**C++编码编码规范**

**一、文件排版方面**

##### 1. 包含头文件

　• 先系统头文件，后用户头文件。  
　• 系统头文件，稳定的目录结构，应采用包含子路径方式。  
　• 自定义头文件，不稳定目录结构，应在dsp中指定包含路径。  
　• 系统头文件应用：#include <xxx.h>  
　• 自定义同文件应用：#include "xxx.h"  
　• 只引用需要的头文件。

##### 2. h和cpp文件

　• 头文件命名为.h，内联文件命名为.inl；C++文件命名为\*.cpp  
　• 文件名用大小写混合，或者小写混合。例如DiyMainview.cpp，infoview.cpp，Diy\_Mainview.cpp。不要用无意义的名称：例如XImage.cpp；SView.cpp；xlog.cpp；  
　• 头文件除了特殊情况，应使用#ifdef控制块。  
　• 头文件#endif应采用行尾注释。  
　• 头文件，首先是包含代码块，其次是宏定义代码块，然后是全局变量，全局常量，类型定义，类定义，内联部分。  
　• CPP文件，包含指令，宏定义，全局变量，函数定义。

##### 3. 文件结构

　• 文件应包含文件头注释和内容。  
　• 函数体类体之间原则上用2个空行，特殊情况下可用一个或者不需要空行。

##### 4. 空行

　• 文件头、控制块，#include部分、宏定义部分、class部分、全局常量部分、全局变量部分、函数和函数之间，用一个空行。

**二、注释方面**

##### 1. 文件头注释

　• 作者，文件名称，文件说明，生成日期(可选)

##### 2. 函数注释

　• 关键函数必须写上注释，说明函数的用途。  
　• 特别函数参数，需要说明参数的目的，由谁负责释放等等。  
　• 除了特别情况，注释写在代码之前，不要放到代码行之后。  
　• 对每个#else或#endif给出行末注释。  
　• 关键代码注释，包括但不限于：赋值，函数调用，表达式，分支等等。  
　• 善未实现完整的代码，或者需要进一步优化的代码，应加上 // TODO …  
　• 调试的代码，加上注释 // only for DEBUG  
　• 需要引起关注的代码，加上注释 // NOTE …  
　• 对于较大的代码块结尾，如for,while,do等，可加上 // end for|while|do

**三、命名方面**

##### １. 原则

　• 同一性：在编写一个子模块或派生类的时候，要遵循其基类或整体模块的命名风格，保持命名风格在整个模块中的同一性。  
　• 标识符组成：标识符采用英文单词或其组合，应当直观且可以拼读，可望文知意，用词应当准确，避免用拼音命名。  
　• 最小化长度 && 最大化信息量原则：在保持一个标识符意思明确的同时，应当尽量缩短其长度。  
　• 避免过于相似：不要出现仅靠大小写区分的相似的标识符，例如"i"与"I"，"function"与"Function"等等。  
　• 避免在不同级别的作用域中重名：程序中不要出现名字完全相同的局部变量和全局变量，尽管两者的作用域不同而不会发生语法错误，但容易使人误解。  
　• 正确命名具有互斥意义的标识符：用正确的反义词组命名具有互斥意义的标识符，如："nMinValue" 和"nMaxValue"，"GetName()" 和"SetName()" ….  
　• 避免名字中出现数字编号：尽量避免名字中出现数字编号，如Value1,Value2等，除非逻辑上的确需要编号。这是为了防止程序员偷懒，不肯为命名动脑筋而导致产生无意义的名字（因为用数字编号最省事）。

##### ２. T,C,M,R类

　• T类表示简单数据类型，不对资源拥有控制权，在析构过程中没有释放资源动作。  
　• C表示从CBase继承的类。该类不能从栈上定义变量，只能从堆上创建。  
　• M表示接口类。  
　• R是资源类，通常是系统固有类型。除了特殊情况，不应在开发代码中出现R类型。

##### ３. 函数名

　• M类的函数名称应采用HandleXXX命名，例如：HandleTimerEvent；不推荐采用java风格，例如handleTimerEvent；

　• Leave函数，用后缀L。  
　• Leave函数，且进清除栈，用后缀LC。  
　• Leave函数，且删除对象，用后缀LD。

##### ４. 函数参数

　• 函数参数用a作为前缀。  
　• 避免出现和匈牙利混合的命名规则如apBuffer名称。用aBuffer即可。  
　• 函数参数比较多时，应考虑用结构代替。  
　• 如果不能避免函数参数比较多，应在排版上可考虑每个参数占用一行，参数名竖向对齐。

##### ５. 成员变量

　• 成员变量用m最为前缀。  
　• 避免出现和匈牙利混合的命名规则如mpBuffer名称。用mBuffer即可。

##### ６. 局部变量

　• 循环变量和简单变量采用简单小写字符串即可。例如，int i;  
　• 指针变量用p打头，例如void\* pBuffer;

##### ７. 全局变量

　• 全局变量用g\_最为前缀。

##### ８. 类名

　• 类和对象名应是名词。  
　• 实现行为的类成员函数名应是动词。  
　• 类的存取和查询成员函数名应是名词或形容词。

##### ９. 风格兼容性

　• 对于移植的或者开源的代码，可以沿用原有风格，不用C++的命名规范。

**四、代码风格方面**

##### 1. Tab和空格

　• 每一行开始处的缩进只能用Tab，不能用空格，输入内容之后统一用空格。除了最开始的缩进控制用Tab，其他部分为了对齐，需要使用空格进行缩进。这样可以避免在不同的编辑器下显示不对齐的情况。  
　• 在代码行的结尾部分不能出现多余的空格。  
　• 不要在"::","->","."前后加空格。  
　• 不要在"，","；"之前加空格。

##### 2. 类型定义和{

　• 类，结构，枚举，联合：大括号另起一行

##### 3. 函数

　• 函数体的{需要新起一行，在{之前不能有缩进。  
　• 除了特别情况，函数体内不能出现两个空行。  
　• 除了特别情况，函数体内不能宏定义指令。  
　• 在一个函数体内，逻揖上密切相关的语句之间不加空行，其它地方应加空行分隔。  
　• 在头文件定义的inline函数，函数之间可以不用空行，推荐用一个空行。

##### 4. 代码块

　• "if"、"for"、"while"、"do"、"try"、"catch" 等语句自占一行，执行语句不得紧跟其后。不论执行语句有多少都要加 “{ }” 。这样可以防止书写和修改代码时出现失误。  
　• "if"、"for"、"while"、"do"、"try"、"catch" 的括号和表达式，括号可紧挨关键字，这样强调的是表达式。

##### 5. else

• if语句如果有else语句，用 } else { 编写为一行，不推荐用 3 行代码的方式。

##### 6. 代码行

　• 一行代码只做一件事情，如只定义一个变量，或只写一条语句。这样的代码容易阅读，并且方便于写注释。  
　• 多行变量定义，为了追求代码排版美观，可将变量竖向对齐。  
　• 代码行最大长度宜控制在一定个字符以内，能在当前屏幕内全部可见为宜。

##### 7. switch语句

　• case关键字应和switch对齐。  
　• case子语句如果有变量，应用{}包含起来。  
　• 如果有并列的类似的简单case语句，可考虑将case代码块写为一行代码。  
　• 简单的case之间可不用空行，复杂的case之间应考虑用空行分割开。  
　• case字语句的大括号另起一行，不要和case写到一行。  
　• 为所有switch语句提供default分支。  
　• 若某个case不需要break一定要加注释声明。

##### 8. 循环

　• 空循环可用 for( ;; ) 或者 while( 1 ) 或者 while( true )

##### 9. 类

　• 类继承应采用每个基类占据一行的方式。  
　• 单继承可将基类放在类定义的同一行。如果用多行，则应用Tab缩进。  
　• 多继承在基类比较多的情况下，应将基类分行，并采用Tab缩进对齐。  
　• 重载基类虚函数，应在该组虚函数前写注释 // implement XXX  
　• 友元声明放到类的末尾。

##### 10. 宏

　• 不要用分号结束宏定义。  
　• 函数宏的每个参数都要括起来。  
　• 不带参数的宏函数也要定义成函数形式。

##### 11. goto

　• 尽量不要用goto。

**五、类型**

• 定义指针和引用时\*和&紧跟类型。  
• 尽量避免使用浮点数，除非必须。  
• 用typedef简化程序中的复杂语法。  
• 避免定义无名称的类型。例如：typedef enum { EIdle, EActive } TState;  
• 少用union，如果一定要用，则采用简单数据类型成员。  
• 用enum取代(一组相关的)常量。  
• 不要使用魔鬼数字。  
• 尽量用引用取代指针。  
• 定义变量完成后立即初始化，勿等到使用时才进行。  
• 如果有更优雅的解决方案，不要使用强制类型转换。

**六、表达式**

• 避免在表达式中用赋值语句。  
• 避免对浮点类型做等于或不等于判断。  
• 不能将枚举类型进行运算后再赋给枚举变量。  
• 在循环过程中不要修改循环计数器。  
• 检测空指针，用 if( p )  
• 检测非空指针，用 if( ! p )

**七、函数**

##### １. 引用

　• 引用类型作为返回值：函数必须返回一个存在的对象。  
　• 引用类型作为参数：调用者必须传递一个存在的对象。

##### ２. 常量成员函数

　• 表示该函数只读取对象的内容，不会对对象进行修改。

##### ３. 返回值

　• 除开void函数，构造函数，析构函数，其它函数必须要有返回值。  
　• 当函数返回引用或指针时，用文字描述其有效期。

##### ４. 内联函数

　• 内联函数应将函数体放到类体外。  
　• 只有简单的函数才有必要设计为内联函数，复杂业务逻辑的函数不要这么做。  
　• 虚函数不要设计为内联函数。

##### ５. 函数参数

　• 只读取该参数的内容，不对其内容做修改，用常量引用。  
　• 修改参数内容，或需要通过参数返回，用非常量应用。  
　• 简单数据类型用传值方式。  
　• 复杂数据类型用引用或指针方式。

**八、类**

##### 1. 构造函数

　• 构造函数的初始化列表，应和类的顺序一致。  
　• 初始化列表中的每个项，应独占一行。  
　• 避免出现用一个成员初始化另一个成员。  
　• 构造函数应初始化所有成员，尤其是指针。  
　• 不要在构造函数和析构函数中抛出异常。

##### 2. 纯虚函数

　• M类的虚函数应设计为纯虚函数。

##### 3. 构造和析构函数

　• 如果类可以继承，则应将类析构函数设计为虚函数。  
　• 如果类不允许继承，则应将类析构函数设计为非虚函数。  
　• 如果类不能被复制，则应将拷贝构造函数和赋值运算符设计为私有的。  
　• 如果为类设计了构造函数，则应有析构函数。

##### 4. 成员变量

　• 尽量避免使用mutable和Volatile。  
　• 尽量避免使用公有成员变量。

##### 5. 成员函数

　• 努力使类的接口少而完备。  
　• 尽量使用常成员函数代替非常成员函数，const函数  
　• 除非特别理由，绝不要重新定义(继承来的)非虚函数。（这样是覆盖，基类的某些属性无初始化）

##### 6. 继承

　• 继承必须满足IS-A的关系，HAS-A应采用包含。  
　• 虚函数不要采用默认参数。  
　• 除非特别需要，应避免设计大而全的虚函数，虚函数功能要单一。  
　• 除非特别需要，避免将基类强制转换成派生类。

##### 7. 友元

　• 尽量避免使用友元函数和友元类。

**九、错误处理**

• 申请内存用new操作符。  
• 释放内存用delete操作符。  
• new和delete，new[]和delete[]成对使用。  
• 申请内存完成之后，要检测指针是否申请成功，处理申请失败的情况。  
• 谁申请谁释放。优先级：函数层面，类层面，模块层面。  
• 释放内存完成后将指针赋空，避免出现野指针。  
• 使用指针前进行判断合法性，应考虑到为空的情况的处理。  
• 使用数组时，应先判断索引的有效性，处理无效的索引的情况。  
• 代码不能出现编译警告。  
• 使用错误传递的错误处理思想。  
• 卫句风格：先处理所有可能发生错误的情况，再处理正常情况。  
• 嵌套do-while(0)宏：目的是将一组语句变成一个语句，避免被其他if等中断。

**十、性能**

• 使用前向声明代替#include指令。Class M;  
• 尽量用++i代替i++。即用前缀代替后缀运算。  
• 尽量在for循环之前，先写计算估值表达式。  
• 尽量避免在循环体内部定义对象。  
• 避免对象拷贝，尤其是代价很高的对象拷贝。  
• 避免生成临时对象，尤其是大的临时对象。  
• 注意大尺寸对象数组。  
• 80-20原则。

**十一、兼容性**

• 遵守ANSI C和ISO C++国际标准。  
• 确保类型转换不会丢失信息。  
• 注意双字节字符的兼容性。  
• 注意运算溢出问题。  
• 不要假设类型的存储尺寸。  
• 不要假设表达式的运算顺序。  
• 不要假设函数参数的计算顺序。  
• 不要假设不同源文件中静态或全局变量的初始化顺序。  
• 不要依赖编译器基于实现、未明确或未定义的功能。  
• 将所有#include的文件名视为大小写敏感。  
• 避免使用全局变量、静态变量、函数静态变量、类静态变量。在使用静态库，动态库，多线程环境时，会导致兼容性问题。  
• 不要重新实现标准库函数,如STL已经存在的。

C++编码规范例子：

StudentInformation.cpp:

#include "base.h"

/\*

作者：宋佳琪

文件名称:学生信息系统

日期：2017年9月3号

\*/

int main ()

{

cout<<"Please choose:"<<endl<<"1.set"<<endl<<"2.add"<<endl<<"3.check"<<endl<<"4.alter"<<endl<<"5.clear"<<endl<<"6.show"<<endl<<"7.file\_in"<<endl<<"8.file\_out"<<endl<<"9.exit"<<endl;

int a;

cin>>a;

base b;

while(a!=9)

{

if(a==1)

b.SetValue();

if(a==2)

b.AddValue();

if(a==3)

b.CheckValue();

if(a==4)

b.AlterValue();

if(a==5)

b.ClearValue();

if(a==6)

b.ShowValue();

if(a==7)

b.FileIn();

if(a==8)

b.FileOut();

cout<<"What else do you want to choose?"<<endl;

cin>>a;

}

return 0;

}

base.h:

#pragma once

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct information

{

int num;

string name;

string sex;

int age;

int clas;

string college;

string major;

information \* next;

};

class base

{

private:

information \* ps;

information \* phead;

information \* pend;

public:

base(void);

~base(void);

void SetValue (); //创建

void AddValue (); //添加

void CheckValue (); //查询

void AlterValue (); //修改

void ClearValue (); //删除

void ShowValue (); //显示

void FileIn (); //文件导入

void FileOut (); //文件导出

};

base.cpp:

#include "base.h"

base::base(void)

{

information \* ps=NULL;

information \* phead=NULL;

information \* pend=NULL;

}

base::~base(void)

{

}

//创建（仅限用户第一次使用该通讯录，输入第一个人的信息）

void base::SetValue ()

{

ps=new information;

cout<<"Please input the information:"<<endl;

cin>>ps->num>>ps->name>>ps->sex>>ps->age>>ps->clas>>ps->college>>ps->major;

phead=ps;

pend=ps;

pend->next=NULL;

}

//添加

void base::AddValue ()

{

ps=new information;

cout<<"Please input the information:"<<endl;

cin>>ps->num>>ps->name>>ps->sex>>ps->age>>ps->clas>>ps->college>>ps->major;

pend->next=ps;

pend=ps;

pend->next=NULL;

}

//查询

void base::CheckValue()

{

cout<<"Please choose;1.according to number 2.according to class "<<endl;

ps=phead;//ps作为查询的指针先指向头指针，从头开始一一对比

int a;

cin>>a;

if(a==1)//按学号查询

{

cout<<"Please input the number:"<<endl;

int n;

cin>>n;

int b=0;

for(;ps!=NULL;ps=ps->next)

{

if(ps->num==n)

{

cout<<ps->num<<" "<<ps->name<<" "<<ps->sex<<" "<<ps->age<<" "<<ps->clas<<" "<<ps->college<<" "<<ps->major<<endl;

b=1;

break;

}

}

if(b==0)

cout<<"Can't find out the information"<<endl;

}

if(a==2)//按班级查询

{

cout<<"Please input the class:"<<endl;

int c;

cin>>c;

int b=0;

for(;ps!=NULL;)

{

if(ps->clas==c)

{

cout<<ps->num<<" "<<ps->name<<" "<<ps->sex<<" "<<ps->age<<" "<<ps->clas<<" "<<ps->college<<" "<<ps->major<<endl;

b=1;

}

ps=ps->next;

}

if(b==0)

cout<<"Can't find out the information"<<endl;

}

/\*delete ps;\*/

}

//修改

void base::AlterValue()

{

cout<<"Pleas input the number:"<<endl;

int n;

cin>>n;

int b=0;

ps=phead;

for(;ps!=NULL;ps=ps->next)

{

if(ps->num==n)

{

cin>>ps->num>>ps->name>>ps->sex>>ps->age>>ps->clas>>ps->college>>ps->major;

b=1;

break;

}

}

if(b==0)

cout<<"infoemation error!"<<endl;

}

//删除

void base::ClearValue ()

{

cout<<"Please input the number:"<<endl;

int n;

cin>>n;

ps=phead;

information \* pre=NULL;//pre作为删除项的前一项

if(phead->num==n)//表示所需要删除的在第一个的情况，将头指针向后挪一个然后删除

{

phead=ps->next;

}

if(phead->num!=n)//表示所需要删除的在除了第一个位置的情况

{

pre=phead;

for(ps=phead->next;ps!=NULL;)

{

if(ps->num!=n)

{

pre=ps;

ps=ps->next;

}

if(ps->num==n)

{

pre->next=ps->next;

break;

}

}

}

}

//（显示通讯录中所有信息）

void base::ShowValue ()

{

ps=phead;

for(;ps!=NULL;ps=ps->next)

cout<<ps->num<<" "<<ps->name<<" "<<ps->sex<<" "<<ps->age<<" "<<ps->clas<<" "<<ps->college<<" "<<ps->major<<endl;

}

//文件导入（从文件中读取若干条通讯录）

void base::FileIn()

{

ifstream infile("Student\_information.txt",ios::in);

if(!infile)

{

cerr<<"open Student\_information.txt error"<<endl;

exit(1);

}

information i[100];

int k=0;

for(;k<100;k++)

{

infile>>i[k].num>>i[k].name>>i[k].sex>>i[k].age>>i[k].clas>>i[k].college>>i[k].major;

if(i[k].num==000)

break;

else

cout<<i[k].num<<" "<<i[k].name<<" "<<i[k].sex<<" "<<i[k].age<<" "<<i[k].clas<<" "<<i[k].college<<" "<<i[k].major<<endl;

}

infile.close();

}

//文件导出（将通讯录输出到文件中）

void base::FileOut()

{

ofstream outfile("Student\_information.txt",ios::out);

if(!outfile)

{

cerr<<"open Student\_information.txt error"<<endl;

exit(1);

}

for(ps=phead;ps!=NULL;ps=ps->next)

{

outfile<<ps->num<<" "<<ps->name<<" "<<ps->sex<<" "<<ps->age<<" "<<ps->clas<<" "<<ps->college<<" "<<ps->major<<endl;

}

outfile.close();

}

**JAVA编码规范**

**一、java文件组织**

文件组织规则：由于超过2000行的程序难以阅读，应该尽量避免出现超过2000行的程序。一个Java源文件都包含一个单一的公共类或接口。若私有类和接口与一个公共类相关联，可以将它们和公共类放入同一个源文件。公共类必须是这个文件中的第一个类或接口。

文件组织顺序：

1、文件注释：所有的源文件都应该在开头有一个注释，其中列出文件的版权声明、文件名、功能描述以及创建、修改记录。

2、包和引入语句：在多数Java源文件中，第一个非注释行是包语句。在它之后可以跟导包语句。

3、 类或接口注释：采用JavaDoc文档注释，在类、接口定义之前应当对其进行注释，包括类、接口的描述、最新修改者、版本号、参考链接等。

4、类或接口的声明

5、类或接口的实现注释：如果有关类或接口的信息不适合作为“类或接口文档注释”，可以在类或接口的实现注释中给出；

6、类的（静态）变量：首先是类的公共变量，随后是保护变量，再后是包一级别的变量（没有访问修饰符），最后是私有变量；

7、实例变量：首先是公共级别的，随后是保护级别的，再后是包一级别的（没有访问修饰符），最后是私有级别的；

8、构造方法；

9、普通方法：方法应该按功能分组，而不应该按作用域或访问权限进行分组。

**二、代码风格**

      1、缩进：程序块要采用缩进风格编写，缩进只使用TAB键，不能使用空格键（编辑器中请将TAB设置为4格）；方法体的开始、类的定义、以及if、for、do、while、switch、case语句中的代码都要采用缩进方式；

      2、对齐：程序块的分界符左大括号"{" 和右大括号"}"都另起一行，应各独占一行并且位于同一列，同时与引用它们的语句左对齐；对齐只使用TAB键，不使用空格键；不允许把多个短语句写在一行中，即一行只写一条语句；if、for、do、while、case、switch、default等语句自占一行。

3、换行：一行的长度超过80个字符需要换行，换行规则如下：

在一个逗号后面断开；

在一个操作符前面断开；

长表达式要在低优先级操作符处划分新行；

新行缩进2个TAB。

4、间隔：类、方法及相对独立的程序块之间、变量说明之后必须加空行；关键字之后要留空格， 象if、for、while  等关键字之后应留一个空格再跟左括号"（"， 以突出关键字；方法名与其左括号"（"之间不要留空格, 以与关键字区别；二元操作符如   " ="、" +="  " >="、" <="、" +"、" \*"、" %"、" &&"、" ||"、" <<" ," ^" 等的前后应当加空格；一元操作符如" !"、" ~"、" ++"、" --"等前后不加空格；xiang"［ ］"、" ." 这类操作符前后不加空格；for语句中的表达式应该被空格分开；强制转型后应该跟一个空格。

三、注释

1、原则：对已经不推荐使用的类和方法需要注明@Deprecated，并说明替代的类或者方法；对于针对集合、开关的方法，要在方法注释中表明是否多线程安全。

2、字段注释: 采用JavaDoc文档注释，定义为public的字段必需给出注释，在类的(静态)变量、实例变量定义之前当对其进行注释，给出该字段的描述等：

3、方法注释：采用JavaDoc文档注释，在方法定义之前当对其进行注释，包括方法的描述、输入、输出及返回值说明、抛出异常说明、参考链接等：

4、单行注释格式//

5、多行注释格式/\*……\*/

**三、命名规则**

1、基本规则:使用可以准确说明变量、字段、类、接口、包等完整的英文描述符；采用大小写混合，提高名字的可读性；采用该领域的术语；尽量少用缩写，但如果一定要使用，当使用公共缩写和习惯缩写等；避免使用相似或者仅在大小写上有区别的名字。

2、包命名：包名一律小写, 少用缩写和长名；采用以下规则：

                                  [基本包].[项目名].[模块名].[子模块名]...

不得将类直接定义在基本包下，所有项目中的类、接口等都应当定义在各自的项目和模块包中。

3、类或接口命名：类或接口名是个一名词，采用大小写混合的方式，每个单词的首字母大写。尽量使你的类名简洁而富于描述。使用完整单词，避免用缩写词(除非该缩写词被更广泛使用，像URL，HTML)。

4、变量命名: 采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写；变量名不应以下划线或美元符号开头；尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为i，j，k，m和n，它们一般用于整型；c，d，e，它们一般用于字符型；不采用匈牙利命名法则，对不易清楚识别出该变量类型的变量应使用类型名或类型名缩写作其后缀；组件或部件变量使用其类型名或类型名缩写作其后缀；集合类型变量，例如数组和矢量，应采用复数命名或使用表示该集合的名词做后缀。

5、常量命名：全部采用大写，单词间用下划线隔开。

6、方法命名：方法名是一个动词，采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写；取值类可使用get前缀，设值类可使用set前缀，判断类可使用is(has)前缀。

**四、声明**

1、类或接口的声名：类、接口定义语法规范如下

[可见性][('abstract'|'final')] [Class|Interface] class\_name

[('extends'|'implements')][父类或接口名]{

//方法体

}

2、方法声明：良好的程序设计应该尽可能减小类与类之间耦合，所遵循的经验法则是：尽量限制成员函数的可见性。如果成员函数没必要公有 (public)，就定义为保护 (protected)；没必要保护 (protected)，就定义为私有 (private)；方法定义语法规范：

[可见性]['abstract'] [‘static’] ['final'] ['synchronized'][返回值类型] method\_name(参数列表)[('throws')][异常列表]{

//方法体

}

声明顺序：构造方法、静态公共方法、静态私有方法、公共方法、友元方法、受保护方法、私有方法、main方法；方法参数建议顺序：(被操作者，操作内容，操作标志，其他)。

3、变量声明：一行一个声明；声明局部变量的同时初始化（在变量的初始值依赖于某些先前发生的计算的特殊情况下可以不用同时初始化）；只在代码块的开始处声明变量，（一个块是指任何被包含在大括号"{"和"}"中间的代码）不要在首次用到该变量时才声明；避免声明的局部变量覆盖上一级声明的变量，即不要在内部代码块中声明相同的变量名；公共和保护的可见性应当尽量避免，所有的字段都建议置为私有，由获取和设置成员函数（Getter、Setter）访问；定义一个变量或者常量的时候，不要包含包名（类似java.security.MessageDigest digest = null），除非是两个包有相同的类名；数组声明时应当将"[]"跟在类型后，而不是字段名后；声明顺序：常量、类变量、实例变量、公有字段、受保护字段、友元字段、私有字段。

**五、异常**

1、捕捉异常的目的是为了处理它

2、多个异常应分别捕捉并处理，避免使用一个单一的catch来处理。

**六、习惯**

1、if、for、do、while等语句的执行语句部分无论多少都要加括号"{}"

2、 每当一个case顺着往下执行时(因为没有break语句)，通常应在break语句的位置添加注释；

3、尽量避免在循环中构造和释放对象

4、在使用局部变量的过程，按就近原则处理。不允许定义一个局部变量，然后在很远的地方才使用；

5、相同的功能不允许复制成N份代码；

6、在处理 String 的时候要尽量使用 StringBuffer 类。

Java编码规范例子：

Cell类：

**package** puzzle;

**import** java.awt.Rectangle;

**import** javax.swing.Icon;

**import** javax.swing.JButton;

@SuppressWarnings("serial")

**public** **class** Cell **extends** JButton{

**private** **static** **int** *IMAGEWIDTH*;//设置按钮的宽度大小

**private** **static** **int** *IMAGEHEIGHT*;

**private** **int** ID = 0;//设置当前按钮的指向坐标

**public** Cell(Icon icon, **int** id, **int** imagewidth, **int** height)//构造函数初始化，传入两个参数，一个是图像的图标，一个是该按钮的数组ID

{

**this**.setIcon(icon);

**this**.ID = id;

**this**.*IMAGEWIDTH* = imagewidth;

**this**.*IMAGEHEIGHT* = height;

**this**.setSize(*IMAGEWIDTH*, *IMAGEHEIGHT*);

}

**public** **void** move(Direction dir)//移动

{

Rectangle rec = **this**.getBounds();//获取当前对象的这个边框

**switch**(dir)

{

**case** *UP*://向上移动，改变坐标

**this**.setLocation(rec.x, rec.y + *IMAGEHEIGHT*);

**break**;

**case** *DOWN*://向下移动

**this**.setLocation(rec.x, rec.y - *IMAGEHEIGHT*);

**break**;

**case** *LEFT*://向左移动

**this**.setLocation(rec.x - *IMAGEWIDTH*, rec.y);

**break**;

**case** *RIGHT*://向右移动

**this**.setLocation(rec.x + *IMAGEWIDTH*, rec.y);

**break**;

}

}

**public** **int** getID() {

**return** ID;

}

**public** **int** getX()

{

**return** **this**.getBounds().x;

}

**public** **int** getY()

{

**return** **this**.getBounds().y;

}

}

Direction类：

**package** puzzle;

**public** **enum** Direction {

*UP*,

*DOWN*,

*LEFT*,

*RIGHT*

}

GameFrame类：

**package** puzzle;

**import** java.awt.BorderLayout;

**import** java.awt.Container;

**import** java.awt.EventQueue;

**import** java.awt.FlowLayout;

**import** java.awt.Font;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPanel;

**import** java.util.Timer;

**import** java.util.TimerTask;

**public** **class** GameFrame **extends** JFrame {

**public** JPanel pane1 = **new** JPanel();

**public** JButton button1 = **new** JButton("游戏开始");

**public** JButton button2 = **new** JButton("游戏结束");

**public** GameFrame()

{

**super**("拼图游戏");

pane1.setLayout(**new** FlowLayout());

pane1.add(button1);

pane1.add(button2);

Container con = **this**.getContentPane();

con.add(pane1,BorderLayout.*NORTH*);

**final** GamePanel gamepane = **new** GamePanel();

con.add(gamepane,BorderLayout.*CENTER*);

**this**.setBounds(300, 300, 620, 620);

**this**.setVisible(**true**);

**this**.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

button1.addActionListener(**new** ActionListener()

{

**public** **void** actionPerformed(**final** ActionEvent e)

{

gamepane.OutOfOrder();

}

});

button2.addActionListener(**new** ActionListener()

{

**public** **void** actionPerformed(**final** ActionEvent e)

{

System.*exit*(1);

}

});

}

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

**new** GameFrame();

}

}

GamePanel类：

**package** puzzle;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseListener;

**import** java.awt.image.BufferedImage;

**import** java.io.File;

**import** java.util.Random;

**import** javax.imageio.ImageIO;

**import** javax.swing.ImageIcon;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

@SuppressWarnings("serial")

**public** **class** GamePanel **extends** JPanel **implements** MouseListener{

**private** Cell BlankCell = **null**;

**private** **int** row = 4;

**private** **int** col = 4;//设置这个拼图的行列

**private** Cell cells[] = **new** Cell[row\*col];//创建一个按钮对象数组

**int** ImageWidth;

**int** ImageHeight;

**public** GamePanel()//构造函数

{

**this**.setLayout(**null**);

init();

}

**public** **void** init()//初始化完成以下功能--完成图像的切割，完成图像到图标的转换，完成按钮图标的添加，将按钮添加到面板上，并且给每一个按钮添加监听机制

{

**int** num = 0;

BufferedImage buf = **null**;

BufferedImage bufnew = **null**;

ImageIcon icon = **null**;

**int** width = 0;

**int** height = 0;

**try**

{

buf = ImageIO.*read*(**new** File("d:/java/smith/BigWork/images/图片2.jpg"));//读取文件图像

ImageWidth = buf.getWidth();

ImageHeight = buf.getHeight();

System.*out*.println("ImageWidth->"+ImageWidth+"ImageHeight->"+ImageHeight);

width = ImageWidth/col;

height = ImageHeight/row;

}**catch**(Exception e)

{

System.*out*.println(e);

}

**for**(**int** i = 0; i < row; i++)

{

**for**(**int** j = 0; j < col; j++)

{

num = i\*col+j;//表示当前这个图像的坐标id，在数组中的下标

**if**(num < row\*col-1)

{

bufnew = buf.getSubimage(width\*j, height\*i, width, height);

icon = **new** ImageIcon(bufnew);//将图像转化成图标

}

**else**//使最后一张图像为空白图像

{

icon = **new** ImageIcon("images/空白.png");//一张空白图像

}

cells[num] = **new** Cell(icon, num, width, height);//添加图标到每一个BUTTON按钮上面

cells[num].setLocation(width\*j, height\*i);

}

}

BlankCell = cells[cells.length-1];//初始化空白格

**for**(**int** i = 0; i < cells.length; i++)

{

**this**.add(cells[i]);//将每一个按钮添加到当前这个面板上面

**if**(i < cells.length-1)

cells[i].addMouseListener(**this**);//空白格不添加监听机制

}

}

**public** **void** OutOfOrder()//乱序----打乱图片的排布顺序

{

Random random = **new** Random();

**for**(**int** i = 0 ; i < cells.length ; i++)

{

**int** index1 = random.nextInt(cells.length);//cells的长度是9，但是他的上限是9，取不到9，所取值范围是0-8

**int** index2 = random.nextInt(cells.length);

**int** x = cells[index1].getX();

**int** y = cells[index1].getY();//获取下标是index1的数组元素按钮的坐标

cells[index1].setLocation(cells[index2].getX(), cells[index2].getY());

cells[index2].setLocation(x, y);

}

}

**public** **boolean** IsWin()//判断游戏玩家是否赢

{

**for**(**int** i = 0; i < cells.length; i++)

{

**int** x = cells[i].getX();

**int** y = cells[i].getY();

**if**(x/(ImageWidth/col) + y/(ImageHeight/row) != i)

{

**return** **false**;

}

}

**return** **true**;

}

**public** **void** mouseClicked(MouseEvent e)

{

Cell t = (Cell) e.getSource();

**int** x = BlankCell.getX();

**int** y = BlankCell.getY();

**if**(t.getY() == y && t.getX() + ImageWidth/col == x)//图像向右走

{

t.move(Direction.*RIGHT*);

BlankCell.move(Direction.*LEFT*);

}

**else** **if**(t.getY() == y && t.getX() - ImageWidth/col == x)//图像向左走

{

t.move(Direction.*LEFT*);

BlankCell.move(Direction.*RIGHT*);

}

**else** **if**(t.getX() == x && t.getY() + ImageHeight/row == y)//图像向上走

{

t.move(Direction.*UP*);

BlankCell.move(Direction.*DOWN*);

}

**else** **if**(t.getX() == x && t.getY() - ImageHeight/row == y)//图像向下走

{

t.move(Direction.*DOWN*);

BlankCell.move(Direction.*UP*);

}

**if**(IsWin())

{

**int** choice = JOptionPane.*showConfirmDialog*(**null**, "恭喜您过关了是否还来一局？", "提示", JOptionPane.*YES\_NO\_OPTION*);

**if**(choice == 0)//表示再来一局

{

**this**.OutOfOrder();

}

**else**//表示退出游戏

System.*exit*(1);

}

}

//@Override

**public** **void** mousePressed(MouseEvent e) {

}

//@Override

**public** **void** mouseReleased(MouseEvent e) {

}

//@Override

**public** **void** mouseEntered(MouseEvent e) {

}

//@Override

**public** **void** mouseExited(MouseEvent e) {

}

}