说明：

该部分为对便签的**功能分析与建模**，还未做进一步细致地流程设计分析，且没有代码实现。如需了解流程设计分析和代码实现请移步**核心流程设计分析**与**高级设计意图**。

场景描述：

使用者在脑中想好需要记录的信息后，打开便签，输入需要记录的信息，保存，关闭。

第一步：正常处理(Normal)

【名称】

使用便签记录信息

【场景】

Who：使用者、便签；

Where：电脑附近；

When：便签打开后。

【描述】

1. 使用者想好需要记录什么，打开便签；
2. 便签询问使用者需要记录新信息，还是查看以往的信息，或是退出；
3. 使用者告诉便签；
4. 便签根据使用者的回复准备不同的页面；
   4.1 如果是记录新的信息，则新建一页提供给使用者；
   4.2 如果是查看以往的信息，则告诉使用者目前记录了哪些信息；
      4.2.1 便签再次询问需要哪一次的信息；
      4.2.2 使用者回复后；
      4.2.3 便签根据回复打开以往的信息给使用者；
   4.3 如果是退出，则退出；
5. 使用者对内容进行编辑；
6. 使用者告诉便签自己编辑好了；
7. 便签保存；
8. 告诉使用者自己保存好了；
9. 使用者告诉便签自己需要退出；
10. 退出便签。

【价值】

使用者能够记录自己可能遗忘的信息；开发者能够熟悉面向对象思想……

【约束和限制】

1.输入必须是中文或英文，以及键盘上存在的符号，因为这是我们常用的语言；

2.便签数目不得大于 999，因为这是一个便签，不是一个记事本，不能占用太多空间。

第二步：异常处理(Exception)

【描述】

1. 使用者想好需要记录什么，打开便签；
2. 便签询问使用者需要记录新信息，还是查看以往的信息，或是退出；
3. 使用者告诉便签；
4. 便签根据使用者的回复准备不同的页面；
   4.1 如果是记录新的信息，则新建一页提供给使用者；
   4.2 如果是查看以往的信息，则告诉使用者目前记录了哪些信息；
      4.2.1 便签再次询问需要哪一次的信息；
      4.2.2 使用者回复后；
      4.2.3 便签根据回复打开以往的信息给使用者；
         4.2.3.1 便签找不到使用者需要的信息，退回到 2；
   4.3 如果是退出，则退出；
5. 使用者对内容进行编辑；
6. 使用者告诉便签自己编辑好了；
7. 便签保存；
   7.1 如果没有进行编辑，则不需要保存；
8. 告诉使用者自己保存好了；
   8.1 如果没有编辑，则也不需要回复；
9. 使用者告诉便签自己需要退出；
10. 退出便签。

第三步：替代处理(Alternative)

【描述】

1. 使用者想好需要记录什么，打开便签；
2. 便签询问使用者需要记录新信息，还是查看、删除以往的信息，或是退出；
3. 使用者告诉便签；
4. 便签根据使用者的回复准备不同的页面；
   4.1 如果是记录新的信息，则新建一页提供给使用者；
   4.2 如果是查看以往的信息，则告诉使用者目前记录了哪些信息；
      4.2.1 便签再次询问需要哪一次的信息；
      4.2.2 使用者回复后；
      4.2.3 便签根据回复打开以往的信息给使用者；
         4.2.3.1 便签找不到使用者需要的信息，退回到 2；
   4.3 如果是需要删除，则告诉使用者目前的记录列表；
      4.3.1 便签再次询问需要删除哪一次的信息；
      4.3.1 使用者告诉便签需要删除的序号；
      4.3.2 便签将其删除；
      4.3.3 退回到 2；
         4.3.3.1 便签找不到需要删除的信息，也退回到 2
   4.4 如果是退出，则退出；
5. 使用者对内容进行编辑；

5.1 可以插入图片等非文本进行编辑；

6. 使用者告诉便签自己编辑好了；
7. 便签保存；

7.1 如果没有进行编辑，则不需要保存；

8. 告诉使用者自己保存好了；

8.1 如果没有编辑，则也不需要回复；

9. 使用者告诉便签自己需要退出；
10. 退出便签。

完整的需求分析：

【名称】

使用便签记录信息

【场景】

Who：使用者、便签；

Where：电脑附近；

When：便签打开后。

【描述】

1. 使用者想好需要记录什么，打开便签；
2. 便签询问使用者需要记录新信息，还是查看、删除以往的信息，或是退出；
3. 使用者告诉便签；
4. 便签根据使用者的回复准备不同的页面；

4.1 如果是记录新的信息，则新建一页提供给使用者；

4.2 如果是查看以往的信息，则告诉使用者目前记录了哪些信息；

4.2.1 便签再次询问需要哪一次的信息；

4.2.2 使用者回复后；

4.2.3 便签根据回复打开以往的信息给使用者；

4.2.3.1 便签找不到使用者需要的信息，退回到 2；

4.3 如果是需要删除，则告诉使用者目前的记录列表；

4.3.1 便签再次询问需要删除哪一次的信息；

4.3.1 使用者告诉便签需要删除的序号；

4.3.2 便签将其删除；

4.3.3 退回到 2；

4.3.3.1 便签找不到需要删除的信息，也退回到 2

4.4 如果是退出，则退出；

5. 使用者对内容进行编辑；

5.1 可以插入图片等非文本进行编辑；

6. 使用者告诉便签自己编辑好了；
7. 便签保存；

7.1 如果没有进行编辑，则不需要保存；

8. 告诉使用者自己保存好了；

8.1 如果没有编辑，则也不需要回复；

9. 使用者告诉便签自己需要退出；
10. 退出便签。

【价值】

使用者能够记录自己可能遗忘的信息；开发者能够熟悉面向对象思想……

【约束和限制】

1. 输入必须是中文或英文，以及键盘上存在的符号，因为这是我们常用的语言；

2. 便签数目不得大于 999，因为这是一个便签，不是一个记事本，不能占用太多空间。

功能提取——将分析描述中的动词挑选出来

【描述】

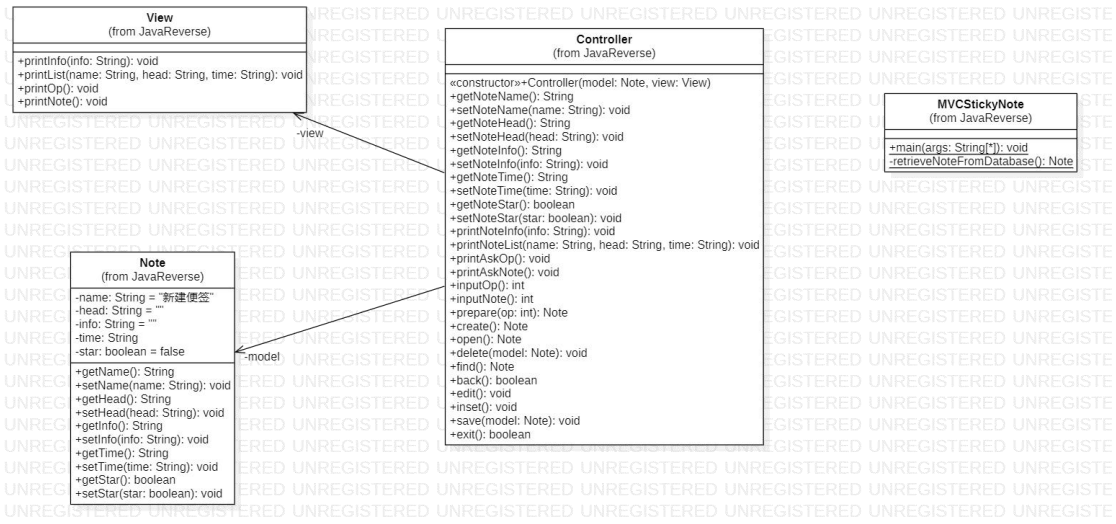
1. 使用者想好需要记录什么，打开便签；
2. 便签询问使用者需要记录新信息，还是查看、删除以往的信息，或是退出；
3. 使用者告诉便签；
4. 便签根据使用者的回复准备不同的页面；
   4.1 如果是记录新的信息，则新建一页提供给使用者；
   4.2 如果是查看以往的信息，则告诉使用者目前记录了哪些信息；
      4.2.1 便签再次询问需要哪一次的信息；
      4.2.2 使用者回复后；
      4.2.3 便签根据回复打开以往的信息给使用者；
         4.2.3.1 便签找不到使用者需要的信息，退回到 2；
   4.3 如果是需要删除，则告诉使用者目前的记录列表；
      4.3.1 便签再次询问需要删除哪一次的信息；
      4.3.1 使用者告诉便签需要删除的序号；
      4.3.2 便签将其删除；
      4.3.3 退回到 2；
         4.3.3.1 便签找不到需要删除的信息，也退回到 2
   4.4 如果是退出，则退出；
5. 使用者对内容进行编辑；
   5.1 可以插入图片等非文本进行编辑；
6. 使用者告诉便签自己编辑好了；
7. 便签保存；
   7.1 如果没有进行编辑，则不需要保存；
8. 告诉使用者自己保存好了；
   8.1 如果没有编辑，则也不需要回复；
9. 使用者告诉便签自己需要退出；
10. 退出便签。

功能提取——将挑选的动词整理合并，形成功能矩阵

功能编号	功能描述	备注
001	询问本次需要进行什么操作	即打印并接受输入，此处接受操作为创建新记录、查看以往记录、删除记录和退出
002	准备不同页面	根据使用者输入选择不同功能
003	新建	新建一张便签
004	打印目前记录的记录列表	打印目前已有的记录列表
005	再次询问	需要知晓对哪一次记录进行操作
006	打开	打开以往的记录
007	删除	删除以往的记录
008	查找	查找使用者需要的记录
009	退回	退回重新询问本次操作
010	编辑	根据使用者输入文本进行编辑
011	插入	插入其他非文本信息
012	保存	保存/不保存目前的记录，并打印相应信息
013	退出	退出便签

由该功能矩阵可以大致写出整个便签的框架，此处选择使用 MVC 模式来对便签进行分离设计，进一步体现面向对象高内聚、低耦合的特性。

在此仅仅给出一个简单的 UML 类图：



其工作流程大致如下：

首先读取本地已经存在的便签存于一个数组中，然后控制器控制视图输出询问信息，控制器在拿到用户输入后，根据不同输入给用户准备便签。用户可以对便签进行增删改查操作。而这一部分均靠使用 MVC 模式实现。在改动完成后，进行保存到本地，并重新读目前的数据，此时就可以进入下一轮操作。

UML 类图中的变量名以及方法名的作用和其名称几乎一模一样，都非常易于理解，且此处只是一个基本框架，在接下来可能会对其进行进一步细致划分，故不再提供详细解释。

根据以上的分析和简单设计，我们大概可以以非常简短的代码实现一个功能样式都极度简朴的便签。但这并不是我们所想要的。我们使用面向对象思想进行编程的一个重要目的是降低编程的复杂度。而以上的初步设计即使不使用面向对象编程也能够轻松解决。因此考虑对该处于萌芽的便签添加一些新功能或者是新样式，以及对其间的关系进行进一步分析。

于是列出了部分上面的框架中存在的主要问题留作后续逐渐解决：

1. 核心处理流程还没有构建，即如何将这几个类较好地联合在一起实现便签需要的功能。这一方面需要完善各个类及其成员的设计；另一方面需要对 MVCStickyNote 类中的 mian 函数控制流程进行精心设计，最好使其只能拿到类的实例，做到低耦合的要求；
2. 控制器的功能太过于复杂，可以将其中功能作用级别相同的部分单独提出来作为一个类，控制器则使用这几个类的功能，这样可以大大增加其余部分的复用，且便于分析；
3. 功能过于简单，比如还可以加入特殊的编辑排版方法，或者加入导入，导出等功能。但是这部分功能目前还触及较少，需要进一步的学习来完善实现。
4. 就便签这一设计来说，如果没有 GUI，感觉始终有些别扭。因此还需要学习如何做一个过得去的 GUI 界面，不过这一部分可以放在后面再来考虑。