Assignment 1

Authors: Elena Ibi, Nour Oujjit, Wassim Mayyasi, Ørjan Johnsen

Introduction

Author(s): Elena Ibi, Wassim Mayyasi

Imagine the scenario where you constantly work on similar programming projects and you have so many code snippets that you have to reuse over and over again. Wouldn't rewriting them multiple times be an annoying tedious task? What if there was a system you could use to save all your code snippets and be able to retrieve them easily whenever needed? The solution to this problem is the code snippet manager which is the project track we chose to go for. A code snippet manager is a very useful tool that programmers and other users can use to keep a record of code snippets and libraries for reusability purposes.

Additionally, the code snippet manager will provide a well-structured and organized system to keep record of all the code snippets that the user will add; this makes the search and update functions easy for the user. Our code snippet manager will allow the user to display, sort the code based on some specified attributes, mark snippets as favorite snippets, copy the code of the snippet onto clipboard aside from providing the regular update functions such as adding and deleting a snippet through our user-friendly graphical interface.

Our target users are any users who work with code on a frequent basis. It does not get any more specific than this since we believe that it is such an important tool that anyone working with code, ranging from professional programmers to aspirant programmers, can benefit from. Furthermore, the graphical user interface that our system will have will make the navigation of the code snippet manager very easy and user-friendly; it will display all the important buttons that the user can click on to have access to the various functions, list all the snippets that have been added so far on the front page. We will also provide a help button whose function is to provide general information on how to navigate the system and how to use it. Generally speaking, we thought that a user-friendly interface was one of the most important aspects for systems such as code snippet managers because of the ease it should provide in the retrieval process. Hence, we decided to implement a graphical user interface in the intent to facilitate the navigation and the general use of the code snippet manager for every type of user.

The decision to implement the code snippet manager was taken because we thought that a way to organize and store code snippets was a useful activity in the long run for anyone who works with code. It stood out as a project to us since it did not tackle the more commonly chosen projects that implement games or similar types of applications that deal more with entertainment but rather provided a useful functionality for anyone who works with code at any level and is useful for the development of any other type of software. There were few sources online we took inspiration from such as this youtube video which gave us an

excellent overview of not only what a code snippet manager should do but also an idea for a possible graphical user interface. The link to this youtube video is the following:
https://www.youtube.com/watch?v=IfWRwwmX9Q8
It is a 2 minutes and 30 seconds video that highlights the important features of an already existing code snippet manager, ‘Code Notes’.
Another source of inspiration is the following website:
https://dev.to/tomlangdon/5-code-snippet-managers-that-will-change-the-way-you-write-code-10ml
This website underlines all the general pros and cons of using such a system and provides a similar overview on different code snippet managers; it mentions five code snippet managers that are widely used worldwide. The similarities and the differences of these code snippet managers are mentioned which helped us gain more knowledge on these systems and gave us a wider perspective design wise. We also thought of some future improvements that the existing code snippet managers could adopt.

Our goal of the final design of this project is an application for individuals who come in contact with programs or code. Those individuals will be able to use this application to store code snippets that they feel are important and possibly reusable in the future. They would add their chosen snippet into the text portion of the application, then add a name they think is appropriate, tags they think are useful to add and search up later, and the programming language the code is written in. All these characteristics or attributes will be used for the user to later extract the needed snippet from the collection of snippets added. There is an editing option as well, it could be used for any mistakes made while adding the snippet, or possibly for updating the code with a better one made. The user will also have a search bar with different options, for example, searching the code snippets currently stored with a certain attribute. In the front page, there will be a display of all the stored snippets, they will be displayed by their date of creation, however, the user can alter this sorting order to different options. The displayed snippet entries will each have some buttons next to them, for favoriting and deleting. When deleting a snippet, there will be a reassurance message for the user just in case of misclicks. Finally, after the user chooses their code snippet, the code snippet will be available on display, and a possible button to copy the code snippet to clipboard is present; that will allow the user to use this code snippet in whatever project they are currently working on.

In the back end, the code snippets will be stored in text files, and the descriptions and attributes of each snippet will be stored in a JSON file. When a certain code snippet is chosen for display, what happens is that the file name of the snippet will be stored in the descriptions of that snippet in the JSON file, so that it has to get extracted first, then access to that file is made to retrieve the actual snippet.

Features

Author(s) : Elena Ibi

Functional features

Making a code snippet manager comes with several already implied features. Adding, editing, displaying and deleting are base functionalities of a snippet manager. They have somewhat obvious functions. Searching, sorting, favorites, copying and the help button are helpful functions that all make the program more user friendly. Search, favorites and sorting are there to make it easy to find the snippets the user is looking for. Copying is there to make it easier to use the snippets elsewhere. And the help button is there to familiarize new users of how the interface of the application works.

Id	Name	Description
F1	Adding	The system shall enable the user to add a snippet. This will be done by providing the user with a template on which the name, description, tags, code language and content of the snippet can be filled in.
F2	Searching	The snippet manager shall allow the user to search for specific snippets based on three options: name, tags and the code language. Then this function will output all the filtered snippets based on the attributes chosen in the search.
F3	Editing	The user shall be able to alter the name, description, tag, language and the content of already existing snippets using a similar template to the one for adding snippets.
F4	Displaying	The snippets shall be displayed on the front page. The system can show the entire collection of existing snippets or a subset thereof based on searching and filtering. The snippets will be displayed by name, description and time of creation.
F5	Deleting	The user shall be able to delete a snippet by clicking on a delete button next to where the snippets will be displayed. The user should be able to click on the snippet and then click on the delete button to remove the snippet from the system.
F6	Help button	The system shall provide a help button that any user can click on. This button will display general information about the system and how the user can navigate and use the system.
F7	Sorting	The code snippets shall be displayed on the front page sorted by the date of creation (default sorting). The user shall then be able to choose to sort the snippets based on the name, language or date of the snippet.
F8	Favorites	The system shall enable the user to select favorite snippets. The user

		will be able to do so by pressing a button next to the displayed snippet. The system should provide a section on one side where all the favorite code snippets will be displayed.
F9	Copying	The system shall provide a copy button that will enable the user to copy the code of the snippet onto clipboard.

Quality requirements

Authors: Ørjan Johnsen, Nour Oujjit

Several quality requirements are required to develop a user friendly snippet manager. The qualities discussed in this section cover the three most suitable quality attributes to our system. These attributes are usability, reliability, and maintainability. Attributes befitting network applications are not covered.

The first two requirements in the table are supposed to ensure the maintainability of the system. QR1 does this by formalizing style constraints whilst QR2 is to maintain the structure of the code. The purpose of QR2 is to prevent having to alter many functions within the code when attributes within classes are changed. The requirement specifies that instead of passing class attributes to a function, an entire object should be passed. In this manner no big changes, such as adding or removing parameters for a large number of functions, would be necessary.

The Reliability requirements are covered by QR3 until QR7. The first two (QR3 and QR4) are to ensure that no data is added, altered or deleted without the user's consent. Requirement 6 (QR6) relates to QR3 and QR4, but is different in the sense that it specifies an extra confirmation, after pressing the delete button, in order to prevent accidental loss of information. Another requirement that is supposed to prevent this is QR7. It states the existence of a termination check whose purpose is to look for any conceptual snippet before terminating. When an “in progress” snippet is found the user is presented with a confirmation message asking whether or not to close.

The final two requirements describe usability constraints. QR8 states that all text elements should be short and concise. The last requirement (QR9) states that there should be a user manual within the program in order to simplify the use of the features the system has to offer.

Id	Name	Quality attribute	Description
QR1	Consistent Style	Maintainability	The system code should follow one consistent code convention to allow for easy addition/extension of features
QR2	Function Parameters	Maintainability	The systems code should use functions whose parameters are class objects in order to limit necessary changes due to alteration of class attributes.
QR3	Intentional Alteration	Reliability	The system shall not remove/alter snippets unless user presses delete/alter button (in order to prevent unwanted loss of information)

QR4	Intentional saving	Reliability	The system shall not save a snippet unless user confirms saving message (to prevent storage of unwanted information)
QR5	Unique Keys	Reliability	System shall ensure uniqueness of snippet names in order to prevent complications with deletion/alteration.
QR6	User Confirmation	Reliability	The system shall ask for confirmation from the user before deletion or alteration of a snippet
QR7	Termination Check	Reliability	Before the system terminates it should check for unfinished/unsaved snippets and ask the user whether to save or discard before closing.
QR8	Descriptive Commands	Usability	The text fields used by the system shall be short and descriptive.
QR9	System instructions	Usability	The System should have an instruction manual built in.

Java libraries

Author(s): Wassim Mayyasi, Nour Oujjit

Name	Description
json-io	We looked for a library that will translate the json file to a Java string we can parse, and also the opposite, from Java string to a json string. We found several libraries that do this parsing, including json-io, however json-io seemed to be the only one that “writes” on the JSON file we will store on. This function will be useful in our case where we can directly store the added snippet’s descriptions immediately onto the JSON file.
JsonPath	We plan on storing the data of the snippets within a Json file. JsonPath is a library that enables the retrieval of JSON data based on the use of specific queries. This library will be useful to realize the searching, filtering, favorites and displaying functionality of the system. We can, for example, extract the data of all snippets that are marked as favorites.
Scene Builder	Scene Builder is a UI designer tool that has a drag and drop functionality. We chose this tool because we did not want to waste too much time on programming the UI, and using this tool makes it easier to have the UI we imagined without too much effort. Scene Builder works with the JavaFX library.
JavaFX	JavaFX is a library that will provide us with all resources needed to develop a graphical user interface for our application. The Scene Builder tool works alongside the JavaFX library, so it was needed after the decision to use Scene Builder was made.
Java Standard	The java standard library has several packages that will have multiple uses for our project. Starting with the java.awt; It contains the needed classes/functions for us to copy to the clipboard for our feature(F9). Next, there is the java.lang; for uses of different data types throughout our project, as Java does not have those data types as primitive, but as classes. Finally, there is the java.util, the utilities package. Using this package, we will have access to Lists and Arrays, to possibly parse the json arrays to java arrays. Also, then we can use the sorting function that is available in this package as well to sort those arrays as needed for our feature(F7). The last usage of this package is the availability of Date and Time classes; we can use those classes to add the attribute of “date and time added” to each snippet.