目录

Day06.Java

1 面向对象

- 人为抽象的一种编程模型
- 类
- 对象
- 引用
- 构造方法
- this
- 重载
- 继承
- super
- 多态
- instanceof
- 抽象类
- static
- final
- 访问控制符
- 对象的创建过程
- 接口
- 内部类

2 类

- 对事物、算法、逻辑、概念等的抽象
- 把相关的属性数据、逻辑运算方法，封装成一个“类”组件
- 类，理解成“模板”、“图纸”

3 对象

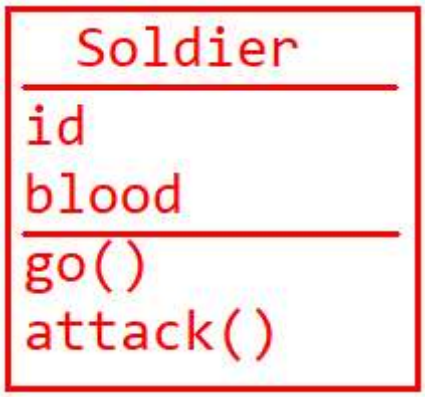
- 从类，创建的具体实例
- 每个对象，都占用独立的内存空间，保存各自的属性数据
- 每个对象，都可以独立控制，执行指定的方法代码

4 引用

- 引用变量，保存对象的内存地址
- 理解成“遥控器”
- 引用变量的特殊值 `null`，空，不保存任何地址

士兵

项目：day0601_士兵
类：day0601.Test1
Soldier



Soldier

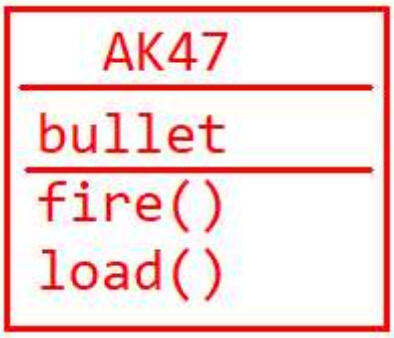
```java
package day0601;

import java.util.Random;

/*
 * 封装
 * 把士兵相关的属性数据，和逻辑运算方法，
 * 封装成一个“类”组件
 */
public class Soldier {
   /*
    * 属性变量
    * 成员变量
    */
   int id; //默认值0
   int blood = 100;

   /*
    * 成员方法
    */
   public void go() {
      System.out.println(id+"号士兵前进");
   }

   public void attack() {
      if(blood == 0) {
         System.out.println("这是"+id+"号士兵的尸体");
         return;
      }
      System.out.println(id+"号士兵进攻");
      //随机的减血量
      int d = new Random().nextInt(10);
      //减血
      blood -= d;
      if(blood<0) {
         blood = 0;
      }
      System.out.println("血量: "+blood);
      //血量是0，阵亡
      if(blood == 0) {
         System.out.println(id+"号士兵阵亡");
      }
   }
}
```

Test1

```java
package day0601;

public class Test1 {
```

```
    public static void main(String[] args) {
        //新建Soldier对象
        //内存地址存到变量 a
        Soldier a = new Soldier();
        Soldier b = new Soldier();

        //用a变量找到第一个士兵的内存空间
        //访问它的id变量
        a.id = 9527;
        b.id = 9528;

        //用a变量控制第一个士兵执行go()方法代码
        a.go();
        b.go();
        b.attack();
        b.attack();
        b.attack();
        b.attack();
    }
}
```

武器



项目：day0602_武器
类：day0602.Test1
AK47

AK47

```
package day0602;
/*
 * 封装
 * 武器相关的数据、逻辑运算方法，
 * 封装一个“类”组件
 */
public class AK47 {
    /*
```

```
     * 属性数据
     * 成员变量
     */
    int bullet = 10;//子弹数量

    /*
     * 成员方法
     */
    public void fire() {
        if(bullet == 0) {
            System.out.println("已经没有子弹了");
            return;
        }
        bullet--;
        System.out.println("发射");
        System.out.println("子弹数量: "+bullet);
    }
    public void load() {
        System.out.println("装载子弹");
        bullet = 10;
    }
}
```

Test1

```
package day0602;

import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        //新建AK47对象
        //把内存地址，存到变量a
        AK47 a = new AK47();
        AK47 b = new AK47();

        System.out.println("按回车进攻");
        System.out.println("输入a，装载a枪");
        System.out.println("输入b，装载b枪");
        while(true) {
            String s = new Scanner(System.in).nextLine();
            if(s.equals("a")) {
                a.load();
                continue;
            } else if(s.equals("b")) {
                b.load();
                continue;
            }
            System.out.println("a枪发射");
            a.fire();
            System.out.println("b枪发射");
            b.fire();
            System.out.println("------------\n\n");
```

```
        }
    }
}
```

电子宠物

项目：day0603_电子宠物
类：day0603.Test1
Dog

Dog

```
package day0603;
/*
 * 封装
 * 电子宠物的属性数据，运算方法，
 * 封装成Dog类
 */
public class Dog {
   String name;//null
   int full;//0
   int happy;//0

   public void feed() {
      if(full == 100) {
         System.out.println(name+"已经吃不下了");
         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(
         "饱食度: "+full+", 快乐度: "+happy);
    }
    public void punish() {
        System.out.println(
         "打"+name+"的pp, "+name+"哭叫: 汪~");
        happy -= 10;
        System.out.println("快乐度: "+happy);
    }
}
```

Test1

```
package day0603;

import java.util.Random;
import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        System.out.print("给宠物起个名字: ");
        String name = new Scanner(System.in).nextLine();
        Dog dog = new Dog();
        dog.name = name;
        dog.full = 50;
        dog.happy = 50;
        System.out.println("按回车执行");
        while(true) {
            new Scanner(System.in).nextLine();
            int r = new Random().nextInt(3);
            switch(r) {
            case 0: dog.feed(); break;
            case 1: dog.play(); break;
            case 2: dog.punish(); break;
            }
        }
    }
}
```

5 构造方法

- 新建对象时执行的特殊方法

```
new Soldier()
new AK47()
new Dog()
```

- 一个类，必须有构造方法
- 如果不定义构造方法，编译器编译时，会添加默认构造方法

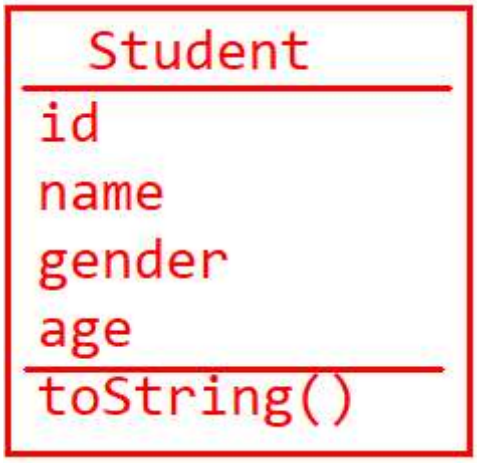
```
class A{
    public A() {
    }
}
```

- 自己定义构造方法，可以添加任意代码，执行意义的运算，完成任何功能都可以
- 构造方法重载（不同参数的多个方法）
  创建对象时，更加灵活、方便

```
class A {
    public A() {}
    public A(int i) {}
    public A(int i,String s) {}
    public A(int i,String s,double d) {}
    public A(String s) {}
}

new A();
new A(5);
new A(5,"abc",3.14)
```

- 一般在构造方法中，给成员变量赋值

学生



项目：day0604_学生
类：day0604.Test1
Student

Student

```
package day0604;

public class Student {
   int id;
   String name;
   String gender;
   int age;

   public Student() {
   }
   public Student(int id,String name) {
      this(id,name,null);
   }
   public Student(int id,String name,String gender) {
      //从构造方法，调用另一个重载的构造方法
      this(id,name,gender,0);
   }
   public Student(int id,String name,String gender,int age) {
      this.id = id;
      this.name = name;
      this.gender = gender;
      this.age = age;
   }

   public String toString() {
      return id+", "+name+", "+gender+", "+age;
   }
}
```

Test1

```
package day0604;

public class Test1 {
   public static void main(String[] args) {
      Student s1 = new Student();
      Student s2 = new Student(9527,"张三");
      Student s3 = new Student(9527,"张三","男");
      Student s4 = new Student(9527,"张三","男",19);

      System.out.println(s1.toString());
      System.out.println(s2.toString());
      System.out.println(s3.toString());
      System.out.println(s4.toString());
   }
```

```
}
```

6 this

- this.xxx
  - 特殊引用，引用当前对象的地址
  - 当前对象，正在调用、正在创建的对象
- this(...)
  - 重载的构造方法之间调用
  - 目的：减少代码重复
  - 一般从参数少的方法，调用参数多的方法
  - 必须是首行代码

7 方法重载 Overload

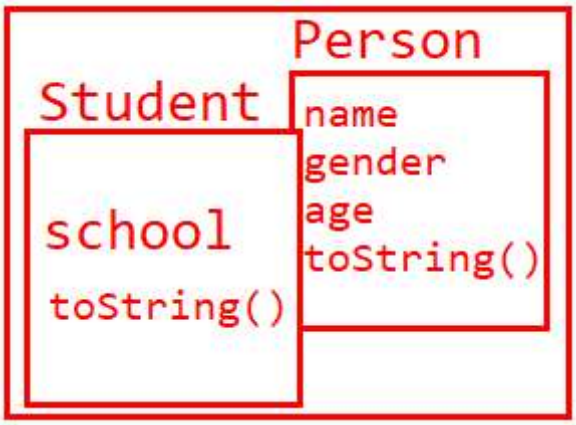
- 同名不同参

```
println()
println(int)
println(double)
println(String)
println(char)
...
```

8 继承

- 作用：代码重用、复用
- 单继承
  - 一个类，只能继承一个父类
  - 一个父类，可以有多个子类
- 不继承：
  - 构造方法
  - 私有成员
- 子类对象
  - 先创建父类对象
  - 再创建子类对象
  - 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

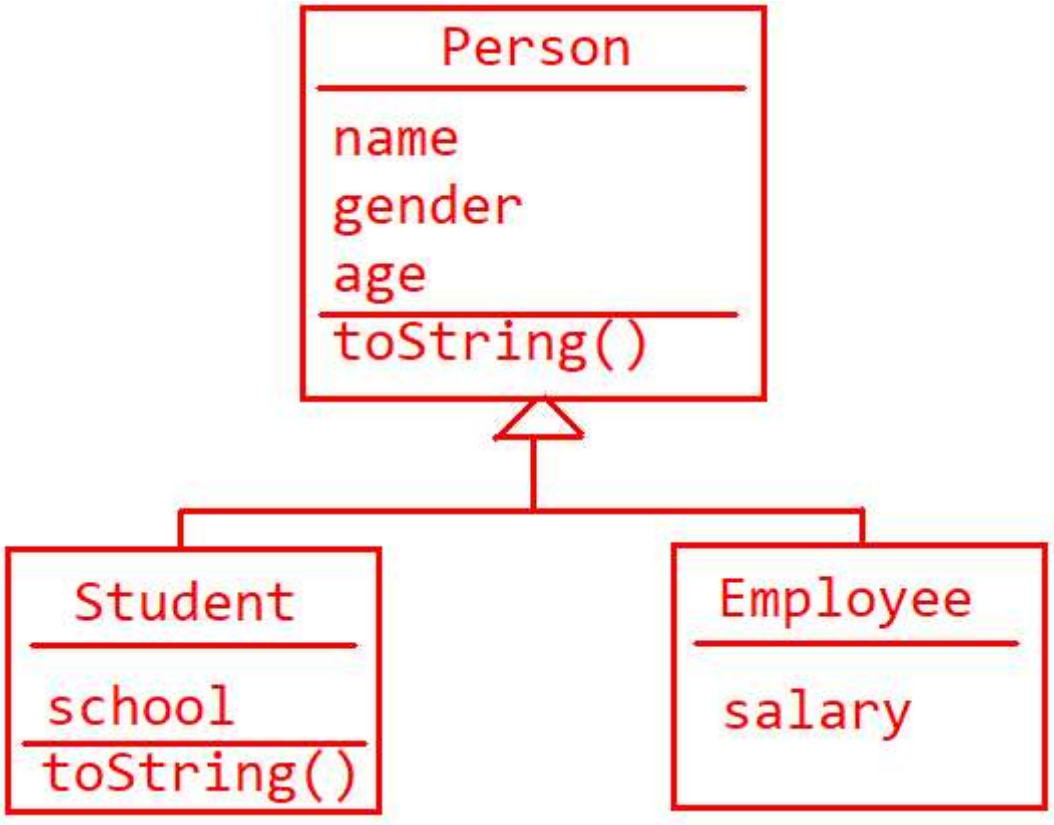


8.1 方法重写 Override

- 继承的方法，在子类中，重新定义，重新编写这个方法（改写）
- 重写方法时， 可以调用父类方法的代码
  `super.xxxxx()`

人类

项目：day0605_人类
类：day0605.Test1
Person
Student
Employee



Person

```
package day0605;

public class Person {
   String name;
   String gender;
   int age;

   public Person() {
   }
   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 day0605;

public class Student extends Person {
   String school;

   public String toString() {
      //super
      //在子类对象中,
      //对父类对象的特殊引用
      return super.toString()+", "+school;
   }
}
```

Employee

```
package day0605;

public class Employee extends Person {
   double salary;
}
```

Test1

```
package day0605;

public class Test1 {
   public static void main(String[] args) {
```

```
        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 = "女";
        e.age = 23;
        e.salary = 9000;
        System.out.println(p.toString());
        System.out.println(s.toString());
        System.out.println(e.toString());

    }
}
```

9 jdk类库中的类

```
new Scanner(System.in)
new Random()
```

- javax.swing.JFrame
  对桌面窗口的封装类
  新建JFrame对象，就相当于创建了一个新的窗口
  可以对窗口的各种属性做设置，
  并显示这个窗口

窗口

项目：day0606_窗口
类：day0606.Test1

```
package day0606;

import javax.swing.JFrame;

public class Test1 {
    public static void main(String[] args) {
        JFrame f = new JFrame();
        //对窗口的属性进行设置
        f.setSize(300,200);//尺寸
        f.setTitle("窗口标题");//标题栏文字
```

```
        f.setResizable(false);//不可改变大小
        f.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);
        //通知窗口服务对象，在屏幕上绘制窗口
        f.setVisible(true);

        //第二个窗口
        JFrame f2 = new JFrame();
        f2.setSize(200,300);//尺寸
        f2.setTitle("第二个窗口");//标题栏文字
        f2.setVisible(true);

    }
}
```

10 作业

- 重写
  - day0603_电子宠物
  - day0605_人类
- 复习面向对象概念
- 预习
  - 继承
  - 多态
  - 抽象类