今日内容介绍
1、面向对象思想
2、类与对象的关系
3、局部变量和成员变量的关系
4、封装思想
5、private,this关键字
6、随机点名器

###01面向对象和面向过程的思想
* A: 面向过程与面向对象都是我们编程中，编写程序的一种思维方式
* a: 面向过程的程序设计方式，是遇到一件事时，思考“我该怎么做”，然后一步步实现的过程。
* b: 面向对象的程序设计方式，是遇到一件事时，思考“我该让谁来做”，然后那个“谁”就是对象，他要怎么做这件事是他自己的事，反正最后一群对象合力能把事就好就行了。

###02面向对象的思想的生活案例
* A: 买电脑（组装机）
* a: 面向过程：自己该怎么做
* b: 面向对象：找人帮我们做

* A: 项目有一个新需求（新增一个接口）
* a: 面向过程：自己该怎么做
* b: 面向对象：PM找开发和测试来完成

###03面向对象好处
* A: 面向对象好处
* a: 面向对象思维方式是一种更符合人们思考习惯的思想
* b: 面向过程思维方式中更多的体现的是执行者（自己做事情），面向对象中更多的体现是指挥者（指挥对象做事情）。
* c: 面向对象思维方式将复杂的问题简单化。

###04大象装进冰箱的代码案例
* A: 需求：把大象装冰箱里
* a: 面向过程
* 自己打开冰箱门
* 自己将大象装进去
* 自己关闭冰箱门
* b: 面向对象
* 分析发现打开、装、关闭都是冰箱的功能。即冰箱对象具 备如下功能
* 冰箱打开
* 冰箱存储
* 冰箱关闭

* B: 通过伪代码描述大象和冰箱
* 描述大象：

```
class 大象
{
}
* 描述冰箱
class冰箱
{
void 打开(){}
void 存储(大象){}
void 关闭(){}
}
```

* C: 使用对象：
* 1、创建冰箱的对象
* 冰箱 bx = new 冰箱();
* 2、调用冰箱的功能
* 对象.功能()；
* bx.打开();
* bx.存储(new 大象());
* bx.关闭();
* D：总结：
* 1、先按照名词提炼问题领域中的对象
* 2、对对象进行描述，其实就是在明确对象中应该具备的属性和功能
* 3、通过new的方式就可以创建该事物的具体对象
* 4、通过该对象调用它以后的功能。

###05定义小汽车类
* A: 分析小汽车的属性和功能
* 属性
* 颜色
* 轮胎个数
* 功能
* 运行
* B: 通过伪代码描述小汽车
* 小汽车{
* 颜色
* 轮胎个数
* 运行(){}
* }
* C：通过JAVA代码描述小汽车
* public class Car {
* String color;
* int number;

* void run() {
* System.out.println(color + ":" + number);

* }
* }

###01测试汽车类
* A: 创见对象的格式
* a: 类名 变量名 = new 类名();
* B: 测试汽车类

```
public class CarDemo {
public static void main(String[] args) {
/*
测试：Car类中的run方法。
*/
// 1,创建Car的对象。给对象起个名字。
Car c = new Car();// c是类类型的变量。c指向了一个具体的Car类型的对象。
// 2,通过已有的对象调用该对象的功能。格式：对象.对象成员;
// 3,可以该对象的属性赋值。
c.color = "red";
c.number = 4;
c.run();
}
}
```

###02对象的内存图
* 见课后资料：对象的内存图.JPG

###03类和对象的关系
* A: 类和对象的关系
* 类是对某一类事物的抽象描述，而对象用于表示现实中该类事物的个体
* B: 举例
* 可以将玩具模型看作是一个类，将一个个玩具看作对象，从玩具模型和玩具之间的关系便可以看出类与对象之间的关系

###04成员变量和局部变量的区别

- 区别一：定义的位置不同
  - 定义在类中的变量是成员变量
  - 定义在方法中或者{}语句里面的变量是局部变量
- 区别二：在内存中的位置不同
  - 成员变量存储在对内存的对象中
  - 局部变量存储在栈内存的方法中
- 区别三：声明周期不同
  - 成员变量随着对象的出现而出现在堆中，随着对象的消失而从堆中消失
  - 局部变量随着方法的运行而出现在栈中，随着方法的弹栈而消失
- 区别四：初始化不同
  - 成员变量因为在堆内存中，所有默认的初始化值

- 局部变量没有默认的初始化值，必须手动的给其赋值才可以使用。

###01方法参数是基本数据类型和引用数据类型
* A.基本类型
class Demo
{
public static void main(String[] args)
{
int x = 4;
show(x);
System.out.println("x="+x);
}
public static void show(int x)
{
x = 5;

```
  }
 }
 基本类型作为参数传递时，其实就是将基本类型变量x空间中的值复制了一份传递给调用的方法show
* B.引用类型
 class Demo
 {
  int x ;
  public static void main(String[] args)
  {

   Demo d = new Demo();
   d.x = 5;
   show(d);
   System.out.println("x="+d.x);
  }
  public static void show(Demo d)
  {
   d.x = 6;
  }
 }
 当引用变量作为参数传递时，这时其实是将引用变量空间中的内存地址(引用)复制了一份传递给了s
 由于是两个引用指向同一个对象，不管是哪一个引用改变了引用的所指向的对象的中的值，其他引用
* C.结论
 * 对于基本类型形式参数改变不会影响到实际参数
 * 对于引用类型形式参数改变会影响到实际参数
```

###02封装的概述
* A.面向对象三大特征
* 封装、继承、多态
* B.封装表现
* 1、方法就是一个最基本封装体

* 2、类其实也是一个封装体
* C.封装的好处
* 1、提高了代码的复用性
* 2、隐藏了实现细节，还要对外提供可以访问的方式。便于调用者的使用。这是核心之一，也可以理解为就是封装的概念
* 3、提高了安全性
###03封装的生活中的举例
* A.封装的生活中的举例
机箱：
一台电脑，它是由CPU、主板、显卡、内存、硬盘、电源等部件组长，其实我们将这些部件组装在一起就可以使用电脑了，但是发现这些部件都散落在外面，很容造成不安全因素，于是，使用机箱壳子，把这些部件都装在里面，并在机箱壳上留下一些插口等，若不留插口，大家想想会是什么情况。
总结：机箱其实就是隐藏了办卡设备的细节，对外提供了插口以及开关等访问内部细节的方式。
* B.总结
* 机箱其实就是隐藏了办卡设备的细节，对外提供了插口以及开关等访问内部细节的方式

###04private关键字
* A.private概述
* private可以修饰成员内容包括成员方法和成员变量
* 被private修饰的内容不能在其他类访问
* B.使用步骤
* 1、通过private修饰属性
* C.完整代码
class Person {
private int age;
private String name;

```
  public void show() {
   System.out.println("age=" + age + ",name" + name);
  }
 }
```

###01get和set方法
* A.get和set方法
* 年龄已被私有，错误的值无法赋值，可是正确的值也赋值不了，这样还是不行，那肿么办呢？
按照之前所学习的封装的原理，隐藏后，还需要提供访问方式。只要对外提供可以访问的方法，让其他程序访问这些方法。同时在方法中可以对数据进行验证。
一般对成员属性的访问动作：赋值(设置 set)，取值(获取 get)，因此对私有的变量访问的方式可以提供对应的 setXxx或者getXxx的方法。
class Person {
// 私有成员变量
private int age;
private String name;

```
 // 对外提供设置成员变量的方法
 public void setAge(int a) {
  // 由于是设置成员变量的值，这里可以加入数据的验证
  if (a < 0 || a > 130) {
   System.out.println(a + "不符合年龄的数据范围");
   return;
  }
  age = a;
 }

 // 对外提供访问成员变量的方法
 public void getAge() {
  return age;
 }
}
* 总结
 * 类中不需要对外提供的内容都私有化，包括属性和方法。
```

以后再描述事物，属性都私有化，并提供setXxx getXxx方法对其进行访问
* 注意
* 私有仅仅是封装的体现形式而已
###02私有化Person类带get,set
* 标准代码
package cn.itcast.demo05;

```
/*
 *   类描述人：
 *     属性：姓名和年龄
 *     方法：说话
 *
 *   私有化所有的属性（成员变量），必须写对应的get/set方法
 *   凡是自定义的类,自定义成员变量,应该私有化,提供get/set
 *
 *   this关键字：
 *     区分成员变量和局部变量同名情况
 *     方法中,方位成员变量,写this.
 */
public class Person {
 private String name;
 private int age;

 // set方法,变量name,age赋值
 public void setAge(int age) {
  this.age = age;
 }

 public void setName(String name) {
  this.name = name;
 }
```

```
  // get方法,变量name,age获取值
  public int getAge() {
   return age;
  }

  public String getName() {
   return name;
  }

  public void speak() {
   String  name = "哈哈";
   int age = 16;

   System.out.println("人在说话  " + this.name + "..." + this.age);
  }
 }
* 标准测试代码
 package cn.itcast.demo05;

 public class PersonTest {
  public static void main(String[] args) {
   Person p = new Person();
   //调用set方法,对成员变量赋值
   p.setAge(18);
   p.setName("旺财");
   p.speak();


   //调用get方法,获取成员变量的值
 //  System.out.println(p.getName());
 //  System.out.println(p.getAge());
  }
 }
```

###03this关键字_区分成员变量和局部变量的同名
* A.什么时候用
* 当类中存在成员变量和局部变量同名的时候为了区分，就需要使用this关键字
* B.代码
class Person {
private int age;
private String name;

```
  public void speak() {
   this.name = "小强";
   this.age = 18;
   System.out.println("name=" + this.name + ",age=" + this.age);
  }
 }

 class PersonDemo {
  public static void main(String[] args) {
   Person p = new Person();
   p.speak();
  }
 }
```

###04this内存图
* A.this内存图
* 见附件：this内存图.jpg

###01this的年龄比较
* A.需求：在Person类中定义功能，判断两个人是否是同龄人
* B.代码
class Person {
private int age;
private String name;

```
 public int getAge() {
  return age;
 }

 public void setAge(int age) {
  this.age = age;
 }

 public String getName() {
  return name;
 }

 public void setName(String name) {
  this.name = name;
 }

 public void speak() {
  System.out.println("name=" + this.name + ",age=" + this.age);
 }

 // 判断是否为同龄人
 public boolean equalsAge(Person p) {
  // 使用当前调用该equalsAge方法对象的age和传递进来p的age进行比较
  // 由于无法确定具体是哪一个对象调用equalsAge方法，这里就可以使用this来代替
  /*
   * if(this.age == p.age) { return true; } return false;
   */
  return this.age == p.age;
 }
}
```

###02随机点名器案例重构
* A.需求：随机点名器，即在全班同学中随机的找出一名同学，打印这名同学的个人信息
它具备以下3个内容：
存储所有同学姓名
总览全班同学姓名
随机点名其中一人，打印到控制台
* B.代码
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

```
/**
 * 思路：
 * 第一步：存储全班同学信息
 * 第二步：打印全班同学每一个人的信息
 * 第三部：随机对学生点名，打印学生信息
 */
public class Test {
 public static void main(String[] args) {
```

```
 public static void main(String[] args) {
  ArrayList<Student> list = new ArrayList<Student>(); //1.1创建一个可以存储多
   //1.存储全班同学信息
  addStudent(list);
   //2.打印全班同学每一个人的信息（姓名、年龄）
  printStudent(list);
   //3.随机对学生点名，打印学生信息
  randomStudent(list);
 }
 public static void addStudent(ArrayList<Student> list) {
  //键盘输入多个同学名字存储到容器中
  Scanner sc = new Scanner(System.in);
  for (int i = 0; i < 3; i++) {
   //创建学生
   Student s = new Student();
   System.out.println("存储第"+i+"个学生姓名：");
   String name = sc.next();
   s.setName(name);
   System.out.println("存储第"+i+"个学生年龄：");
   int age = sc.nextInt();
   s.setAge(age);
   //添加学生到集合
   list.add(s);
  }
 }
 /**
  2.打印全班同学每一个人的信息（姓名、年龄）
  */
 public static void printStudent (ArrayList<Student> list) {
  for (int i = 0; i < list.size(); i++) {
   Student s = list.get(i);
   System.out.println("姓名："+s.getName() +",年龄："+s.getAge());
  }
 }
 /**
  3.随机对学生点名，打印学生信息
  */
 public static void randomStudent (ArrayList<Student> list) {
  //在班级总人数范围内，随机产生一个随机数
  int index = new Random().nextInt(list.size());
  //在容器（ArrayList集合）中，查找该随机数所对应的同学信息（姓名、年龄）
  Student s = list.get(index);
  System.out.println("被随机点名的同学："+s.getName() + "，年龄:" + s.getAge(
 }
}

/**
 * 学生信息类
 */
public class Student {
 private String name; // 姓名
 private int age; // 年龄
```

```
  public String getName() {
   return name;
  }

  public void setName(String name) {
   this.name = name;
  }

  public int getAge() {
   return age;
  }

  public void setAge(int age) {
   this.age = age;
  }
 }
```

###03总结

- A.类与对象
  - 类，用于描述多个对象的共同特征，它是对象的模板。
  - 对象，用于描述现实中的个体，它是类的实例。
  - 类的定义：使用关键字class来定义java中的类
  - 格式：

```
class 类名 {

 //属性

 数据类型 变量名;

 …

 //方法

 修饰符 返回值类型 方法名(参数){    }

 …

}
```

  -

- B.创建对象：
  - 格式：
  - 类名 对象名 = new 类名();
- C.封装（private关键字）
  - 封装，把对象的属性与方法的实现细节隐藏，仅对外提供一些公共的访问方式
  - 封装的体现：
  - 变量:使用 private 修饰，这就是变量的封装
  - 方法:也是一种封装，封装了多条代码
  - 类： 也是一种封装，封装了多个方法
- D.private关键字，私有的意思
  - 它可以用来修饰类中的成员(成员变量，成员方法)
  - private的特点：
  - private修饰的成员只能在当前类中访问，其他类中无法直接访问
- E.this关键字
  - this关键字，本类对象的引用
  - this是在方法中使用的，哪个对象调用了该方法，那么，this就代表调用该方法的对象引用
  - this什么时候存在的？当创建对象的时候，this存在的
  - this的作用：用来区别同名的成员变量与局部变量（this.成员变量）
  - public void setName(String name) {
  - this.name = name;
  - }