●面向对象(构造方法 Constructor 概述和格式)(掌握)

A:构造方法概述和作用

给对象的数据(属性)进行初始化

B:构造方法格式特点

a:方法名与类名相同(大小写也要与类名一致)

b:没有返回值类型，连 void 都没有

c:没有具体的返回值 return;

```
public class TestStudent {
    public static void main(String[] args) {
        Student s = new Student();//是用无参数的构造方法创建对象
       System.out.println(s); //Student@b4c966a ： 其实从这个结果可以参数构造方法返回了一个内存地址
    }
}
class Student{
    /*
    构造方法：没有返回值，也没由 void,作用用来创建对象（在内存中开辟内存），
    如果这个没有参数的构造方法如果不写，那么系统会自己隐式加上
    没有 return
     */
    public Student(){

    }

    private String name;
    private int age;

    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;
    }
    public void show(){
        System.out.println(this.getName()+"--"+this.getAge());
    }
}
```

●构造方法的重载及注意事项(掌握)

A:案例演示

构造方法的重载

重载:方法名相同,与返回值类型无关(构造方法没有返回值),只看参数列表

B:构造方法注意事项

a:如果我们没有给出构造方法，系统将自动提供一个无参构造方法。

b:如果我们给出了构造方法，系统将不再提供默认的无参构造方法。

注意：这个时候，如果我们还想使用无参构造方法，就必须自己给出。建议永远自己给出无参构造方法

```
public class TestStudent {
    public static void main(String[] args) {
        Student s = new Student();//是用无参数的构造方法创建对象
       System.out.println(s); //Student@b4c966a ： 其实从这个结果可以参数构造方法返回了一个内存地址

       Student stu = new Student("小明",19);//创建对象的同时，给属性进行了赋值操作
       System.out.println(stu);
         stu.show();
    }
}
class Student{
    /*
    没有参数的构造方法：没有返回值，也没由 void,作用用来创建对象（在内存中开辟内存），
    如果这个没有参数的构造方法如果不写，那么系统会自己隐式加上
    没有 return
     */
    public Student(){

    }

    private String name;
    private int age;

    //带参数的构造方法
   public Student(String name,int age){  //如果写了带有参数的构造方法，那么没有参数的构造方法系统不会自动加了，必须手动写上，开发中推荐无参的构造方法，无论什么时候都加上
       this.name = name;
         this.age = age;
    }
    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;
    }
    public void show(){
        System.out.println(this.getName()+"--"+this.getAge());
    }
}
```

●给成员变量赋值的两种方式的区别

A:setXxx()方法

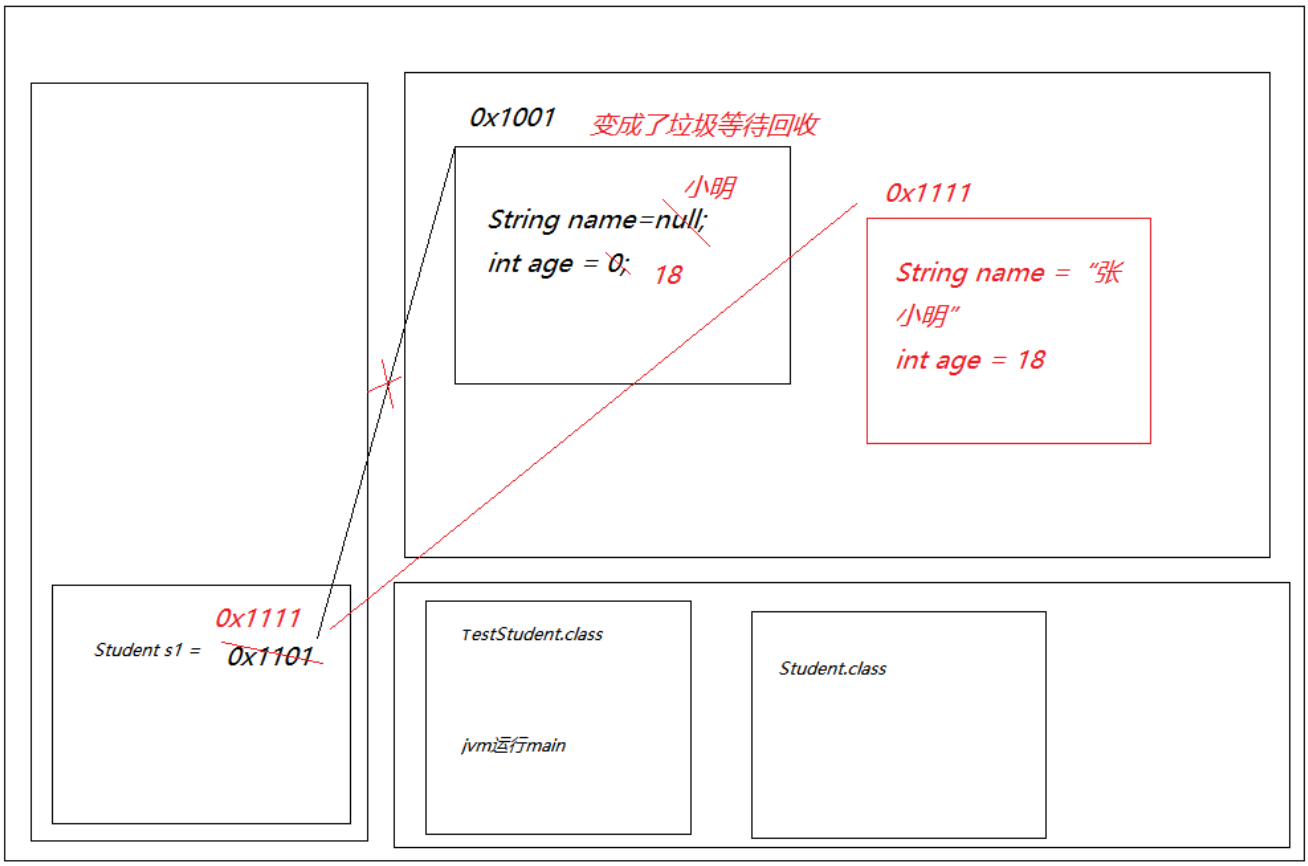
修改属性值

B:构造方法

给对象中属性进行初始化

```
public class TestStudent {
    public static void main(String[] args) {
        //在创建对象的时候初始化
       Student s1 = new Student("小明",18);
        //修改属性信息
       System.out.println("修改前："+s1);
        s1 = new Student("张小明",18);//不是修改小明的姓名，是新创建了一个新对象

       System.out.println("修改后："+s1);
        s1.setName("张大明");//修改姓名
       s1.show();



    }
}
class Student{
    /*
    没有参数的构造方法：没有返回值，也没由void,作用用来创建对象（在内存中开辟内存），
    如果这个没有参数的构造方法如果不写，那么系统会自己隐式加上
    没有return
     */
    public Student(){

    }

    private String name;
    private int age;
```

```
    //带参数的构造方法
   public Student(String name,int age){   //如果写了带有参数的构造方法，那么没有参数的构造方法系统不会自动加了，必须手动写上，开发中推荐无参的构造方法，无论什么时候都加上
       this.name = name;
        this.age = age;
    }
    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;
    }
    public void show(){
        System.out.println(this.getName()+"--"+this.getAge());
    }
}
```



●学生类的代码及测试(掌握)

A:案例演示

学生类：

成员变量：私有化

name，age

构造方法：

无参，带两个参

成员方法：

getXxx()/setXxx()

show()：输出该类的所有成员变量值

B:给成员变量赋值：

a:setXxx()方法

b:构造方法

C:输出成员变量值的方式：

a:通过 getXxx()分别获取然后拼接

b:通过调用 show()方法搞定

●手机类的代码及测试(掌握)

A:案例演示

模仿学生类，完成手机类代码

```
public class TestPhone {
    public static void main(String[] args) {
        Phone p1 = new Phone();
        p1.setBrand("华为");
        p1.setPrice(3888.0);
        p1.show();

        Phone p2 = new Phone("vivo",3999.0);
        p2.show();
    }
}
class Phone{
    private String brand;
    private double price;
    public Phone(){

    }
    public Phone(String brand,double price){
        this.brand = brand;
        this.price = price;
    }
```

```
    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void show(){
        System.out.println("手机的信息："+brand+"牌"+"价格："+price);
    }

}
```

●长方形案例练习(掌握)

A:案例演示

需求：

定义一个长方形类,定义 求周长和面积的方法，

然后定义一个测试类进行测试。

```
public class TestRect {
    public static void main(String[] args) {
        Rect r = new Rect(23,45);
//        System.out.println(r.cicre());//如果调用方法是没有返回值的用户再打印语句中
      r.cicre();
        System.out.println("面积是："+r.getArea());
}
}
//创建一个矩形类
class Rect{
    private double width;
    private double high;
    public Rect(double width,double high){
        this.width = width;
        this.high = high;
    }
    //求周长
   public void cicre(){
        System.out.println("周长是："+(width+high)*2);
```

```
    }

    //求面积
    public double getArea(){
        return width*high;
    }
}
```

●员工类案例练习(掌握)

A:案例演示

需求：定义一个员工类 worker

自己分析出几个成员，然后给出成员变量

姓名 name,工号 id,工资 salary

构造方法，

空参和有参的

getXxx()setXxx()方法，

以及一个显示所有成员信息的方法。并测试。

```
public class TestWorker {
    public static void main(String[] args) {
        Worker w = new Worker("李工",100,20000);
        w.showMsg();
    }
}
class Worker{
    private String name;
    private int id;
    private double salary;
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return name;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getSalary() {
        return salary;
    }
```

```
    public void setSalary(double salary) {
        this.salary = salary;
    }

    public Worker(){

    }
    public Worker(String name,int id,double salary){
        this.name = name;
        this.id = id;
        this.salary = salary;
    }
    public void showMsg(){
        System.out.println(name+"-"+id+"-"+salary);
        System.out.println(this.getName()+"--"+this.getId()+"--"+getSalary());
    }
}
```

用数组装对象：

```
import java.util.Arrays;

public class TestWorker {
    public static void main(String[] args) {
        Worker w = new Worker("李工",100,20000);
        w.showMsg();
        Worker[] ww = new Worker[5];//创建一个数组用来存储5 个员工对象
       ww[0]= new Worker("张工",101,12000);
        ww[1]= new Worker("谭工",102,13000);
        ww[2]= new Worker("老工",103,12500);
        ww[3]= new Worker("刘工",104,12700);
        ww[4]= new Worker("打工",105,18000);
//         System.out.println(Arrays.toString(ww));
        for (int i = 0; i <ww.length ; i++) {
            Worker wor = ww[i];//从数组取出来的都是对象的引用（对象的内存地址）
          wor.showMsg();
        }
    }
}
class Worker{
    private String name;
    private int id;
    private double salary;
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return name;
    }
```

```
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public Worker(){

    }
    public Worker(String name,int id,double salary){
        this.name = name;
        this.id = id;
        this.salary = salary;
    }
    public void showMsg(){
        System.out.println(name+"-"+id+"-"+salary);
//        System.out.println(this.getName()+"--"+this.getId()+"--"+getSalary());
    }
}
```

学生管理系统

用数组装所有的学生信息，然后可以进行添加学生，删除学生，修改学生

```
public class Demo1 {
    public static void main(String[] args) {
        //判断两个String类型的值是否相同
       String s1 = "长沙";
        String s2 = "长沙";  //常量在常量池中
       boolean flag = s1 ==s2; //用来对比String类型两个值结果不一定是true(学String类再详细讲)
       System.out.println(flag);


        boolean flag2 = s1.equals(s2);//开发中用这个equals来比较字符串是否相等
       System.out.println(flag2);
    }
}
```

●static 关键字及内存图(了解)

A:案例演示

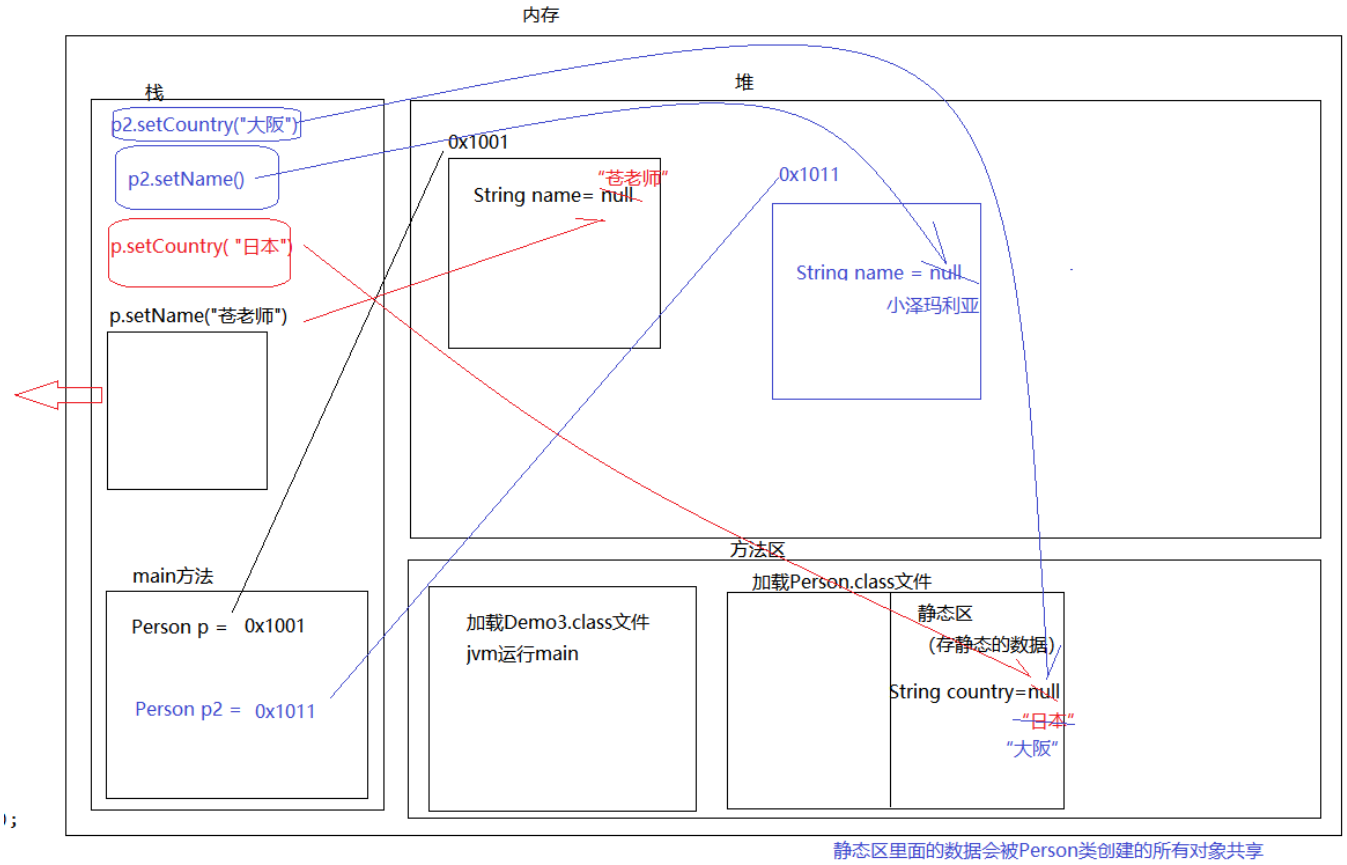
通过一个案例引入 static 关键字。
人类：Person。每个人都有国籍，中国。

B:画图演示

带有 static 的内存图

```
public class Demo3 {
    public static void main(String[] args) {
        Person p = new Person();
        p.setName("苍老师");
        p.setCountry("日本");
        p.show();
        System.out.println("--------");

        Person p2 =new Person();
        p2.setName("小泽玛利亚");
        p2.show();
        p2.setCountry("大阪");

        System.out.println("---------");
        p.show();
        p2.show();
    }
}
class Person{
    private String name;
    static String country;//国籍

   public String getName() {
        return name;
    }

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

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        Person.country = country;
    }

    public void show(){
        System.out.println(name+"--"+country);
```

```
    }
}
```




```
public class Demo3 {
    public static void main(String[] args) {
        Person p = new Person();
        p.setName("苍老师");
        Person.country = "日本";  //可以通过类型.变量名访问静态的成员变量，也是它不需要创建对象也可以访问，开发中推荐直接用类名.类访问
       p.show();

    }
}
class Person{
    private String name;
    static String country;//国籍

   public String getName() {
        return name;
    }

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

    public String getCountry() {
        return country;
```

```
    }

    public void setCountry(String country) {
        Person.country = country;
    }

    public void show(){
        System.out.println(name+"--"+country);
    }
}
```

●static 关键字的特点(掌握)

A:static 关键字的特点

static 修饰的变量或方法

a:随着类的加载而加载

b:优先于对象存在

c:被类的所有对象共享

举例：咱们班级的学生应该共用同一个班级编号。

其实这个特点也是在告诉我们什么时候使用静态?

如果某个成员变量是被所有对象共享的，那么它就应该定义为静态的。

举例：

饮水机用静态

水杯不要静态

(共性用静态,特性用非静态)

d:可以通过类名调用

其实它本身也可以通过对象名调用。

推荐使用类名调用。

静态修饰的内容一般我们称其为：与类相关的，类成员

B:案例演示

static 关键字的特点

```
class Demo {
    int num1 = 10;                          //非静态的成员变量
    static int num2 = 20;                   //静态的成员变量

    public void print1() {                  //非静态的成员方法
        System.out.println(num1);
        System.out.println(num2);
    }

    public static void print1() {           //静态的成员方法
        System.out.println(num1);
        System.out.println(num2);
    }
}
```

```
public static void main(String[] args) {
    Demo d = new Demo();
    d.print1();
}
```

既可以访问静态成员　也可以访问非静态成员

静态方法不能访问非静态的成员　　Demo.Print();　报错

```
public class Demo4 {
    static int num = 10;
    public static void main(String[] args) {
        show();
        Demo4.show();
        System.out.println(num);//问什么报错？

       Demo4 d = new Demo4();
        d.display();
    }
    public static void show(){
        System.out.println("show 方法");
    }

    public void display(){  //非静态的方法
       System.out.println("display 方法"+num);
        show();

    }
}
/*
   静态的方法中只能调用静态的方法和属性
   非静态的方法中可以调用静态的方法和属性（后进内存的可以访问先进内存的）
 */
```

●static 的注意事项(掌握)

A:static 的注意事项

a:在静态方法中是没有 this 关键字的

如何理解呢?

静态是随着类的加载而加载，this 是随着对象的创建而存在。

静态比对象先存在内存中。

b:静态方法只能访问静态的成员变量和静态的成员方法

静态方法：

成员变量：只能访问静态变量

成员方法：只能访问静态成员方法

非静态方法：

成员变量：可以是静态的，也可以是非静态的

成员方法：可是是静态的成员方法，也可以是非静态的成员方法。

简单记：

静态只能访问静态。非静态可以访问静态（关键点：搞清楚谁先进入内存的问题）

B:案例演示

static 的注意事项

```
public class Demo4 {
    static int num = 10;
    public static void main(String[] args) {
        Demo4 d1 = new Demo4();
        d1.getMsg();
        System.out.println(d1);
       Demo4.show();//静态的方法可以被对象的引用调用也可以直接用类型.调用

    }
    public static void show(){
        System.out.println("show 方法");
    }
    public static void show2(){
//        this.show();//为什么前面没有隐含 this?
        //show2 是个静态的方法，随着 Demo4 类加载而加载进了内存，此时还没有创建对象，this 是指对象的引用，
那么这里不能有 this
    }

    public void display(){  //非静态的方法
      System.out.println("display 方法"+num);

    }
    public void getMsg(){  //非静态的方法
      System.out.println("getMsg 方法");
        this.display();//非静态的方法调用非静态方法
        System.out.println("this 的值："+this);
    }
}
```

●静态变量和成员变量的区别(掌握)

静态变量也叫类变量 成员变量也叫对象变量

A:所属不同

静态变量属于类，所以也称为为类变量

成员变量属于对象，所以也称为实例变量(对象变量)

B:内存中位置不同

静态变量存储于方法区的静态区

成员变量存储于堆内存

C:内存出现时间不同

静态变量随着类的加载而加载，随着类的消失而消失

成员变量随着对象的创建而存在，随着对象的消失而消失

D:调用不同

静态变量可以通过类名调用，也可以通过对象调用

成员变量只能通过对 象名调用

生活举例：

静态与非静态的区别

静态方法里面不能调用非静态的数据(属性，方法)

非静态的方法里可以调用静态的数据(属性，方法)

理解方法：理清楚谁先进入内存，后进入内存的可以调用先进入内存的数据，反之不行

●main 方法的格式详细解释(了解)

A:格式

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

B:针对格式的解释

public： 公共的，因为 main 要被 jvm 调用，需要是公共的

static ：静态的，main 是程序的入口，那么需要被 jvm 还没有对象的时候可以直接通过类名.main 来调用，jvm 内部默认绑定了 main 方法

void：没有返回值，运行 main 方法，不需要返回任何数据给 jvm

main：是个方法名，是关键字，java 底层绑定这个单词

String[ ] args ：字符串数组

C:演示案例

通过 args 接收键盘例如数据

```
public class Demo5 {
    public static void main(String[] args) {
        for (int i = 0; i <args.length ; i++) {
            System.out.println(args[i]);
        }
    }
}
```

```
E:\>javac Demo5.java

E:\>java Demo5 在吗 我在
args[0]在吗
args[1]我在
```

●工具类中使用静态(了解)

A:制作一个工具类

ArrayTool

1,获取整型数组中的最大值元素

2,数组的遍历

3,数组的反转

```
/**
 * 这是一个工具类：用来操作数组功能
 * @author :TeacherLiu
 * @since :14.0
```

```
 * @version 1.0
 */
public class ArrayTools {
    /*
    工具类一般情况下推荐用户使用类名.方法名调用，不希望用户创建工具类的对象来调用所以，
    把工具类的无参的构造方法私有化，不让用户创建对象
     */
    private ArrayTools(){}
    /**
     * 这是一个获取整型数组中的最大值元素的方法
    * @param arr :传入一个整型的数组
    * @return :返回数组中最大的元素
    */

    public static int getMax(int[] arr){
        int max = arr[0];
        for (int i = 0; i <arr.length ; i++) {
            if(max < arr[i]){
                max = arr[i];
            }
        }
        return max;
    }
    /**
     * 这是一个用来数组的遍历的方法
    * @param arr :用来接受一个整型的数组
    */

    public static void printArr(int [] arr){
        for(int a:arr){
            System.out.print(a+"\t");
        }
    }
    /**
     * 用来处理数组的反转的方法
    * @param arr :传入一个整型的数组
    * @return :返回一个反转之后的新数组
    */
    public static int[] reverse(int []arr){
        for (int i = 0; i <arr.length/2 ; i++) {
            int temp = arr[i];
            arr[i] = arr[arr.length-1-i];
            arr[arr.length-1-i] = temp;
        }
        return arr;
    }
}
```

●如何使用 JDK 提供的帮助的 API 文档(了解)

A:找到文档，打开文档
B:点击显示，找到索引，出现输入框
C:你应该知道你找谁?举例：Scanner
D:看这个类的结构(需不需要导包)
成员变量 字段
构造方法 构造方法
成员方法 方法

```
import java.util.Scanner;

public class Demo7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
//        int a = sc.nextInt();
//        double b = sc.nextDouble(); //从键盘接收的数据返回了double 类型
//        System.out.println(a);
//        System.out.println(b);
        String s = sc.nextLine();//读取一行文本
      System.out.println(s);

    }
}
```

●学习 Math 类的随机数功能(了解)

A:Math 类概述
含用于执行基本数学运算的方法
B:Math 类特点
由于 Math 类在 java.lang 包下，所以不需要导包。
因为它的成员全部是静态的,所以私有了构造方法
C:获取随机数的方法
public static double random():返回带正号的 double 值，该值大于等于 0.0 且小于 1.0。
D:我要获取一个 1-100 之间的伪随机数（“伪”：不是真实存在,计算出来的），肿么办?
int number = (int)(Math.random()*100)+1;

```
import java.util.Random;

public class Demo8 {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        int max = Math.max(a,b);
        System.out.println(max);
          int min  = Math.min(a,b);
        System.out.println(min);

        for (int i = 0; i <20 ; i++) {
            double d=  Math.random();//返回一个0.0~1.0 之间的随机数
```

```
            System.out.print(d+"\t");
        }
        System.out.println();
        //如果需要一个0~9之间的随机整数
       int num = (int)(10*Math.random());
        System.out.println(num);

        Random r = new Random();
        int num2 = r.nextInt(10);//0~9之间的随机数
       System.out.println(num2);
    }
}
```

●猜数字小游戏案例(了解)

A:案例演示

需求：猜数字小游戏(数据在 1-100 之间)

```
import java.util.Scanner;

public class Demo9 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while(true){
            System.out.println("请输入一个整数：");
            int num = sc.nextInt();
            guessNum(num);
        }

    }
    //猜数字小游戏(数据在1-100之间)
    static int n = getNum();
    public static void guessNum(int num){
        if(num>n){
            System.out.println("大了");
        }else if(num<n){
            System.out.println("小了");
        }else{
            System.out.println("猜对了，你要猜的数字是："+n);
            System.exit(0);
        }
    }

    //返回一个1~100之间的随机数
   public static int getNum(){
        double a = Math.random();//0.0~1.0之间的随机数
       int num = (int)(a*100+1);
        return num;
```

```
    }

}
```