GOLauncher项目编码规范

|  |  |  |  |
| --- | --- | --- | --- |
| 版本修订记录 | | | |
| 日期 | 版本 | 修订人 | 修改内容 |
| 2011-11-28 | 1.00 | 骆培焕 | 初稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 命名规范

1. 包（package）名全部使用小写，单词间使用“.”分开。
2. 类（Class）名使用混合大小写的形式，每个单词的首字母大写，而其他字母均为小写。使用完整的单词，除非是规范的单词缩写。

|  |
| --- |
| class MessageManager  class DataProvider |

1. 接口（Interface）可以有以下2种命名方式：
2. 首字母用I来代替，其他每个单词首字母都要大写。

|  |
| --- |
| interface INetOperator |

1. OnXXXListner，XXX部分表示事件的名称，例如onClickListener.
2. 方法（Method）名使用动词或是惯用短语描述，第一个单词首字母小写，其他每个单词首字母都要大写。

|  |
| --- |
| public void run()  public int getContainerId() |

1. 变量的命名：
2. 所有非public、非静态的成员变量以m开头。
3. 静态成员变量使用s开头。
4. 局部变量第一个单词首字母小写，其后的单词首字母大写。
5. 静态的final变量，整个单词都要大写，单词之间用下划线“\_”分开。

|  |
| --- |
| public class MyClass {  public static final int SOME\_CONSTANT = 42;  public int publicField;  private static MyClass sSingleton;  int mPackagePrivate;  private int mPrivate;  protected int mProtected;  } |

1. Exception的子类以“Exception”为类名的结尾

|  |
| --- |
| class AccessException extends Exception  {  :  } |

1. 常见的缩写词作为变量、方法和类名时应当被作为单词进行大小写：

| **Good** | **Bad** |
| --- | --- |
| XmlHttpRequest | XMLHTTPRequest |
| getCustomerId | getCustomerID |
| class Html | class HTML |
| String url | String URL |
| long id | long ID |

1. 类的方法名不应当包含该类的名称

|  |
| --- |
| class Line{  public int getLength(){ *// NOT: getLineLength()*  //…  }  } |

1. 布尔类型的变量避免使用**否定**形式命名

|  |
| --- |
| boolean isError; *// NOT: isNoError*  boolean isFound; *// NOT: isNotFound* |

1. 单例类应该通过方法getInstance返回唯一的单例

|  |
| --- |
| class UnitManager  {  private final static UnitManager instance\_ = new UnitManager();  private UnitManager()  {  ...  }  public static UnitManager getInstance() *// NOT: get() or instance() or unitManager() etc.*  {  return instance\_;  }  } |

# 2.声明（Statements）

1. 类或接口的开头使用JavaDoc说明开发人员、功能说明和重要的更改记录
2. 着重补充点

|  |
| --- |
| /\*\*  \* 视图管理器,用于排版以及显示  \*  \* **@author** luopeihuan  \* **@version** 1.0  \*/  **public** **class** ViewManager{  } |

1. 在Eclipse中使用Ctrl+Shift+O添加import声明，消除警告去除不需要的import，不要使用xxx.\*;的导入形式
2. 类和接口使用以下顺序进行声明：
   1. 类/接口文档（JavaDoc说明）
   2. 类/接口定义
   3. 类静态变量，按照public, protected, package (默认权限), private的顺序声明
   4. 类成员变量，按照public, protected, package (默认权限), private的顺序声明
   5. 构造函数
   6. 方法
3. 方法/变量的修饰词使用以下的顺序：

**<access> static abstract synchronized <unusual> final native**

<access>指访问权限：public\protected\private\package(默认)

<unusual> 包括 volatile 和transient.

|  |
| --- |
| public static double square(double a); *// NOT: static public double square(double a);* |

1. 变量必须在声明的时候初始化，并且尽量在最小的作用范围内声明。
2. 每行声明一个变量，不要在一行代码内同时声明多个变量。

|  |
| --- |
| int width;  int height; *// NOT: int width, height;* |

1. 数组的中括号应靠近类型

|  |
| --- |
| double[] vertex; *// NOT: double vertex[];*  int[] count; *// NOT: int count[];*  public static void main(String[] arguments)  public double[] computeVertex() |

1. 不要在子类中定义与父类同名的变量

# 3.语句格式

1. if, if-else, if-else-if-else 语句

|  |
| --- |
| // only if  if (condition) {      statements;  }  // if-else  if (condition) {      statements;  }  else {      statements;  }  // if-else-if-else  if (condition) {      statements;  }  else if (condition) {      statements;  }  else if (condition) {      statements;  } |

1. for 语句

|  |
| --- |
| for (initialization; condition; update) {      statements;  } |

1. while语句

|  |
| --- |
| while (condition) {      statements;  } |

1. do-while语句

|  |
| --- |
| do {      statements;  }  while (condition); |

1. switch语句，必须有default分支

|  |
| --- |
| switch (condition) {  case ABC:  statements;  break;  case DEF:      statements;      break;  case XYZ:      statements;      break;  default:      statements;      break;  } |

1. try-catch语句

|  |
| --- |
| try {      statements;  }  catch (ExceptionClass e) {      statements;  }  finally {      statements;  } |

注意：在finally里面关闭io、文件、cursor等。

**\*重要：**

1. 以上的if，while和for循环语句块内的代码（不管多少）必须包含在花括号{}内；
2. 在switch语句块中，一定要有default分支（即使无默认行为）；
3. Object的wait()方法必须放入while循环。在被唤醒后再次判断条件，不满足则重新进入wait状态。

# 4.空格/空行的使用

1. 所有的二元运算符，除了"."，应该使用空格将之与操作数分开，一元操作符负号("-")、自增("++")和自减("--")与操作数之间不加空格。
2. Java的保留字（reserved words也称关键字）后紧跟一个空格。
3. 使用单行注释 // 或者 块注释/\*\*/，说明文字和注释标识符//或者\*间要有一个空格

|  |
| --- |
| a = (b + c) \* d; *// NOT: a=(b+c)\*d*  while (true) { *// NOT: while(true){*  ...  doSomething(a, b, c, d); *// NOT: doSomething(a,b,c,d);*  for (i = 0; i < 10; i++) { *// NOT: for(i=0;i<10;i++){*  *...* |

1. 方法（Methods）间使用空行隔开
2. 单行注释和块注释之前使用空行，例外情况：注释上一行为左花括号{。
3. 方法内的两个逻辑段之间，使用空行
4. 着重补充点
5. 隔开，以提高可阅读性。

|  |
| --- |
| private void initial(){  *// Create a new identity matrix*  Matrix4x4 matrix = new Matrix4x4();  *// Precompute angles for efficiency*  double cosAngle = Math.cos(angle);  double sinAngle = Math.sin(angle);  }  private void doSomething(){  } |

# 5.内存管理

1. 要尽早的释放无用对象的引用。如果该对象不用了，可以把它设置为null。需要**注意**其他异步（Hanlder、Thread等方式）使用到该对象的情况，避免空指针。
2. 避免使用finalize函数。因为它会加大GC的工作量，因此尽量少用finalize方式回收资源。
3. 如果需要使用经常用到的图片，可以使用SoftReference类型，它可以尽可能将图片保存在内存中，供程序调用，而不引起OutOfMemory。
4. 注意集合数据类型，包括数组、树、图、链表等数据结构，这些数据结构对于GC来说，回收更为复杂。另外，要注意那些全局变量，静态变量，这些对象往往容易引起悬挂对象，造成内存浪费。
5. 尽量避免在类的默认构造器中创建、初始化大量的对象，防止在调用其子类的构造器时造成不必要的内存资源浪费。
6. 尽量避免强制系统做垃圾内存回收（通过显式调用方法System.gc())，增长系统做垃圾回收的最终时间，降低系统性能。
7. 尽量避免显式申请数组空间，当不得不显式申请数组空间时尽量准确的估计出其合理值，以免造成不必要的系统内存开销。
8. 尽量避免在Activity里面创建非静态内部类，尤其是你无法控制内部类生命周期的时候。使用静态内部类，或者使用WeakReference引用外部类代替。因为内部类的生命周期一旦超过Activity的生命周期，Activity就会因为没有及时回收导致内存泄漏。
9. 尽量避免在View/ViewGroup的onLayout()、onMeasure()、onSizeChanged ()、onTouchEvent()和dispatchDraw()方法里面创建对象。
10. 尽量避免在setOnXXXListener里面直接创建内部类的方式处理事件，应该使用继承onXXXListener的方式。例如在Activity里面处理View的click事件，Activity继承onClickListener，然后调用view的setOnClickListener设置观察者为Activity。
11. 尽量使用context-application 代替context-activity，需要注意对话框、view等需要使用context-activity。
12. 在getView中合理使用convertView提高效率，避免重复创建view。

# 6.其他

1. 严禁使用System.out.println() (或者在native code 里面使用printf() )， System.out 和 System.err 会将输出重定向到/dev/null，打印的内容实际上是看不到的，打印信息使用Log类。
2. 没有特殊理由尽量避免使用反射调用方法。反射调用的方法经过混淆打包后可会导致调用失败。
3. 避免使用魔法数字（magic number），非0和1的数字应该定义成一个常量，在其他地方引用。

|  |
| --- |
| private static final int TEAM\_SIZE = 11;  Player[] players = new Player[TEAM\_SIZE]; *// NOT: Player[] players = new Player[11];* |

1. 编辑代码时请与周围的代码风格保持统一。

# 要点补充：

1.变量命名：

所有非public、非静态的成员变量以m开头

静态成员变量使用s开头

局部变量第一个单词首字母小写，其后的单词首字母大写

静态的final变量，整个单词都要大写，单词之间用下划线“\_”分开

2.注释：

新增类必须有作者注释；

新增接口/公开方法，必须有参数注释；

新增魔数定义，必须有含义注释；

新增工具方法，必须有适用范围注释；

1. log输出：

不要使用中文；

不要在频繁打印输出；

不要将链接地址打出；

1. 统一定义：

LauncherEnv.java：公共常量定义类，路径地址/包名/服务器地址；

Machine.java：硬件设备相关类，网络状态/版本号/设备状态/区域判断；

AppUtils.java：应用相关类，安装包判断/杀进程/发通知/下载/跳转；