**建立分支**

1.主线名为master  
2.分支名为branch1;分支是在主线建议

切换到分支

1. android studio中下载checkout主线版本
2. 默认然后点击 保持master为最新
3. 切换到branch1分支

**修改分支**

1. 确保当前项目已经切换到了branch1上



1. 然后再分支上执行各种操作
2. Commit分支并且push到远程上

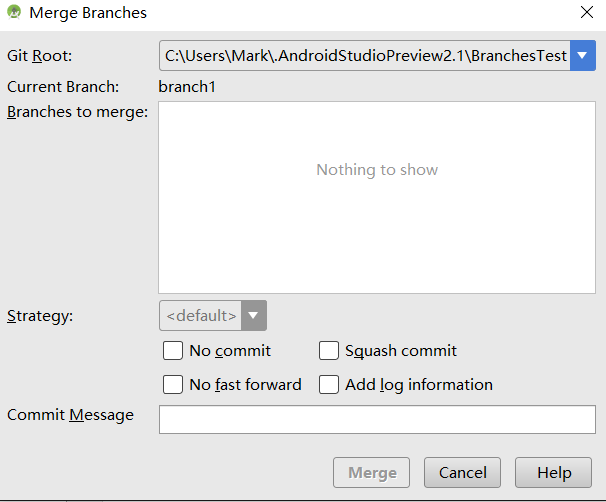
当前状态

一共有两个版本，1个是主线master一个是branch1.并且branch1经过修改版本要大于master. 本地也有两个版本一个master一个branch1这两个版本和远程的两个版本是同步一致的。

**合并分支**

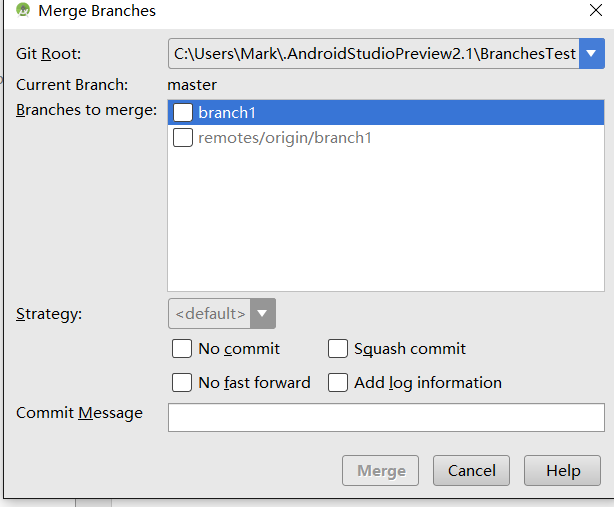
主线master的版本要小于branch1版本换句话说就是branch1的修改要放到master上。这样相当于master要包含branch1中的改变。用比较有歧义的词就是合并。把branch1的分支合并到主线master上。

如果我们不这么想。非要在分支上操作合并会什么效果呢?



分支上看不到任何可合操作的对象（分析是因为当前分支的版本是最高的。所以他不存在任何可融入的版本。也就是只要老的版本才需要和新的内容合并）

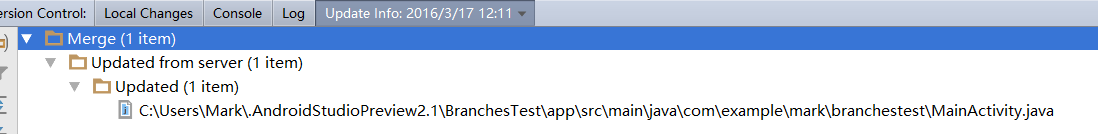
切换到主线master上去执行Merge Change..



主线上可以看到有2个branch1可以操作。到这一步已经证实了之前的推测。只有老版本才需要与新版本合并。并且必须先切换到老版本分支上

2个branch1的解释，因为git是分布式管理。所以任何可操作的版本都有他的实体版。因此以上步骤共生成了4个实体。远程上有master and branch1 本地也有master and branch1 所以远程的在这里的写法是remotes/origin/xxxx表示这两个是在服务器上

我们可以确认branch1和remotes/origin/branch1实际是一个版本。那么我可以随意选择一个了。.选择后点击”Merge“按钮



提示合并成功没有冲突。

这里需要注意的是当前操作的master合并了branch1都是在本地完成的。所以还要将本地的master和远程的master同步。

因此再在master上执行push 把合并的修改结果提交到服务器。这样。4个版本就都一致了。

命令行合并操作顺序

//checkout和修改branch1的步骤省略

git checkout master

git merge --no-ff branch1

git push origin master