Developing a modern collections library

The research I am aiming to undertake is the development of a modern collections library. Allowing for development to feel more streamlined in the future. There are a few features that I will be considering for development in this language: Sets, Sequences and Maps. The idea is to have a few very dynamic and powerful collections at the fingertips of developers. Additionally, improving the runtime on java’s native collections library. This would help in the development of more efficient java programs going forward.

Usage figures suggest that 56% of all initialisations of a collection library object in java are for some kind of list, with maps and sets being 28% and 15% respectively. This backs up the view that only three collection types are needed. For any others they can be implemented by the three mentioned with relative ease. The library will fix a gap in the java market where there are no small and powerful collections libraries. The libraries may be quick in action when the right method is selected but if the correct methods are hard to find and seldom used, with a very generic method or class used instead, then the benefits of this library are negated.

Another feature that would be useful to implement would be secure and efficient parallel processing of collections. This is done securely by java but not in the most efficient way. “*The simplest solution to ensure correctness is to provide synchronization wrappers as done by Java and C#. However, this approach is rarely efficient and often lacks support for performing multiple operations safely together*”. This would allow for efficient multi-threaded operations to be carried out in the java language something that is currently missing. The java parallel support is secure but slow something that needs to be built on.

The plan for development would be to first get the basic functionality of a collections library completed. Then efficiency improvements can be made, and parallel processing support added. As a general rule the collections should be as efficient as the native java implementation or ideally faster. There would be no point developing something that already has a better implementation as standard. Once the basics are down testing will commence based on the results of these improvements will be made until the algorithm meets the requirements.

One of the ways that the library will be tested is through the conversion of old programs developed by myself to use the new collections library. This will demonstrate its ease of use and efficiency bonuses. It may also demonstrate the ease of refactoring code to use the new library.

A timeframe for completion of the research would be about 7 months from October 2024 to April 2025. The first three months would be spent developing the library. The next two would be spent fine tuning and getting everything up to scratch. The final two months spent writing up results as well as final benchmark tests and finishing touches being made.

The library will be used instead of the built-in java collections library in projects going forwards. Anything that needs parallel processing support and needs to run efficiently will use this library as it will have the features needed to achieve this.

Further research could perhaps delve into code discoverability which seems to be an issue in languages with vast libraries. When there are too many collections to choose from it can become hard to pick the correct one for the specific implementation at hand. So some research into this would be useful in aiding library designers.