“一叶账目”小组规程

一叶账目小组

一、大纲

第0条 这是一个高逼格的规程文件。不许笑。笑什么笑。认真读。听到没。嗯。

第一条 本规程为“一叶账目”小组内部使用规程，其作用为保证团队开发“一叶账目”软件过程中的严谨性，作用时间为“一叶账目”软件生命周期内。

第二条 在没有其余组员同意的情况下，组内每位成员必须遵守该规程所有的约定。由于个人违反规程造成的损失，将由个人全部承担。

第三条 本规程所有解释权归一叶账目小组所有。

二、软件简介

第四条 软件中文名称“一叶账目”，英文名称“OneBill”，适用于 Andriod 平台，最低 Android 系统版本要求为 Android 4.0。

第五条 软件开发采用 Android Studio 平台进行开发。

第六条 版本控制采用 Git 分布式版本控制系统，使用 GitHub 作为远程端。

三、开发环境搭建

第七条 小组成员需完成 Android Studio 的安装与 JDK 的下载。

四、版本控制环境搭建

第八条 小组成员应按以下步骤完成版本控制环境的搭建。

1、注册 GitHub 账号，将用户名与邮箱保存并进行组内整合。

2、下载 msysgit（网址 http://msysgit.github.io/），默认选项安装，或者下载 GitHub For Windows（或 Mac）（网址 https://desktop.github.com/）并安装。二者均有命令行及可视化两种方式。以下环境搭建内容针对 msysgit 编写。

3、安装完成后，找到安装目录下 Git->Git Bash，键入以下内容设置用户名和邮箱：

```
$ git config --global user.name “用户名”
$ git config --global user.email “邮箱”
```

4、键入以下内容，一路回车，生成 SSH Key：

```
$ ssh-keygen -t rsa -C "邮箱"
```

5、在电脑的“用户->你的用户名->.ssh”目录下，复制“id_rsa.pub”文件的内容。

6、登录 GitHub，在设置->SSH Keys 界面下，输入任意 title，在 Key 文本框内粘贴内容，

Add Key，远程仓库设置完成。

7、找一个建立工程的位置。路径要求不含中文字符、且路径名中不包含空格。

8、在该处右键，选择“Git Bash Here”，键入命令以下命令将远程仓库拷贝至本机：

```
$ git clone git@github.com:ghh3809/OneBill.git
```

请注意，该命令只将 master 分支拷贝至本机。还需要拷贝 dev 分支：

```
$ git checkout -b dev origin/dev
```

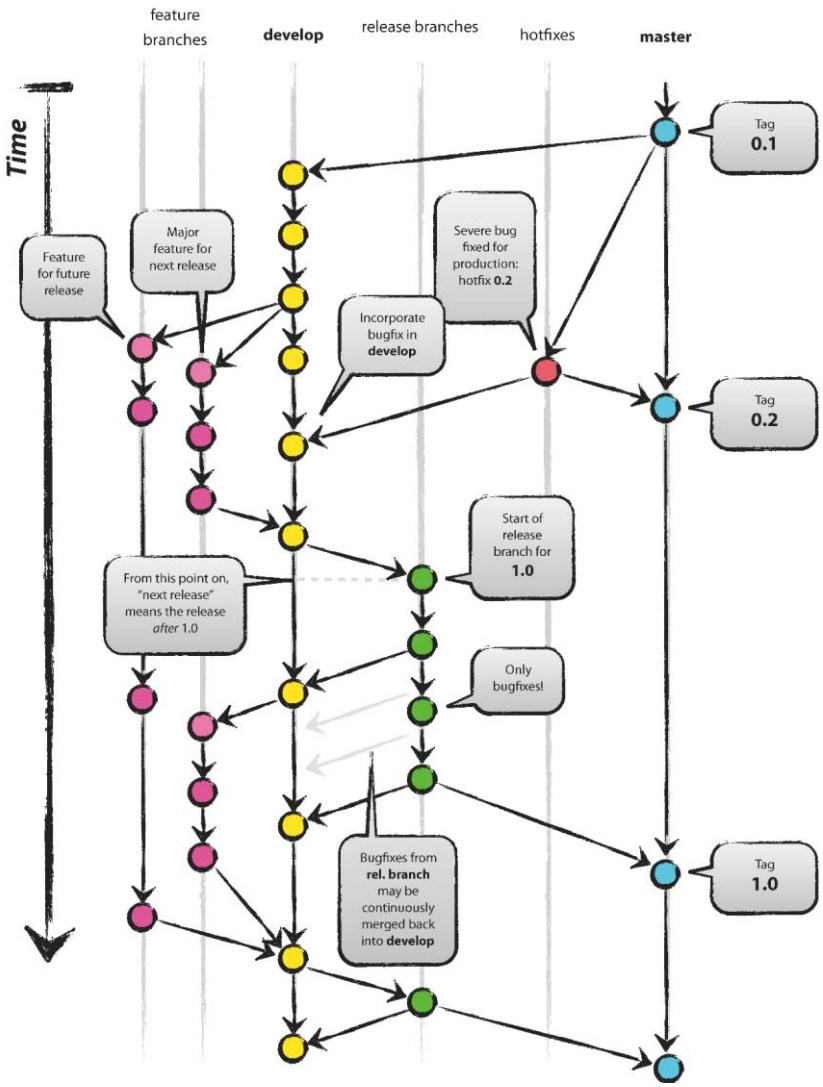
此时 dev 分支拷贝完成，并进入 dev 分支。

9、用 Android Studio 打开该工程，提示检测到 Git 使用。选择 Configure，再添加当前路径，此时利用 Android Studio 更改文件时即可发现变化。

10、如果使用 GitHub For Windows，同样可以把工程在其中打开。至此，版本控制环境搭建完毕。

五、版本控制约定

第九条 本工程拟采用以下版本控制模型，其具体说明详见以下规程。



第十条 对于 master 分支，应遵守如下约定：

含义：主分支，正式发布的分支

周期：永久

来源：只能通过 release 分支或 hotfixes 分支合并至 master 分支得到

要求：①在任何情况下，不得对 master 分支作任何非发布修改，而必须通过合并得到。

②在任何情况下，不得删除 master 分支。

③对每一 master 分支上的 commit，必须以标签注明其版本号，形式为“Vx.x”或“Vx.x.x”。

④只有 master 分支上的 commit 可以正式发布至市场。

⑤每一个 master 分支上的 commit 必须有经过小组商讨后的注释。

第十一条 对于 dev 分支，应遵守如下约定：

含义：开发分支，小组共同协作的分支

周期：永久

来源：通过 master 分支拷贝或 feature 分支及 hoxfixes 分支合并得到，或对 dev 分支之前的 commit 的更改。

要求：①在任何情况下，不得将 dev 分支直接合并至 master 分支。

②对 master 分支的 commit 进行开发时，必须先拷贝到 dev 分支，再允许在 dev 分支上进行工作。

③对 hotfixes 分支的 bug 进行修改后，需要与 dev 分支的 commit 进行合并。

④对 dev 分支的直接 commit 修改只限 bug 调整。对于任何需要添加的新特性，必须新拷贝至 feature 分支，再在开发完毕后合并至 dev 分支。

⑤不允许为 dev 分支的 commit 打上版本标签，允许设置本地其他标签，但不能上传至远程端。

⑥每一个 dev 分支的 commit 必须有对应格式的注释，方便起见，具体规定如下：

第一部分（若有）：merge from feature-xx 或者 merge from hotfixes-xx

第二部分（必须）：简要描述所做的更改

第三部分（必须）：by 姓名

各部分之间使用||分割，以下为一个实例：

“merge from feature-graphing||add graph section||by ghh3809”

⑦只允许存在统一的 dev 分支，不得自行设定 dev 分支。

第十二条 对于 feature 分支，应遵守如下约定：

含义：开发新特性需要的分支

周期：某一新特性开发的时间段

来源：从 dev 分支拷贝或用户自行开发

命名：所有 feature 分支的命名模式为：feature-xx 特性

要求：①feature 分支开发完成，只能合并至 dev 分支。

②feature 分支属有生命周期的分支，特性开发完毕合并至 dev 分支后，应及时删除该分支。

③若为集体开发同一个 feature，可上传至远程仓库，开发完后从远程仓库删除，否则不应上传至远程仓库。

第十三条 对于 release 分支，应遵守如下约定：

含义：即将发布版本的分支

周期：即将发布某版本前期至正式发布

来源：只能从 dev 分支合并或 release 之前 commit 的 bug 修改

命名：所有 release 分支的命名模式为：release-x.x

要求：①当 dev 分支的 commit 趋于稳定，即可创建并合并至 release 分支。

②release 分支的创建要在小组全体人员同意下完成。

③对 release 分支所做的修改，只能是对 bug 的修复，不允许添加新特性，否则应拷贝至 dev 分支，再创建新的 feature 分支进行开发。

④对 release 分支所作的修改必须及时上传远程仓库。

⑤每一个版本号只能对应单独的 release 分支，当该版本合并至 master 分支后，该分支应删除。

⑥release 分支上的 commit 不应标注版本标签。

⑦release 分支的 commit 应注明是否进行了调整以及是否适合发布。

第十四条 对于 hotfixes 分支，应遵守如下约定：

含义：正式版本 bug 修复分支

周期：bug 修复阶段

来源：只能从 master 分支拷贝得到

命名：所有 hotfixes 分支的命名模式为：hotfixes-某 bug

要求：①只允许临时创建，只允许在发布版出现 bug 时创建。

②必须放下手头工作处理 bug 的问题，尽快合并回 master 分支。

③合并回 master 分支后，需要重新打版本标签。

第十五条 总而言之，所有的开发工作在 dev 分支进行，开发新特性使用 feature 分支，开发稳定后进入 release 分支，最后合并如 master 分支进行发布。坚决杜绝在 master 分支进行直接修改。

第十六条 各小组成员务必遵循以上控制模型的要求，否则一切后果均由个人承担。

一叶账目小组

2015 年 10 月 29 日