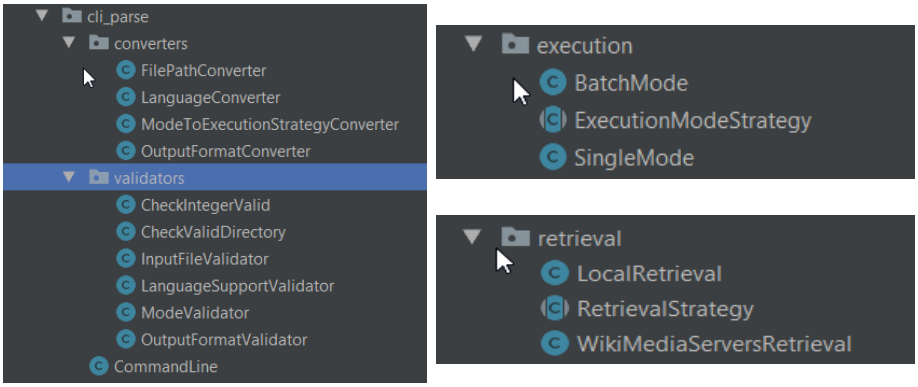
What changed?

Over the course of my architectural change these are the main classes I created and changed to improve the quality and architecture of the system.

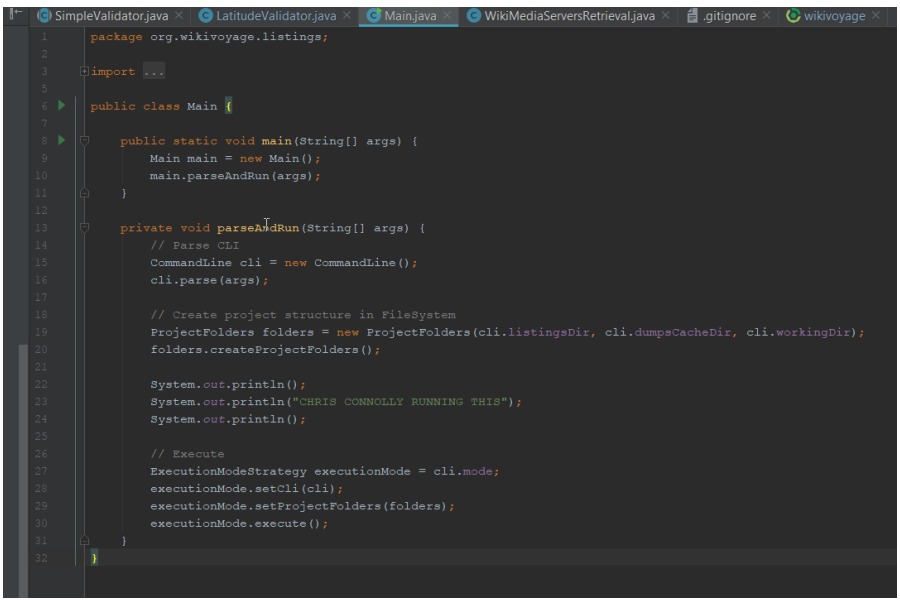


The execution package is completely new. It contains an abstract class called ExecutionModeStrategy which represents different strategies for executing the command line tool. The ExecutionModeStrategy is now implemented by different forms of execution styles, for instance SingleMode and BatchMode. Previously these execution strategies were stored in a long, confusing 'Main' class which made the extensibility and usability of the system difficult to manage.

By abstracting away the execution modes, it makes it much easier to extend the system without worrying about other parts breaking.

This was implemented by using the strategy pattern, where the Main class has a reference to some ExecutionModeStrategy and just calls execute on that strategy and doesn't need to worry about the functional requirements of that execution.

This change did several things, it cleaned up the main class from over 250 lines of code to 30. It improved the execution extensibility and adhered to single responsibility principle – that is, the execution modes now only execute each execution strategy, the main class had too much responsibility and needed to be split.



Another architectural abstraction I created was the retrieval package. I recognised in my architectural essay that there could be multiple methods for data retrieval, be it by connecting to the WikiMedia servers or by connecting to the file system and retrieving data locally this way. Like the execution package, I created an abstract class that represented different data retrieval strategies and recreated the two retrieval methods. This increased extensibility of the system – if we wanted to download from another server this was easily possible with another implementation of RetrievalStrategy.
To do this, I mixed composition and inheritance. Compositionally, the ExecutionModeStrategy contains a RetrievalStrategy so it's easy to change what retrieval method is used. RetrievalStrategies are currently implemented by LocalRetrieval and WikiMediaServersRetrieval.

Another change I made was to make rebuild the command line parsing system to make the software follow Martin Fowler's principle of 'failing faster'.
In the original implementation of the command line parsing, they were doing no input validation until much later in the execution of the system. For instance, only certain supported input files like .xml or .bz2 are allowed to be used. Before my change there was no validation occurring on the input file types until the data had been downloaded and retrieved and parsing was going to occur on the input file (it could've been an .exe!). I added validation checking and type conversions (under the new cli_parse package) which makes the system fail faster. As Martin Fowler says, 'failing faster sounds like it would make your software more fragile, but it actually makes it more robust. Bugs are easier to find and fix, so fewer go into Production'.
So hence with these changes I improved the robustness and reliability of the architecture – reliability to fail that is.