Design report for Smart Folder Programming Project

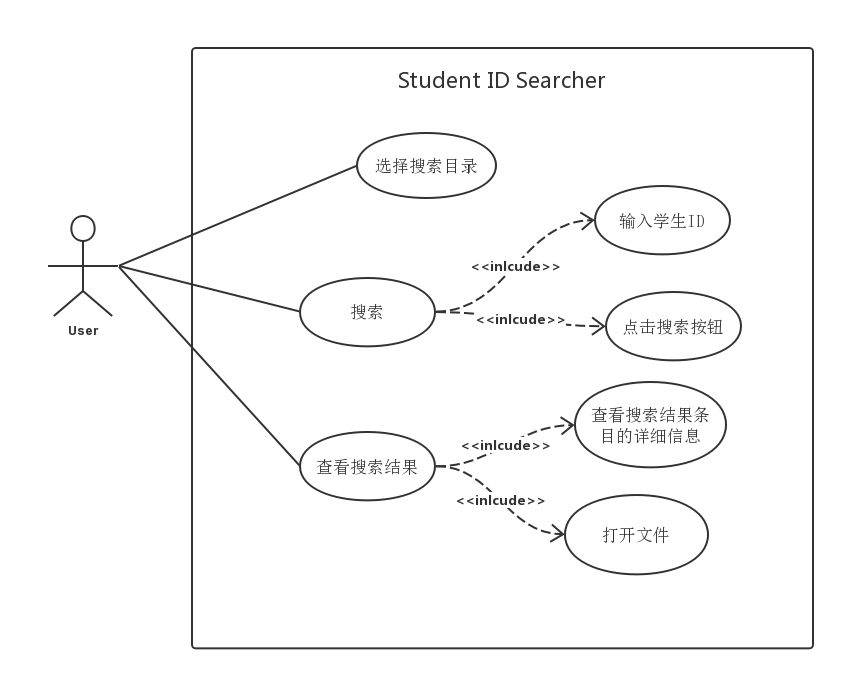
Smart Folder Programming Project is a typical GUI program based on C++, there are 2 main parts of design phase, which are system design and evaluation design.

1.System design

本项目将主要采用面向对象设计方法，在本设计文档中，我将绘制用例图，交互图，界面设计图，并给出重要函数的伪代码和UML类设计图。

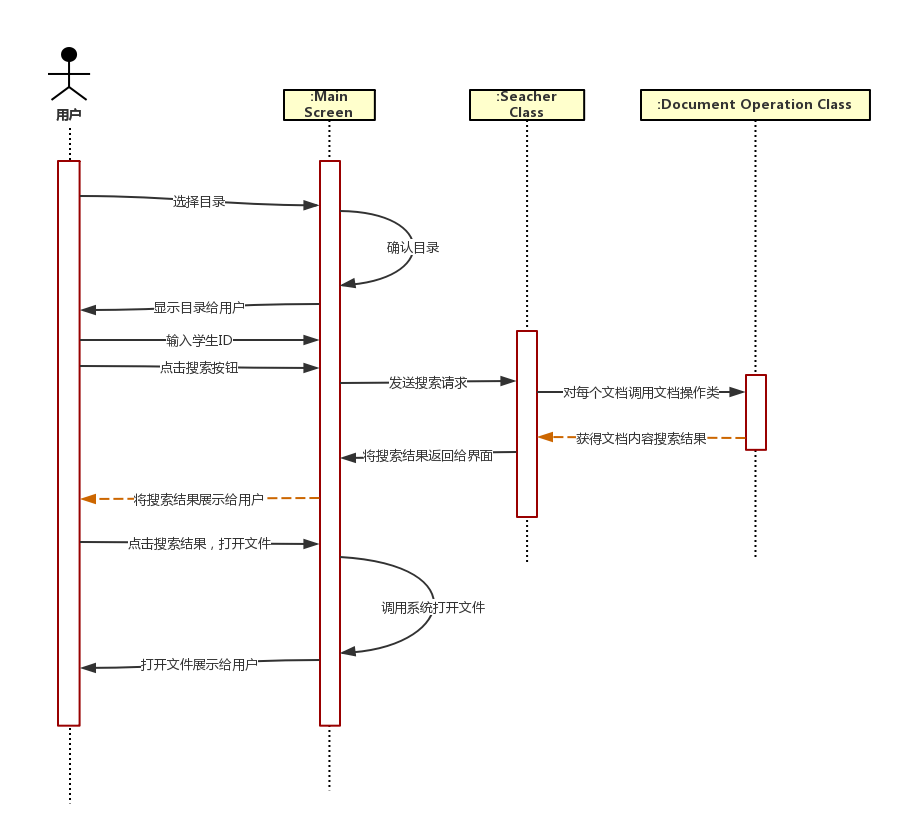
* Use-case diagram

本项目的用户用例并不复杂，用户首先选择要搜索的目录，然后开始搜索，搜索包括输入关键字和点击搜索按钮两个操作。最后，用户可以查看搜索结果，其中，用户可以在搜索结果面板上查看每个条目的详细信息，还可以点击条目，以打开文件。

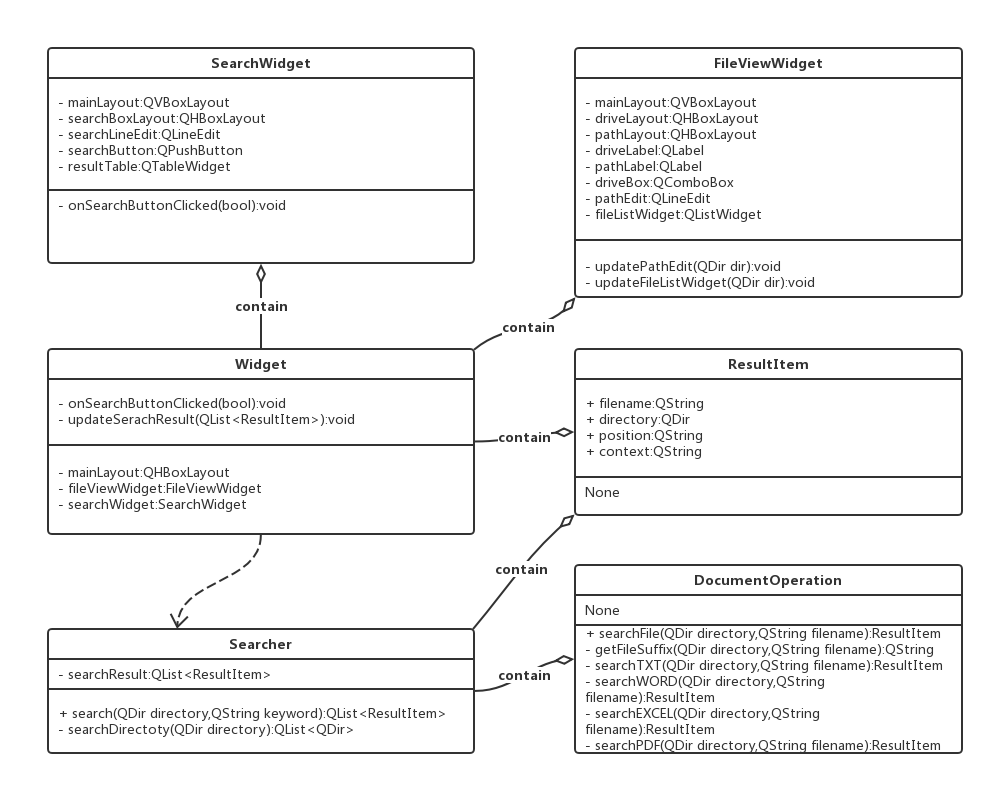
用例图如下：

* Interaction chart

用户的交互过程如下图所示，首先，用户选择目录，确认后将在界面上显示需要搜索的目录，之后用户输入关键字并点击搜索按钮，用户界面将发送请求到Searcher 类的实例，由Searcher完成搜索，Searcher类实现了查询目录中所有文件的搜索算法，对于各类文档文件，Searcher 类的实例将调用DocumentOperation 类的实例来打开TXT, Word, Excel等文档并提取文本内容，以完成对单个文档的搜索。最后，Searcher汇总搜索结果，将结果一并返回到用户界面。用户在查看搜索结果时，可以点击搜索结果中的条目，用户界面将会调用系统默认的应用来打开文件。

考虑到每一次搜索可能会占用较长时间，为避免用户界面被搜索操作阻塞，我将用户界面和搜索操作分离到两个线程中，即下图的中的Main Screen为主要线程，Searcher类的实例和DocumentOperation 类的实例都运行在工作线程中。线程间使用Signal/Slot实现通信。

* UML Class diagram

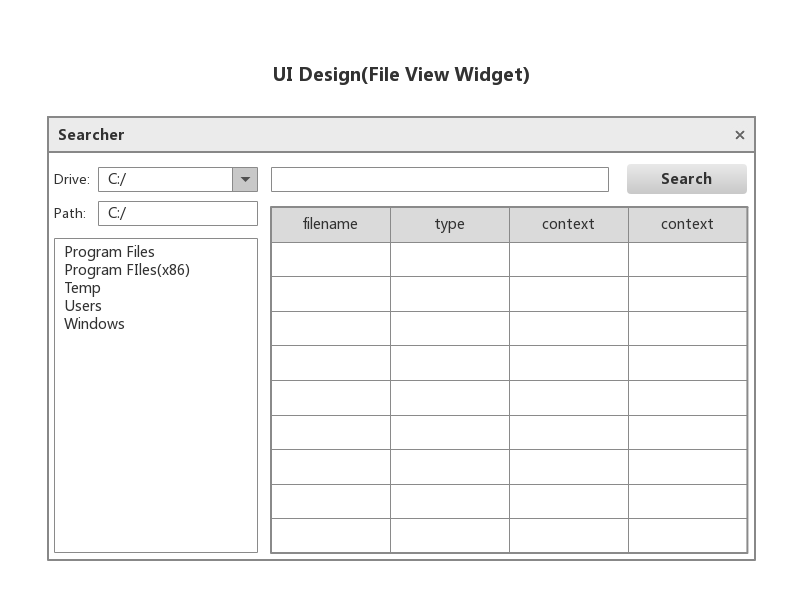
类和类关系的设计如下：

如图所示，Widget类构成用户界面，其中Widget类包含一个SearchWidget类的实例和FileViewWidget类的实例。SearchWidget类实现关键字输入框，搜索按钮，以及搜索结果展示面板，FileViewWidget用于浏览目录并选择搜索目录。

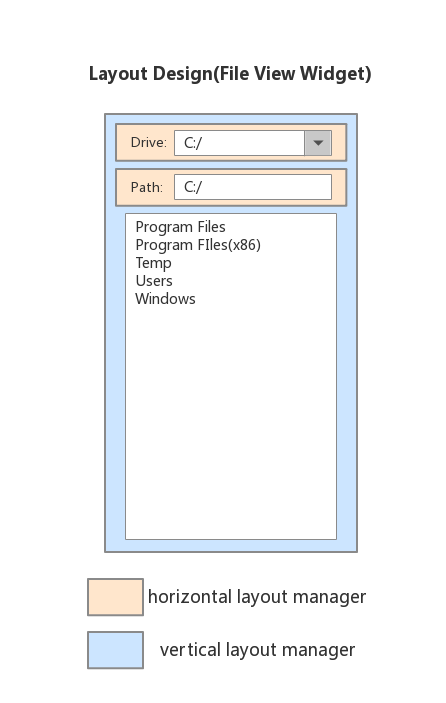
Searcher类实现了查询目录中所有文件的搜索算法，其中search方法接收两个参数，一个为目录，另一个为关键字，search方法将调用类中的私有方法SearchDirectory,获得目录下的文件列表，对于每个文档文件，search方法将调用DocumentOperation 类的实例来打开TXT, Word, Excel等文档并提取文本内容，以获取每个文档文件的搜索结果。

ResultItem类为搜索结果条目，用于在主线程（界面线程）和工作线程间通信，该类只包含数据，无任何方法。其中包含文件名，文件目录，关键字在文件中出现的位置，关键字的上下文。每个ResultItem实例表示一个搜索结果。Search类将通过一个ResultItem的列表向主线程传递所有搜索结果。

* Interface design

用户界面设计如图所示：

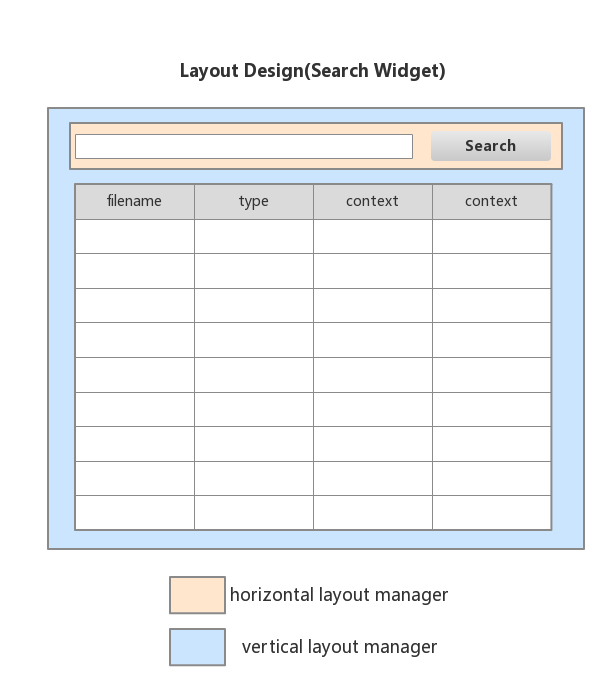
界面的左半部分用于浏览并选择目录，即上文中FileViewWidget实现的内容，界面右半部分用于输入关键字，搜索并展示搜索结果，即上文中SearchWidget类实现的内容。

FileViewWidget的详细布局设计如下：

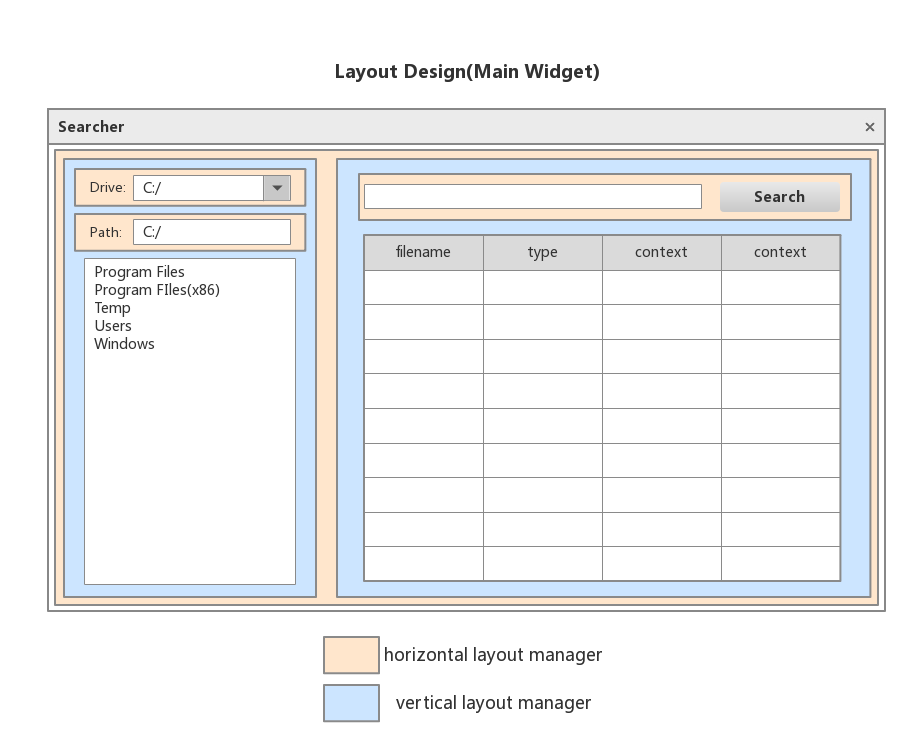
蓝色部分表示垂直布局管理器，橙色部分表示水平布局管理器。颜色仅仅用于图示 ，并不代表界面实际效果

该控件包含一个垂直布局管理器，它作为主要布局管理器。由上至下，包含两个水平布局管理器和一个目录浏览框，第一个水平布局管理器用与选择硬盘驱动器，第二个水平布局管理器用于查看具体目录，目录浏览框用于选择目录。

SearchWidget的详细布局设计如下：

该控件包含一个垂直布局管理器，它作为主要布局管理器。由上至下，包含一个水平布局管理器和一个搜索结果浏览框，水平布局管理器用于输入关键字并搜索，搜索结果浏览框用于查看搜索结果。

最终的界面布局设计如下：

主界面由一个水平布局管理器将FileViewWidget和SearchWidget包含起来。

* Search algorithm design

我们通常使用深度优先搜索和广度优先搜索遍历树形数据结构（文件系统就是一种树形结构），考虑到文件目录可能过深的问题，我使用广度优先搜索来遍历目录中的所有文件。

从选定目录出发，获取目录下所有文件和目录，将目录加入待搜索队列中，将文件加入文件列表中。

从待搜索队列中提取一个目录，重复上一步骤，直到待搜索队列中没有任何目录。这时就获得了选定目录下所有文件的列表。

本算法伪代码如下：

1. QList<QDir> searchDirectory(QDir directory)
2. {
3. QList<QDir> fileList;
4. QQueue<QDir> dirQueue;
5. dirQueue.append(directory);
7. **while**(!dirQueue.empty())
8. {
9. QDir curDir = dirQueue.pop();
10. QFileInfoList curDirList = curDir.entryInfoList();
11. **for**(**int** i=0;i<curDirList.size();i++)
12. {
13. **if**(curDirList[i] is Directory) // curDirList[i] is a directory.
14. {
15. dirQueue.push(curDirList[i])
16. }
17. **else** // curDirList[i] is a file.
18. {
19. fileList.append(curDirList[i])
20. }
21. }
22. }
23. **return** fileList;
24. }

2. Evaluation design

本项目将主要采用面向对象设计方法，在本设计文档中，我将绘制用例图，交互图，界面设计图，并给出重要函数的伪代码和UML类设计图。

本项目的评价系统设计将分为特性完成度评估，白盒测试，第三方黑盒测试三部分。

* 特性完成度评估

根据规格文档和本设计文档，期望在项目完成时实现的特性包含在如下表格中：

|  |  |  |
| --- | --- | --- |
| 特性 | 未完成 | 已完成 |
| 界面友好 |  |  |
| 目录浏览/选择 |  |  |
| 关键字搜索 |  |  |
| 搜索结果展示 |  |  |
| 显示关键字位置 |  |  |
| 显示关键字上下文 |  |  |
| 从搜索结果中打开文件 |  |  |

项目实现后，我将逐一核对每个特性的完成度。

* 白盒测试

白盒测试部分主要由我编写白盒测试用例来测试程序的各个重要组成。

测试用例如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 场景步骤 | 步骤描述 | 预期结果 |
| 1 | 目录浏览/选择 | 在用户界面选择目录 | 程序成功获取被选择的目录 |
| 程序弹出对话框显示已选择的目录 |
| 2 | 获取关键字 | 在用户界面输入关键字 | 程序成功获取用户输入的关键字 |
| 点击搜索按钮 |
| 程序弹出对话框显示获取的关键字 |
| 3 | 测试搜索算法 | 给定目录，在程序中调用Searcher类中的searchDirectory函数 | 成功获取制定目录下的所有文件 |
| 获取所有文件 |
| 4 | 测试文档内容搜索 | 给定关键字，分别调用word, excel, txt, pdf的文档内容搜索函数 | 成功获取关键字在文档中的位置和上下文 |

* 黑盒测试

黑盒测试将由我和第三方（我的朋友和家人）共同完成，下图为黑盒测试用例，我将在第三方使用软件后发放问卷来获得测试结果。

|  |  |  |  |
| --- | --- | --- | --- |
| 用例编号 | 场景步骤 | 步骤描述 | 预期结果 |
| 1 | 目录浏览/选择 | 在用户界面选择目录 | 目录框显示当前已选择的目录 |
| 在目录框查看当前目录 |
| 2 | 搜索 | 在用户界面输入关键字 | 搜索结果准确显示包含关键字的文档，以及关键字在文档中出现的位置 |
| 点击搜索按钮 |
| 查看搜索结果 |
| 3 | 在搜索结果中打开文件 | 在搜索结果中选择条目 | 该文件被对应的软件正确打开 |
| 点击条目 |