

面向对象基础

核心：一切行为，让对象自己做自己的事情，其他人不能手动操作

前提案例：

案例演示：

结构图：

案例代码：

1、类和对象

2、对象属性

属性的类型

属性名称

3、对象方法

模拟语境训练：

变量

什么是变量

变量的分类

变量的修饰类型

基本变量类型

探讨整型（byte, short, int, long）取值范围

原码，补码，反码

注意1：

注意2：

注意3：

字符型

浮点型

布尔类型

字面量=字面值

整数字面值

浮点类型字面量

字符和字符串字面量

命名规则

作用域

字段，属性，field—（成员变量—类—模板的组件）

参数

局部变量

作业

Final

声明时候赋值

在声明的时候没有赋值

Final可以修饰类也可以修饰方法

参数中也可以使用final

表达式

代码块

变量的初始化TODO

操作符

算术操作符

关系操作符

逻辑操作符

位移操作符

赋值操作

三元操作符：三目运算

控制流程（选择语句+循环）

if语句

If语句的坑

If else

else if

Switch

选择语句练习题：

循环

面向对象基础

核心：一切行为，让对象自己做自己的事情，其他人不能手动操作

前提案例：



案例演示：

需求：创建一个女神泪



女神泪装备类:

名称

价格

效果

Left-装备

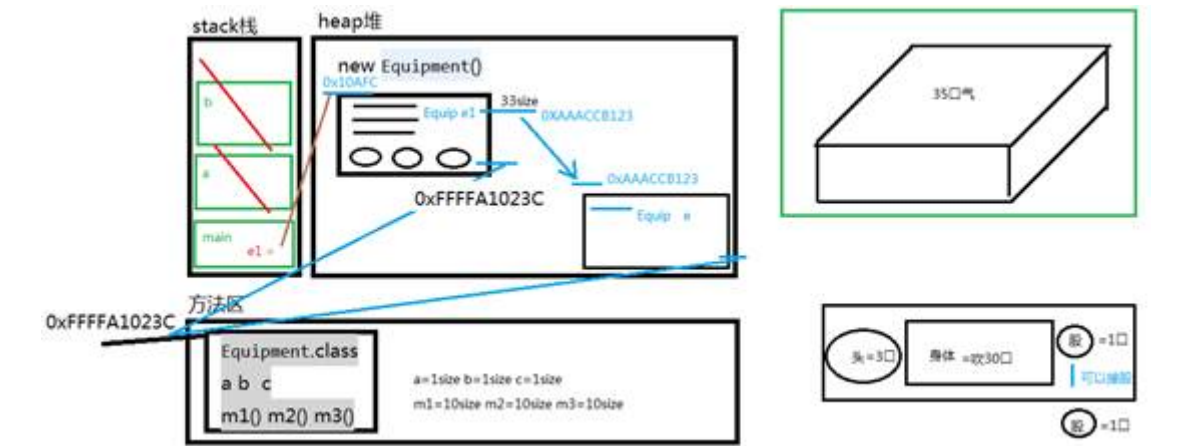
Right-装备

被动功能: f(){}

主动功能

默认功能: 合成其他装备

结构图:



案例代码:

```
1  /**
2   * <p>Title: Demo5.java</p>
3   * <p>Description:
```

```

4  女神泪装备类:
5  名称
6  价格
7  效果
8  Left-装备
9  Right-装备
10 被动功能: f(){}
11 主动功能
12 默认功能: 合成其他装备
13  * </p>
14  * <p>Copyright: Copyright (c) 2017</p>
15  * <p>Company: com.haoyu</p>
16  * @author 大师
17  * @date 2019年7月31日
18  * @version 1.0
19  */
20 public class Demo5 {
21     public static void main(String[] args) {
22         // a();
23         //蓝水晶装备
24         Equipment leftE=new Equipment();
25         leftE.name="蓝水晶";
26         leftE.price=10;
27         leftE.main="加法力量";
28         //仙女护符装备
29         Equipment rightE=new Equipment();
30         rightE.name="仙女护符";
31         rightE.price=10;
32         rightE.main="恢复法力";
33         //女神泪装备
34         Equipment girlEyewater=new Equipment();
35         girlEyewater.name="女神泪";
36         girlEyewater.price=375;
37         girlEyewater.main="加250法力";
38         girlEyewater.fix(leftE, rightE);
39         //合成一个大天使
40         Equipment bigAngle=new Equipment();
41         bigAngle.name="大天使";
42         bigAngle.fix(girlEyewater.getOut(),null);//如果没有对象, 可以使用null代
替
43
44     }
45     static void a() {
46         b();
47     }
48     static void b() {
49
50     }
51 }
52 //需要创建装备对象, 需要装备模型
53 class Equipment{
54     //模型有了, 现在要搞他的组件
55     // 名称
56     String name;
57     // 价格
58     int price;
59     // 效果
60     String main;

```

```

61 // Left-装备, 理论上应该有这个类型的其他装备, 实际上现在没有任何东西
62 //类属性 类组件 类成员变量
63 Equipment leftEquip;//
64 // Right-装备
65 Equipment rightEquip;
66 //主动
67 void f1() {
68     System.out.println("主动功能");
69 }
70 //被动
71 void f2() {
72     System.out.println("被动功能");
73 }
74 //自己与其他装备融合
75 void fix(Equipment leftEquip, Equipment rightEquip) {
76     //代码有一个就近原则, 谁离这个变量近, 这个变量就是谁
77     //为了让成员变量, 组件, 起作用, 使用this关键字
78     this.leftEquip=leftEquip;
79     this.rightEquip=rightEquip;
80 }
81 //把自己提供给其他装备, 供其融合
82 Equipment getOut() {
83     return this;//this=0x10AFC=new Equipment()=(girlEyewater=new
Equipment())
84 }
85 }
86

```

1、类和对象

类: class 对象模型, 模板

对象: new出来的 具体的东西

2、对象属性

属性: 这里主要强调模板属性, 能够提供特别信息和功能的组件

属性的类型

String: 字符串—用来存储一串文字

Int: 用来存储整数

Float: 用来存储小数 (浮点数)

属性名称

自定义——驼峰命名格式, 首字母小写

3、对象方法

方法: 函数, 特殊功能, 执行内容—method-function—js

特点:

返回值——一个方法执行之后要返回信息，将信息存进变量中，提供这个有值的变量，供其他地方使用

方法名称——驼峰命名，首字母小写

方法参数——自定义传不传参数，传多少个，传的类型，传递的对象的顺序

模拟语境训练:

游戏：组件—价格，房间号，被开被关 方法—我嘿你 你嘿我

游客：组件—体重 方法—开，关门 跑 嘿嘿，被嘿

教练：组件—衣服 方法—tip, run, 被嘿，嘿嘿

阐述需求：

游客给钱，选择 门，开门，关门，教练tip, run，嘿嘿，游客被嘿

```
1 package com.haoyu;
2
3 public class Demo6 {
4     public static void main(String[] args) {
5         Game game=new Game();
6         game.run(1500);
7     }
8 }
9 //顾客
10 class Customer{
11     //体重
12     int weight=1000;
13     //名字
14     String name="游客";
15     //方法
16     //开门
17     void openDoor(Room room) {
18         System.out.println("游客开门");
19         room.openDoor();
20     }
21     //关门
22     void closeDoor(Room room) {
23         System.out.println("游客关门");
24         room.closeDoor();
25     }
26     //跑
27     void run(Game game) {
28         System.out.println("游客运动之前的体重"+this.weight);
29         System.out.println("游客跑起来");
30         //if 如果
31         if(game.price==500) {
32             this.weight-=5;//this.weight=this.weight-5;
33         }else if(game.price==1000) {
34             this.weight-=10;
35         }else {
36             this.weight-=20;
37         }
38     }
39     //主动技能
```

```

40     void function1(waiter waiter) {
41         System.out.println(this.name+" 我嘿你: "+waiter.name);
42         System.out.println("游客体重为: "+this.weight);
43     }
44     //被动技能
45     void function2(waiter waiter) {
46         System.out.println("你嘿我: ");
47         waiter.function1(this);
48     }
49 }
50 //服务员
51 class waiter{
52     //cloth衣服Status 状态  boolean =true false
53     boolean clothStatus=true;
54     String name="服务员";
55     //tip
56     void tip(Game game) {
57         if(game.price==1500) {
58             System.out.println("如果我追到你, 我就把你嘿嘿嘿!!!");
59         }else {
60             System.out.println("你追我, 嘿嘿嘿!!!");
61         }
62     }
63     void run() {
64         System.out.println("waiter--run");
65     }
66     //主动技能
67     void function1(Customer customer) {
68         System.out.println(this.name+" 我嘿你: "+customer.name);
69         System.out.println("游客体重为: "+customer.weight);
70     }
71     //被动技能
72     void function2(Customer cutomer) {
73         System.out.println("你嘿我: ");
74         cutomer.function1(this);
75     }
76 }
77 //游戏
78 class Game{
79     //价格 500-5  1000-10  1500-20
80     int price=500;//默认是0
81     //房间
82     Room room=new Room();
83     //游客
84     Customer c1;
85     //服务员
86     waiter w1;
87     //一切让游戏自动地运转起来
88     void run(int price) {
89         this.price=price;
90         before();
91         start();
92         after();
93     }
94     //加上private之后, 你看都看不到, 屁不屁?
95     //游戏的开始
96     private void before() {
97         //先准备好游客和服务员

```

```

98         c1=new Customer();
99         w1=new Waiter();
100        c1.openDoor(this.room);
101    }
102    //游戏过程
103    private void start() {
104        c1.run(this); //this--game new game
105        w1.run();
106        //游戏过程应该调用游戏方法本身
107        if(this.price==1500) {
108            wHeiC(w1,c1);
109        }else {
110            cHeiW(w1,c1);
111        }
112    }
113    //游戏的结束
114    private void after() {
115        c1.closeDoor(this.room);
116    }
117    //两种游戏方法
118    //w嘿c
119    void wHeiC(Waiter waiter, Customer customer) {
120        //游戏方法真正起作用的是玩家和服务员
121        waiter.function1(customer); //w hei c
122    }
123    //c嘿w
124    void cHeiW(Waiter waiter, Customer customer) {
125        customer.function1(waiter);
126    }
127 }
128 //房间
129 class Room{
130     //房间号
131     String rnum;
132     //房间开门状态 true 开启门 false 关闭门
133     boolean doorStatus=false;
134     //被开门
135     void openDoor() {
136         System.out.println("门自己开了");
137         this.doorStatus=true;
138     }
139     //关门
140     void closeDoor() {
141         System.out.println("门自己关闭");
142         this.doorStatus=false;
143     }
144 }
145

```

变量

看效果


```
//标题
String title;
//内容
String content;
//版本
float version;
//改版原因
String reason;
```



```
Demo7.java
4 public static void main(String[] args) {
5     Story luozha1=new Story();
6     luozha1.setTitle("震惊! 陈塘关6岁小孩被亲生父亲残忍逼死");
7     luozha1.setContent("xxxxooooo");
8     luozha1.setVersion(1.0f);
9     luozha1.setReason(null);
10    Story luozha2=new Story();
11    luozha2.setTitle("家长必看, 魔童熊小孩, 为救百姓顶住大冰雹");
12    luozha2.setContent("非限制级");
13    luozha2.setVersion(2.0f);
14    luozha2.setReason("人伦惨剧太惨了, 需要正能量");
15
16    System.out.println(luozha1);
17    System.out.println(luozha2);
18 }
19 }
```

Story [title=震惊! 陈塘关6岁小孩被亲生父亲残忍逼死, content=xxxxooooo, version=1.0, reason=null]
Story [title=家长必看, 魔童熊小孩, 为救百姓顶住大冰雹, content=非限制级, version=2.0, reason=人伦惨剧太惨了, 需要正能量]

什么是变量

变量的定义：用来命名一个数据的标识符

1949年，这个1949代表的的是一个年份

如果要给这个1949取名字，year=1949，int year=1949;

加法运算：1+2=3;

Int a=1;

Int b=2;

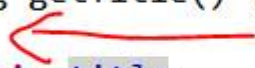
Int c=a+b;

C=3

变量的分类

局部变量：在方法中的变量


```
public String getTitle() {
    int a=1;
    return this.title;
}
```



全局变量-成员变量

```
class Story{
    //标题
    String title;
    //内容
    String content;
    //版本
    float version;
    //改版原因
    String reason;

    public String getTitle() {
        int a=1;
        return this.title;
    }
}
```



变量的修饰类型

类类型

自定义以及系统自带的类(例如: java.lang.String)

基本数据类型

Byte short char int float double long boolean

整型 (byte, short, int, long) , 字符型 (char) , 浮点型 (float, double) , 布尔型 (boolean)

基本变量类型

探讨整型 (byte, short, int, long) 滴取值范围

原码, 补码, 反码

计算机所有的运算都是补码层面进行运算, 原码层面进行呈现

正数: 原码, 反码, 补码组成内容一样

负数: 原码, 反码: 原码符号位不变, 其他位取反, 补码+1

案例:

十进制+1

原码: 0-0000001

反码: 0-0000001

补码: 0-0000001

十进制-1

补码: 1-11111111

最大值: 127:0-1111111

类型	默认值	长度	数的范围
byte		0 8位；2个字节	· -128~127: $-2^7 \sim (2^7-1)$
short		0 16位；4个字节	· -32768~32767: $-2^{15} \sim (2^{15}-1)$
int		0 32位；8个字节	· -自己去看
long		0 64位；16个字节	· -自己动脑静

例如: `byte a2=128;`

思考题:

```

1      int a=130;
2      byte b=(byte)130;
3      System.out.println(b); //-126
4      /*
5          计算机中的一切运算都是补码运算
6          int a=130;
7          a: 32位二进制 0-00000000000000000000000000000000
8                      0-000000000000000000000000010000010
9          b: 8位二进制10000010--补码
10                     1-0000010
11                     1-0000001--反码
12                     1-1111110--原码      -126
13
14      */

```

```
byte a=1;//int 1 (byte)1 基本数据类型自动向下转型 //等价于(byte) float a2=
(float)3.3;//3.3 double float a3=3.3f;
```

注意3:

一个常量是不可以被改变的

例如:

12

'A'

true

字符型

Char类型用于存放一个字符, 使用单引号表示 (双引号表示的是字符串)

其长度和short一样是16位: 二进制位 1010101010101010

单引号之内只能存放一个字符, 超过就会报错

```
//字符
static void m2() {
    char c='中';
    char c2='国';
    char c3='中国';
}

String c4="中国"
```

浮点型

Float: 32位

Double: 64位

注意: 默认的小数值就是double类型

Float f=54.321会产生编译错误,

在后面加f, 解决编译问题

float	4	32	1.4E-45~3.4028235E38	
double	8	64	4.9E-324~1.7976931348623157E308	
char	2	16	0~65535	0~2 ¹⁶ -1
boolean	1	8	true或false	true或false

布尔类型

Boolean a=true;//1

Boolean b=false;//0

作业：

请大家分别为下面的数据查找合适的类型

3.14

2.567473

365

'吃'

False

"不可描述"

答案：

```
double a=3.14;
double a2=2.567473;
int a3=365;
char a4='吃';
boolean f=false;
String s="不可描述";
```

字面量=字面值

```
Hero h=new Hero();
```

```
int a=1;
```

解释：创建一个英雄对象会用到new关键字，但是给一个基本数据类型变量赋值是不需要new关键字滴，基本类型的变量在java中是一种特别的内置数据类型，并非某个对象

定义：给基本类型变量赋值的方式就叫做字面量或者字面值

```
Float hp=120;
```

```
int armor=10;
```

整数字面值

```
1 long val=26L; //以L结尾代表long型
2 int decVal=26; //默认常数就是int
3 int hexVal=0x32; //16进制--十进制50
4 int oxVal=032; //8进制--十进制26
5 int binVal=0b1101011; //二进制写法--十进制107
6 System.out.println(binVal);
```

当以L或者l结尾的时候，一个整型的字面量是long类型，否则就是int类型。建议用L而非小写的l，容易和1混淆。

Byte, short,int,long 的字面量都可以通过int类型的字面量来创建，整数字面量可以用四种类型来表示：

二进制

八进制

十进制

十六进制

浮点类型字面量

```
1 float f1=1234.1F; //以F结尾的字面量表示浮点型
2 double d1=123.3;
3 double d2=1.2e2; //科学计数法 1.2e2--120--en 10^n
4 System.out.println(d2);
5
```

浮点类型尽量用F

浮点类型可以用科学计数法 e2表示10的2次方

字符和字符串字面量

```
1 String name="啦抓";
2 char a='拿';
3 //以下是转义字符
4 char tab='\t';
5 char enter='\r'; //回车
6 char newline='\n'; //换行
7 char doubleq='"'; //双引号
8 char singleq='\''; //字符类型单引号
9 char backslash='\\';
10 System.out.println(backslash);
```

\ 表示转义，将特殊的带有格式效果的格式符号转换成字符效果的符号

作业：

给出变量类型。赋予合法的字面量

Byte b;

Short s;

Int l;

Long L;

Float f;

Double d;

Char c;

String str;

命名规则

变量名称只能用 字母，数字，\$ _

变量名第一个字母不能使用数字

```
static void m1() {
    int a=5;
    int a_1=5;
    int $a_1=5;
    int _$a_99_$_=5;    编译会出问题
    int a234=5;
    int 1sss=1;|
}
```

使用完整英文单词名称，而非缩写

```
class Hero{
    int hp;
    String name;
    float armor;
    int moveSpeed;
}
class H{
    int h;
    String n;
    float a;
    int m;
}
```

不能使用关键字

```
int class=1;
int static=2;
int public =3;
int void =5;|
```

凡是系统自带的具有特殊意义的关键字，通通不能作为变量名称

关键字的列表

说明	类型
异常	try catch finally throws throw
对象相关	new extends implements class instanceof this super
字面值常量	1 2 3... false true null
方法	void return
包相关	package import
保留字	goto const

其他关键字

八种基本数据类型不可以

说明	类型
循环	for do while break continue
分支关键字	if else else if switch case default
方法	private public protected final static abstract synchronized transist volatile strictfp

中文词组也可以当做变量名只是不常用，建议，不要用

```

12      苍老师 苍姐=new 苍老师();
13      苍姐.开展教学工作();
14
15  }
16  static void m1() {
17      int a=5;
18      int a_1=5;

```

Markers Properties Servers Data Source Explorer

<terminated> Demo3 (1) [Java Application] C:\Program Files\jdk

牙佳虫藏虫蝶

作用域

字段，属性，field—（成员变量—类—模板的组件）

只要是成员变量就可以被整个作用域给访问到

作用域：一对大括号

参数


```

9      m3(3);
10     m4(4);
11     System.out.println(L);
12 }
13 static void m3(int k) {
14     System.out.println(k); //3
15 }
16 static void m4(int k) {
17     System.out.println(k); //4
18 }
19 static void m5() {
20     System.out.println(k);
21 }
22 static int k=2;
23 static int L=k; //2
24

```

Markers Properties Servers Data Source Explorer

<terminated> Demo4 [Java Application] C:\Program Files\jdk1.8.0_1

3
4
2

如果一个变量，是声明在一个方法上，就叫做参数

参数的作用域是该方法内所有的代码——{}包含的

其他方法和类中的其他位置，都是不可以访问这个参数的

局部变量

```

14 static void m4() {
15     int m=5; //作用范围只在15-23行，其他区域不可调用该变量
16     System.out.println(m);
17     { //代码块--区分作用域
18         System.out.println(m); //可以访问，父级作用域，包含当前的作用域
19         int n=6;
20         System.out.println(n); //可以访问，当前作用域
21     }
22 // System.out.println(n); //不能够访问到n，作用域，生存周期到21行就结束，21--}
23 }
24 {
25     {
26
27     }
28 }
29 {
30
31 }

```

定义在方法内的变量就叫做局部变量

作用域就是包含他的那对大括号{}

作业

取什么值？

```

1  public class HelloWorld{
2      int i=1;
3      public void method1(int i){
4          System.out.println(i);
5      }
6      public static void main(String[] args){
7          new HelloWorld().method1(5); //匿名类表达方式
8          //打印出来是多少
9      }
10 }
11

```

Final

声明时候赋值

```

3  public class Demo5 {
4
5      public void method1() {
6          final int i=5;
7          i=10; //i在6行已经被定义，由于是final修饰，不可再赋值
8      }
9
10 }

```

Final代表最终状态，不可以被修改

在声明的时候没有赋值

```

9      public void method2() {
10         final int i;
11         i=10; //i在10行没有被赋值，所以可以在11行被赋值
12         // i=11; //i在11行已经被赋值，由于是final修饰，最终状态，所以会出现编译错误，不可再赋值
13     }

```


如果在声明的时候没有赋值，可以在后面的代码中赋值唯一一次——final修饰的

Final可以修饰类也可以修饰方法

```

final class Person{
    final void m1() {
    }
}
class P extends Person{
}

```



Final修饰的类与方法，均不可以被继承与修改

参数中也可以使用final

```
m1(final int a)
```

表达式

以分号结尾的代码就是一个表达式

```
final int i;  
i=10; } /i在10行  
i=11; } /i在11行
```

单一的；分号，就是一个完整的表达式

```
;;;  
;  
;  
;;  
;;;;;
```

代码块

```
{ }
```

可以在一个类中的任意位置，任意嵌套，任意并排排列

变量的初始化TODO

操作符

+ - % / = > < >> << >>> ~ ^ ++ -- += -= == *= /= !=

算术操作符

+ - * / % ++ --

a++, 先做（）中a的呈现，再在括号外做自增1 ++a

++a,先自增1，然后再做呈现（参与运算）

同理--

```
1  int a=1+2;  
2      int b=a+1;// - * /  
3  
4      int c=a%2;//取余数 取模  
5      System.out.println(c);//1    [3 / 2=1---] <=> 3%2--(1)=c  
6  
7      //a=3  
8      //++ -- 前后区别即可  
9      //如果没有参与表达过程，++ -- 在前在后都完全等于 a=a+1  b=b-1  
10     // ++a;//4
```

```

11 //      System.out.println(++a);// 5
12 //      System.out.println(++a);// 6----++a 等于 每次自增1
13      System.out.println("-----");
14      System.out.println(a);//3
15      //System.out.println(a++);//3 (a)-a++
16      //a++ 呈现为原来没有增加的值 当前表达式完成后值发生了变化
17      //(a)之后, 分号; 之前 --?对的, 因为如果正确(1+a++)--(1+a)-a++
18
19      //所以推到出来a++, 先做( )中a的呈现, 再在括号外做自增1 ++a
20      System.out.println(1+a++);
21      System.out.println(a);//4
22
23      System.out.println(a++);
24      System.out.println(a);
25 //      System.out.println(a++);//4 5 6
26

```

关系操作符

关系操作符生成的是一个Boolean结果

关系操作符包括(<、>、<=、>=、==、!=)

```

1 //<、>、<=、>=、==、!=
2      System.out.println(1<2);
3      System.out.println(1>2);
4      System.out.println(1!=2);
5      System.out.println(1==2);
6      System.out.println(1<=2);
7      System.out.println(1>=2);
8 //      true
9 //      false
10 //      true
11 //      false
12 //      true
13 //      false
14

```

逻辑操作符

&& (短路与)、& (与)、|| (短路或)、| (非短路或)

值

在二进制层面，对二进制位进行移动和取舍操作，强调，所有的位移操作都是补码层面上进行的

111111111111111111111111111111111111=补码

* 111111111111111111111111111110=反码

[illegible]

* /

```
System.out.println(-5 << 3);
```

/ *

[illegible]

$11111111111111111111111111111010 = \text{反码}$

* 11111111111111111111111111111011=补码

$1111111111111111111111111011000 = \text{补码}$

```

29      *   11111111111111111111111101011=反码
    1000000000000000000000000101000=-40=原码
30      */
31      System.out.println(~2);
32      /*
33      *   0000000000000000000000000000010
    111111111111111111111111111101
34      *   111111111111111111111111111100
    10000000000000000000000000011=-3
35      */
36      System.out.println(3 ^ 8);
37      /*
38      *   000000000000000000000000000011
    00000000000000000000000000001000
39      *   0000000000000000000000000001011=11
40      */
41    }
42 }
43 /*
44 * <<表示左移，不分正负数，低位补0；
45 *
46 * 注：以下数据类型默认为byte-8位
47 *
48 * 左移时不管正负，低位补0
49 *
50 * 正数：r = 20 << 2
51 *
52 * 20的二进制补码：0001 0100
53 *
54 * 向左移动两位后：0101 0000
55 *
56 * 结果：r = 80
57 *
58 * 负数：r = -20 << 2
59 *
60 * -20 的二进制原码 ：1001 0100
61 *
62 * -20 的二进制反码 ：1110 1011
63 *
64 * -20 的二进制补码 ：1110 1100
65 *
66 * 左移两位后的补码：1011 0000
67 *
68 * 反码：1010 1111
69 *
70 * 原码：1101 0000
71 *
72 * 结果：r = -80
73 *
74 * >>表示右移，如果该数为正，则高位补0，若为负数，则高位补1；
75 *
76 * 注：以下数据类型默认为byte-8位
77 *
78 * 正数：r = 20 >> 2
79 *
80 * 20的二进制补码：0001 0100
81 *
82 * 向右移动两位后：0000 0101

```

```

83  *
84  * 结果: r = 5
85  *
86  * 负数: r = -20 >> 2
87  *
88  * -20 的二进制原码 : 1001 0100
89  *
90  * -20 的二进制反码 : 1110 1011
91  *
92  * -20 的二进制补码 : 1110 1100
93  *
94  * 右移两位后的补码: 1111 1011
95  *
96  * 反码: 1111 1010
97  *
98  * 原码: 1000 0101
99  *
100 * 结果: r = -5
101 *
102 * >>>表示无符号右移, 也叫逻辑右移, 即若该数为正, 则高位补0, 而若该数为负数, 则右移后高位
    同样补0
103 *
104 * 正数:  r = 20 >>> 2
105 *
106 * 的结果与 r = 20 >> 2 相同;
107 *
108 * 负数:  r = -20 >>> 2
109 *
110 * 注: 以下数据类型默认为int 32位
111 *
112 * -20:源码: 10000000 00000000 00000000 00010100
113 *
114 * 反码: 11111111 11111111 11111111 11101011
115 *
116 * 补码: 11111111 11111111 11111111 11101100
117 *
118 * 右移: 00111111 11111111 11111111 11111011
119 *
120 * 结果: r = 1073741819
121 */
122

```

赋值操作

= ! = > > = < < = > > =

三元操作符: 三目运算

```

1      System.out.println(1>2?true:false);
2      System.out.println(1>2?false:true);
3      System.out.println(1>2?"难受":"舒服");
4      int a=1;
5      //三目运算中的每一个表达式都可以进行无限的三目运算下去
6      System.out.println( (true&&false) || (a<=2&&true!=false)?
1>2?"a":"b":true );//b
7

```

控制流程（选择语句+循环）

- 1, if语句
- 2, switch
- 3, while, do while
- 4, for, 增强for循环 (foreach)
- 5, continue, break

if语句

```

1      boolean flag=true;
2      //最简单表达式
3      if(flag) {
4          System.out.println("ojbk");
5      }
6      //简单表达式
7      if(1>2 || 2>1) {
8          System.out.println("ojbk2");
9      }
10     //复杂表达式
11     int a=1;
12     if(a+1>2?true:true) {
13         System.out.println("ojbk3");
14     }
15

```

If语句的坑

```

1      boolean b=false;
2      if(b);//这个分号是个坑，； 代表表达式结束
3      System.out.println("yes");
4

```


If else

```
1      int score=80;
2      if(score>90) { //如果
3          System.out.println("A");
4      } else { //其他
5          System.out.println("B");
6      }
7  
```

else if

```
1      int score=80;
2
3      if(score>90) { //如果
4          System.out.println("A");
5      } else if(score>80 && score<=90) { //其他
6          System.out.println("B");
7      } else if(score>70 && score<=80) {
8          System.out.println("C");
9      } else {
10         System.out.println("D");
11     }
12 
```

注意：如果满足多个条件，第一个执行后就结束判断

```
1      int score=99;
2      if(score>80) { //如果
3          System.out.println("B");
4      } else if(score>90) { //其他
5          System.out.println("A");
6      }
7      //B
8  
```

Switch

```
1      char level = 'C'; // A B C
2      switch (level) {
3          //case 小案子 接收的内容
4          case 'A':
5              System.out.println("90-100");
6              break;
7          case 'B':
8              System.out.println("80-90");
9              break;
10         default: //等同于 if else 中的 else
11             System.out.println("80以下");
12             break;
13     }
14 
```

```
13     }
14 }
```

```
1 //byte short int char String jdk1.7之后, 可以传String enum 枚举类
2 /* 枚举类的样子
3     color 黑色=Color.黑色;
4     //color-颜色
5     enum Color{ 枚举类
6         黄色,白色,黑色;
7     }
8     */
```

Switch是if else if else if else的一种变形，格式更加简洁，操作更加方便，但是不能适应复杂表达式，只能简单判断单一的值

选择语句练习题：

```
1 //需求：排位选英雄 王者荣耀
2 //5v5
3 //每一方一个上单，中单，打野，adc，辅助
4 //现在大家水平砖石守门员，我选了你别想选
5 //108个梁山好汉
6 //一盘对局
7 //水晶被推失败
8
9 //主题：人工智能打王者荣耀
10 //1，选英雄阶段
11 //对108个英雄随机分配默认位置
12 //确定选择英雄了之后，控制台打印出这个英雄的属性，能力，默认位置，第几楼
13 //可以跟队友交换英雄，可以嘲讽队友
14 //2，对局阶段
15 //每个位置敌我双方英雄随机pk，随机死亡，当一方死亡总次数比另一方多30个头，判定数据小的一方为败方
16
```

循环

1+2+3+...+100=?

```
1 static void m2() {
2     //和
3     int sum=0;
4     //i<101
5     //i<=100
6     //int i=0
7     //i<101
8     //sum+=i;
9     //i++
10    for(int i=0;i<101;i++) {
11
12        sum+=i;
```

```

13     }
14     }
15     System.out.println(sum);
16 }
17

```

打印一个九九乘法表

1*1=1

12=2 22=4

13=3 23=6 3*3=9

解决思路

```

1 //从最简单的入手，一次改或者加一个需求，最终达到完善
2 /*
3     1，找到分析的关键对象，2*3=6以及整个关键对象所代表的大范围1*3=3 2*3=6 3*3=9
4     注释：2*3=6视为关键，原因：3个数都是不同的数，便于观察左右变化，发现，值都是左面两个变量的乘积，不用考虑
5     要考虑的是左面两个变量，由于左面的第二个变量值不变，所有只观察左面第一个变量，既然是
1 2 3 所以搞一个for循环，i从1-3
6     for(int i=1;i<=3;i++){
7         System.out.println(i+"*" +j+"="+i*j+" ");
8     }
9     结果是：1*3=3
10         2*3=6
11         3*3=9
12     2，解决上面的一个问题：不要换行如何做？由于现在我还会不换行的写法，百度查一下换行，
java--得出 print就是不换行
13     for(int i=1;i<=3;i++){
14         System.out.print(i+"*" +j+"="+i*j);
15     }
16     1*3=32*3=63*3=9
17     3，解决上面的一个问题：所有的内容挤成了一堆，表达式与表达式之间应该有一个间隔
18     解决方案：制表符'\t'但是，我忘记了怎么写，两种思路解决，第一种百度搜一下制表符，第二种先用空格简单替代一下
19     for(int i=1;i<=3;i++){
20         System.out.print(i+"*" +j+"="+i*j+" ");
21     }
22     4，解决上面的一个问题：我需要打印出三行，例如：
23     1*1=1
24     1*2=2 2*2=4
25     1*3=3 2*3=6 3*3=9
26     我先打印三行相同的内容再说
27     for(int k=1;k<=3;k++){
28         for(int i=1;i<=3;i++){
29             System.out.print(i+"*" +j+"="+i*j+" ");
30         }
31         System.out.println();
32     }打印如下
33     1*3=3 2*3=6 3*3=9
34     1*3=3 2*3=6 3*3=9
35     1*3=3 2*3=6 3*3=9
36     5，解决上面一个问题：我需要有层次性地打印33乘法表，思路就是，那个j值必须有变化，j值的变化应该是从1*3（从上到下的从1*3）

```

```

37  例如:
38      1*1=1
39      1*2=2  2*2=4
40      1*3=3  2*3=6  3*3=9  的每个式子的右面那个变量值是从上到下从1-3
41      解决思路就是把j放进一个for循环, 从1-3
42      所有最终办法是把k值变成j值, 从1-3
43      //i+"*" +j=i*j
44      for(int j=1;j<=3;j++) {
45          for(int i=1;i<=j;i++) {
46              System.out.print(i+"*" +j+"="+i*j+" ");
47          }
48          System.out.println();
49      }
50      即成功打印出33乘法表
51  6, 解决上面一个问题: 由于要打印的是九九乘法表, 所以把j值的最值3改成9
52      //i+"*" +j=i*j
53      for(int j=1;j<=9;j++) {
54          for(int i=1;i<=j;i++) {
55              System.out.print(i+"*" +j+"="+i*j+" ");
56          }
57          System.out.println();
58      }
59      得到正解
60      */
61      static void m3() {
62          //i+"*" +j=i*j
63          for(int j=1;j<=9;j++) {
64              for(int i=1;i<=j;i++) {
65                  System.out.print(i+"*" +j+"="+i*j+" ");
66              }
67              System.out.println();
68          }
69      }
70

```

作业:

请输入一个任意数字, 打印一个对应上行数的完整菱形

例如:

```

3
 *
***
*****
***
 *
1
2
 *
***
 *

```

即, 上面三角形的行数, 下面三角形的行数是上面的行数减去一行的菱形表达形式

