**C# 代码规范**

1.规范目的：

（1.1）增强代码可维护性。代码的编写不是一次性就能写得很完美的，需要不断的修复bug，修改或增加功能，重新设计整体架构等。这时就需要进入代码中去做修改，如果没有良好的代码规范，时间久了自己阅读起来就很费力。

（1.2）提高团队开发效率。大多数项目的代码都不是由一个人编写的，其他成员也许会因为项目的交接需要接手管理你所编写的代码，如果没有良好的代码规范，他人便无法快速轻松的理解你的代码。

（1.3）提高个人编码效率。最开始可能会觉得被规范约束，后来反而会因为有规范而有依靠感，不必再为起什么名字而犹豫。

2.规范说明：

（2.1）代码的规范不是绝对的，每个公司都会有自己的一套代码规范，每个新人刚进入都要最先学习代码规范，才能加入到团队的开发中来。

3.命名规范：

（3.1）基本规则（对于类、字段、方法、参数、变量）：

【规则1-1】英文单词命名。禁止使用拼音或无意义的字母命名。

【规则1-2】直观易懂。使用能够描述其功能或意义的英文单词或词组。

【规则1-3】除了常量的命名以外，不要采用下划线命名法。

//命名表示 汽车类型 的字段：

int carType //正确：英文词组、能描述其意义。

int qicheleixing //错误：拼音。

int akhj //错误：无意义。

int car //错误：不能描述其意义。

int car\_type //错误：下划线命名。

【规则1-4】约定俗成的变量可以 依次 简单命名。

//命名循环里的递增（减）变量：

int i,j,k;

【规则1-5】命名空间名、类名、接口名、非私有字段名、枚举名及枚举值名、方法名采用 大驼峰式命名法 ，即每一个单词的首字母都大写。

namespace Car {} //命名空间名

public class SendManager; //类名

public interface IState; //接口名

public string FirstName; //非私有字段名

enum CarDriveType {FrontWheelDrive, RearWheelDrive, FourWheelDrive}//枚举名及枚举值名

public void SendMessage(string message) {} //方法名

【规则1-6】私有字段名、方法参数名、局部变量名采用 小驼峰式命名法 ，即第一个单词首字母小写，其余单词首字母大写。

private string mFirstName; //私有字段名

public void FindByFirstName(string firstName) {} //方法参数名

string firstName; //局部变量名

（3.2）特殊规则：

【规则2-1】接口的命名要以“I”开头。

public interface IState;

【规则2-2】类的私有变量命名以“m”开头。

private string mFirstName;

【规则2-3】常量的命名单词之间用“\_”隔开，所有字母均大写。

private const float MAX\_SPEED = 180.0f;

【规则2-4】若方法返回一个成员变量的值，则方法名一般为Get + 成员变量名。

public string GetFirstName()

{

}

【规则2-5】若方法修改一个成员变量的值，则方法名一般为Set + 成员变量名。

public void SetFirstName(string firstName)

{

}

【规则2-6】若方法返回的值是bool变量，则方法名一般根据含义以Is/Can/Has/Try作为前缀。

public bool IsEmpty()

{

}

public bool CanEmpty()

{

}

【规则2-7】避免方法内的局部变量与类的字段命名相同。

4.编码规范：

【规则3-1】声明变量时，一行只声明一个变量。

private string mFirstName;

private string mLastName;

【规则3-2】声明变量时，依次简单变量允许一行声明多个。

int i,j,k;

【规则3-3】局部变量声明时，采用就近原则，即将使用该变量时再声明。

【规则3-4】“｛”与“｝”的位置如下所示，均独占一行。

public void Example()

{

int i = 5;

int j = 6;

i += j;

int k = 3;

k = k \* 2;

}

【规则3-5】 两个方法之间使用一个空行。

public void ExampleA()

{

}

public void ExampleB()

{

}

【规则3-6】 类的字段声明统一放置于类的最前端，并且最后一个声明与类的第一个方法之间使用一个空行。

public class Student

{

private string mFirstName;

private string mLastName;

public string GetFirstName()

{

}

}

【规则3-7】方法参数列表中的“，”后面使用一个空格。

public void SetName(string firstName, string lastName)

{

}

【规则3-8】二元操作符，除了“.”在其两边各使用一个空格。

a = c + d;

【规则3-9】for语句表达式子中“；”后面使用一个空格。

for(i = 0; i <= 3; i++)

{

}

【规则3-10】强制转换后面使用一个空格。

char c;

int a = 4;

c = (char) a;

【规则3-11】如果if、for、while语句内容只有一行，可以不加“｛｝”，但是必须和if、for、while语句位于同一行。

if(i < 5) i = 5;

【规则3-12】一行代码长度不要超过屏幕宽度。如果超过了，将超过部分换行。特殊情况下不强求，如定义了海量内容的常量数组（比如所有敏感词）或上下列整齐的公式。

5.注释规范：

【规则4-1】代码头部注释，包含以下内容:

文件名称：文件的名称。

功能描述：文件的功能描述与大概流程说明。

作者：创建并编写的人员。

日期：创建并编写的日期。

修改记录：若类有所修改，则需要有修改人员的名字、修改日期及修改理由。

// 文件名称：UserInput.cs

// 功能描述：玩家输入按键的定义

// 编写作者：张三

// 编写日期：2017.7.16

// 修改记录：

// R1:

// 修改作者：李四

// 修改日期：2017.7.17

// 修改理由：使玩家可以自定义输入按键

Using System;

【规则4-2】方法注释，采用 /// 形式自动产生XML标签格式的注释。包括方法功能，参数含义，返回内容。

/// <summary>

/// 设置场景的名字.

/// </summary>

/// <returns><c>true</c>, 场景名字设置成功, <c>false</c> 场景名字设置失败.</returns>

/// <param name="sceneName">场景名字.</param>

public bool SetSceneName(string sceneName)

{

}

【规则4-3】类变量注释，采用 /// 形式自动产生XML标签格式的注释变量含义。

/// <summary>

/// 场景的名字

/// </summary>

private string mSceneName;

【规则4-4】局部变量注释，在变量声明语句的后面注释，与前后行变量声明的注释左对齐，注释与代码间以Tab隔开。

string firstName; //姓

string lastName; //名

【规则4-5】代码行注释，注释位于代码上行，与代码开始处左对齐，双斜线与注释之间以空格分开。

//设置场景的名字。

this.mSceneName = sceneName;