COMP0004 Coursework
Java Web Application

Notes App

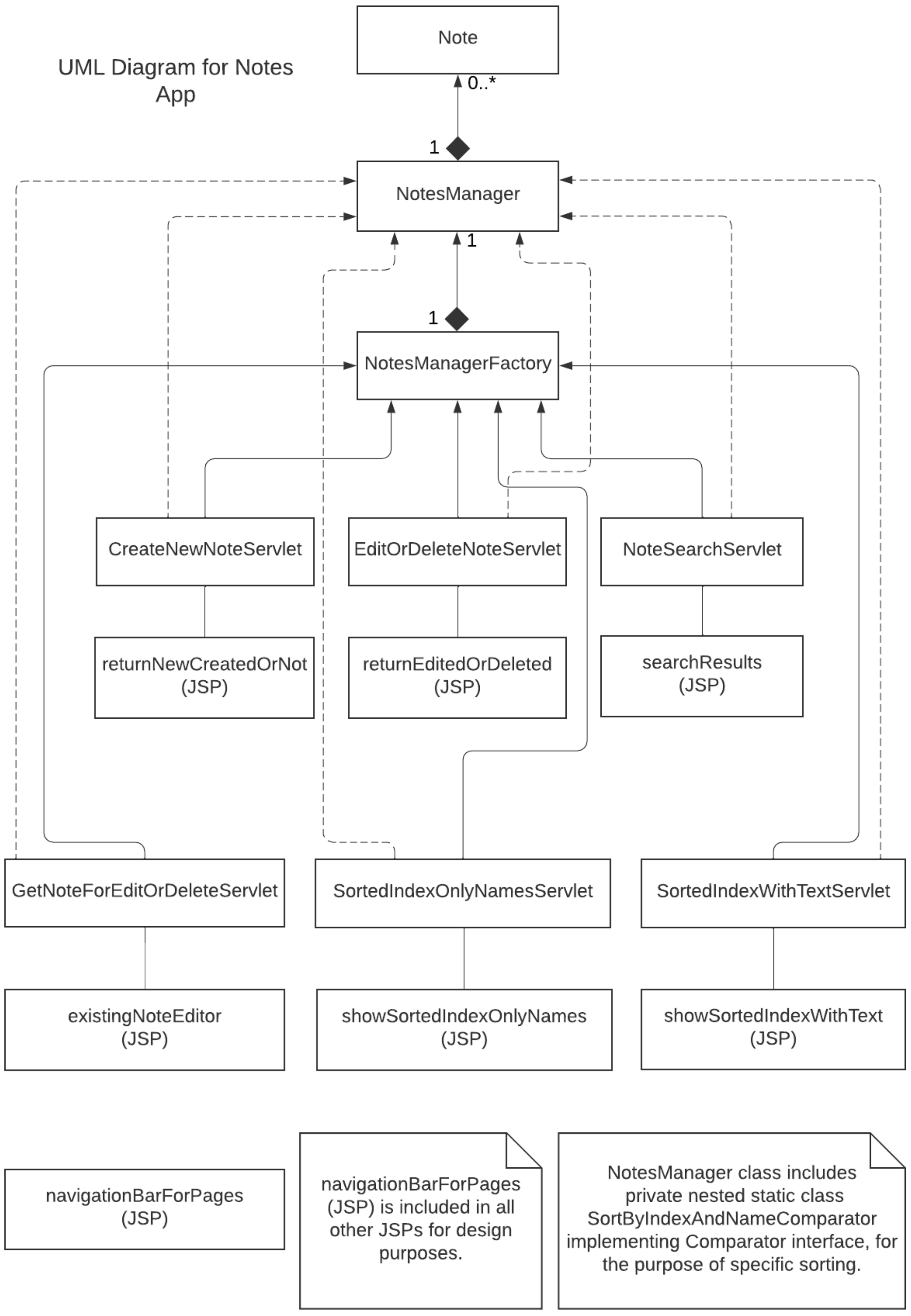
Main features of the program

Notes app gives users the ability to create, edit, delete, search, as well as lookup a sorted list of notes. Note itself, suggests of an index, name, URL, image and/or text. For creating or editing notes, user is given the opportunity to choose what he wants to store in it, meaning none of the note editor fields are required to be filled. For instance, if user wants to specify only image and text in the note, rest of the fields will be set to default values.

Furthermore, through navigation bar, user can choose to see full sorted list of notes. Indexes are sorted alphabetically, while note names in the indexes are further sorted the same way. Names of notes presented suggest of a link each, which opens their corresponding note editor for straightforward preview/edit/delete. User can choose between two options: seeing sorted list with or without text from the notes.

Moreover, user can search for the index and the notes in the index by inputting search keyword. Once the search finishes, results page is displayed to the user. Results page suggests of two parts. First part displays a list of matches by index with their notes' names. Second part displays list of matches by note name. This allows for only one search for both index or note, therefore easier use of the app. Clicking on the corresponding note name (in both parts) opens a note editor for its preview, edit or deletion.

Important feature of the app is that after stopping and rerunning the app, all notes will be again available to the user. This feature is possible since, upon creating/editing/deleting a note, all note files are being created/updated/deleted accordingly. Furthermore, this way user does not need to explicitly load or save to a file, which makes it much simpler for everyday use.


UML Diagram for Notes App
Note
0..*
1
NotesManager
1
1
NotesManagerFactory
CreateNewNoteServlet
EditOrDeleteNoteServlet
NoteSearchServlet
returnNewCreatedOrNot (JSP)
returnEditedOrDeleted (JSP)
searchResults (JSP)
GetNoteForEditOrDeleteServlet
SortedIndexOnlyNamesServlet
SortedIndexWithTextServlet
existingNoteEditor (JSP)
showSortedIndexOnlyNames (JSP)
showSortedIndexWithText (JSP)
navigationBarForPages (JSP)

navigationBarForPages (JSP) is included in all other JSPs for design purposes.

NotesManager class includes private nested static class SortByIndexAndNameComparator implementing Comparator interface, for the purpose of specific sorting.

Description and evaluation of design and programming process

Program operates using MVC (Model-View-Controller) pattern. Model system suggests of three core classes: Note, NotesManager (represents Model) and NotesManagerFactory (represents Model Factory). Note is a class which describes a "universal" note, which has five private variables corresponding to its index, name, URL, image, and text. Note provides setter and getter methods, which are created alongside private variables for the purpose of encapsulation. NotesManager is a class which holds collection of notes in an ArrayList using a private variable. That implementation gives the user ability to store large number of notes, without any program restriction.

The NotesManager class provides methods for searching, sorting, as well as note file input/output, for use of created notes even after rerunning the program. Since NotesManager is being used every time user inputs a request, it is appropriate to have only one NotesManager object throughout the entire lifetime of the program. That is why, NotesManagerFactory was introduced (NotesManagerFactory provides static function returning a reference to an initialised NotesManager (Model) object), following the Singleton pattern design. To support sorting alphabetically for both index and note names from that index at the same time, private nested static class implementing Comparator interface was added inside the NotesManager class. Note files created or edited are being saved as text files in dedicated directory (data), first four lines indicating different properties of the note, with fifth line onwards representing text property of the note, for same representation of text on the webapp as well as in the file itself (newlines will be at same places, where the user specified in textarea at the note editor page of the app). In fourth line of the text file, filename of the image uploaded in the app is specified for getting its path in the imageData directory, where the images themselves are stored.

Furthermore, apart from the Model, there is also Controller system, which suggests of different Servlets, all with their own separate purposes. Servlets get an instance to the manager object through the Singleton pattern, and by calling function on it, either update the Model with new user input, or get the data from the Model for its display on the app. For displaying pages on the app, Servlets use Java Server Pages (JSP), which represent the final View system (Fulfilling the specification for MVC pattern). JSPs are primarily used for displaying, but also for taking the input from the user. Apart for listing out searched and sorted lists of notes, they are especially important for allowing user to edit existing notes, without having to start writing them all over again. For example, once the user clicks on the corresponding note entry from a list (also displayed through JSP), another JSP will be used for displaying the corresponding input form of the note, filled with previous contents and an image alongside it. This allows for straightforward preview/edit/delete.

In summary, Notes App has all features user could need in a note editor, while implementing all requirements specified. Comments (documentation) are included in the files, explaining the classes and methods, which allows for easier understanding and further use of the code. Finally, HTML and CSS are used for tidy look of the app, therefore rounding up the project into one user-friendly, useful, and easy to use app.