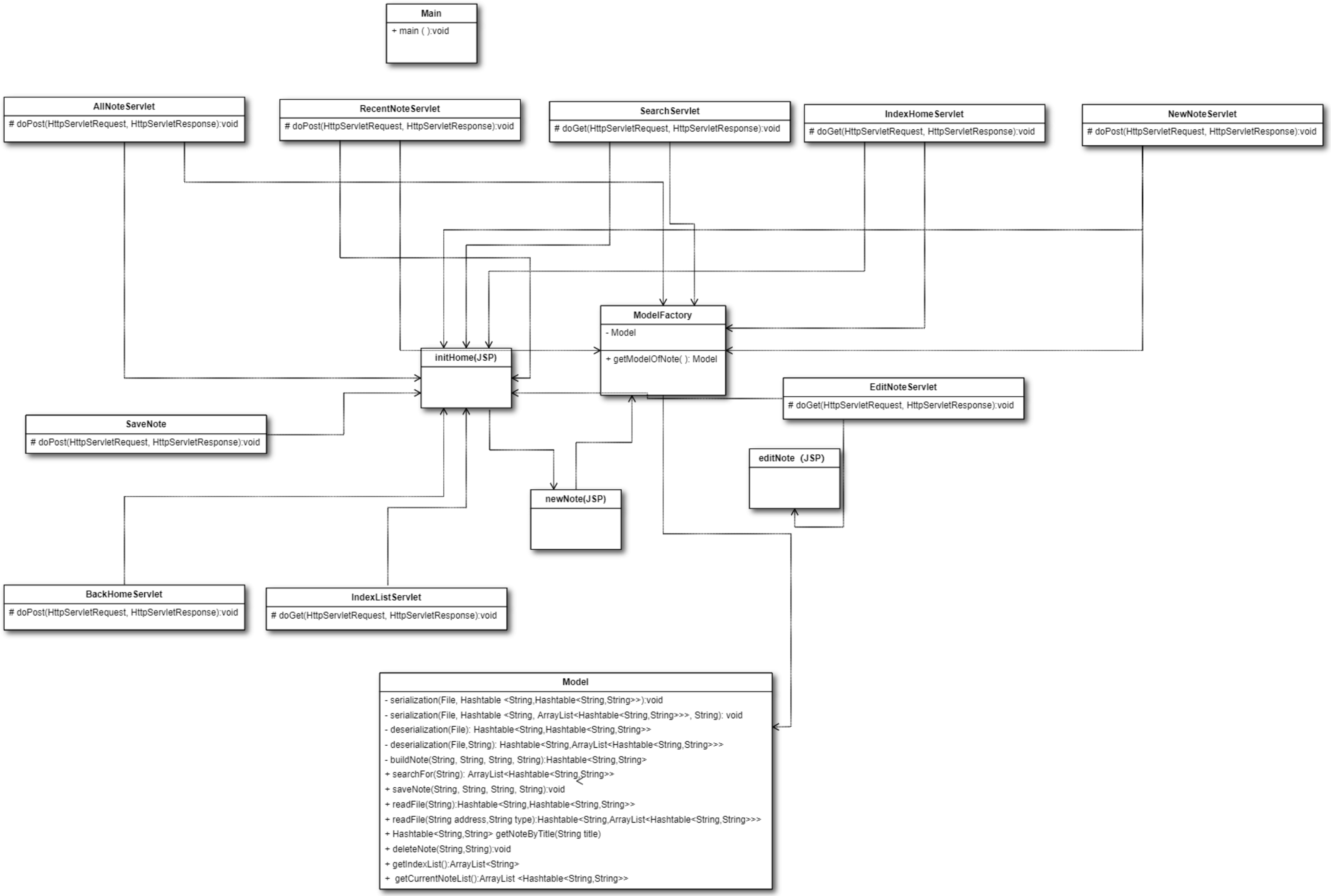
COMP0004 Coursework Report

As a note manager application, my program certainly supports creating new notes. For a new note, you need to define its title, content and the index to hold it. The name of notes should not be the same, but they can have the same index to hold them. After you click on the ‘save’ button, it will jump back to the home page. On the left side, there is a list of indexes in alphabet order. Just click on it, and the list of notes under this index appears on the right side. The notes are shown in blocks that contain their title, create time, and index. By clicking on the note, you can edit it and save it again. There is no limit to which term you can change. Title, content, and index can be changed. There are also two choices of display all notes on the left side. The current note shows that all notes begin with the newest notes. All notes will just randomly show the notes. Under each note, a delete icon is used to delete the note. At the top of the page, there is a search bar. The search can not only search by index, but also by name and content. All related notes will be shown.

The structure of the web application follows the MVC structure. The model class is used to arrange data in the special files in a data file folder and the model factory class is used to automatically build needed model classes. The model class packages the detailed data operations. With the model class, the servlets can deal with the data by using the method instead of rewriting code for that. It reduced the repeatability of code. The servlets work as the controller parts. They build models and get needed data from them. Then they sent the data to the correct JSP pages. Because most of the functions are on the home page, there are many servlets for the home page. For the edit note and new note page, each one has its own servlet. The view parts are the JSP pages. They accept data sent by the servlet. The home JSP page refreshes itself depending on where did the users come from. This feature allows the home page to have many different statuses, so its structure of it can be changed.

I am not quite satisfied with the structure I designed. Although the home page can change its status, the code on the JSP page is mixed and unclear. The servlets are too separate. There is some repeated code that appears again and again.


Main
+ main ( ):void
AllNoteServlet
# doPost(HttpServletRequest, HttpServletResponse):void
RecentNoteServlet
# doPost(HttpServletRequest, HttpServletResponse):void
SearchServlet
# doGet(HttpServletRequest, HttpServletResponse):void
IndexHomeServlet
# doGet(HttpServletRequest, HttpServletResponse):void
NewNoteServlet
# doPost(HttpServletRequest, HttpServletResponse):void
ModelFactory
- Model
+ getModelOfNote( ): Model
initHome(JSP)
EditNoteServlet
# doGet(HttpServletRequest, HttpServletResponse):void
SaveNote
# doPost(HttpServletRequest, HttpServletResponse):void
editNote (JSP)
newNote(JSP)
BackHomeServlet
# doPost(HttpServletRequest, HttpServletResponse):void
IndexListServlet
# doGet(HttpServletRequest, HttpServletResponse):void
Model
- serialization(File, Hashtable <String,Hashtable<String,String>>):void
- serialization(File, Hashtable <String, ArrayList<Hashtable<String,String>>>, String): void
- deserialization(File): Hashtable<String,Hashtable<String,String>>
- deserialization(File,String): Hashtable<String,ArrayList<Hashtable<String,String>>>
- buildNote(String, String, String, String):Hashtable<String,String>
+ searchFor(String): ArrayList<Hashtable<String,String>>
+ saveNote(String, String, String, String):void
+ readFile(String):Hashtable<String,Hashtable<String,String>>
+ readFile(String address,String type):Hashtable<String,ArrayList<Hashtable<String,String>>>
+ Hashtable<String,String> getNoteByTitle(String title)
+ deleteNote(String,String):void
+ getIndexList():ArrayList<String>
+ getCurrentNoteList():ArrayList <Hashtable<String,String>>