iOS 开发规范

1. 命名规范

a) 类的命名

视图控制器命名尽量保持“xxxVC”的统一格式，以标准英文含义命名；

一些基类以“Basexxx”格式命名；

自定义的 view 类与所在的控制器类名尽量保持一致；

b) 常量 / 变量的命名

变量命名遵循驼峰标示法则，以小写字母开头，尽量以下划线“_”开头，不要与 OC 中的特殊字符冲突；

常量的命名统一化以小写 k 命名，同变量一样遵循驼峰标示；

c) 方法的命名

方法的命名以包含动词和名次的英文含义为标准，各页面之间相同功能的方法名保持统一，eg:

初始化页面- (void)initView;

加载网络数据- (void)loadData;

d) 宏定义命名

以大写字母表示，多个词时以下划线连接;

2. 目录结构

目录结构采取 MVC 或 MVVM 的架构模式保持清晰;

一级目录以标签栏子标签设置目录，在一级目录下设置 Base 文件夹包括各种基类，设置 Tool 文件夹包含各种自定义工具类;

二级目录以每个子标签下控制的所有功能模块设置目录;

三级目录是每个功能模块下设置 Model、View、ViewController;

3. 注释

a) 对一些必要变量进行注释

b) 对一些方法进行”#pragma mark—<方法说明>“标识，或者用多行注释标识出方法的作用以及各形参的含义,eg:

```
#pragma mark - <ResponseDelegate>
```

c) 一些可以优化的地方用“#warning—备注”标识，eg:

```
#warning text----测试
```

d) 对每一个视图控制器类，在.h 中固定位置用中文标示出代表的是哪个界面，方便查看和维护，eg:

```
/// 登录 Api
@interface LoginApi : HKBaseApi
```

e) 一些常用的工具类文件，导入.pch 文件

4. 版本控制

使用 CocoaPods 做为版本控制工具，由固定的一人管理第三方的下载或更新。

5. 代码规范

a) 局部变量定义与实现时，严格按照以下格式：

```
NSMutableArray *itemListArray = [NSMutableArray arrayWithArray:itemArray];
```

b) 全局变量声明时，严格按照以下格式：

```
NSString *_minRentPrice;
```

c) 属性的声明时，严格按照以下格式：

```
@property (nonatomic,copy)NSString *customerKeyId;
```

d) 代理的声明

错误实例：

```
@property (nonatomic,strong)id<ChannelFilterDelegate> delegate;
```

正确格式：

```
@property (nonatomic,assign)id <ChannelFilterDelegate>delegate;
```

e) 方法的声明或实现时，形参和方法体严格按照以下格式：

```
- (NSArray *)removeObject:(NSArray *)itemArray;
```

特别注意：空格！空格！还是空格！

6. 代码提交

协同开发过程中，不要频繁提交代码，也不要长时间不提交代码，在写完一个独立的功能块或者是与你的队友的工作紧密关联的功能后，进行提交，每次 commit 代码之前，保险起见，都要先 update 一下。