● Cover page details:

Input API: Marvel

Output API: SendGrid

Extension: The user is asked for a number from 3 to 50 when the application starts. This number is a 'popularity threshold'. If a character that is searched has more than the threshold of comics, the application should pop up an alert saying 'This character is popular!'[END]

Feature Set: Hurdle

Implemented features:

- Loading page including a "splash" image and a bottom process bar
- Background music which can be turned off/on
- String localization

1. Code Modifications Made

a) *OnlineEngine*, *DummyEngine*, *DummyInEngine*, *DummyOutEngine* Class

i. Member variable changes

I have added two member variables to these classes, respectively *thresholdListener* and *alertFactory*.

The reason for adding thresholdListener is to use it to verify if a character is popular when the user intends to search a character.

The reason for adding alertFactory is that currently the engine classes create alerts by themselves which violates Model-View separation. After considering remaining Model-View violation or raising a better design thought, I chose the latter option. Therefore, the new alertFactory class has been added to be responsible for creating the newly added alerts, while the existing alerts have not been modified and remains being created by the engine.

ii. Method changes

ThresholdLoad() method has been created for asking **viewFactory** to initialize the newly added threshold input page.

findCharacter() method now will additionally trigger the **thresholdListener**'s **verify(JsonObject)** method to determine if the searched character is popular. And if so, the **alertFactory** would be called to create the corresponding alert.

getAlertFactory() has also been added for other classes to get current alert factory.

characterComics(int id) has been added to generate the number of comics a character has.

iii. Constructor changes

I have updated the engines' constructor to make it initialize the newly added thresholdListener and alertFactory.

b) ViewFactory class

A new **BuildThreshold()** method has been added. As the factory class is supposed to contain all the entrance for creating each product. A new page coming with a new entrance is a reasonable change.

c) Routes class

As the number of comics is required by "total" instead of "available", the data cannot be found using existing requests. Therefore, a new **characterComics()** has been developed to request the total number of comics a specific character has.

d) ModelTest

As **findCharacter()** has been modified to make one more http request, the corresponding test has been updated as well. A new mocked http response has been created to handle the **characterComic()**'s request.

2. Feature Modifications Made

a) Start process

Previously the start of the application is the splash image loading page and after that is the search page, now the threshold input page is between them. The user must enter the threshold after loading then it comes to the original previous search page.

b) A new alert

Previously when the user tries to navigate a specific character, only "cache hit"

alert might appear before entering the character detail page. Now, an additional popularity alert might appear after the “cache hit” alert, and the user must close the alert to go to the character detail page. Overall, one possible step has been added here.

3. Base Code Discussion

a) Pros

i. Factory pattern

The **ViewFactory** class using factory patter has benefited the extension. Though it might not be implemented professionally, it still gives me a clear view of where my new extended page and its controller should be initialized and how it should be created. Any new page could be easily added by following the process of creating a fxml file, creating a controller and adding an entrance in the **ViewFactory**.

However, the factory could be improved for not having so many duplicate codes. The related codes could have been reused better if the inheritance of controller being designed better.

ii. No pattern but good design

The routes class have been helpful for me to easily establish new http request. Without knowing how it is working, it has still been easy to make extension. This class has good extensibility.

Though it might be better to handle the response in this class. Instead of just returning http response, it might be better to check the status code then throw exceptions or return the body text.

b) Cons

i. Factory pattern

All the alerts are created by the engine class, which not only impacts the Model-View separation, also increase the difficulty to extend and maintain the code resulting from the overburdened responsibility of engine class. When I was to implement the extended alert, I had to choose between using a new design or modify the existing code, which is typically a sign of poor extensibility and maintainability.

In order to increase the maintainability and extensibility by retrieving this responsibility from engine classes, an alert factory could have been used to be responsible for initializing all the possible alerts.

ii. Observer pattern

When I need implement the threshold listener's trigger in the character search, the only way is to put the trigger in the search method. This modification to the existing code could have been avoided if there is a well-applied observer pattern. If there is an observer interface and the engine updates all the observers after each search action, all the features requiring observation could be easily extended by creating an observer and attach it to engine.

All in all, the observer pattern should be applied to achieve better extensibility.

iii. Code reuse

There currently are four engine classes respectively being responsible for different online/offline settings. Consequently, when I modify anything in the engine class, I must modify all the four of them. Better use of inheritance could significantly increase the maintainability and modularity.

iv. Builder pattern

With the increasing responsibility and complexity of engine class, it might be beneficial to apply a builder pattern here to make engine's creation easier and more customizable. Applying a builder pattern for creating engine class would not only benefit the testing process, but it would also increase the maintainability and extensibility of this class.

v. Lack of comments

The lack of comments makes the functionality hard to understand. It also makes the code hard to read. As a result, this crucially increases the difficulty to extend and maintain existing code.

4. Resulting code discussion

a) Positive impacts

The addition of alert factory has increased the maintainability, extensibility, and modularity. All the alerts in the factory could be easily maintained and modularized. And new alerts could be extended easily by adding it to the factory.

b) Negative impacts

As the newly added alert factory and threshold listener class are both being managed by the engine class, which increases the difficulty in maintaining the engine class, this has reduced the maintainability of the engine class.

The newly added threshold listener also decreases the maintainability of the whole project because it did not follow any existing design structure. This is a new single class created, it would have been better if there is existing observer pattern, so the threshold listener could just be a new concrete observer.

The newly added view factory entrance and fxml file slightly reduce the maintainability, which may be considered unavoidable and reasonable.

5. Process discussion

The below processes are in the order of my development.

a) Understand what feature needs to be achieved

Firstly, I conclude that two features need to be achieved, respectively an alert occurring when the searched character is popular, and a window at the start of the application for user to input the threshold.

b) Implement the window

As a matter of fact, I am the designer of this project, which means I need not to get familiar with the structure. And I am clear in mind that to implement a new window I could follow the existing design straightly by creating a new fxml file and creating a new controller.

i. Fxml file

I used the javafx scene builder to design the window and generate the fxml file. Afterwards, I gave necessary element an id for being referenced by the controller, and I also set the button's action's name to reference the event in the controller.

ii. Controller

Like existing controllers, the new threshold controller inherits from the **contollerImpl** class. And apart from this, as I understand the text field should only accept numbers, I add a listener to the text field to make it only accepts numbers. Then, I implemented the method when the confirm button is clicked. It

will alert if the input is illegal (ie >50 or <3), otherwise it would process the number to threshold listener.

iii. Initialization

As I mentioned in above sections, the initialization process of this window is put in the view factory. I add a new entrance for this window and changed the start page of the application from splash image loading to threshold input.

c) Alert Implementation

The alert has been implemented along with an alert factory. Also, as I mentioned above, I could have added it in the engine's search character method just like other existing alerts. But as this approach has already severely affected the maintainability and extensibility. I decided to implement an alert factory to manage the alerts without modifying existing codes. And the new alert has been implemented in the alert factory.

d) Threshold Listener implementation

As there were no observer pattern implemented, I implemented **thresholdListener** class to be responsible for the logic of deciding whether a character is popular. Also, as I do not want to modify existing code to let controller accesses **thresholdListener** class through the engine, I made the threshold number static and could be set through a public method.

e) Update string localization

I have updated the string localization file to have the text for extensions.

f) Add new http request

After knowing the required number of comics is "total" instead of "available", as normally there is no such data from existing character query, I developed a new

request to retrieve specific character's total number of comics.

g) Update test

Due to the change to some existing code, the test needs to be updated to successfully run. I updated the test file to support the modified methods of engine class, especially regarding the *findCharacter* method.

6. References

None.