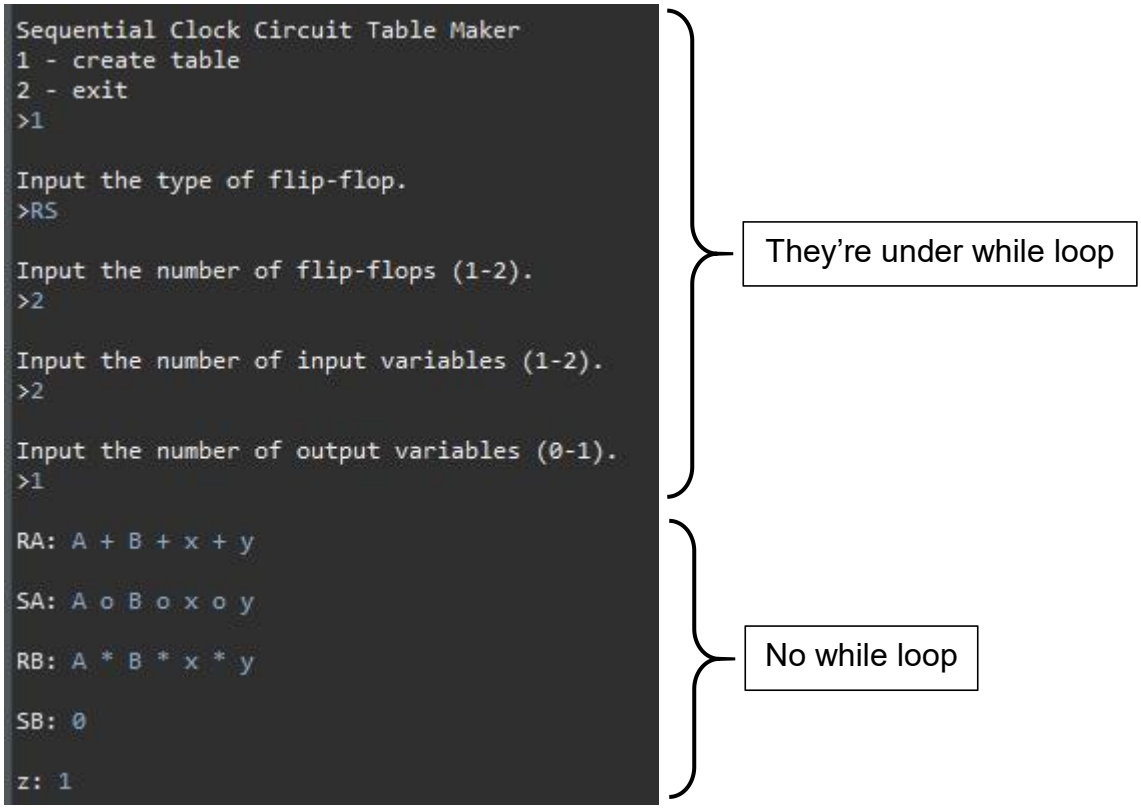
The code is written in Java starts with some variables such as these to be used later on the program. “ff” stands for flip-flop “uc” stands for uppercase.

```
public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        MoreMethods mm = new MoreMethods();

        String choice;

        String ff;
        String ff_uc;
        int ff_num;
        int in_num;
        int out_num;

        int table_length;
        int table_height;
```

Under a huge while loop (meant to handle incorrect inputs, and prompts you to input again) are the choices for (1) making the table which is the main part of the code, (2) terminating the program that exits the while loop or incorrect input prompting you to input again. If (1) is selected, multiple while loops are used for every part where you have to input something except for the part where you input the expressions for the flip-flop input function where if there’s a mistake, an error occurs and the program terminates. Refer to the image below:



A table is also created depending on the number of flip-flops and variables. An additional two rows are created for labeling such as:

				and	J	K	J	K
A	B	x			A	B	A	B

Below that are the 1's and 0's. The minterm binary is made for the first few columns (again depending on the number of flip-flops and variables).

After that, the boolean calculation happens per row. Firstly, the expression converts all the letters into either 1's or 0's depending on what row they're in. Next, the compliment is taken first by using "takeCompliment(String)" from MoreMethods.java before evaluating the string. After that the string is evaluated using "evaluate(String)" from EvaluateString.java which converts inputted expressions and outputs and number. I modified that code a bit for "or" and "and", and added another symbol for the "xor" operator which is the letter "o". After that, it's placed in the table and then moves to the next row. If that entire row is done, it moves to the next column for the operations. If there's an output variable, it goes to the last column and then operates per row.

After that, the table is almost done, it just needs the next state. Depending on the type of flip-flop, it will check results of the flip-flop input functions per row, and then out put it in the next state until the last row. Now the table is complete, and it is printed using “printTable(char[][])” from MoreMethods.java and the menu appears again in which you can make another table or exit. This the table from input functions above.