**Pseudocode for Data Extraction (Code A)**

1. Define a package named "org.example".

2. Import the required classes.

3. Define a class named "CodeA".

4. Define a public method named "fetchAPI" that throws an exception.

5. Declare three String variables: "keywords", "apiKey", and "httpsUrl".

6. Assign appropriate values to the "keywords", "apiKey", and "httpsUrl" variables.

7. Create a new URL object named "myurl" using the "httpsUrl" variable.

8. Create a new BufferedReader object named "in" by calling "openStream ()" method on "myurl" and wrapping it in an InputStreamReader.

9. Declare a String variable named "inputLine".

10. Create a new StringBuffer object named "response".

11. While there is input available from the BufferedReader, read each line of input into the "inputLine" variable and append it to the "response" StringBuffer.

12. Close the BufferedReader.

13. Create a new instance of CodeB class named "codeB".

14. Call the "dataProcessing" method on the "codeB" instance, passing the "response" StringBuffer as a parameter.

15. Create a new instance of CodeC class named "codeC".

16. Call the "transformData" method on the "codeC" instance.

**Flowchart of Transformation Engine**

**Diagram

Description automatically generated**

**Explanation of Flowchart**

Before the function transformation data is called we would already have fetched the data from the API and processed it to store in the files with utmost 5 article in each file. After that we would read and clean all the articles from the files by removing URLS and emoticons from the title and content part of the files with the help of regex. Each article consisting of title with its respective content will be stored in the list of articles. After that we will push the data as a document in the MongoDB database, myMongoNews we made.

**Note**

Once the program has been executed it will successfully execute Code A, Code B, Code C without the manual creation of files and cleaned and transformed data would be stored in MongoDB database. But when you run the program again it would again fetch the data from the API and store it in the existing file, hence duplicity of data in file. I could have designed the program to delete all existing files of data in APIData folder, but one wouldn’t be able to see the data again even after the data is successfully stored in MongoDB database.