

# 这样学Java不枯燥

# 第一章 入门

数据类型      接口      抽象类

内部类      override      多态      overload      final      设计模式

泛型      装箱      匿名类

静态类

**学Java！好难呀！**

```
D:\Program Files\Xinox Software\JCreatorV3\G
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
Press any key to continue...
```

```
C:\Documents and Settings\Administrator\桌面\
请输入一个整数，来确定显示的行数：11
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
0 10 45 120 210 252 210 120 45 10 1
Press any key to continue
```

学Java！好枯燥呀！



如鹏  
rupeng.com

在校不迷茫 毕业即辉煌



下定决心，不怕  
牺牲，排除万难，  
去争取胜利。

毛泽东



如鹏

rupeng.com

在校不迷茫 毕业即辉煌

# 何必呢！

1998年





# 2页的Basic说明书



# 没有网络

# 一个暑假

# 无师自通



如鹏  
rupeng.com

在校不迷茫 毕业即辉煌

```
OK
LIST
1000 CLS:SPRITE ON
2000 LOCATE 9,9:PRINT "RUPENG.
3000 M=LOCATE 6,10:PRINT "CHAOJI
4000 L=
5000 X=35:Y=130
6000 FOR I=0 TO 7
7000 DEF FN MOVE(0)=SPRITE(0,I+1,
8000 1,30)
9000 POSITION 0,X,Y:MOVE 0
1000 IF MOVE(0)=-1 THEN 90
1100 X=XPOS(0):Y=YPOS(0)
1200 NEXT I
OK
```



RUPENG.COM  
CHAOJIMALI

苦苦寻找  
适合现今的  
实用  
的入门编程平台

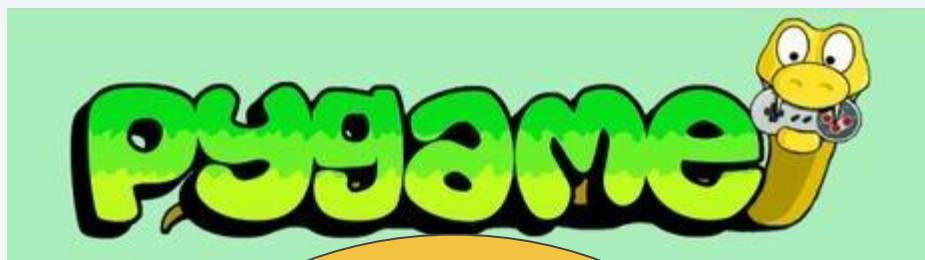




www.kidscod.cn

FLogo语言		初始	CS	HTC	智能	背图	存图	帮助	编程	X
背	前	1	恢复	CT	hom	PU	MP3	图片	执行	停止!
<pre>repeat 3 [repeat 6[fd 50 rt 360/6] rt 120]</pre>										





# Why not Swing?



如鹏  
rupeng.com

在梦不迷茫

毕业即辉煌



**没有枪，没有炮，我们自己造！**

# RupengGame

## 游戏引擎

**有用**→ Java语言占有率第一的编程语言

**有趣**→ 精灵、图片、文本、输入、音乐

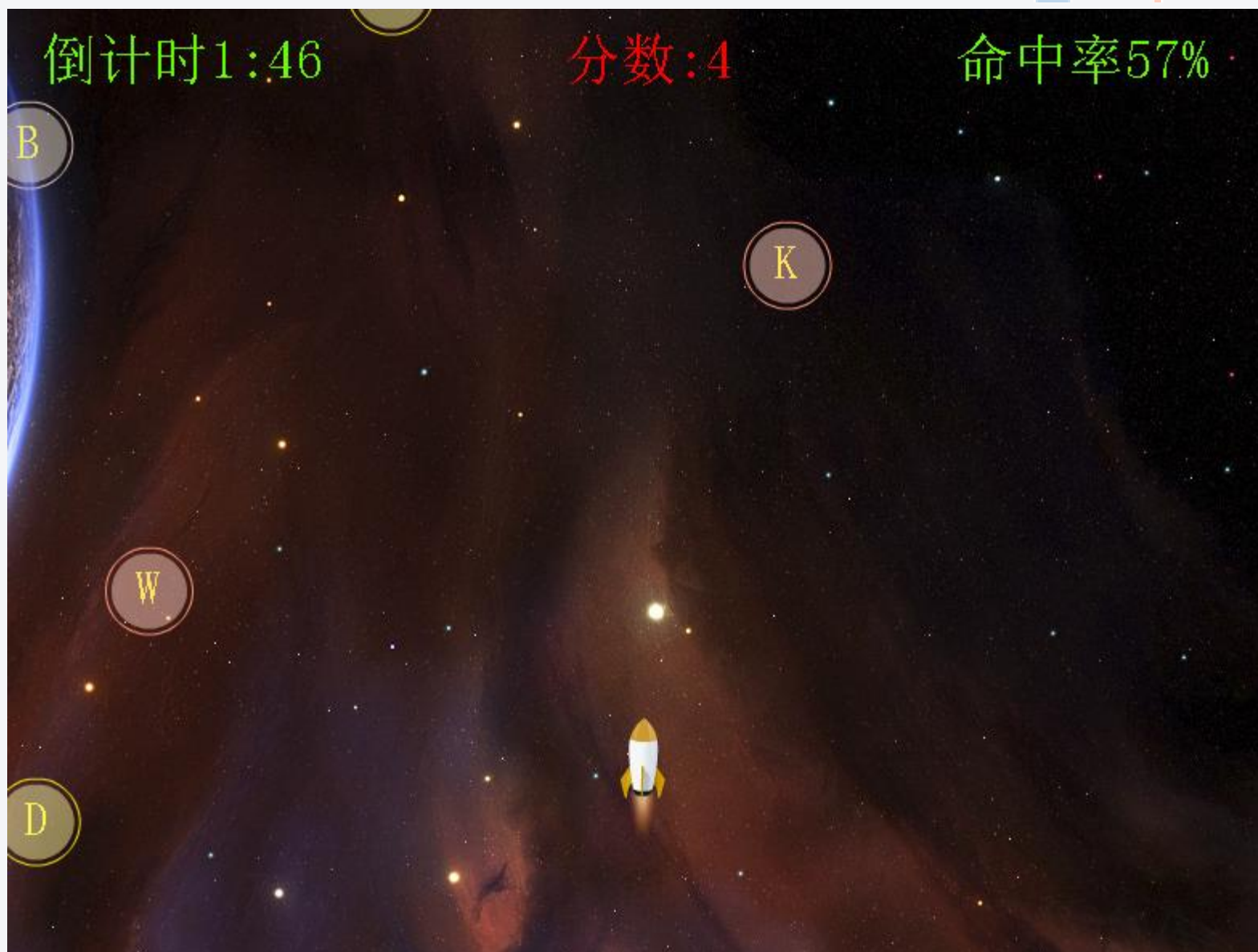
**极简**→ No “鸡生蛋、蛋生鸡”，0基础可学

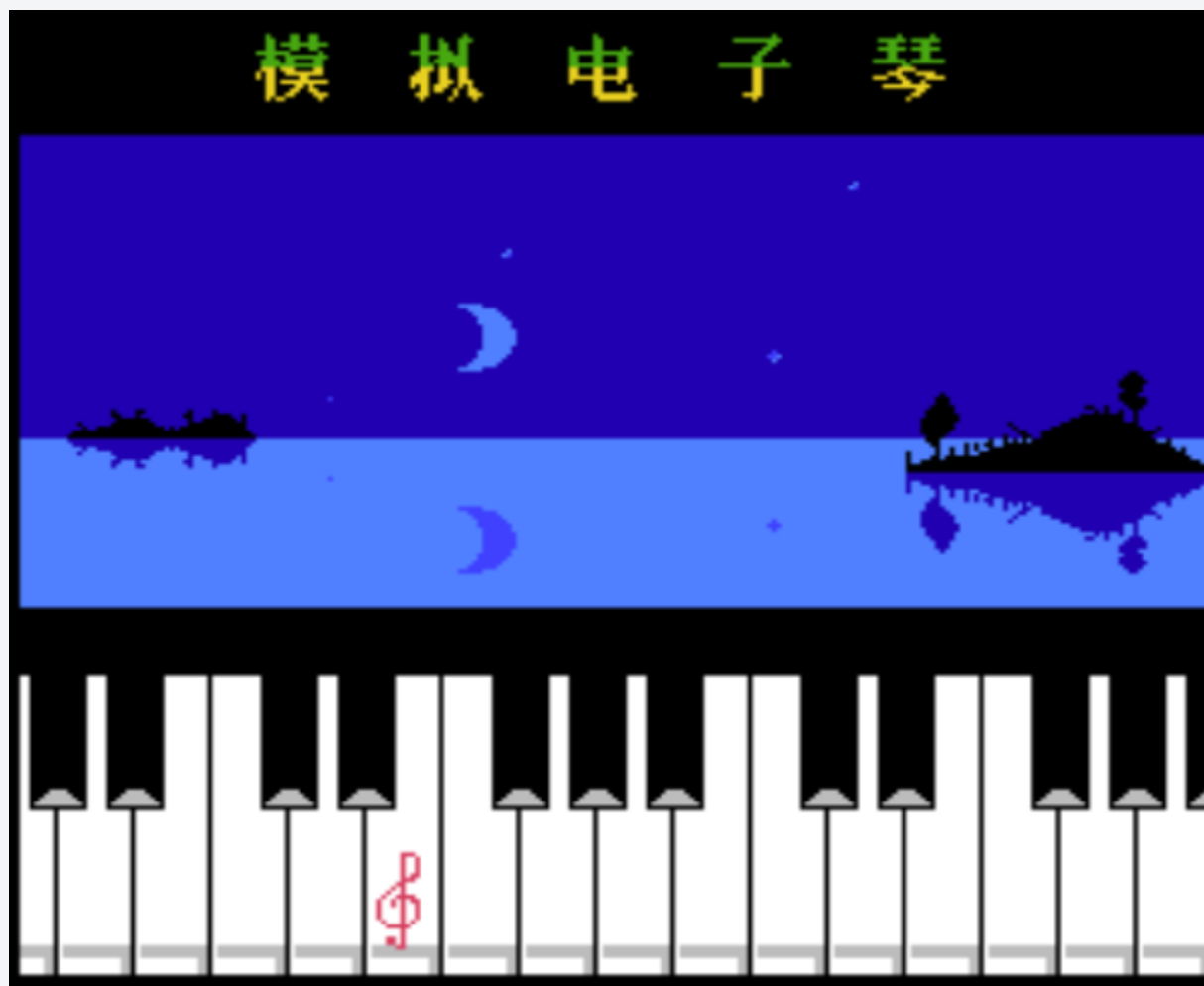


倒计时1:46

分数:4

命中率57%









# 音乐欣赏

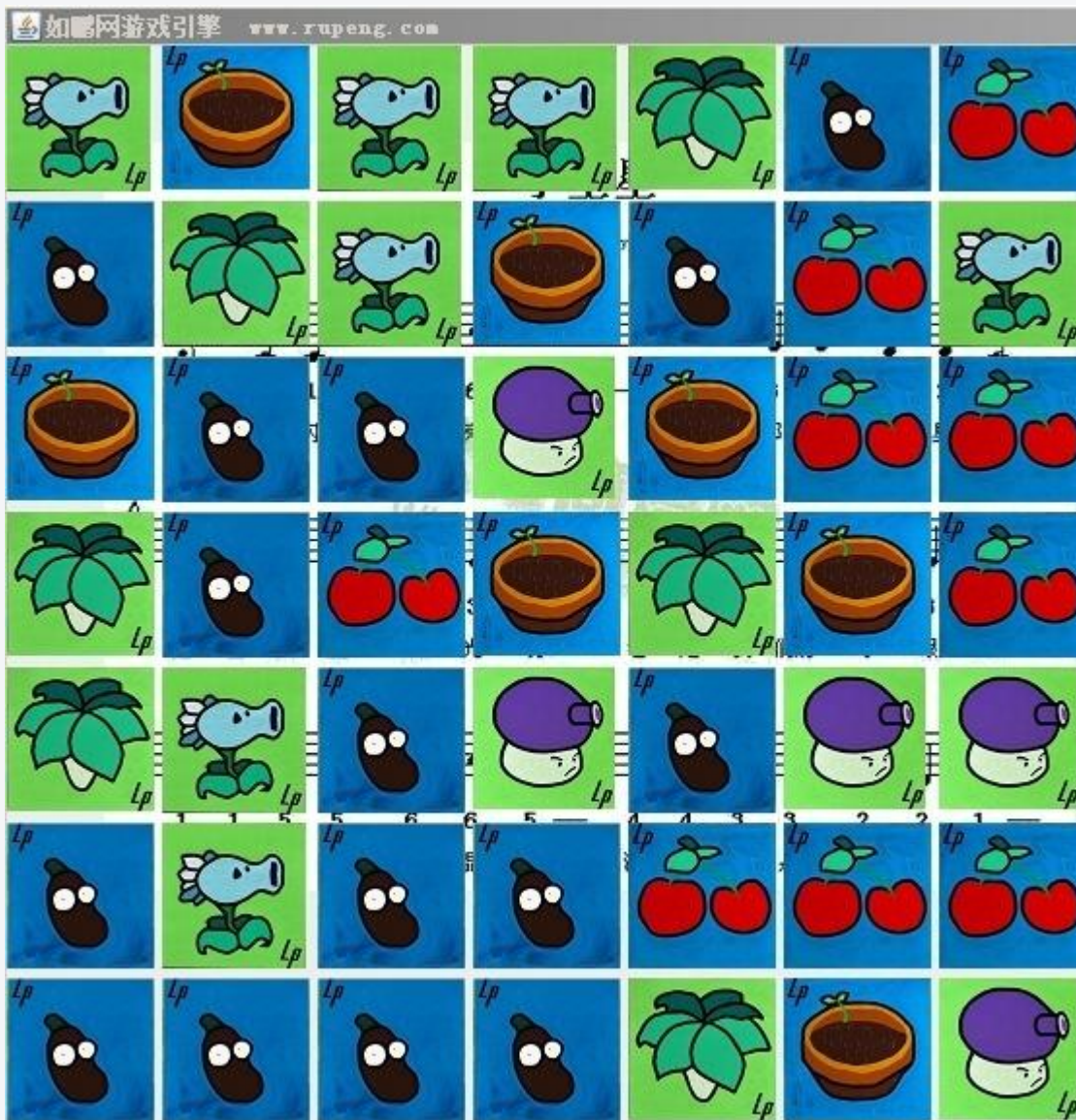


1. 蓝色多瑙河
2. 小夜曲
3. 红河谷
4. 绿袖子
5. 樱花
6. 归苏莲托
7. 啊，朋友
8. 故乡恋情歌
9. 康定情歌













- 不是完备的游戏引擎，不能开发所有的游戏；
- 掌握这个引擎不是目的，学习游戏开发不是主要目的，主要目的是**通过开发游戏来掌握编程**。除了GameCore的类，其他都是标准Java的东西，做 JavaEE 、 Android 等 的 时 候 一 样 的 。 GameCore的东西也是“一通百通”的。
- 这个游戏引擎和Unity、SDL等游戏引擎用法非常像；

# 开发工具

- 专业Java开发者需要掌握“命令行+记事本”写程序的方法，但是初学者难度太高。因此先学傻瓜化的开发环境。（下载地址在如鹏网在线视频右侧区域中，请看如鹏网的官网课程）

1、下载JDK、解压Eclipse。都分32位和64位。

2、如果配置遇到问题，

- (\*) MyEclipse和Eclipse

- 解压、运行、选择工作空

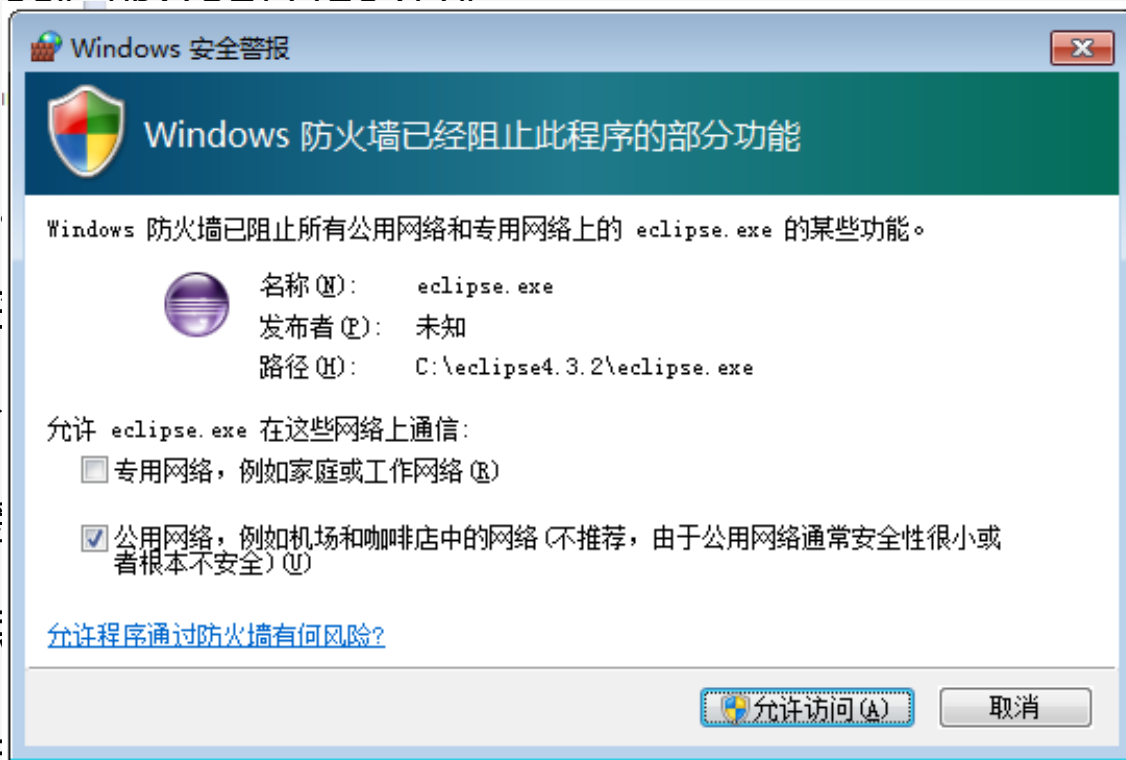
- 建项目：文件（File）→

- 在src文件夹下点右键“?

- 在文件上或者编辑器里点

（Java Application）“

部分功能”对话框，点击



# Eclipse个性化配置



- 中文字体太小：窗口（Window）→首选项（Preferences）→常规（General）→外观（Appearance）→颜色和字体（Colors and Font）→基本（Basic）→双击“文本字体（Text Font）”在弹出窗选择“Courier New”
- 我喜欢大括号在同一行：首选项→Java→代码样式（CodeStyle）→格式化程序→编辑（Edit）→花括号（braces），都选“下一行”。写一个名字保存
- 配置项目的Java编译器版本：项目右键→属性→Java编译器（Compiler）→启用特定于项目的设置，选择相应版本，建议选择1.7以上，后续讲课也默认1.7。
- 配置java的源代码，方便研究java类库源码、文档：首选项→Java→已安装的JRE→双击JRE→展开rt.jar→源代码链接，选择JDK中的src.zip
- 快捷键：Ctrl+shift+F格式化代码(如果没反应可能是和别的软件热键冲突，比如搜狗拼音，在输入法条中点击右键“设置属性”→“按键”→“系统功能快捷键”，看到了吗，把所有都取消吧)；Alt+/智能提示；Ctrl+鼠标查看定义。



# RupengGame引擎使用



- 新建一个Java项目
- RuPengGame.jar、 j1.0.jar复制到项目里，在两个文件上点右键  
→ “构建路径” → “添加到构建路径”

```
import com.rupeng.game.GameCore;
public class Main1 implements Runnable
{
    public static void main(String[] args)
    {
        GameCore.start(new Main1());
    }
    public void run()
    {
    }
}
```

# 代码结构

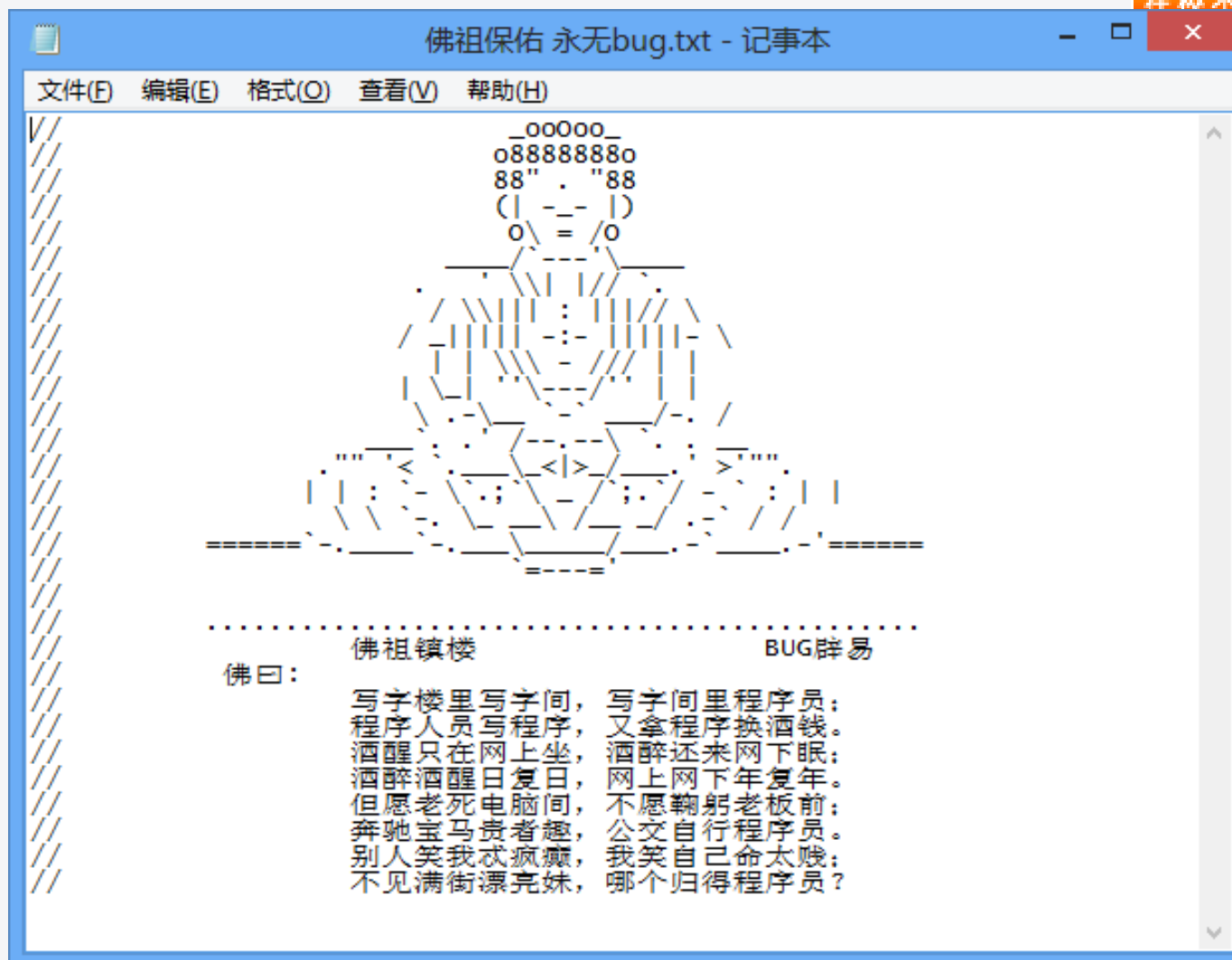
- main是程序入口。
- run 是游戏逻辑控制代码的入口，run执行结束游戏就退出。所以窗口一闪而过。
- GameCore.setGameTitle( “哥的第一个如鹏游戏” );修改游戏窗口的标题
- GameCore.pause(n);可以让执行过程暂停n毫秒
- setGameSize(int width, int height)：设置游戏窗口尺寸
- GameCore.exit()：程序立即直接结束

# 常见错误

- 语句结束应该是英文分号，而不是中文分号。
- 字符串声明应该是英文引号，而不是中文引号。
- 拼写错误（ mian ）；Java是大小写敏感：Main和main不一样。
- 如果调试启动失败，看编辑器或者“问题（ Problem ）”视图中的错误信息。出了错误别慌，认真先看“问题（ Problem ）”视图。

# 注释

- 被注释的代码编译器忽略。作用：说明代码的作用：程序中适当加注释；屏蔽无用的代码。
- 使用 “//” 注释一行代码，//之后的代码编译器会忽略（给例子代码加注释）。
- /\* \*/ 注释多行代码。
- 注释不用太多、也不能太少



# 顺序执行

- 1、每句代码以分号结束。
- 2、从上向下一条执行，上一条执行结束后下一条才会执行。
- 3、有的方法执行完成的慢，有的完成的快。  
`setGameTitle`、`setGameSize`很快完成，`pause`则要等暂停时间到了之后才完成。

验证：

```
GameCore.pause(1000);  
GameCore.setGameTitle("如鹏网");  
GameCore.pause(1000);  
GameCore.setGameTitle("rupeng.com");  
GameCore.pause(10000);
```

注释掉`pause`呢？

练一下：窗口宽度从400开始每次增长100，增长到800

# 概念

- **方法**（函数）：让计算机去做什么事情的“**指令**”。
- **参数**：调用“方法”（发布指令）的时候给的数据。“暂停多少毫秒？”、“设置为什么标题”
- **数据类型**：  
字符串（string） “rupeng.com”  
整数（int） 3
- “33” 和33不一样。
- 问：alert("12+21");alert(12+21);

# 加载背景图

- `GameCore.loadBgView(String imgName)`：加载游戏背景图，参数是图片文件的全名（包含后缀名）。
- 图片必须放到src的**Images**包（src上点右键→新建→包）下。



# 消息框

- `alert(Object msg)` 显示弹出框消息：  
`alert( "rupeng.com" )`
- 每个方法都是“指令执行结束后再执行下一条指令”，  
`alert`是“对话框关闭后执行结束”。
- 显示消息什么类型都可以：`GameCore.alert(5)` (\*)因为  
`alert`的参数是`Object`类型，是所有类的父类（面向对象时候会讲）。

# 播放音乐

- 音乐必须放在src的Sounds文件夹下，只支持mp3格式文件（只是把后缀改成mp3没卵用）
- playSound(String soundName, boolean repeat) 第一个参数为音乐文件名；repeat是boolean类型（只有true、false两个取值的类型），false表示只播放一次，true表示一直重复播放（比如游戏背景音乐）。测试：播放“狗叫.mp3”。
- playSound不等音乐播放结束后才结束，而是“启动播放”后结束，但是音乐还在继续。测试：播放“Hot.mp3”，然后alert。
- closeSound(String soundName) 关闭音乐的播放（包括重复播放的）。

# 关于项目（ Project ）



- 什么时候新建一个项目：相关功能放到一个项目中；（\*）一个游戏一个项目。
- 工作空间（WorkSpace）：相关的项目放在一个工作空间中，避免一个工作空间中项目太多，也不便于管理。
- 切换工作空间的方法：退出重启，如果勾选了【将此值用作初始值并且不再询问】，则文件(File)→切换工作空间(switch workspace)，从“工作空间”列表中选择（之前打开过的）或者浏览选择工作空间根目录（包含.metadata）。试一下：“资源下载”下载我的工作空间。
- 如何导入别人的项目到工作空间中：包资源管理器（Package Explorer）→导入（Import）→常规（General）→现有项目到工作空间中（Existing Projects into Workspace），对话框中选中“选择根目录（Select Root Directory）”→【浏览】→选择项目的根目录（包含.project文件）或者WorkSpace，选择要导入的项目即可。

## 第二章 变量和数据类型

# 你怎么不讲XXX?

# 我们的教学理念



- 由浅入深，不一开始就讲复杂的原理。符合认知规律：学习→练习→总结提升到理论体系→提高。
- 先学习零散的知识点，再总结。

规避对初学者来讲不必要的“陷阱”，降低学习的

# 挫败感

程序由**指令**和**数据**组成。

```
GameCore.setGameSize(1000, 800);  
GameCore.setGameTitle("哥的第一个如鹏游戏");  
GameCore.alert("你好");  
GameCore.loadBgView("大片草地.png");  
GameCore.pause(20000);
```



# 初识“变量”

- 数据放到“内存”中，有时为了重复使用某个数据，可以给这块数据贴一个标签



我名字

```
String myname = "如鹏网";
```

```
GameCore.setGameTitle(myname);
```

```
GameCore.pause(3000);
```

```
GameCore.alert(myname);
```

```
GameCore.pause(3000);
```

# 区分“变量名”和“数据”

```
String myname = "tom";  
GameCore.alert(myname);  
GameCore.alert("myname");  
GameCore.alert(tom);//改成定义一个  
tom变量
```



# 变量声明多个

```
String myname = "如鹏网";  
GameCore.setGameTitle(myname);  
GameCore.pause(3000);  
GameCore.alert(myname);  
GameCore.pause(3000);  
String home="北京";  
GameCore.setGameTitle(home);  
GameCore.pause(3000);
```

变量定义不能“重名”，也就是不能重复“定义”。

# 到处贴的“标签”

我女友

今天“我女友”来找我吃饭。



# 到处贴的“标签”

今天“我女友”来找我吃饭。





# 到处贴的“标签”

今天“我女友”来找我吃饭。



# 到处贴的“标签”

- 在一定范围内，“标签”可以到处贴，但不可以重复定义。
- 在一定范围内，“变量”可以重复使用，但不可以重复定义。

```
String myname = "如鹏网";
```

```
int pauseTime = 2000;
```

```
GameCore.setGameTitle(myname);
```

```
GameCore.pause(pauseTime);
```

```
GameCore.alert(myname);
```

```
GameCore.pause(pauseTime);
```

```
myname= "Rupeng" ;//标签贴到新的数据上；变量指向新的数据上。
```

```
GameCore.setGameTitle(myname);
```

```
GameCore.pause(pauseTime);
```

```
GameCore.alert(myname);
```

```
GameCore.pause(pauseTime);
```

过去的就过去了，变量指向新的数据只会影响之后指令的执行效果。不会“回头算账”！

# 关键字 ( Keyword )

- java定义了一些关键字(public/static/void/class/int)，这些关键字是构成java基本语法用的，这些关键字没必要去背，一边学一边就掌握了。这些关键字的特征就是在Eclipse中颜色是紫色。
- goto、const在java中没有意义，但是是保留的，变量、类的名字也不能用这两个，他们也被叫做“保留字”，不用知道goto、const是什么意思的。
- main、String、System等这些都不是关键字。



# 标识符

- 标识符是用来给类、方法、变量等命名用的。
- 命名规则：1 ) 由字母、中文（不推荐）、数字、下划线 \_、\$组成；2 ) 不能以数字开头；3 ) 不能是关键字。
- java语言是大小写敏感的：demo和Demo是两个东西；
- 课上思考：下列哪些是合法的标识符？a1、1a、a\_b、\_1a、@c5、\$a。
- 驼峰命名法：每个单词第一个字母大写。ageOfBaby、openComputer、createMenuItem。
- 标识符要有意义；标识符的命名规范，不是强制规定，而是“潜规则”：类名：大写开头；变量名、方法名：小写开头。

# 变量的类型

- 我们可以把price变量指向内存的值改为3、5、10，但是能不能改成“tom”呢？显然“一斤菜hello元”是没意义的，所以这块内存中能够放什么类型的数据是有限制的，这就是“变量类型”。
- 定义一个变量的格式：变量类型 变量名=初始值；





# 变量作用域初步

- 变量在{}范围之内定义不能重名，定义一个int b=3；再定义一个int b=10;编译出错。
- int b=3;b=5;是可以的为什么？

局部变量的作用域就是所在的{}，出了作用域就不认识了。学到if、for等时候就会碰到这个问题。



# 变量的初始化

今天和“我女友”去宾馆检查工作



# 变量的初始化

今天和“我女友”去宾馆检查工作



# 变量的初始化

- 错误代码：

```
String name;  
GameCore.alert(name);
```

- 正确的代码1：

```
String name= "rupeng.com" ;  
GameCore.alert(name);
```

- 正确的代码2：

```
String name;  
name= "rupeng.com" ;  
GameCore.alert(name);
```

- 局部变量**使用前**必须赋给初值。声明时赋值或者使用前赋值都可以。



# 数值类型

	数据类型	占用字节	取值范围
整数类型	byte(字节)	1	$-2^7$ 到 $2^7-1$
	short(短整型)	2	$-2^{15}$ 到 $2^{15}-1$
	int(整型)	4	$-2^{31}$ 到 $2^{31}-1$
	long(长整形)	8	$-2^{63}$ 到 $2^{63}-1$
小数类型	float(单精度浮点数)	4	不用记
	double(双精度浮点数)	8	不用记

- byte是“字节”类型，代表一个8位二进制，也就是一个字节。
- 整数常量默认是int类型，小数常量默认是double类型。
- 选用数据类型的时候在考虑到数据的可能范围之后，选择最小范围的类型，这样节省资源。

# 整数类型



- [illegible]

# 整数类型转换(Type cast)

- `byte b=3;int i=b;//隐式类型转换`
- 其实并没有把这个东西从这种类型"转换"成另一种类型，它只是根据原来的内容创建一个新东西。
- 如何把int赋值给byte（编程经常用）  
`int i = 999;`  
`byte b = i;//不能从 int 转换为 byte`
- 下面的呢？  
`int i = 3;`  
`byte b = i;`
- 怎么办？`byte b = (byte)i;//显式类型转换`
- “`int i=999;byte b = i;`” 报错可以理解。为什么 “`byte b=3`” 可以， “`int i=3;byte b=i;`” 就不可以？为什么`byte b=(byte)i;`又可以？

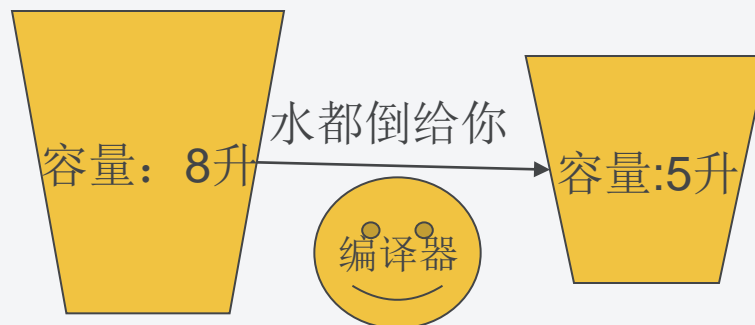
# 为什么呢？

● byte b=3;



YES

◎ int i=3; byte  
b=i;



NO

◎ int i=3; byte b=(byte)i;



YES

# 小数数据类型

- 声明常量(double>float) :

double d1 = 3.14D;//ok , 3.14d也可以

double d2 = 3.14;//ok,一般不用写 “D”

double d3 = 3;//ok

float f1 = 3.14;//error:不能从 double 转换为 float。小数默认是double

float f2 = 3.14F;//ok,3.14f也可以

# 小数类型转换

- `double d1 = 3.14;`
- `float f1 = 3.14F;`
- `double d2 = f1;`//ok , 隐式类型转换
- `float f2=d1;`//error : 不能从 double 转换为 float
- `float f3=(float)d1;`//ok , 显式类型转换
- `int i1 = d1;`//error
- `int i2 = (int)d1;`//ok , 显式类型转换 , 会丢失小数部分
- `double d3 = i1;`//ok , 隐式类型转换

# 小数类型计算问题

- 加(+)减(-)乘(\*)除(/)
- 计算5除以8 :  
GameCore.alert(5/8);  
GameCore.alert(5.0/8.0);  
GameCore.alert(5.0/8);  
GameCore.alert(5F/8F);  
GameCore.alert(5D/8D);
- alert改成*showTypeName*看看计算结果是什么类型
- 再看看哪个错，结果是：  
int i1 = 5/8;  
int i2 = 5F/8F;  
float f1 = 5F/8F; int i2 = (int)5F/8F; //错误 //int i2 = (int)(5F/8F); //正确
- 思考：下面的结果是？ GameCore.alert((1/3)\*3);怎么改。

# 结论

- “大范围” = “小范围”：隐式转换
- “小范围” = “大范围”：显式转换，编译器担保不出事
- 都是整数参与乘除运算，结果还是整数。整数常量参与乘除计算一般加f声明为小数类型，避免精度损失。



# 常用数据类型

- boolean:true/false
- String : ""是什么?  
“\n”与转义符：“你好\n如鹏网”。  
如何表示 "" : “\” : “我叫\“杨中科\””  
如何表示\ : “a\\b”。表示文件路径的时候一定要注意。
- int
- int转换为String : Integer.toString(i) ; String转换为int : Integer.parseInt("33")。显式类型转换仅限整数、小数类型之内。
- 其实并没有把这个东西从这种类型"转换"成另一种类型，它只是根据原来的内容创建一个新东西。
- char : 'a'和"a"的区别 ; '2'和2的区别。

# 基本运算符

- 运算符就是加(+)减(-)乘(\*)除(/)等符号，英文键盘输入 $\times \div$ 很麻烦，所以用 $*$  / 表示乘除。
- 求余数运算符 “%”：5%4为1、5%5为0、22%7为1。
- 自增： $++$ 是对一个变量进行自增运算  

```
int x=3;  
x++;
```
- 自减： $--$
- $+$ 还可以用来对字符串进行拼接：“abc”+“cde”
- $+$ 还可以与其他类型拼接：“hello”+5
- “hello”+5+5、“hello”+(5+5)、5+5+“hello”的区别。

# 总结提问

- Java中表达式从左向右扫描进行扫描运算，一旦遇上一个字符串，之后的运算就变成字符串了。

提问：

$3+5+"hello"+5+3$  结果是什么

# 赋值运算符

- `int i=5;`

应该读成“声明int类型变量i，并且把5赋值给i”。

- `int x=10;x=x+5;`

看`x=x+5`从数学上来讲是一个错误，但是按照“赋值”的读法就可以理解了。

- 因此`x++`等价于`x=x+1`；

- `x+=5`等价于`x=x+5`;

`x++`和`x+=1`一样

还可以`-=`、`*=`、`%=`

# 基本数据类型的赋值

- 基本数据类型是复制传递

```
int i=10;
```

```
int j=i;
```

```
i=20;
```

```
GameCore.alert(j);//结果是？
```

- 如何交换两个int变量的值。

```
int i=10;
```

```
int j=20;
```

```
//写代码，不能使i=20;j=10;
```

```
GameCore.alert("i="+i+"j="+j);//输出i=20;j=10;
```



i=紫;j=黄;



声明变量temp



temp=i;



i=j;



j=temp;

# 分析

```
int i=10;  
int j=20;  
int temp = i;//temp:10;i=10;j=20;  
i=j;//temp:10;i=20;j=20;  
j=temp;//temp:10;i=20;j=10;  
GameCore.alert("i="+i+";j="+j);
```

验证：每句话之后都打印一下i、j、temp三个变量的值

# 赋值表达式也是值

赋值运算本身也是一个值，这个值就是赋值之后左边的值。

```
int a;  
int b=(a=6);  
System.out.println(b);  
System.out.println(b=b+9);
```

i++和++i的区别：

```
int i=1;  
System.out.println(i++);  
int a=1;  
System.out.println(++a);
```

同理：i--和--i的区别。

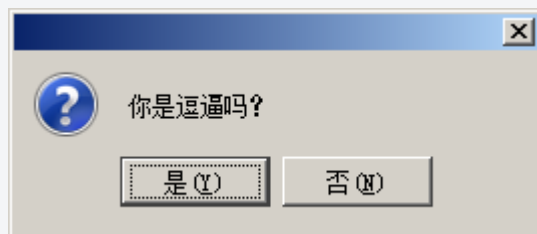


# 问

```
int i=1;  
System.out.println(i++);  
System.out.println(++i);
```

## 第三章 GameCore

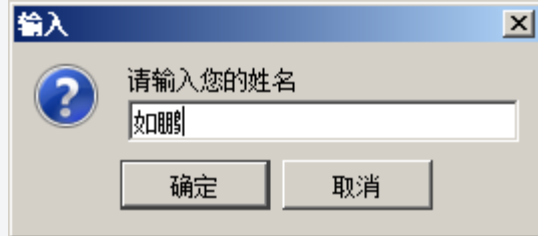
# 确认对话框



- **boolean confirm(Object msg)** 显示有【是/否】按钮的对话框；msg为显示的提示消息；返回值为boolean，true为点了【是】，false为点了【否】。
- 我们学的第一个有“返回值”（方法**执行结束之后**，获取执行的结果）的方法，可以通过变量获取返回值。  

```
boolean isMale = GameCore.confirm("你是男人吗？");  
GameCore.alert("是否是男人：" + isMale);  
boolean isStudent = GameCore.confirm("你是学生吗？");  
GameCore.alert("是否是学生：" + isStudent);
```
- 没有返回的方法是void。

# 输入对话框



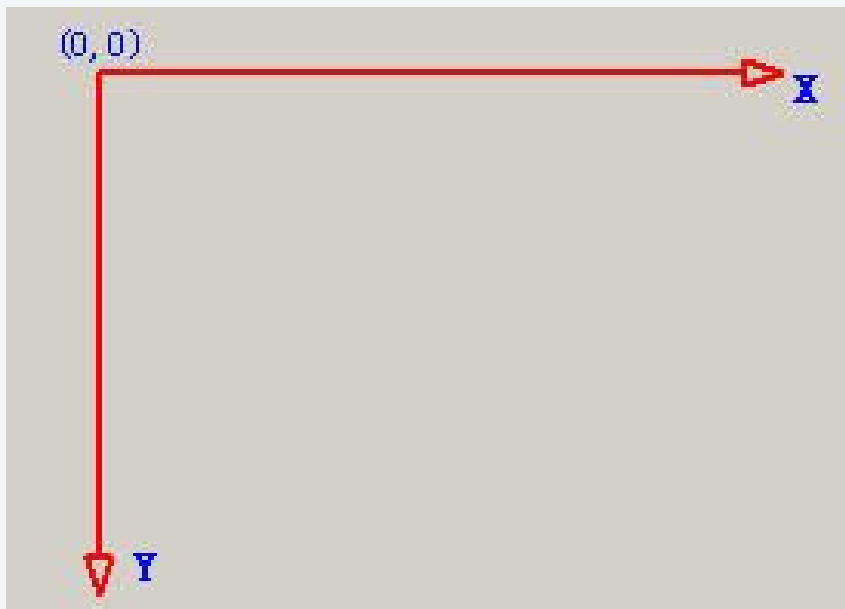
- **String input(Object value, Object msg)** value为默认值；msg为提示消息；返回值为用户输入的文字，如果用户点击了取消，则返回null（以后详细讲）

```
String name = GameCore.input("如鹏", "请输入您的姓名");
```

```
GameCore.alert("你好" + name);
```

- 实现一个加法计算器：提示用户输入第一个数，再提示用户输入第二个数，计算两个数的和。
- 练习：提示用户输入他的名字，然后再分别输入游戏的宽度和高度，然后改变游戏的标题为“某某某的游戏”，并且把游戏窗口的尺寸修改为用户输入的。提示：String转换为int使用Integer.parseInt、修改游戏尺寸是setGameSize。

# 窗口坐标



网页、Swing、Android、游戏引擎等几乎所有图形环境中，都是以屏幕左上角为原点坐标，x向右为正向，y向下为正向。

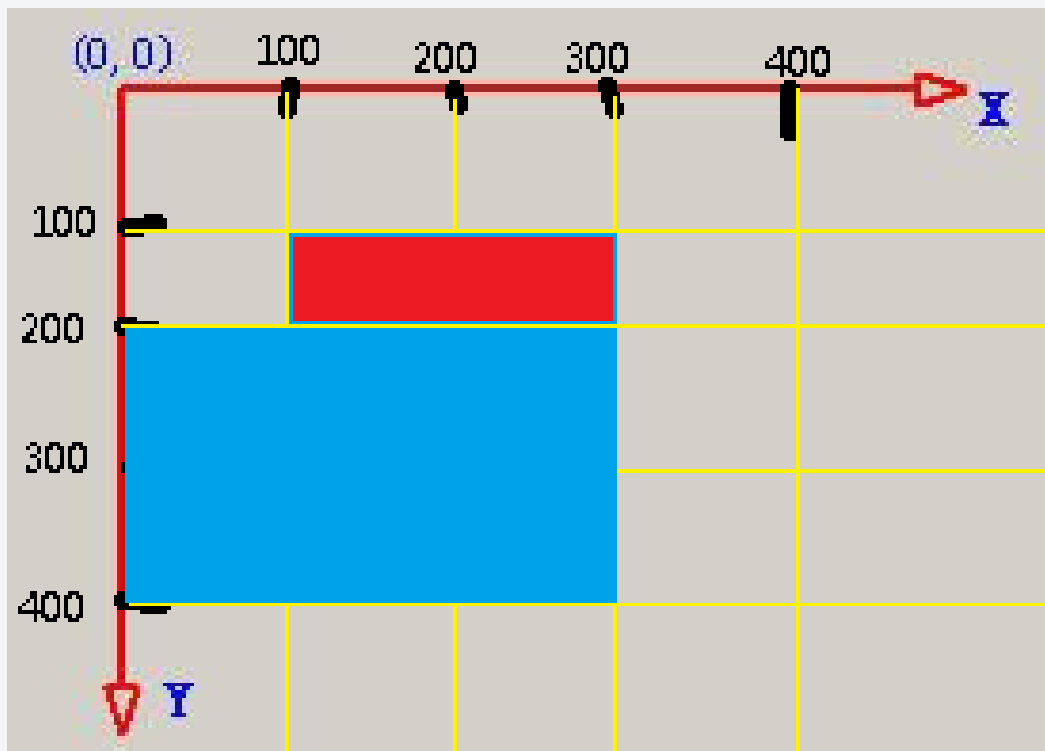
像素：图形内容都是由一个个的像素点组成的。1024\*768；1280\*800。

问：让一个点从屏幕左侧向右侧飞，那么是它的\_\_坐标值逐渐变\_\_

问：让一个点从底部向上飞，那么是它的\_\_坐标值逐渐变\_\_

# 矩形区域的位置

描述一个矩形区域的位置指的是“**左上角**”的坐标



问：说出红色矩形的位置、大小；  
问：说出绿色矩形的位置、大小；

# 文本Text



- void createText(int txtNum, String text)。创建只读文本。txtNum唯一的编号(易错：重复)； text初始显示的文本。
- void setTextPosition(int txtNum, int x,int y)。修改文本的位置，默认位于(0,0)。txtNum （你要修改哪个文本，因为可以有多个。易错：不存在）案例：实现上面的效果。
- void setText(int txtNum, String text)。修改只读文本的内容。text修改后的文本。
- void setColor(int txtNum, Color color) 。 设置文本的颜色。比如 setColor(1,Color.red);
- void setFontSize(int txtNum, int size)。设置文本的**字体大小**。
- Point getPosition(final int num)，获得文本的位置。
- Dimension getSize(final int textNum)。获得文本的大小。
- hideText(final int labelNum)、 void showText(final int labelNum)隐藏或者重新显示文字。

# 案例：超级玛丽启动界面





# 练习

格式工厂 3.7.0

[www.pcfreetime.com](http://www.pcfreetime.com)

欢迎在不随意修改的前提  
自由传播格式工厂

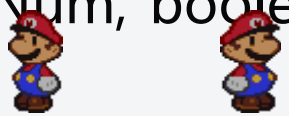
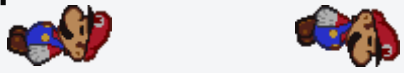
开发者：陈俊豪

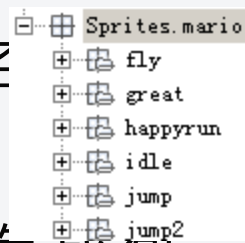
感谢封面图标设计者 Serede

# 图片

- void createImage(int num, String imgName) 创建一张图片控件，num还是图片控件的唯一编号，imgName为初始显示图片的文件名（**必须放在src的Images包下**）。也可以调用void createImage(int num)创建一个什么都不显示的图片。
- void setImageSource(int num, String imgName) 修改显示的图片文件。
- setImagePosition(int num, int x, int y) 修改坐标
- Point getImagePosition(final int num) 获得坐标
- Dimension getImageSize(final int num) 获得大小
- void hideImage(int num)、 void showImage(int num) 隐藏、显示图片

- 以 "ll" 和 "coin" 为例讲解 "精灵(sprite)" 和动作(Animation)。在图片浏览器中使用方向键快速播放查看效果。汤姆猫
- 精灵必须放在src的Sprites包下，第一级包为精灵的名字，第二级包为动作的名字，动作图片按照数字进行排序。
- void createSprite(int num, String spriteName) 创建精灵，num精灵唯一编号，spriteName精灵的名字
- void playSpriteAnimate(int spriteNum, String animateName, boolean repeat) 播放编号为 spriteNum 的精灵的名字为 animateName 的动作，repeat 表示是播放一次，还是重复播放。playSpriteAnimate 是启动动画之后就结束，不会等动画播完。
- void setSpritePosition(int spriteNum, int x, int y) 设置精灵的位置
- Point getSpritePosition(int spriteNum) 得到精灵的位置

- Dimension getSize(int spriteNum) 获得精灵的尺寸
- void hideSprite(int spriteNum) 、 void showSprite(int spriteNum) 隐藏和显示精灵
- void setSpriteFlipX(int spriteNum, boolean flipX) 设置是否进行X轴方向内容翻转。  

- void setSpriteFlipY(int spriteNum, boolean flipY ) 设置是否进行Y轴方向内容的。  

- 这种情况怎么放第二个精灵，如果直接粘贴会造成包结构错误，在节点上点击右键【新建】→【包】，改名称。
- 案例：每alert一次换一个动作。
- 案例：每隔500毫秒，精灵向右移动10像素，10次。
- 练习：每隔500毫秒，精灵向右下角移动(x、y各10像素)，10次。



\*的代表：  
即使学不会也不影响后续学习

# 精灵文件的生成(\*)

- 这节不重要：如果生成有问题就直接用我给大家提供的现成的就可以。版权问题说明。来源 <http://www.sprites-resource.com> 等
- 美工生成的都是整张图片（展示“未处理”文件夹下的），需要使用工具切割成精灵图片资源。这种工具有很多，我用的爽的就是SpriteSheets（需要.Net Framework）。生成的图片必须是png格式的。
- 1、把图片拖到SpriteSheet软件中。用鼠标拖界面上下滚动。
- 2、识别出来的精灵图片用蓝色框包围。按顺序点击一组动作图片，然后点击【Export Selected】保存到文件夹。如果点错了就【Deselect All】。



后退，我要开始装逼了

# 程序打包 (\*)



- Eclipse只是一个辅助开发的工具而已，Eclipse的运行以及编译出的程序运行还是要靠“Java运行时环境” (JRE)。
- 把程序发给别人运行的时候，对方电脑上需要安装JRE，很麻烦。我们可以把jre直接打包到程序包中。
- 打包步骤（如果失败就不用纠结，这不是重点）：
- 把bin和两个jar拷贝到单独的文件夹下；
- 把jre拷贝到文件夹，去掉src.zip、lib\ext\jfxrt.jar、bin\server、bin\jfxwebkit.dll，减小尺寸（还可以继续减小）
- 创建一个.bat文件（去掉资源管理器的隐藏扩展名功能），内容是  
start jre\bin\javaw.exe -cp .\bin\ -Djava.ext.dirs=. Main1。  
Main1代表入口类，如果有报名则要写全名。
- 压缩一下，发给对方，解压双击“bat”文件即可





骚年你确定还要继续装逼么



# 制作自解压exe(\*)不重要



- 双击自动启动安装界面、在开始菜单、桌面创建图标。
- 首先用WinRAR压缩成.zip或者.rar文件，用WinRAR打开这个文件，点击主菜单【工具】→【压缩文件转换为自解压格式】→【高级自解压选项】。
- 【常规】中【解压路径】给程序安装文件夹一个名字，【设置】→【解压后运行】填写【启动游戏.bat】；
- 添加开始菜单、桌面快捷方式。【高级】→【添加快捷方式】，【源文件】为点击快捷方式要启动的文件，就是【启动游戏.bat】，快捷方式名比如【启动牛逼游戏】
- 【文本和图标】→【自解压文件窗口标题】、【自解压文件窗口中显示的文本】。【许可】是安装协议显示的内容。
- 更深入的定制自解压exe（改图标、卸载等）自己研究。更高级的安装包制作工具：InstallShield等。

# 第四章 逻辑运算和判断

# 比较运算符

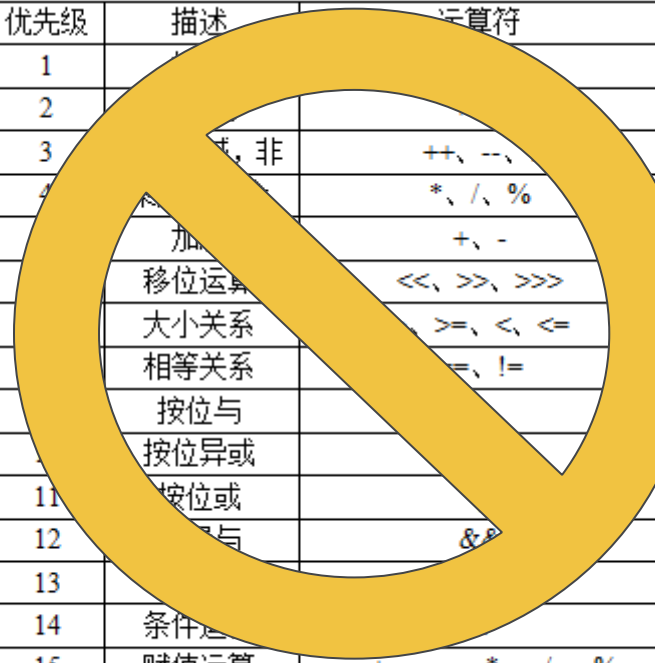
- 比较运算符（又称关系运算符）用来进行值的真假性判断，结果是boolean
- ==等于；!=不等于；>大于；>=大于或者等于；<小于；<=小于或者等于。  
GameCore.alert(3==5);  
int i=9;  
GameCore.alert(i>=3);  
GameCore.alert(i==3);  
也可以把比较的结果赋值给boolean类型变量；  
int a=4;  
int b=5;  
boolean flag=a==b;  
GameCore.alert(flag);
- 易错：除了int、boolean等这些基本类型之外，字符串等对象的相等判断要用equals方法，以后讲为什么。s1.equals("yzk")

# 运算符优先级

- `boolean flag=a==b;`这句话怎么理解呢？
- `int i=3+5*3-6;int j=3+5*(3-6);`分别是什么？

# 运算符优先级

- 运算符有优先级，优先级高的优先“结合”。优先级表，神最右→→→
- 不用记这些，程序不仅是给编译器看的，更是给程序员看的(自己看不懂、别人看不懂。讲程序员最不愿意接手别人项目的段子)要让人能看懂。
- 记住伟大的“()”就ok了！
- 可读性高是重要的，写出别人看不懂的代码不是高手。



优先级	描述	运算符
1		
2		
3	自增、自减、非	++, --, !
4	乘、除、取模	*, /, %
5	加、减	+, -
6	移位运算	<<, >>, >>>
7	大小关系	>, <, <=
8	相等关系	==, !=
9	按位与	
10	按位异或	
11	按位或	
12	逻辑与	&&
13		
14	条件运算	
15	赋值运算	=, +=, -=, *=, /=, %=
16	位赋值运算	&=,  =, <<=, >>=, >>>=

# 不要把==写成=

- 下面程序的执行结果是什么？

```
int a=3;  
int b=4;  
GameCore.alert(a==b);  
GameCore.alert(a=b);
```

- 赋值运算本身也是一个值，这个值就是赋值之后左边的值。

```
int i=(a=b)+3;  
GameCore.alert(i);
```

- 这个b1呢？boolean b1 = (a=b);

- 这个b3呢？

```
boolean b1 = false;  
boolean b2 = true;  
boolean b3=(b1=b2);
```



# 逻辑运算符

- 逻辑运算符用来对布尔类型的值进行运算的，主要有：  
&&(与/并且)；||(或)；!(非)
- 二元运算符(+、-、>等)、一元运算符(++等)
- &&：二元运算符，两边都是true结果才是true。“我是中国人” “我是男人” 只要有一个为假，那么“我是中国人 并且 我是男人” 就是假。
- ||：二元运算符，只要有一个是true结果就是true。“我是中国人” “我是男人” 只要有一个为真，那么“我是中国人 或者 我是男人” 这个判断就是真。
- !：一元运算符，取反，真的变假，假的变真。
- 提前写好程序测试：备注。

# 短路运算

- 下面程序执行结果是什么？
- `int i=8;`
- `boolean b = (i>5)&&((i=i+5)>10);`
- `GameCore.alert(b);`
- `GameCore.alert(i);`
- 把第二句改成：`boolean b = (i>10)&&((i=i+5)>10);`呢

# 短路运算

- 结论：&&当第一个为false的时候，整个运算结果一定是false，就没必要算第二个了。
- 干什么用？
- `int i=8;`
- `int j=3;`
- `boolean b= (j!=0&&(i%j==0));`
- `GameCore.alert(b);`

# 短路运算

- `||`：短路或。直接猜结论：`||`当第一个为true的时候，整个运算结果一定是true，就没必要算第二个了。
- 测试一下：

```
int i=8;  
//boolean b = (i<10)||((i=i+5)>10);  
boolean b = (i>10)||((i=i+5)>10);  
GameCore.alert(b);  
GameCore.alert(i);
```
- 非运算(!)就一个参与运算，没有短路的问题。

# 三元运算符

- 条件表达式?表达式1:表达式2
- 如果“条件表达式”为true，则表达式的值为“表达式1”，否则为“表达式2”  

```
int i=9;  
int j= (i>15?888:666);  
GameCore.alert(j);
```
- 案例：GameCore.alert((i%3==0)?"i能被3整除":"i不能被3整除");
- 练习：输出i是否是偶数。
- 案例：confirm，点击【是】播放一首歌，【否】播放另一首歌。

# If

```
if(比较表达式)
{
    //一行或者多行语句
}
```

当“比较表达式”为true的时候执行{}中的代码。

案例：判断年龄，根据如果大于18岁，则提示“是时候给你看一些.....”、并且显示某些图片、播放某些音乐。

练习：GameCore.input()提示用户输入一个数字，如果数字 $\leq 0$ ，则提示用户“请输入正数”

# 易错

下面的程序运行结果是什么：

```
int i=10;  
if(i=10)  
{  
    GameCore.alert( "我是10" );  
}
```

**if**中必须是条件表达式，**i=10**是赋值表达式，赋值表达式的值是赋值后的结果，也就是**10**，**if(10)**当然是有问题的，要写成**==**，这是初学者经常犯的错误

# 提问

- 结论：“if中不能用=，只能用==”！
- 对吗？
- 下面的程序能不能编译通过，如果能运行结果是什么？

```
boolean flag = false;  
if(flag=true)  
{  
    GameCore.alert(“我是处”);  
}
```

因为**flag=true**虽然是赋值表达式，但是恰好这个赋值表达式的值是**boolean**类型，而且是**true**值。



# if语句中的大括号写不写？

- 如果if语句的大括号中只有一句话，那么可以省略大括号。
- 但是不建议这样做，因此容易出错。举例。
- 建议：哪怕只有一句话，也要写大括号。坏的编程习惯并不会一定导致出错，但是增加的出错的概率。

# 易错：if后多写了个;

- `if(age>18);`
- .....
- 为什么？

# else

if语句还可以带else：

```
if(比较表达式)
```

```
{
```

```
    //比较表达式为true时执行的代码块
```

```
}
```

```
else
```

```
{
```

```
    //比较表达式为false时执行的代码块
```

```
}
```

- 例子： $\geq 18$ 岁显示一个图片， $\leq 18$ 岁显示另外一个图片
- 练习：根据选择的性别，如果是男，则显示“mario”精灵，否则显示“girl”精灵。

# else if

```
if(条件表达式1)
{
}
else if(条件表达式2)
{
}
else if(条件表达式3)
{
}
.....
else//可以不用最后的else，建议加上，更加严谨
{
}
```

- 案例：提示用户输入女明星的姓名，根据姓名加载图片，未知的姓名则显示默认图片。

# else if

- 例子：根据输入的月份显示属于“春夏秋冬”哪个季节，假定：  
春：3、4、5  
夏：6、7、8  
秋：9、10、11  
冬：1、2、12
- 注意点：如果匹配了上面的一个条件，那么即使复合下面的else if条件，else if也不会执行“先来后到”

# if嵌套

- if..else ..else if 中都开可以嵌套if.....可以无限层级
- 用if嵌套实现 “根据输入的年龄和性别显示阶段的名字” 这个练习。

# if和三元运算符

- 分别用if和三元运算符实现下面的效果：如果年龄大于等于18岁，则显示“成年人”，否则显示“未成年人”。
- 练习，分别用if和三元运算符实现下面的效果：如果年龄大于等于18岁，则显示一张图片，否则显示另一张图片。
- 三元运算符能够实现的，基本if都能实现；if能实现的，三元运算符不见得能够实现。所以基本是“简单的判断然后根据判断的boolean结果进行简单的赋值”才使用三元运算符

# switch

switch(表达式)

```
{  
    case 取值1:  
        //语句;  
        break;  
    case 取值2:  
        //语句;  
        break;  
    ...  
    default:  
        //语句;  
        break;  
}
```

使用switch case实现“月份季节判断”

- 表达式可以是byte、short、int、char类型。(\*)JDK5以后还可以是枚举类型，JDK7以后还可以是String类型。
- 易错：switch只能进行离散值的判断，不能进行范围的判断；switch只能进行常量的比较，不能进行变量的比较
- break意味着switch的结束。default相当于if的else，当所有case都不匹配的时候，执行default



# switch其他问题

- 多个switch条件合并的问题：当多个case条件的代码一样的情况下，可以合并，最后加一个break；
- 当进行单个离散值判断的时候，switch可以用来替代if。switch能做的if都能做，反之不一定。switch 效率比if高一丁丁丁丁点儿。

# 第五章 循环控制

# for循环

- 精灵每次向右移动10像素，移动5次，就写五次setSpritePosition。那输出100次呢？如果根据用户输入的数据来动态决定输出多少次呢？
- 这就最好使用循环结构，循环结构有：for、while、do.....while三种语法。  
for(初始化表达式;循环条件表达式;循环后的操作表达式)  
{  
    //循环体语句  
}
- 最开始先执行“初始化表达式”，然后循环执行：先判断“循环条件表达式”，如果为true，则执行“循环体语句”，然后执行循环后的操作表达式；

# 分析一个经典的for循环

- 执行下面的程序：

```
for(int i=1;i<=10;i++)  
{  
    //显示到Text中  
    GameCore.pause(1000);  
}
```

- 分析下面的程序执行过程“打印1到10”（[【播放多媒体课件：ForDemo.exe】](#)）
- 易错：<=还是<
- 案例：精灵从左(0,100)往右(300,100)逐渐移动
- 案例：精灵从下(100,500)往上(100,0)逐渐移动。

# for案例

- 先练后讲：使用for计算1到10的和
- 先练后讲：要求用户输入一个正整数N，计算1到N的和。
- 上一题更好的做法。
- 练习：10、9、8.....0倒计时

# for案例

- 从左向右排列精灵，纵坐标为200，横坐标从0开始每次新增80，放置8个精灵。
- 从上向下排列精灵，横坐标为100，纵坐标从100每次增加77，最多增加到到500。
- 从(0,0)开始向右下角(500,500)平均放置9个精灵。

# for 练习

- 练习：精灵从上(100,0)往下(100,500)移动。
- 练习：精灵从右(500,100)往左(100,100)移动。
- 练习：超级玛丽走正方向，走不同方向的时候做翻转
- 练：输入一个正整数，精灵从(0,10)移动到(n,10)这个位置。如果用户输入数 $\leq 0$ ，则提示用户“输入的比如是正整数”

# 代码不仅可以写一句哟

```
for(int x=0,y=0;x<300&&y<300;x=x+3,y=y+1)
{
    GameCore.setSpritePosition(bird,(int)x, (int)y);
    GameCore.pause(30);
}
```



# for三部分都可以省

- for(初始化;是否继续循环判断;一次循环后执行的) 这三部分都可以省略，第一部分省略就没有初始化，第二部分省略则默认为true，第三部分则执行后不执行什么。

第一种常见用法：

```
int x=5;
```

```
for(x=9;x<20;x++)
```

第二种常见用法：for(;;)，可以实现“游戏永远不退出”，但是最好加上pause，否则cpu占用率会很高。

# for变量作用域的问题

```
for(int x=0;x<10;x++)
```

和

```
int x=0;
```

```
for(x=0;x<10;x++)
```

的区别是什么

# for案例

- 案例：小球弹跳
- 案例：沿着屏幕四个边缘弹跳
- 练习（\*）：小球横向飞抛物线；

# 循环的嵌套

```
int num=1;
for(int i=0;i<500;i+=50)
{
    for(int j=0;j<300;j+=50)
    {
        GameCore.createImage(num, "doge.png");//精神污染一下
        GameCore.setImagePosition(num, i,j);
        num++;
        GameCore.pause(100);
    }
}
```

i, j嵌套调换一下, 看变化

练习：改成Sprite

# 循环的嵌套案例

案例：字体一直不断的放大、缩小。把下面的语句放到for(;;)中

```
for(int fontSize=10;fontSize<100;fontSize++)  
{  
    GameCore.setTextFontSize(txt,fontSize);  
    GameCore.pause(10);  
}  
for(int fontSize=100;fontSize>0;fontSize--)  
{  
    GameCore.setTextFontSize(txt,fontSize);  
    GameCore.pause(10);  
}
```

练习：一边放大缩小一边从左往右移动  
循环嵌套层数可以更多，但是建议不要超过两层。

# break、continue

- 之前学的for循环循环都是从头执行到末，能否实现满足某个条件的时候终止循环呢？就要使用break或者continue。
- break和continue可以用到for、while、do while这些循环语句中。区别是break是终止整个循环，continue是终止当次循环
- 对于循环嵌套来讲break continue针对的是“内层循环”

```
for(int i=0;i<10;i++)  
{  
    if(i==3)  
    {  
        //break;  
        continue;  
    }  
    GameCore.alert(i);  
}
```

案例：要求用户一个个的输入数字，一直到输入ok为止，计算所有数的和，如果输入的是负值，则忽略。用break/continue和不用break/continue两种做法。

## 案例：Sprite队伍一齐向前走



## 案例：数字时钟





# 获取用户输入

- 使用GameCore.getPressedKeyCode();可以获得用户当前的按键，和KeyEvent中的值做比较。数字键：VK\_0、VK\_1.....；字母键：VK\_A、VK\_B；功能键：VK\_F1、VK\_F2.....；回车：VK\_ENTER；空格：VK\_SPACE；ESC：VK\_ESCAPE；方向键：VK\_LEFT、VK\_UP、VK\_RIGHT、VK\_DOWN。
- 案例：键盘控制精灵上下左右移动。
- 案例：键盘控制精灵移动，当距离一个目标精灵少于10的时候就认为碰到目标了。
- 空格弹跳？这种“一次性触发”的动作最好用后面的“事件驱动”，否则实现起来很麻烦，getPressedKeyCode()只适合于“连续动作”。

# while

```
while(条件表达式)
{
    循环体;
}
```

- 每次循环前判断“条件表达式”，如果为true，则执行“循环体”，然后再进行下一次循环判断。
- 用while实现打印五次“我爱Java”（逐语句执行分析）

```
int y=0;
while(y<5)
{
    GameCore.alert(“我爱Java”);
    y++;
}
```

for和while在实现上可以互相代替，一般for用的频率比while高，在io、JDBC中有用到while

# while案例

- 案例：写一个程序打印100到200的值
- 案例：：用while循环计算1到100的和。

普通青年：

```
for(int i=1;i<=100;i++){sum+=i;}
```

文艺青年：

```
sum=100*(100+1)/2;
```

二逼青年：

```
sum=1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+  
23+24+25+26+27+28+29+30+31+32+33+34+35+36+37+38+39+40+41+42+43  
+44+45+46+47+48+49+50+51+52+53+54+55+56+57+58+59+60+61+62+63+  
64+65+66+67+68+69+70+71+72+73+74+75+76+77+78+79+80+81+82+83+84  
+85+86+87+88+89+90+91+92+93+94+95+96+97+98+99+100;
```

# while案例



案例：while实现从左走到右

练习：把for的案例、练习都用while实现一遍

# do while

```
do
{
    执行语句
}
while(条件表达式);
```

先执行一次“执行语句”再判断“条件表达式”决定是否进入下次循环。

演示，使用do while实现打印五遍“我爱Java”

```
int x=0;
do
{
    GameCore.alert(“我爱Java” );
    x++;
}
while(x<5);
```

do...while和while的主要区别：do...while的执行语句至少执行一次，while的执行语句可能一次都不执行。do...while用的更少，到具体应用时再说。

# break/continue

while/do while中也可以用break/continue



# 数组



- `int i=5;String s ="tom";`可以定义单个的数据，要存储多个数据怎么办呢？数组是存储多个数据的容器。
- 类型[] 数组名 = new 类型[个数];
- 比如`int [] arr = new int[5];`
- 读写的方式：`int x=arr[0];int y=arr[3];arr[2]=9;`
- 在读写的时候数组名后的[]叫序号，又叫下标，指的是对第一个元素进行操作，从0开始，所以最后一个元素序号是“个数-1”。
- 第二种定义方法，适合于定义时就确定元素个数和内容，用的较少：`int[] arr={3,5,6,9,11};`

# 数组的遍历

```
for(int i=0;i<5;i++)  
{  
    GameCore.alert(arr[i]);  
}
```

- 可以通过arr.length获得数组的长度，改写如下，这样避免了5写死的问题（避免“魔法数”）

```
for(int i=0;i<arr.length;i++)  
{  
    GameCore.alert(arr[i]);  
}
```

练习，遍历一个int数组，计算数组元素的和。



# 数组使用常见问题

- 对于int、double、float等数组，没有赋值的元素被初始化为0；对于String、Integer等引用类型（后面讲）初始化为null
- java.lang.ArrayIndexOutOfBoundsException 数组下标越界异常。不要超过数组的大小，不要取length。
- 能否动态增加数组的大小？不能，要用LinkedList、ArrayList等。

# 数组案例

案例1：有一个String[]数组，比如String [] strs={"123","3721","888"}; 把它转换为int数组。

案例2：有一个String[]数组，比如String [] strs={"如鹏","阿里","腾讯"}，根据这个数组生成一个逗号分隔的字符串："如鹏,阿里,腾讯"。

案例3：有一个String[]数组，比如String [] names={"王大力","刘小明","张三丰"}，有一个长度和names一样的bool数组**boolean[]** scores={true,false,true} 和names一一对应，代表每个人的成绩，要求生成一个String数组{"王大力及格","刘小明不及格","张三丰及格"}

案例4：把两个int[]数组拼接到一起，比如int[] nums1={3,5,8}，int[] nums2={88,9,25,66,77} 生成{3,5,8, 88,9,25,66,77}

案例5：String[] strs = {"如鹏网","百度","阿里","腾讯"};循环输出strs的值。备注

# 数组练习



练习1：给定一个int[]数组，比如int[] nums={123,3721,888}; 把它转换为String数组。

练习2：有一个String[]数组，比如String [] names={"王大力","刘小明","张三丰"}，有一个长度和names一样的int数组int[] scores ={90,50,70} 和names一一对应，代表每个人的成绩，要求生成一个String数组{"王大力优","刘小明不及格","张三丰及格"}

练习3：计算一个长度相等的int数组的逐位和，int[] nums1={3,5,8}，int[] nums2={88,9,25}，结果{91,14,33}

练习4：反向循环输出一个数组的值。

# 数组案例



案例：给一个String数组，把他们作为Text画到界面上，变红的行从上往下移动（循环）。

练习：给一个String数组，把他们作为Text画到界面上，变红的行从下往上移动（循环）。

# 综合游戏案例：吃金币

案例：一个精灵，8个coin放到界面上（num、位置通过nums、xData、yData三个数组指定），然后用户控制精灵移动，当距离一个coin少于10的时候认为是吃到了coin，然后增加1分，显示到界面上。所有币都吃了游戏结束。

备注



# 练习：吃金币游戏完善



- 1) 吃到地雷就死
- 2) 超过100秒钟没有吃完游戏就结束
- 3) (\*) 随机生成金币位置。GameCore.rand
- 4) 不能走墙壁

注意：完善无止境！

# 数组案例

1、颠倒数组。易错：又颠倒回来了；奇偶数。

$\{1, 8, 22, 5, 7\} \rightarrow \{7, 5, 22, 8, 1\}$

2、找出一个int数组中的最大值。易错：判断一次就结束

3、练习：找出数组中的最小值

4（\*）冒泡排序；二分查找法..... evil

# 二维数组

```
int[][] nums = new int[5][3]; // 声明二维数组, 5行3列
```

```
nums[0][0] = 3;
```

```
nums[0][1] = 5;
```

```
nums[1][2] = 6;
```

```
for(int i=0; i<nums.length; i++) // 一共多少行
```

```
{
```

```
    for(int j=0; j<nums[i].length; j++) // 这行一共有多少个
```

```
{
```

```
        System.out.print(nums[i][j] + "\t");
```

```
}
```

```
    System.out.println();
```

```
}
```

3	5	0
0	0	6
0	0	0
0	0	0
0	0	0



# 二维数组另一种声明

可以为不定长数组。

```
int[][] nums = {{5,3,8,9,2},{2,0,8,2,4,9},{3,4,9,2}};
```

尝试输出一下

# 二维数组案例

- 根据二维数组的值来布置战场：根据不同的值加载不同的图片。用数组的话就避免了代码来绘制地图的麻烦。备注
- 练习：做一个连连看的布局。
- 碰撞检测：碰到色块的时候不能继续移动，课上实现右侧碰撞检测，让学生实现其他方向的检测

# 多维数组（\*）

二维在app开发、系统开发、网站开发中用的很少，多维数组用的更少。

# 方法简介

- 如果程序很多地方都要计算一个数组的最大值、计算两个矩形是否相交，难道每次都要写一堆代码吗？抽象出方法(Method)/函数(Function)。
- 方法(一段可以重复使用的相同或者类似的代码段)主要格式：  
返回值类型 方法名(参数类型 参数1,参数类型 参数2.....)  
{  
}
- 目前来讲所有方法都先加上static，以后讲why
- 方法的声明时参数的名字和调用时候变量名字没有关系，调用的时候甚至可以~~不传变量~~。（\*）~~实参、形参~~
- 封装一下getMax方法。

# 方法

- 方法就是一堆可以重复使用（复用）的代码段。方法执行过程中无法确定的数据以参数形式传递过来；方法的执行结果以返回值进行返回。
- 方法可以没有参数；方法可以没有返回值（void）
- 方法中用return返回处理结果，执行return之后方法内部的代码就执行结束了
- 如果没有返回值，则在需要执行结束的地方return;如果没写return，则在最后一句return；
- 不能在方法中定义方法。
- 方法中的局部变量不能被调用的方法访问，如果真要访问只能通过参数传值，并且只是相当于变量赋值，方法内部对参数的赋值不会影响调用者。
- “不是所有路径都有返回值”

# 方法案例



- 编写方法的步骤（难点）：根据方法的用户定“方法名”、“参数”、“返回值”：做这件事情需你要给我什么→参数，我返回给你什么→返回值。定参数、定返回值。
- 案例：封装一个方法，返回给定两个int的和。定参数、定返回值。
- 案例：写一个方法，返回int数组的最小值。定参数、定返回值。
- 案例：编写一个方法，将两个String字符串数组合并为一个字符串数组，比如String[] strs1={"aaa","bbb","ccc"}和String[] strs2={"123","321"}合并为"aaa","bbb","ccc","123","321"。定参数、定返回值。

# 方法练习



- 练习：写一个方法，返回给定int数组的和。定参数、定返回值。
- 练习：写一个方法，给定一个字符串**s**和一个整数**n**，返回一个新字符串，字符创内容为把**s**重复**n**次，比如**s="abc",n=3**，则返回**"abcabcabc"**
- 方法的嵌套调用：`repeat(Integer.parseInt("333"))`

# 方法案例



- 案例1：封装一个方法，让指定的精灵在当前位置向右移动一个像素。并且调用个方法测试一下。
- 案例2：封装一个方法，可以让指定的精灵向上下左右方向移动若干像素。编写一个“小动画”，感受一下方法封装的方便。
- 案例3：使用案例2封装的方法封装一个走矩形回到原点的方法。
- 案例4：使用案例2封装的方法编写一个效果：mario从左跑到右，从右跑到左循环往复的“游戏加载中”的效果。
- 案例：封装一个方法，矩形相交判断。定参数、定返回值。
- 使用判断矩形相交的方法改造精灵在地图中移动，以及碰到地雷就死、碰到金币就加分。方法可以判断精灵是否和image，以及精灵是否和精灵相交



- 练习：编写一个原地起跳的方法，指定起跳的精灵编号、起跳的初始速度。

编写一个使用这个方法的案例：界面上两个经理。精灵在平地上走，按空格键精灵1原地起跳，按j键精灵2原地起跳。

# 方法重载

- 编写一个获得String[]最长元素的方法getMax；编写一个活的int[]最大元素的方法getMax；编写一个获得两个int最大值的方法getMax；
- 重载（OverLoad）：在同一个类中，允许存在一个以上的同名方法，只要其参数个数或者参数类型不同即可。返回值是否一样无关和参数名是否一样无关。
- 编译器怎么样知道应该调用哪个？
- 案例：封装方法，计算int、float等类型中两个数的最小值，用三元运算符
- **方法重载**可以实现“方法参数默认值的效果”，案例：编写一个方法moveRight向右移动，可以不指定移动多少步，则默认是移动1步。

# 方法重载的陷阱

- 为什么参数名字不一样不能构成重载。
- 为什么返回值类型不一样不能构成重载。
- `f1(int i);f1(long l);`

# 可变长度参数

- 编写一个sum方法，计算一个int数组的和。

```
int sum(int[] nums)
```

- 每次调用都要sum(new int[]{5,3,9,10})。改成：

```
int sum(int... nums) //JDK 1.5后支持
```

就可以这样自由调用sum(3,5);sum(3,9,11,22,5);

这些参数其实还是以数组形式传递过去，在sum看来也是数组，只不过编译器会自动把参数封装为数组，简化了调用。

- 可变参数可以声明为各种类型：test1(String... values)
- 可变参数前面还可以有其他参数，只要保证**可变参数是最后一个**就可以：sum(String name,object... data)。

# 面向对象基础

# 面向对象概念

- 很多教程都想在刚讲解面向对象的时候就让学生理解面向对象的类和对象、封装继承多态是怎么回事。不用阿猫阿狗讲面向对象。
- 面向对象编程（OOP，Object Oriented Programming）是相对于面向过程编程说的，之前我们写的代码基本都是纯的面向过程编程的，当项目复杂了，纯面向过程代码实现会很复杂，面向对象可以简化代码的结构和组织关系。面向对象不是替代面向过程的，宏观是面向对象，微观仍然是面向过程。优势以后你会懂得。
- 类(Class)和对象(Object)：人、诸葛亮。“是” “是一个”
- (\*)面向对象三大特征：封装、继承、多态。

# 类的定义

最简单的类：class Person{

```
class Person{
    private int age;//成员变量
    private String name;
    public void setAge(int age)
    {
        this.age = age;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public void sayHello()
    {
        System.out.println("大家好，我是"+name+"，我今年"+age+"岁了");
    }
}
```

类由“数据”和“行为”组成。数据（成员变量，Field，字段）设置为private，访问数据通过get\*\*\*、set\*\*\*方法进行，只有set\*\*\*没有get\*\*\*则只能读不能写。

- 一个java文件中只能定义一个public的class，且文件名一般和public类一样。

# 对象的实例化

```
Person yzk = new Person();  
yzk.setName("杨中科");  
yzk.setAge(18);  
yzk.sayHello();
```

```
Person lzy = new Person();  
lzy.setName("林志颖");  
lzy.setAge(80);  
lzy.sayHello();
```

//两个对象的内存分配：根据模板拷贝两份。

```
yzk.sayHello();
```

new出的每个对象都是一个单独的实例，两个对象之间的成员变量是独立的两份。new出来的叫类对象或者实例(Instance)。

画内存图。



# 类对象是引用传递

- 复习：int i=10;int j=i;i=20;之后j是多少？

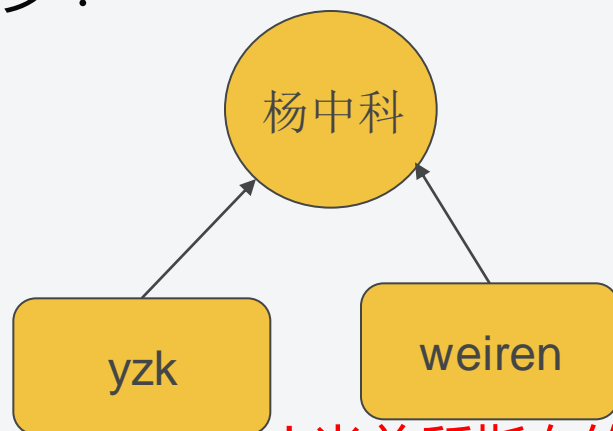
- 下面的程序呢？

```
Person weiren = yzk;
```

```
weiren.setAge(22);
```

```
weiren.sayHello();
```

```
yzk.sayHello();
```



- 解释：“Person weiren = yzk;” **weiren指向yzk当前所指向的对象**

- yzk = lzy;

```
yzk.sayHello();
```

```
weiren.sayHello();
```

- 结果是什么？
- 只有new才会创建新的对象，有多少new就有多少对象。
- 把对象传递进方法中，方法中setAge，外面会变吗？为什么？

# 成员变量和局部变量

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

- 1、局部变量必须初始化，成员变量声明时默认初始化，基本数值类型默认初始值为0、false，String等非基本类型初始化为null。
- 2、当成员变量和局部变量（函数参数也可以看做局部变量）重名的时候，被看做局部变量，因此为了避免混乱，建议访问类成员变量的时候加上“this.”，在类内部this代表当前对象。this.也可以调用当前类的成员方法。

# null和GC

- null是什么：没有对象。
- 栈内存和堆内存。
- 对象在堆内存中，变量在栈内存中。方法结束后栈内存自动释放，但是堆内存不会自动释放。用收餐具举例子（公用餐具，一次性餐具）
- GC：垃圾回收 Gabbage Collection。Java中一般不需要程序员进行内存的回收（别人帮你收餐具）。当没一个对象没有任何变量指向它的时候就“可以”被销毁，局部变量退出后就消失了就不再指向了
- `new Person().sayHello();`创建一个匿名对象，然后调用。输出结果是？

# private/public

- 我们可以把age成员变量声明为public，也可以把setAge声明为private，这样就只能在类内部调用private成员（再写一个方法调用private的setAge）
- 结论：public成员可以被类内部或者外部访问，private成员只能被类内部访问。这样可以保护不希望外接调用的内部成员(Member，包含字段Field/变量、方法)不被外界访问。**封装数据、暴露行为。**
- 直接通过public的age设置年龄，不通过setAge赋值，这样有什么坏处？-1。
- 字段(Field)/成员变量(Member Variable)一般声明为private

# 构造函数(Constructor)

- 成员变量初始化的两种方式。
- 构造函数是创建类对象，并且在创建完成前对类进行初始化的特殊函数。如果定义类时没有声明构造函数，默认会给出一个无参构造函数，如果定义了任意一个构造函数，将不会提供默认的无参构造函数。
- 构造方法格式及特点：
  - 方法名必须和类名一致
  - 没有返回值类型
- 构造函数可以重载，`Person(String name,int age)`

# JavaBean规范



- JavaBean的规则，不是一种技术，只是一种规则。
  - 1) 必须有无参构造函数；
  - 2) 字段不能是public；
  - 3) 类的字段必须通过get和set 方法来访问。 set\*\*\*、get\*\*\*。只有get没有set则是 “只读”
  - 4) boolean类型属性名避免使用 “is” 开头的名称
- Eclipse中根据私有变量自动生成set、set的方法；根据成员变量生成构造函数的方法。

# 对象之间的关系

- Person设定一个Father属性，声明一个SpeakFatherName的方法，注意不能在内部再new一个Person，否则是两个“爹”了。画内存图。
- 编写一个电脑类、U盘类、光驱类、显示器类、电源类，电源一旦关掉，所有设备都无法访问。画内存图。

# 包(package)

- 用文件系统的文件夹解释避免文件名重复的问题。
- 包语法：  
通过package定义在页面最顶部；  
在磁盘上的保存路径和package一致，可以单级、可以多级。
- 包名 “潜规则”  
包名小写  
“公司域名反着写.产品名.模块名”：com.rupeng.crm.user、com.sun.media.sound
- 当前包中的类无需引用；其他包中的类：  
import com.rupeng.crm.user.\*；(不推荐，一次引用太多，容易冲突)  
import com.rupeng.crm.user.Person；(推荐)  
使用的时候java.sql.Array，不用import，适用于同时使用多个重名的类；
- java.lang下的内容不用手动import。



# 静态 static

- 一些场景下会要求一个类的多个实例共享一个成员变量；有时候要定义“常量”，但是java没有提供常量；有时候想定义一些不和具体对象关联、不需要new就调用的方法
- 举例：GameCore中的所有方法几乎都是static方法。
- static方法不需要new就可以直接通过类名调用。
- static变量不需要new就可以直接通过类名调用。static变量是共享一块儿内存空间，非static变量则是各对象隔离的。
- static 方法中无法使用this关键字，因为static独立于对象存在，不是任何人的唯一。
- static成员中只能访问static成员，不能直接访问非static成员。非static成员可以访问static成员。
- 画内存图。

# 单例模式

- 有的类在系统中只能有一个对象（\*，资源管理器、缓存管理等），这时就要使用“单例模式”（singleton）。实现单例模式有很多方法，先介绍最简单、最实用的“饿汉式”。
- 构造函数声明为private，这样避免外界访问
- 定义一个private final static的对象实例，static成员的初始化只在类第一次使用的时候执行一次。
- 定义一个public static的getInstance方法，返回唯一实例
- 案例：封装一个NumberManager作为游戏元素编号的统一管理器，为了简化，不同类型的游戏元素编号也不重复。用NumberManager为RPSprite等提供无参构造函数。
- 为什么要private，为什么要final

# 案例：用面向对象封装GameCore



- 封装一个编号生成器NumberManager
- RPSprite、RPText、RPImage，可能有多重重载的构造函数。封装基本的GameCore中的操作。
- 再增加moveLeft、moveRight、moveUp、moveDown等方法。封装向上跳的功能。封装相交判断的功能。
- 封装RPSoundPlayer

# 枚举

- 为什么要有枚举：限定一个数据的可选值范围，比如“方向”。
- 枚举用法: `enum Dir{Left,Right,Up,Down}`
- 枚举的原理，模拟实现枚举：构造函数设置为private，把几个对象设置为public static final实例。
- 静态成员只会初始化一次。

# 类的静态代码块

```
class MyTest
{
    static
    {
        System.out.println("zi 静态代码块");
    }
    public MyTest()
    {
        System.out.println("zi 构造方法");
    }
}
```

```
MyTest t1 = new MyTest();
```

```
MyTest t2 = new MyTest();
```

静态代码块在类第一次被使用的时候执行一次，在构造函数执行之前执行。一般用于对类进行初始化，以后会用

只要一旦开始“接触一个类”，静态成员、static代码块就会被执行

# 继承(inherit)

- RPSprite、RPIImage、RPText等每个类都要封装show、hide、读取大小等，很麻烦。
- Java中一个类可以“继承”自其他类，如果A继承自B，则A叫做B的子类，B叫做A的父类(基类)。子类会从父类继承所有非private成员。子类还可以有子类。
- Java中一个类只能有一个父类（单继承），如果没指定父类，则Java内置的Object为父类。

```
class FuLei
```

```
{  
    private void method1()  
    {  
    }  
    public void method2()  
    {  
    }  
}
```

```
ZiLei zl1 = new ZiLei();  
zl1.method1();  
zl1.method2();  
zl1.method4();  
zl1.toString();
```

```
class ZiLei extends FuLei
```

```
{  
    private void method3()  
    {  
    }  
    public void method4()  
    {  
    }  
}
```

# 继承中的构造函数调用顺序

- 子类的构造方法默认都去访问了父类的无参构造方法：在子类中的构造方法中都有一行默认语句:super()

```
class Fu
{
    public Fu()
    {
        System.out.println("fu");
    }
}
class Zi extends Fu
{
    public Zi()
    {
        super(); //不管是否显式调用，控制台都会输出fu
        System.out.println("zi");
    }
}
Zi z = new Zi();
```

- 先执行父类的构造函数把父类初始化完成，再初始化子类的。

# 调用有参构造函数

可以通过super(参数)去访问父类中的有参构造函数。可以通过this(参数...)去访问本类中的其他构造函数。

```
class Fu
{
    public Fu(int a)
    {
        System.out.println("fu"+a);
    }
}
class Zi extends Fu
{
    public Zi()
    {
        //super();
        super(20);
        System.out.println("zi");
    }
    public Zi(int a)
    {
        //super();
        super(a);
        System.out.println("zi" +a);
    }
}
```



# 总结

- 1、子类对象创建的时候会先调用父类的构造函数，再调用子类的构造函数。
- 2、在子类的构造函数中，如果没有显式的调用父类的构造函数，则相当于在最开始默认调用父类的无参构造函数。`super();`（额外，`super()`必须是第一行代码）
- 3、如果父类没有无参构造函数，那么子类必须显式调用父类的其他的构造函数。

# private/public/protected

- private成员无法被子类访问，子类只能通过父类的非private方法“间接”访问父类的private成员。这样保证了父类private成员的安全性。
- 子类的对象相当于把父类的成员都拷贝一份出来，对象就是“类”的模子扣出来的，继承是“模子”的集成，数据不会继承。
- protected成员只能被自己以及子类（直接或者间接）以及兄弟（同package包中）访问，无法被“外姓人”访问。
- default（什么都不写）：同一个package包中可以访问。

# 重载问题

- `f1(String s1,Object obj),f1(String s1,String s2)` , 的 `f1("aaa","bbb")` 匹配问题。如果非要匹配 `f1(String s1,Object obj)` 呢？

# 多态

- 面向对象三大特征：封装、继承、多态。多态是面向对象最强大的一个特征，也是最难的一个特征，设计模式等都是多态的体现。大型项目架构也大量应用多态。
- 子类中定义和父类中完全一样（名字、参数、返回值）的方法就叫“重写(Override)或覆盖”

```
class DiQiuRen
{
    public void speak(){System.out.println("我是地球人");}
}
class Chinese extends DiQiuRen
{
    @Override
    public void speak() {System.out.println("我是中国人");}
    public void baiNian() {System.out.println("过年好! ");}
}
DiQiuRen dqr1 = new DiQiuRen();
dqr1.speak();
Chinese zgr1 = new Chinese();
zgr1.speak();
```

- 下面的执行结果是什么？

```
Chinese zgr2 = new DiQiuRen();//地球人不一定是中国人
```

```
DiQiuRen dqr2 = new Chinese();
```

```
dqr2.speak();
```

```
DiQiuRen dqr2 = zgr1;
```

```
dqr2.speak();
```

- 父类变量可以指向子类对象，内存中也还是子类对象，有且只有这一个对象。变量是什么类型没关系，到底执行谁的方法主要取决于内存中的对象是什么类型。
- 变量类型是“把对象看成什么”，`DiQiuRen dqr2 = new Chinese()`是把`Chinese`对象看成是“地球人”，因为“地球人”不一定有`baiNian`方法，因此`dqr2.baiNian()`编译失败。
- 能够调用什么方法由变量类型决定，执行谁的方法由实际指向的对象决定。
- 关于`@Override`和中间父类中定义`override`方法的问题
- 再测试：方法声明为返回`DiQiuRen`，返回一个`Chinese`实例

# 类型转换

- 下面程序的执行结果是什么？

```
Chinese zgr5 = (Chinese)dqr2;
```

```
zgr5.sayHello();
```

```
zgr5.baiNian();
```

```
Chinese zgr5 = (Chinese)dqr1;
```

- ()类型转换可以把“以父类类型角度观察的对象的实例”重新以“子类类型角度观察”。(显式类型转换/强制类型转换)
- 如果对象就是父类对象，当以子类类型观察的时候会失败，抛出运行期异常，编译器无法发现这个错误。
- 类型转换只能在有父子关系的类中进行
- 关于super关键字。
- 思考：
  - 1、定义一个方法 `void test1(DiQiuRen dqr){dqr.sayHello();}`  
如下调用`test1(new Chinese());`可以吗？运行结果是什么？
  - 2、`String s = (String)zgr1;`可以吗？

# 更多多态效果

- 子类中调用super调用谁？怎么调用爷爷的方法？
- 在父类方法中调用虚方法，是执行的父类的方法还是子类的方法？

# 两个问题

- final方法
- final类
- 如果父类中的某些方法不希望被子类Override，比如kill，那么标记为final即可。好处：安全，可以保证父类定义的行为的执行(必须“被杀死”)；效率高，避免了查询虚方法表(\*)。
- 如果某些类不希望被继承，类也可以声明为final。
- final 类的好处主要是安全（不会有冒名顶替的“不肖子孙”）。String类就是final。



# 抽象类

- DiQiuRen的sayHello输出“我是地球人”显然不合理，因为无法确定怎么说，也就是DiQiuRen不知道如何sayHello，只有具体到中国人、日本人、美国人才知道如何sayHello
- 把DiQiuRen的sayHello的方法体去掉，并且方法增加abstract修饰，类也修饰为abstract：  

```
abstract class DiQiuRen
{
    public abstract void speak();
}
```
- 抽象方法没有方法体；一旦类中定义了抽象方法，类必须被修饰为抽象；抽象类无法实例化(new)。

# 抽象类子类

- 抽象类的抽象方法主要用来限制子类“必须实现这些抽象方法”；子类也可以不实现，那么子类也要是抽象类，由子类的子类.....去实现。
- Eclipse自动实现方法的助手：左边的错误提示，初学先自己写。
- Eclipse中可以通过Alt+/或者主菜单“源码”→“覆盖/实现方法”实现覆盖/实现父类方法。
- 提问：抽象类中必须有抽象方法吗？
- 怎么看起来好像创建了抽象类的对象？

# 善用调试

- 如果通过断点发现问题，通过断点查看变量信息。
- 如何通过设置断点、猜测，去发现bug。
- 条件断点。
- 如何通过调试方式运行，程序运行中改代码直接就起作用（HotSwap）。但是如果修改了类结构、加了方法、改了方法、改了签名等就要重启。



# 案例：改进游戏引擎



- 把show\*\*\*、hide\*\*\*、set\*\*\*Position、get\*\*\*Position等方法封装到一个RPGGameObject抽象类中，由具体子类去实现。
- 蛋疼？NO，试着在RPGGameObject中写一个moveLeft方法，set\*\*\*Position、get\*\*\*Position是虚的，借助着两个虚的moveLeft可以干实事儿，通过调试器看看执行过程。
- 虚方法定义“能干什么”，子类中的方法定义具体“干什么”。
- 提供一个Image和Sprite的父类，这个父类封装getSize方法，把矩形相交判断加入这个父类中。
- 使用OO游戏引擎编写一个“潘金莲和武大郎”的动作片
- 使用OO游戏引擎重写“吃金币”游戏。金币数据、墙数据都写成自定义类。
- 实现“打砖块”游戏

# 匿名内部类

- 有时候创建一个类的子类只是为了创建一次他的对象，声明一个子类很麻烦，可以直接**new 类名()** {

**//类体**

- **};**
- 这样就是相当于声明了一个这类的子类，然后并且创建这个类的对象。注意：不是创建这个父类的对象。
- 可以直接在类体中**override**，因为类体中定义的特有方法“很难”被调用，所以一般不这样做。
- 常见用法：直接声明一个抽象类的子类，然后**new**。注意：创建的不是抽象类的对象。

# 匿名内部类访问局部变量（闭包）



```
final int i = 5;
Person p1 = new Person() {
    public void speak()
    {
        System.out.println(i);
    }
};
p1.speak();
```

局部变量（参数）必须被标注为final，防止他变

匿名内部类中this的问题：类中this指的是匿名内部类，如果要想指外部的对象，要用“类名.this”

看似这样没意义，以后就会知道他的意义了

# 接口

接口是一种用来声明“能力”的类型，不提供具体实现  
不提供实现方法，连{}都不能有。

接口无法实例化，只能被类“实现”（implements）

```
public interface Speakable
```

```
{
```

```
    public void speak();
```

```
}
```

```
public class TeacherCang implements Speakable
```

```
{
```

```
    ...
```

```
}
```

既可以使用接口类型变量又可以使用类类型变量调用speak

接口的意义是定义“做什么”，类定义“怎么做”

# 接口

- 接口中只能声明static 成员变量，不能声明普通成员变量。因为static 可以被外界用，一个没有实现代码的接口中声明普通成员变量没意义。而且必须要是public。
- 接口中可以定义多个方法，也可以不定义任何方法(\* 标识接口)。
- 接口只是“能力”不是“实现”，因此不能也必要定义构造函数。
- 类只能有一个父类，类可以实现多个接口。测试一下：Speakable、Walkable。
- 接口可以继承其他接口：extends。
- 接口也有多态、类型转换。
- 接口和抽象类的异同。



# 接口案例

- 定义一个Flyable接口（内有fly方法），飞机（Plane）和鸟（Bird）都实现这个接口。定义一个ToyCreator内有createFlyable(string name)，根据name的不同返回实现了Flyable接口的不同类的对象，但是都可以调用fly方法。
- 注意：这不是new了接口，只是返回了一个实现了这个接口的类的对象。
- 飞机和鸟之间没有继承关系，所以要使用接口。
- 当然不能直接调用Bird独有的方法。

# 接口案例

- 编写一个可以对任意类型数据数组都进行排序的方法  
`sort(RPComparable a, Object[] values);` //加个RP前缀避免和Java内置的类名、接口名冲突。

```
interface RPComparable
```

```
{  
    int compareTo(object obj2);  
}
```

注意，不能给values参数传递int[]，但是可以传递Integer[]

- 使用sort来对各种数据类型做排序。
- 体会优点：可以对自己都未知的类型排序。
- 体会接口的用途：约束类有哪些方法。
- 练习：编写一个可以对任意类型数据数组都进行获取最大值的方法  
`Object getMax(RPComparable a, Object[] values);`
- 接口可以有多个方法，也可以一个方法都没有。

# 补充

- java中==就是用来比较是否指向同一个对象，如果要进行内容的比较，则需要提供 equals 方法或者实现 Comparable接口。举例：比较两个Person
- 为什么String比较要用equals，而有时候使用==也能进行相等判断（反编译）

# 接口和匿名内部类

- 和匿名内部类非常类似。
- 看看我们的GameCore.start吧

# (\*)游戏应用：asyncRun



- 要用新版RupengGame
- 代码运行在单独的线程中，不会阻塞主线程。而且可以写到单独的逻辑中。  
run方法是谁调用的？把这段代码给你，你给我在另外一个线程中执行。
- 把检测是否击中、NPC（怪物等）的控制使用asyncRun，这样思路更简单
- 案例：马里奥在路上走，老虎用asyncRun进行AI控制（满多少步之后随机进行下一个方向的行走），一旦被老虎碰到就死了。

# (\*)键盘事件

- 看不清呀，调整eclipse控制台字体。
- 如果没有在读取getKeyPressed，那么是不知道“有按键按下”的，而且还要反复的读取，很累，用事件可以在用户按下按键的时候再执行我们的代码。
- “回调”：Don't call me,I will call you
- `GameCore.addKeyListener` 重复按键中keyReleased只会调用一次。
- 关于KeyAdapter抽象类，不错的设计。
- 使用keyReleased实现弹跳会更好。实现一下。
- 案例：实现微信打飞机，控制左右移动，空格发射炮弹。

# (\*)鼠标事件

- (\*) 要升级GameCore.jar包版本（新版本才支持**get\*\*\*PositionOnScreen()**），**get\*\*\*PositionOnScreen()**用来获取一个元素在屏幕上的位置
- `GameCore.addMouseListener`，根据**getPositionOnScreen()**；计算一个元素有没有被点到。
- `MouseAdapter`和`MouseListener`很好的结合。
- 封装出游戏对象的OnClick方法
- 案例：做一个播放器，播放鼠标点击的歌曲（给**GameObject**增加一个**setTag**、**getTag**方法简化开发）；点击【静音】图标。
- (\*) 熊猫跑酷素材、坦克大战资源、超级玛丽资源。