Two source files: ***source\_to\_text.py*** and ***tokenize\_method\_body.py***

Modules of ***source\_to\_text.py:***

* read\_keyword\_ops(keyword\_set\_file): build a set using keywords to be ignored, which is later used for ignoring the words
* build\_op\_dict(op\_dict\_file): build a dictionary with operators and the text they need to be replaced with
* build\_abbr\_dict(abbr\_file\_path): build a dictionary with abbreviations and the expansions they need to be replaced with
* read\_pickle\_input(input\_file\_path,output\_file\_path): read the inputs from the pickle file and convert them to plain text, one by one, eventually writing them as a dictionary with their corresponding comments into the output pickle file
* camel\_to\_snake(input\_string): substituting camel casing with snake casing
* parse\_input(input\_method): read the method body and break it into three parts: (1) function declaration; (2) function parameters; (3) function body
* parse\_method\_decl(func\_decl): take the function declaration part (before arguments) and removes static/final as well as access specifiers. Also calls clean\_header() method.
* clean\_header(header\_dict: removes unnecessary symbols like angular brackets and splits function name into separate words, replacing abbreviations.
* replace\_abbr(abbr\_str): replaces abbreviations in function header by their expansions
* expand\_params(func\_params): remove tokens that need to be ignored

Modules of ***tokenize\_method\_body.py:***

* split\_statements(method\_body): convert each statement into a line in the method body with same steps: replacing abbreviations, substituting operators and removing keywords to be ignored
* alter\_for\_loop(for\_statement): change “for” loop into text by splitting the initialization, condition checking and incrementing sections.

How to run:

1. Modify **ignore\_token\_list.txt** to add tokens (each token on a new line) that need to be ignored in the Java source code
2. Modify **operator\_list.txt** to add operators that need to be translated into text. Each operator on a new line in the following format: ***<operator>:=<operator\_substitution>.*** Eg: ***&&:=and***.
3. Modify **abbreviations.txt** to add any abbreviations and their corresponding expansions, in the following format: ***<abbreviation>:=<abbreviation\_substitution>***
4. Modify **source\_to\_text.py** to take in the above parameters (hard-coded for now). Scroll to the bottom of the file and modify the parameters in quotes. Change input/output file:
   1. Modify arguments of *read\_pickle\_input(<input\_pickle\_file>,<output\_pickle\_file>)*.
5. Run ***source\_to\_text.py***.

Some Suggested sFuture Improvements:

1. Nested **if-else** statements can be converted to a form wherein each line runs along the format:

*If < first\_condition> then follow the chain: If<first\_condition> then <first\_action> else if….*

1. try-catch blocks can be retained if they are the only constructs in a method:

*check regex to find pattern* ***try {\*} catch(\*){\*}*** *and replace with “runs statements that perform operations such as “\*\*\*\*” and on finding exceptions does “\*\*\*”*

1. Nested while/for loops can also be identified and then expanded as per the original for-loop implementation recursively.
2. Improving to expand type descriptors with reference to Java documentation for Java libraries