RSS Reflection

Holly French, Andrew Bacon, Veronica Lynn

During our code review, there were three things we discussed, mostly pertaining to the notion of code readability, especially depending on who our audience is.

1. Long, convoluted conditionals: we had a number of long convoluted conditionals that ended up dealing with a lot of edge cases regarding date and time in our RSS parser. These conditionals helped us determine when to print, but ultimately, they checked a lot of edge cases. The conditionals were more than one long line long, so they ended up looking like brute-force solutions to our problem. Instead, we decided to move the conditionals to another method called *isPrintable* and try to clean them up there. Creating this new method provided an additional bonus: it provided a convenient place to check for regex title matches (if necessary), which was previously being handled inconsistently.

2. Use of the ternary operator: In our code review, we discussed how readable the ternary operator was. One thing we talked about was our audience and the role they play in reading our code and influencing our decisions to use certain Java features. For those who are unfamiliar with Java, the ternary operator can be confusing. It isn't inherently readable, but it is much shorter and succinct, yet expressive. It can be handy in some situations while unwieldly in others, so we decided to use it on a case-by-case basis, optimizing readability over our own convenience.

3. Useless comments: We had a lot of comments that described code that was already more-or-less self-documenting. When we were writing Java functions to write to and read from a file, it was useful to have comments (especially since some of us weren't as familiar with a lot of these Java Objects to interact with files). However, once the code was completed, the comments were basically redundant, beacuse our function and variable names basically expressed what the code was doing. Similarly, we had some comments that had not been updated when a method was updated, resulting in outdated comments that did not reflect what the code actually did.

4. Better seperating functionality into classes: A large portion of our main class, RSSReader, was taken up by various methods dealing with how to display the posts for various parameters. This was both messy and did

not fully represent what the purpose of the RSSReader did. To fix this, we created a new class, FeedDisplayer, whose main purpose was to handle the RSSReader's output.

We also made a number of fixes that we did not discuss in the Code Review, but we thought that they would be useful to a better user experience, and they would help improve the program with more robust error handling. We also realized that some parts of the assignment were implemented incorrectly, so we had to fix those as well.

1. Better, concise user feedback: We ran into this difficulty a few times: various feeds and entries would be missing specific attributes. For example, three of the feeds we tested did not have dates. We discussed how to present dateless feeds to the user, but we ended up ignoring date if there was none provided.

2. More robust error handling: What happens if the user types in command line arguments that are not valid? We decided to try to add in some error handling so that the program would not totally crash when there weren't valid command line arguments. We tried to add more error handling that printed warnings/errors to the screen with descriptive messages. For example, we had some sections of the code that were just wrapped in giant try/except blocks, so we tried to break those down where possible to provide more targeted handling of all possible errors at each step.

3. More beautiful output: We got rid of those ugly html tags in the descriptions!

4. Bug Fixes: We realized that we had implemented some parts of the project incorrectly. For example, even if the title REGEX was speficied, it was only sometimes being checked. As mentioned earlier, doing a better job of creating methods to handle certain things helped us to handle this correctly. We also had some minor issues with alphabetization of feeds, which was easy to fix but was a problem nonetheless.