day07.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811296)

[1     回顾](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811297)

[2     继承](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811298)

[2.1       方法重写 Override](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811299)

[2.2       父类的构造方法](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811300)

[2.3   super](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811301)

[练习1   继承练习](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811302)

[练习2   二维点、三维点](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811303)

[练习3   电子宠物](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811304)

[练习4   冒泡排序](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811305)

[3     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day07.htm#_Toc5811306)

**Day07.  Java**

**1   回顾**

  数据类型

  基本类型（8种）

  基本类型变量，直接保存值

  引用类型

  引用类型变量，保存对象的地址

  基本类型

  byte1,short2,int4,long8

  float4,double8

  char2， Unicode

  boolean1

  运算规则（5条）

  3/2, 1

  byte a = 2;

a = a+a; //int+int

  Integer.MAX\_VALUE+1

  2-1.9, 0.1000000000009

4.35\*100, 434.999999999999994

  Infinity, 无穷大, 3.14/0

NaN, Not a Number, Math.sqrt(-6)

  运算符

  +-\*/

  +

  %

  == != > >= < <=

  基本类型 a == b，比较值

  引用类型 a == b，比较地址值

  && || !

  ++, --

  1 ? 2 : 3

  =

  += /= ...

  ()

  流程控制

  if-else if-else

  switch-case-default

  byte,short,char,int

  enum

  jdk1.7, String

  for

  while，先判断，再执行

  do-while，先执行，再判断，至少执行一次

  break，中断循环和switch

  continue

  循环命名

outer:

for() {

   for() {

      break outer;

      continue outer;

}

}

  数组

  创建

int[] a = new int[6];

int[] a = {6,3,6,8};

a = new int[]{3,7,1,3,8,5};

  数组长度属性 a.length

  长度不可变

  最大下标 a.length-1

  二维数组，数组中存放数组

  int[][] a = new int[3][2];

  int[][] a = new int[3][];

  int[][] a = {{}, {}, {}};

  Arrays

  toString()

  sort()

  binarySearch()

  copyOf()

  System.arraycopy()

  对象

  类，模板、图纸

  对象、实例

  引用，遥控器

  构造方法，新建对象时执行

  不定义，有默认

  this

  this.xxx

  特殊引用，引用当前对象的地址

  this(...)

  构造方法之间调用

  首行代码

  方法重载 Overload

  同名不同参

**2   继承**

作用：代码重用、复用

Java的继承是单继承，只能继承一个父类

不能继承：

  构造方法

  私有成员

子类对象

  先新建父类对象

  再新建子类对象

  两个对象绑定在一起，整体作为一个对象

  调用成员时，先找子类，再找父类

**2.1     方法重写 Override**

继承的方法，在子类中重新定义重新编写

可以理解成“覆盖”、“改写”

**2.2     父类的构造方法**

新建子类对象时，会先新建父类对象，也会先执行父类的构造方法

  默认执行父类的无参构造，默认隐含调用 super()

new Student()， 默认先执行父类的无参

new Student("","",0,"")， 默认先执行父类的无参

  手动调用父类有参构造，super(参数)

  当父类没有无参构造，必须手动调用有参构造

**2.3     super**

**super.xxxxx()**

  方法重写时，调用父类中同一个方法的代码

**super(参数)**

  调用父类构造方法

  默认调用父类无参构造 super()

  手动调用父类有参构造 super(参数)

  必须是首行代码

**练习1  继承练习**

人类 Person

{

   name,gender,age

   toString()

}

学生 Student 继承 Person

{

   school

}

员工 Employee 继承 Person

{

   salary

}

项目：day0701\_继承

类：day0701.Test1

             Person

             Student

             Employee

Person

**package** day0701;

**public** **class** Person /\*extends Object\*/ {

String name;

String gender;

**int** age;

**public** Person() {

**this**("无名", "不明", 16);

}

**public** Person(String name,String gender,**int** age) {

**this**.name = name;

**this**.gender = gender;

**this**.age = age;

}

**public** String toString() {

**return** name+", "+gender+", "+age;

}

}

Student

**package** day0701;

**public** **class** Student **extends** Person{

String school;

**public** Student() {

    //super();//默认的隐含调用

}

**public** Student(String name,

               String gender,

**int** age,

               String school) {

    //super();//默认的隐含调用

**super**(name,gender,age);

    this.school = school;

}

@Override

**public** String toString() {

    /\*

     \* super.toString()

     \* 目的是减少代码重复，

     \* 一般在方法重写时，

     \* 调用父类同一个方法的代码

     \*/

**return** **super**.toString()+", "+school;

}

}

Employee

**package** day0701;

**public** **class** Employee **extends** Person{

**double** salary;

@Override

**public** String toString() {

**return** **super**.toString()+", "+salary;

}

}

Test1

**package** day0701;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    /\*

     \* p = {

     \*     name:"张三",

     \*     gender:"男",

     \*     age:22

     \* }

     \*

     \* s = {

     \*     name:"李四",

     \*     gender:"女",

     \*     age:21,

     \*

     \*     school:"牛蹄筋大学"

     \* }

     \* e = {

     \*     name:"王五",

     \*     gender:"男",

     \*     age:20,

     \*

     \*     salary: 9000

     \* }

     \*

     \*/

    Person p = **new** Person("张三","男",22);

    Student s = **new** Student();

    Employee e = **new** Employee();

    s.name = "李四";

    s.gender = "女";

    s.age = 21;

    s.school = "牛蹄筋大学";

    e.name = "王五";

    e.gender = "男";

    e.age = 20;

    e.salary = 9000;

    //println()打印方法中，

    //默认调用了对象的 toString() 方法

    System.***out***.println(p.toString());

    System.***out***.println(s.toString());

    System.***out***.println(e.toString());

}

}

Test2

**package** day0701;

**public** **class** Test2 {

**public** **static** **void** main(String[] args) {

    //默认执行 Person()

    Student s1 = **new** Student();

    //默认执行 Person()

    Student s2 = **new** Student("a","b",0,"c");

    System.***out***.println(s1.toString());

    System.***out***.println(s2.toString());

}

}

**练习2  二维点、三维点**

Point 类

{

   x,

   y,

   toString(),   //x,y

   distance()   //x,y

}

Point3D 类

{

   z,

   toString(),

   distance()

}

项目：day0702\_二维点三维点

类：day0702.Test1

             Point

             Point3D

Point

**package** day0702;

**public** **class** Point {

**int** x;

**int** y;

**public** Point(**int** x, **int** y) {

**this**.x = x;

**this**.y = y;

}

**public** String toString() {

    // "(3, 4)"

**return** "("+x+", "+y+")";

}

**public** **double** distance() {

**return** Math.*sqrt*(x\*x + y\*y);

}

}

Point3D

**package** day0702;

**public** **class** Point3D **extends** Point{

**int** z;

**public** Point3D(**int** x, **int** y, **int** z) {

    //父类没有无参构造，必须手动调用有参构造

**super**(x, y);

**this**.z = z;

}

@Override

**public** String toString() {

**return** "("+x+", "+y+", "+z+")";

}

@Override

**public** **double** distance() {

**return** Math.*sqrt*(x\*x + y\*y + z\*z);

}

}

Test1

**package** day0702;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    /\*

     \* p1 = {

     \*     x: 3,

     \*     y: 4

     \* }

     \*

     \* p2 = {

     \*     x: 3,

     \*     y: 4,

     \*     z: 5

     \* }

     \*/

    Point p1 = **new** Point(3, 4);

    Point3D p2 = **new** Point3D(3, 4, 5);

    System.***out***.println(p1.toString());

    System.***out***.println(p1.distance());

    System.***out***.println("--------------");

    System.***out***.println(p2.toString());

    System.***out***.println(p2.distance());

}

}

**练习3  电子宠物**

Pet 类

{

   name,

   full,

   happy,

   feed(),

   play(),

   punish()

}

Dog 类 继承 Pet

Cat 类 继承 Pet

项目：day0703\_电子宠物

类：day0703.Test1

             Pet

             Dog

             Cat

Pet

**package** day0703;

**public** **class** Pet {

String name;

**int** full;

**int** happy;

**public** Pet(String name) {

**this**(name, 50 ,50);

}

**public** Pet(String name, **int** full, **int** happy) {

**this**.name = name;

**this**.full = full;

**this**.happy = happy;

}

**public** **void** feed() {

**if** (full == 100) {

        System.***out***.println(name+"已经吃饱了");

**return**;

    }

    System.***out***.println("给"+name+"喂食");

    full += 10;

    System.***out***.println("饱食度："+full);

}

**public** **void** play() {

**if** (full == 0) {

        System.***out***.println(name+"饿得玩不动了");

**return**;

    }

    System.***out***.println("陪"+name+"玩耍");

    happy += 10;

    full -= 10;

    System.***out***.println("快乐度："+happy);

    System.***out***.println("饱食度："+full);

}

**public** **void** punish() {

    //不同的代码，就调用方法

    System.***out***.println(

     "打"+name+"的pp，"+name+"哭叫："+cry());

    happy -= 10;

    System.***out***.println("快乐度："+happy);

}

**public** String cry() {

    //无意义代码，必须在子类中重写这个方法

**return** "此处有哭叫声~";

}

}

Dog

**package** day0703;

**public** **class** Dog **extends** Pet{

**public** Dog(String name) {

**super**(name);

}

**public** Dog(String name, **int** full, **int** happy) {

**super**(name,full,happy);

}

@Override

**public** String cry() {

**return** "汪~";

}

}

Cat

**package** day0703;

**public** **class** Cat **extends** Pet{

**public** Cat(String name) {

**super**(name);

}

**public** Cat(String name, **int** full, **int** happy) {

**super**(name,full,happy);

}

@Override

**public** String cry() {

**return** "喵~";

}

}

Test1

**package** day0703;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    System.***out***.println("1. 猫");

    System.***out***.println("2. 狗");

    System.***out***.print("选择：>");

**int** c = **new** Scanner(System.***in***).nextInt();

    System.***out***.print("给宠物起个名：");

    String n = **new** Scanner(System.***in***).nextLine();

    Cat cat = **null**;

    Dog dog = **null**;

**if** (c == 1) {

        cat = **new** Cat(n);

*f*(cat);

    } **else** {

        dog = **new** Dog(n);

*f*(dog);

    }

}

**private** **static** **void** f(Pet pet) {

    System.***out***.println("按回车执行");

**while**(**true**) {

**new** Scanner(System.***in***).nextLine();

**int** r = **new** Random().nextInt(9);

**switch**(r) {

**case** 0: pet.feed(); **break**;

**case** 1: pet.play(); **break**;

**default**: pet.punish();

        }

    }

}

}

**练习4  冒泡排序**

Arrays.sort(数组)

基本类型数组，优化的快速排序

引用类型数组，优化的合并排序

项目：day0704\_冒泡排序

类：day0704.Test1

**package** day0704;

**import** java.util.Arrays;

**import** java.util.Random;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    //获得一个随机乱序数组

**int**[] a = *suiJi*();

    System.***out***.println(Arrays.*toString*(a));

    System.***out***.println("----------------");

*sort*(a);//对a数组排序

    System.***out***.println("----------------");

    System.***out***.println(Arrays.*toString*(a));

}

**private** **static** **int**[] suiJi() {

    //产生一个 5+ [0,6) 范围的整数存到变量 n

    //新建 int[] 数组a，长度 n

    //遍历数组，填入 100 内的随机整数

    //返回数组a

**int** n = 5+ **new** Random().nextInt(6);

**int**[] a = **new** **int**[n];

**for** (**int** i = 0; i < a.length; i++) {

        a[i] = **new** Random().nextInt(100);

    }

**return** a;

}

**private** **static** **void** sort(**int**[] a) {

    /\*

     \*                         j

     \* [2, 11, 24, 51, 46, 95, 33, 45, 64, 72]

     \*             i

     \*

     \*                                j

     \* [1, 5, 5, 33, 40, 48, 71, 50, 77]

     \*                i

     \*

     \*

     \*

     \* [1, 5, 5, 33, 40, 48, 50, 71, 77]

     \*                    i

     \*

     \*          j循环开始之前 flag = false，标记没有交换

     \*

     \*             if(! flag) {

     \*

     \*             }

     \*/

    //i循环从头到尾遍历

**for** (**int** i = 0; i < a.length; i++) {

**boolean** flag = **false**;//没有交换

        //j循环，把较小值向前交换，最小值交换到i位置

        //j循环从尾部向前递减到 >i

**for** (**int** j = a.length-1; j > i; j--) {

           //j和j-1位置的值比大小

**if** (a[j]<a[j-1]) {

**int** t = a[j];

              a[j] = a[j-1];

              a[j-1] = t;

              flag = **true**;//有交换

           }

        }

        //flag还是false，没有修改成true

        //没有执行交换，数据位置都是正确的

**if** (! flag) {

**break**;

        }

        System.***out***.println(Arrays.*toString*(a));

    }

}

}

**3   作业**

  重写

  day0703\_电子宠物

  Pet,Dog,Cat,Test1

  day0704\_冒泡排序

  day0703\_电子宠物

  添加新的宠物类型 Bird,Fish,Mouse,Snake

明天内容：

Object

String

Integer

二进制位运算