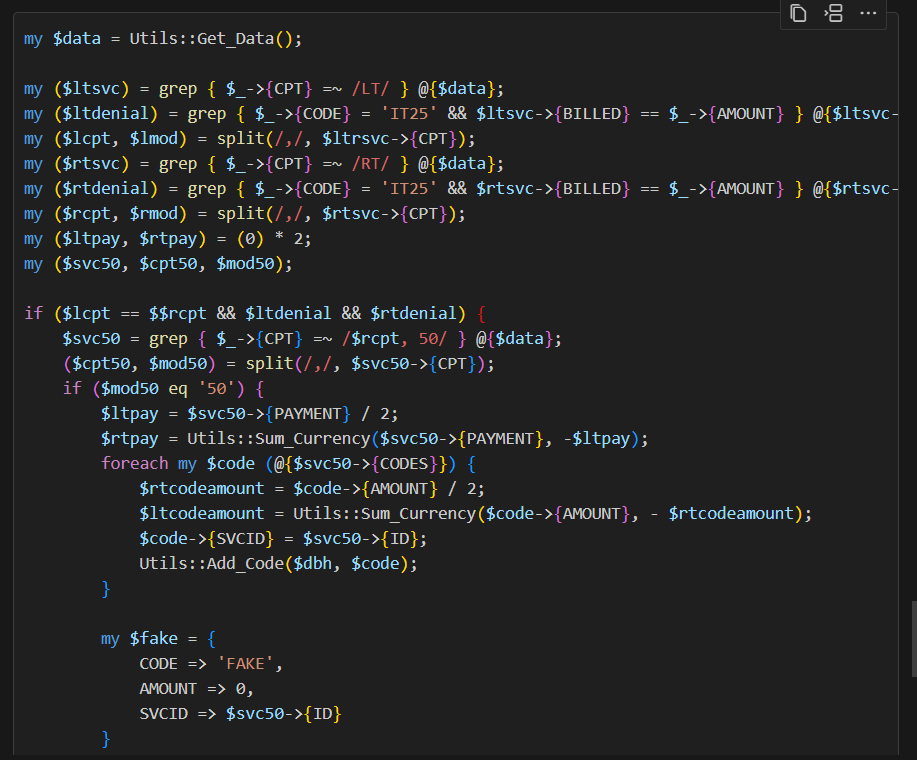
**How Perl to Java Code Migration is done in posting Modernization? These are the steps.**

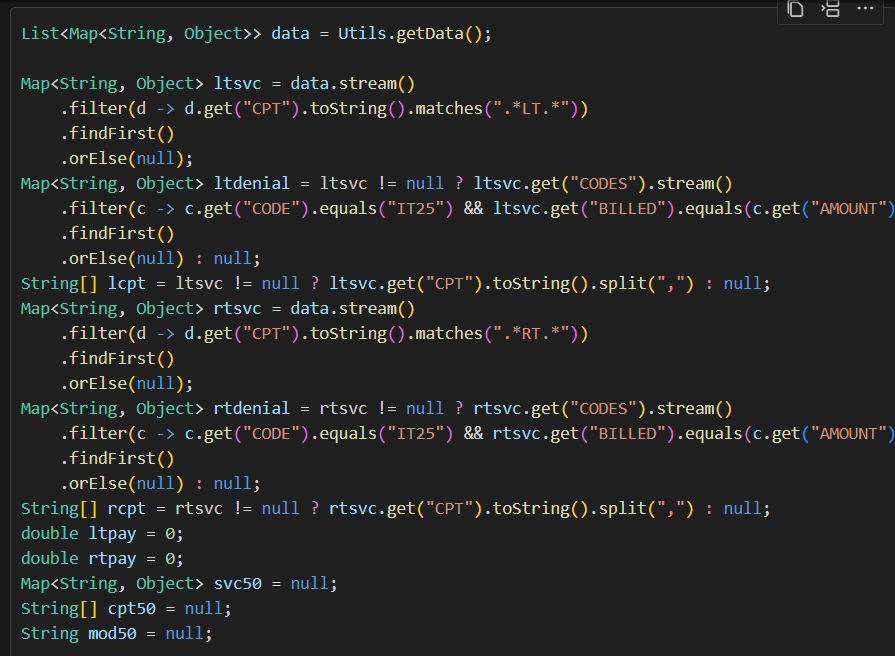
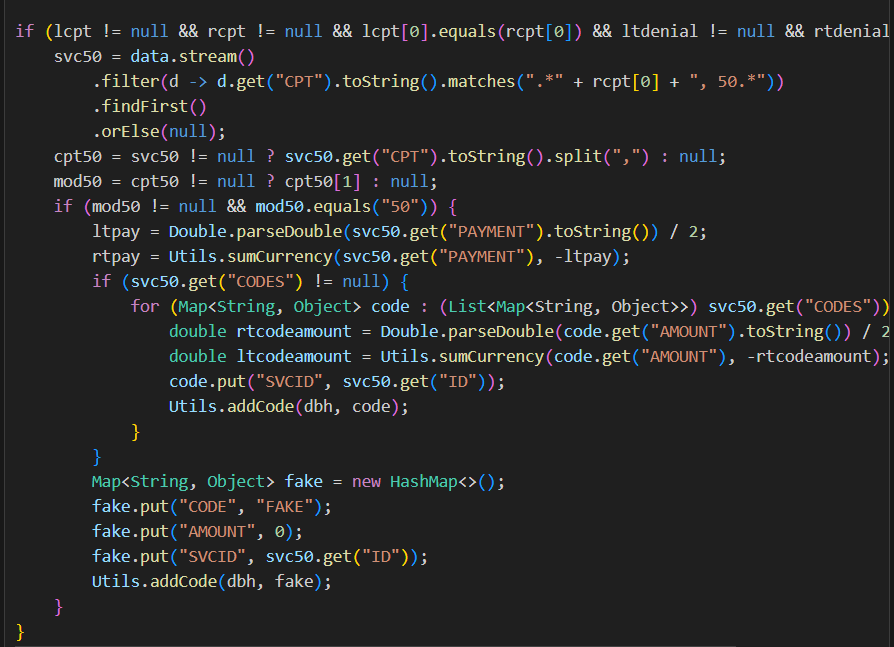
1. Identify the independent code blocks which could constitute a business logic.
2. Translate the identified piece of code into simple English which explains the business logic using healthcare terminologies in a way the business team person would understand.
3. Business team person would review the logic and decides whether it should be migrated or not.
4. Developer goes through the logic & samples remittance data to decide whether this logic should be done as a DMN or a Code after taking the complexity & limitations of DMN into consideration.
5. Complex logics are not usually done through a DMN as the idea of using DMN is to enable Business configurability.
6. The logics which are decided to be coded will be migrated to the 2.0 system as java code which follows a microservice architecture.
7. The entire migration is not done in a lift and shift manner. Instead, the whole code is split to small chunks, reviewed, refactored (if required) and migrated only once the whole approach is confirmed based on the new design.

This updated POC has been done to understand whether we can leverage Github Copilot for code migration. if yes, how much improvement we can expect in the existing development cycle.

**Code Conversion from PERL to JAVA**

1. Level 1 code conversion



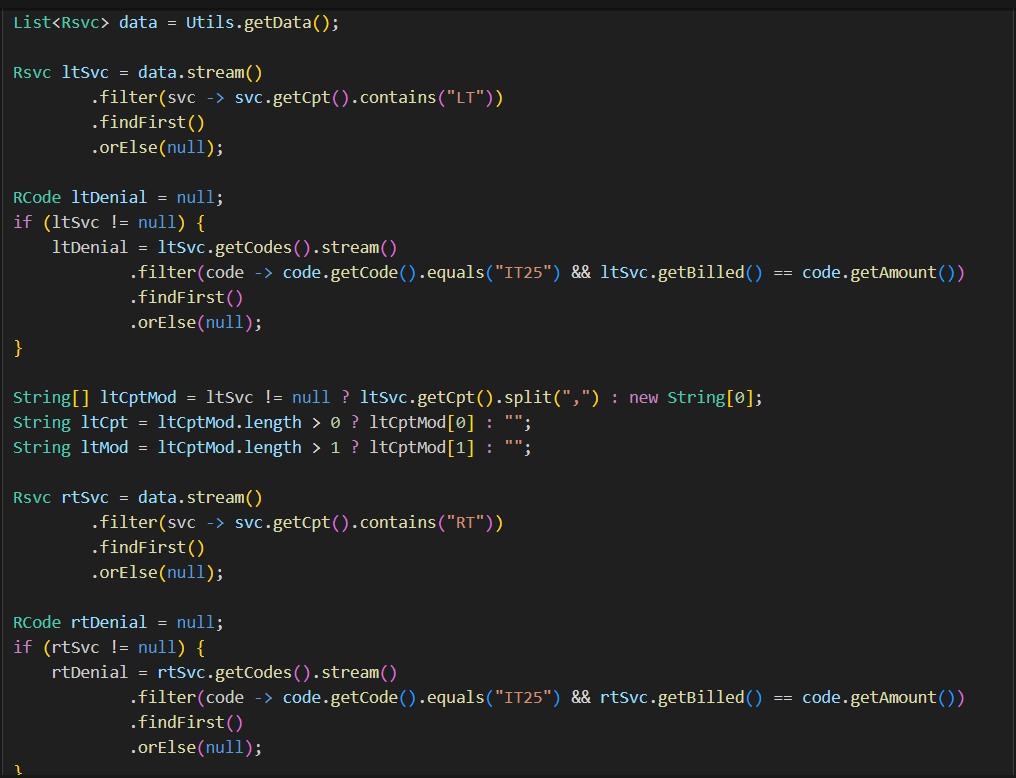
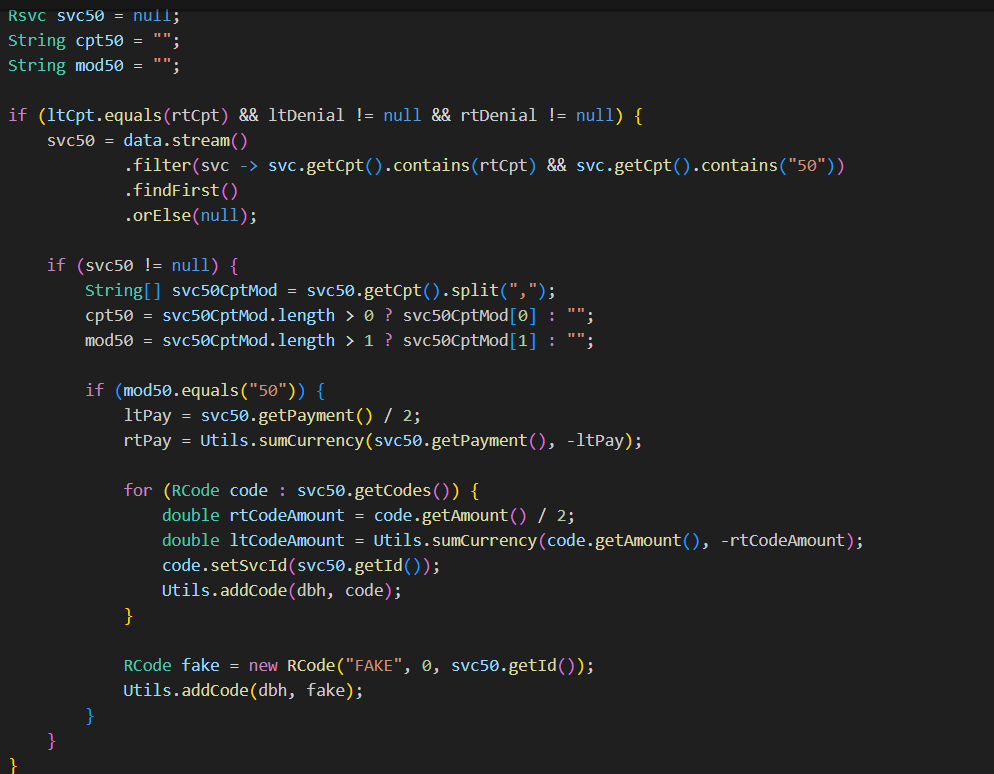
 

**Note :**

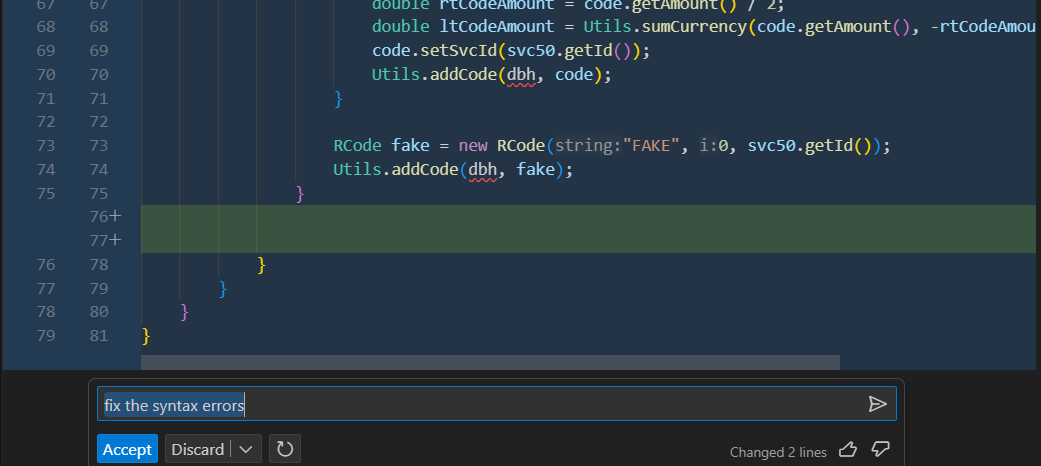
* Converted java code is more like a generic code
* Every attempt generates a different kind of code which sometimes tend to make things more complex.

1. Level 2 Code Conversion : Improved prompt to use specific datatypes.

Prompt : **Do a simple conversion of this code to java in an optimized manner and use the class RCode wherever the object code is being referenced and use Class Rsvc wherever the object svc is used.**

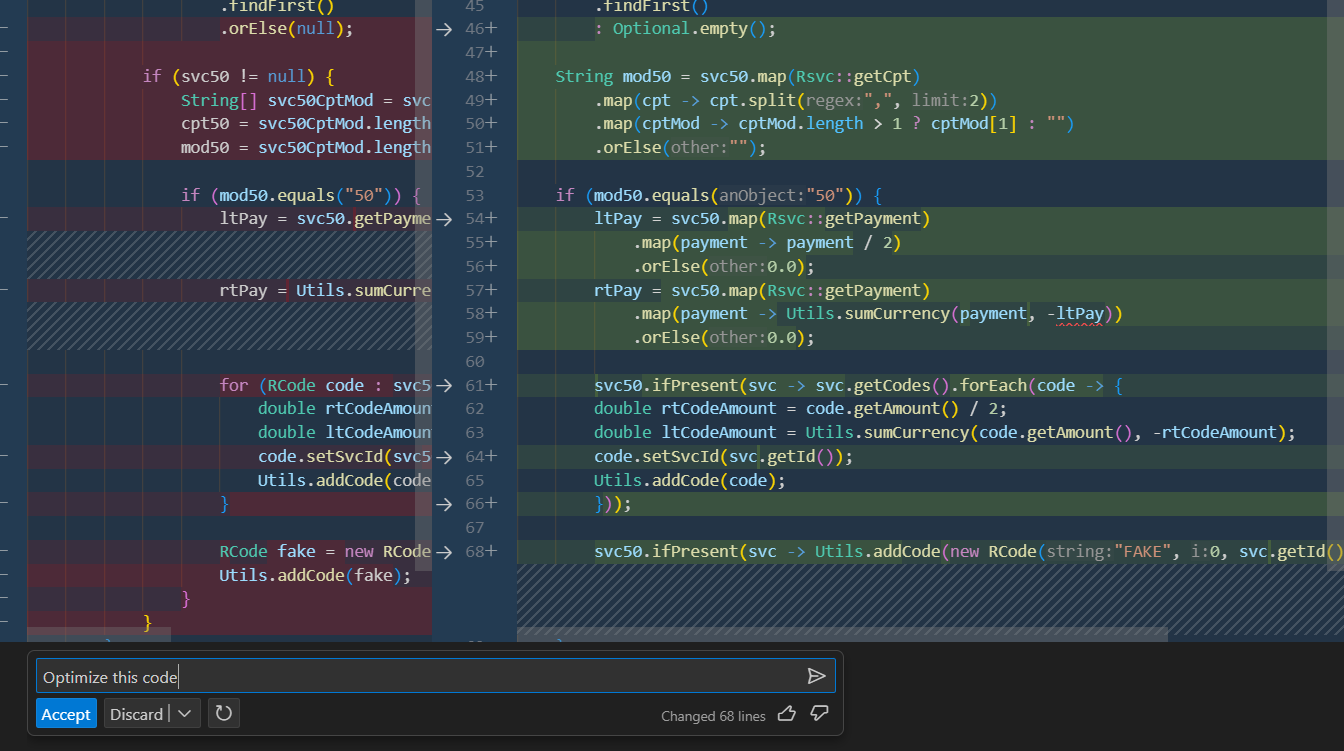
 

**Observation** :

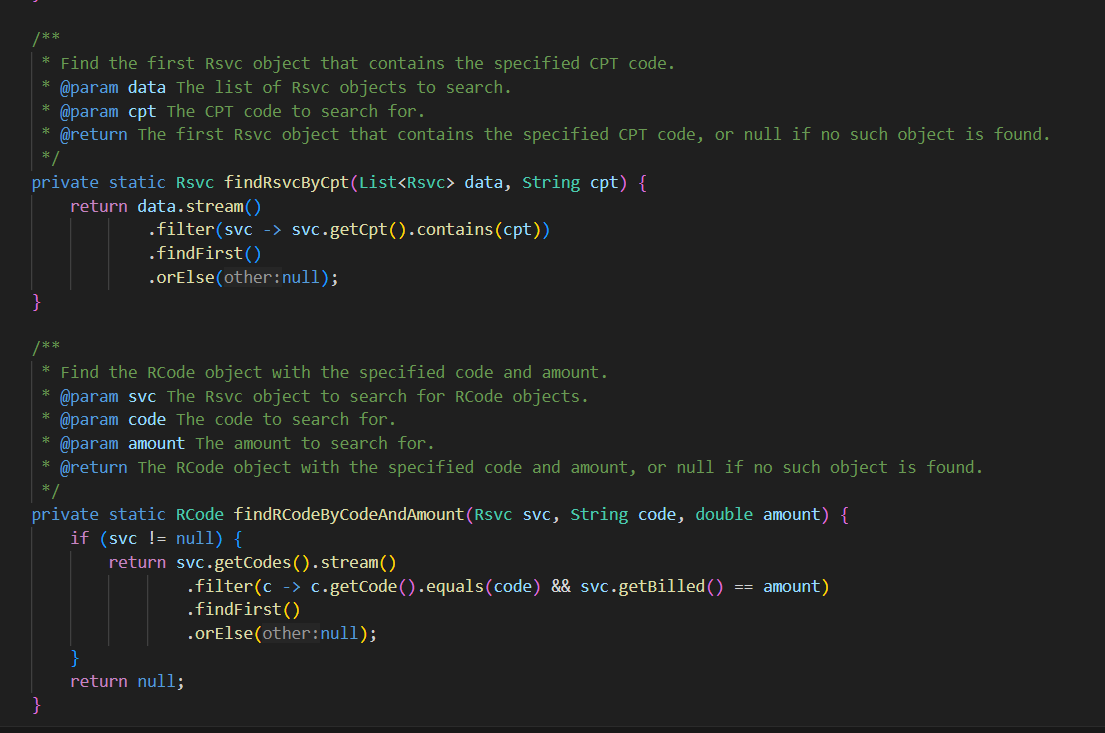
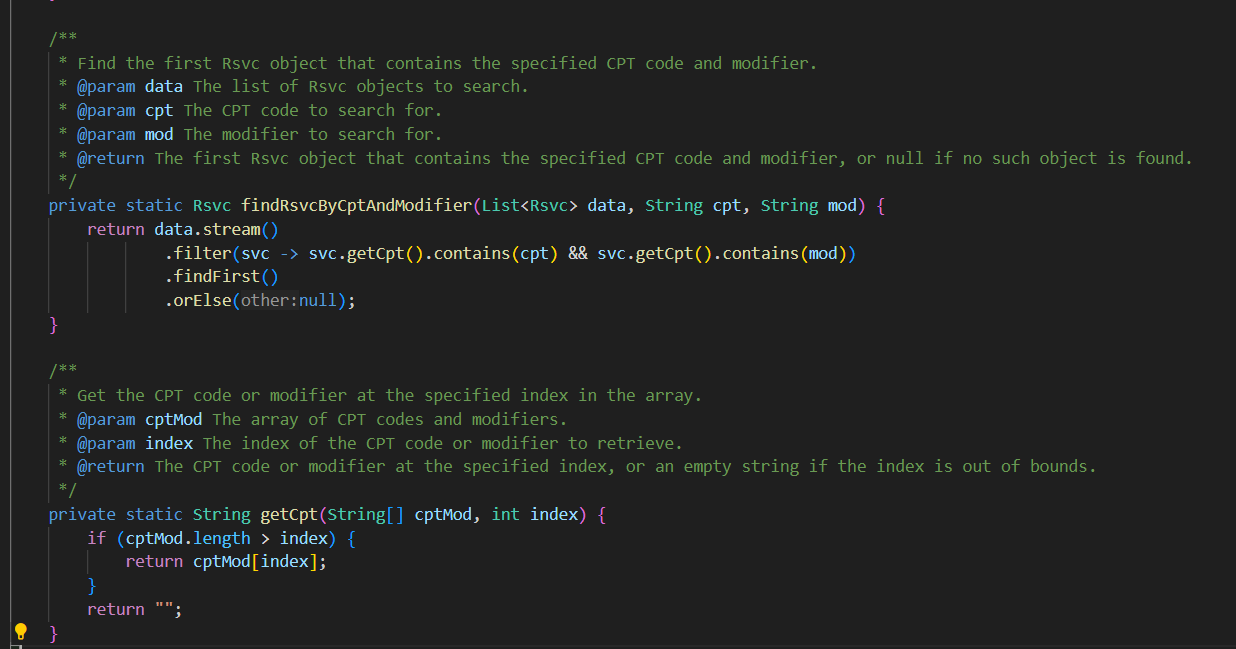
* Created a Util.addCode(RCode code) to check copilot is able to fix the syntax error
* Copilot was not able to fix the syntax errors even after prompting it to do (Code scan not done)  
    
  

1. Prompt to optimize the java code

**Attempt 1** :



**Attempt 2** : Prompted to optimize and use reusable functions wherever possible

**Observation** :

* Didn’t make a significant change in the first couple of tries, Just changed some List into Optional types.
* Upon prompting in a detailed way and mentioning to use reusable functions. Copilot generated the reusable functions along with documentation and used it.

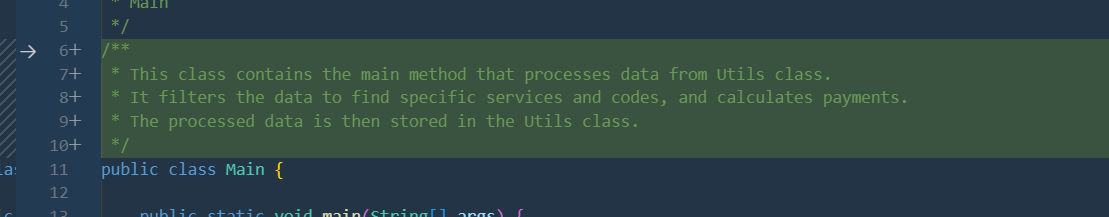
1. Prompt to optimize and remove the unused variables

**Observation** :

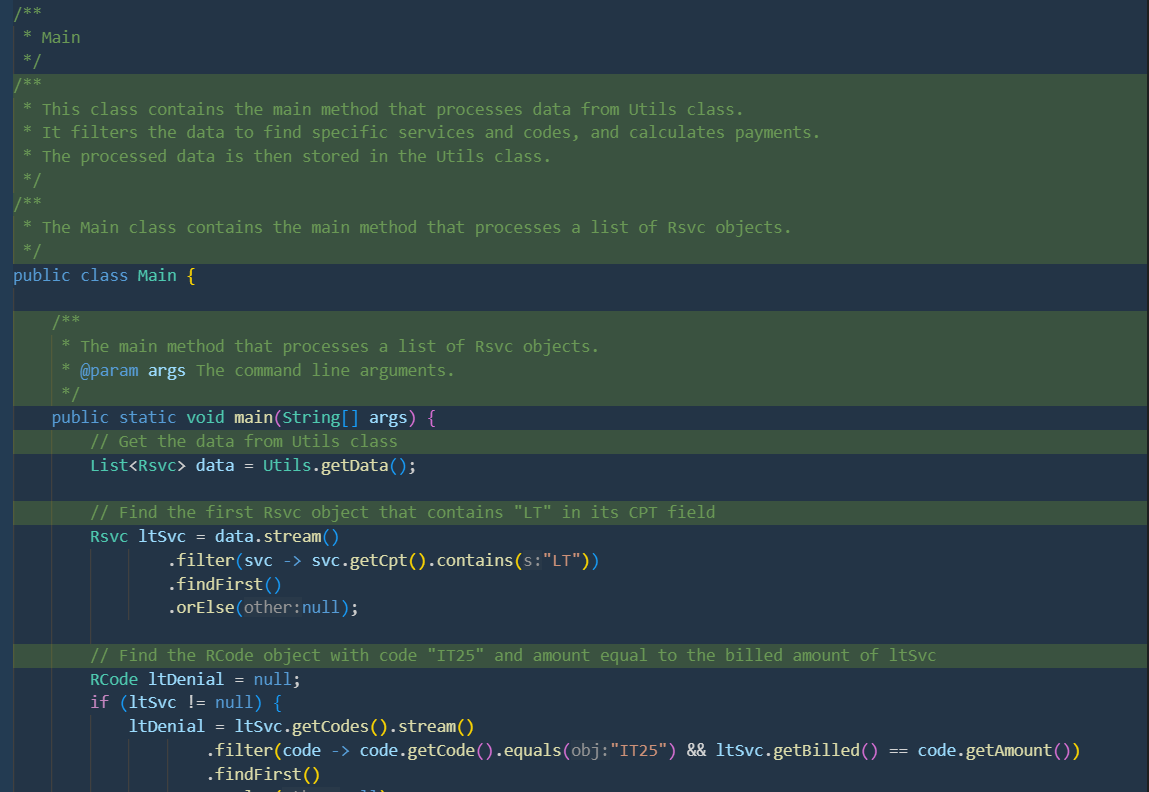
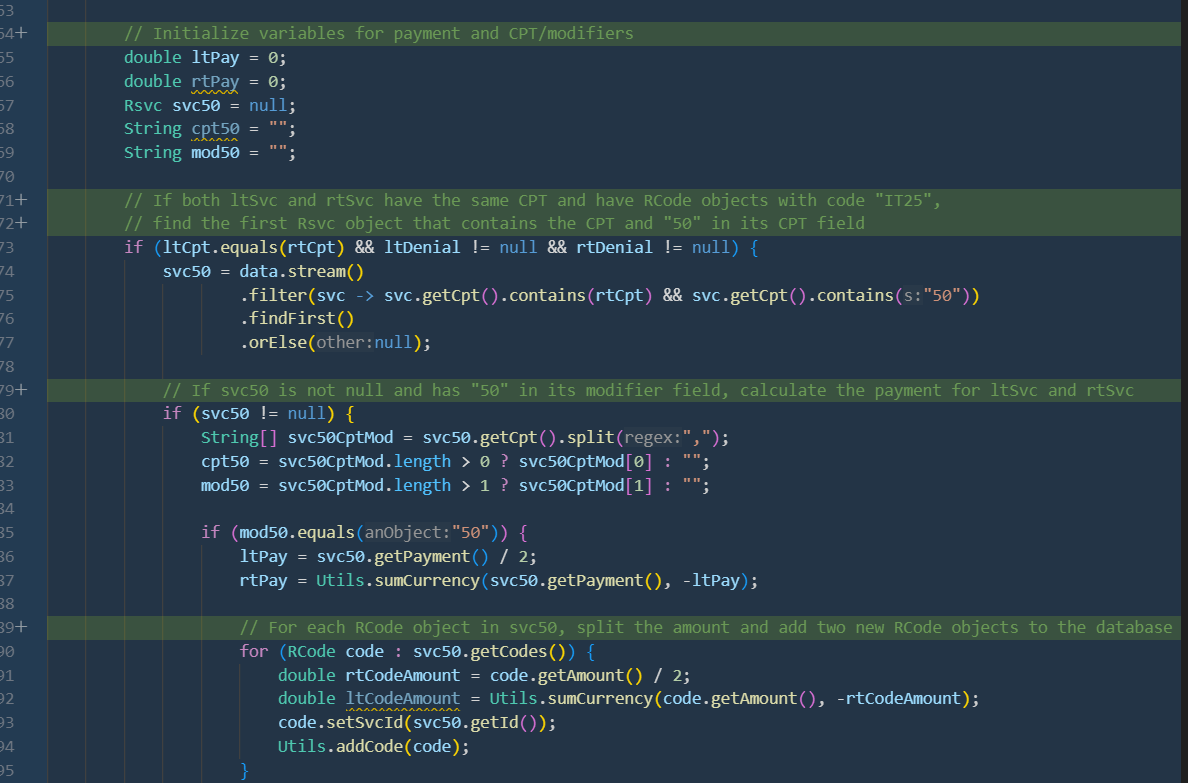
* Outcome was the same as #3
* Unused variables were not removed even though it existed.

1. Documentation of the generated java code

**Attempt 1** :



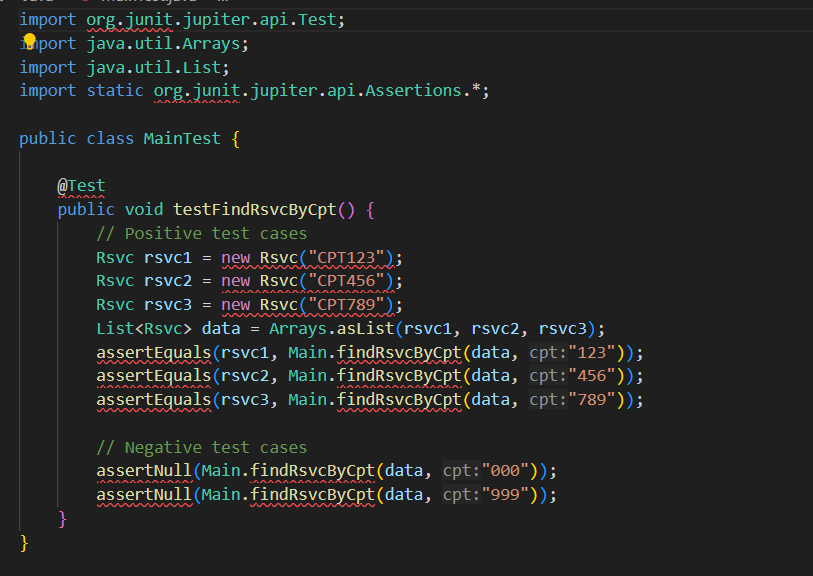
**Attempt 2** :

**Observation** :

* Generated decent enough comments for almost all blocks of code, However the overall documentation was not up to the expectation. It could probably generate a better documentation on further attempts.

1. Junit Test :



**Observation** :

* Generated Both positive and negative test cases for the reusable function.

**CONCLUSION**

1. **Multiple scenarios like optimization, Detailed prompting etc. Were done to understand the behavior of Github copilot for the process of code migration.**
2. **After detailed prompting, it was able to generate code which is somewhat close to how the target code should like however it included many rounds and variations of prompts along with manual fixes.**
3. **Tried to make a syntax fix with copilot prompt However it didn’t recognize it as the codebase scan is not available.**
4. **One major observation is that the user should know how both the code bases are designed in order to use a detailed prompt for copilot to generate code in the desired manner. Generic prompts will generate only generic code.**
5. **We were able to see that copilot is more like an assist the developer and works best for few use cases like Unit test generation, generate code logics which doesn’t involve codebase scans, Code comments etc.**
6. **Its more like stack overflow converted into an AI assistant which could help the developer write code easily.**
7. **Copilot works does not work in a centralized manner. Instead depends on the developer who uses it.**
8. **With the current capabilities of copilot, improvements in developer productivity does not seem to be very significant, rather it would probably be around 10 to 15% which would still vary, subject to detailed prompts and existence of very minimal to no bugs.**