The project deals with designing the interpreter for the project language after parsing the program given as an input and tokenizing it.

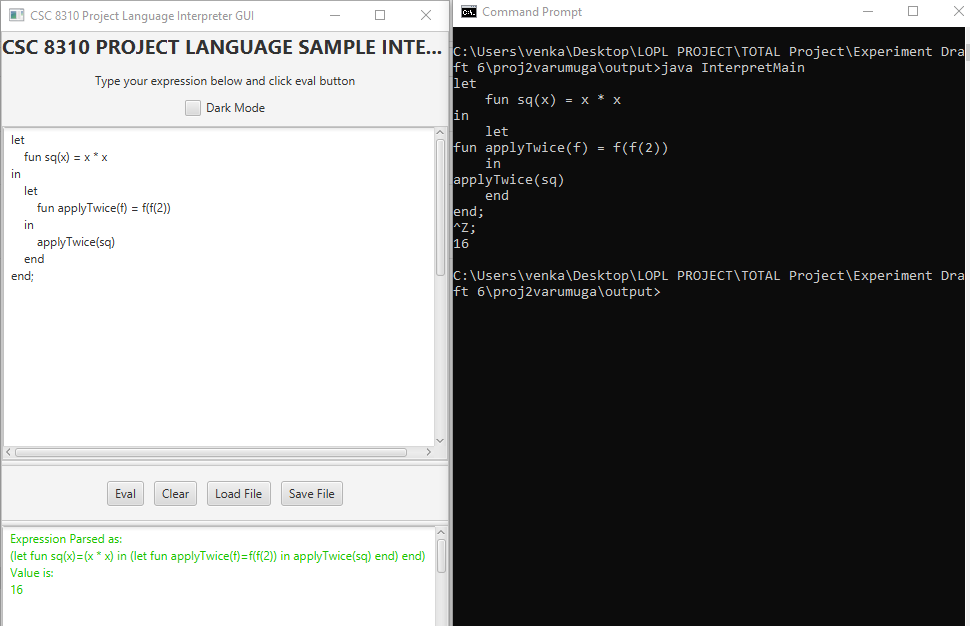
The following are the steps involved in the process:

1. The ProjLang.jj, javacc file initially only parsed the program given as an input and appropriately tokenized it. The only information it gave was if the parse was successful or not. It’s return value was void. This javacc file is now modified and a special feature called **actions** are included. Actions are fragments of java code embedded inside “{“ and “}” and can be used to perform operations on the values returned by the functions described in the non-terminals in the productions or on some of the data inputs themselves. They are combined and held together as an object whose class would be the respective non-terminal. Thus, the return value of each production is an object, inside which data members are either values or objects of other functions that have been returned by other productions. The objects can be of type **Expr,** which describes the respective program constructor **Value,** which describes the data type.
2. Since the program contains various program constructs like Assignment, If Expressions, While Expressions and also due to the fact that each production will return an object, the final return value of **input(),** the initial function that invokes the entire parser and returns its result, will be an object. The object so developed, in which all other constructs of the program are held as objects is called as an **Abstract Syntax Tree.**
3. This abstract syntax tree is fed into the process of evaluation. Now, the steps for the evaluation is outlined in a method called eval(), which is specific to every class. It always returns a variable of type **Value.** As every object is evaluated, other objects embedded inside it is unwrapped and evaluated. The final evaluated output(the result of the program) will be of type **Value.**

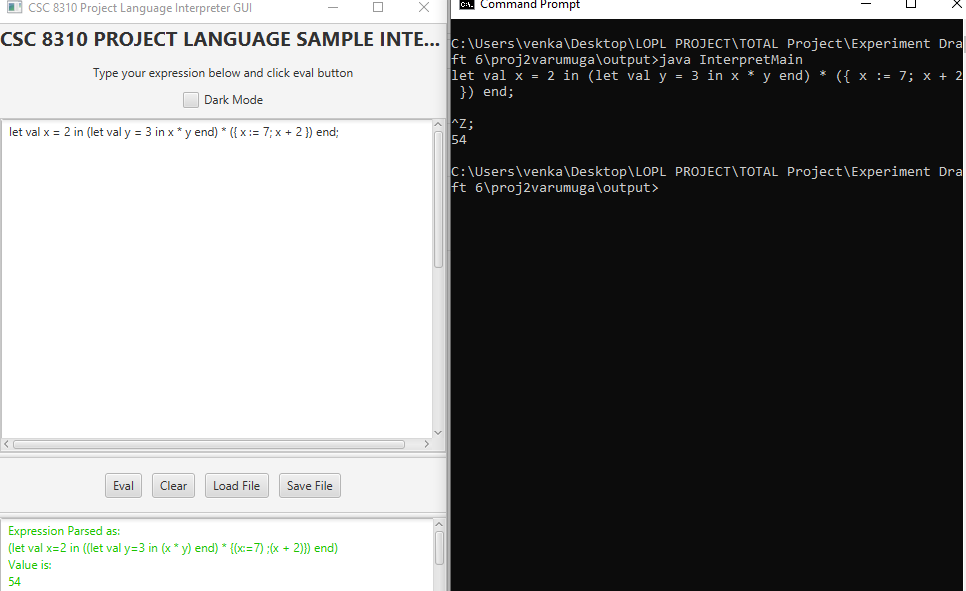
**Sample Test Cases and results:**

Following are some programs that have been executed in the Sample Interpreter pre-supplied and the Interpreter that has been designed.

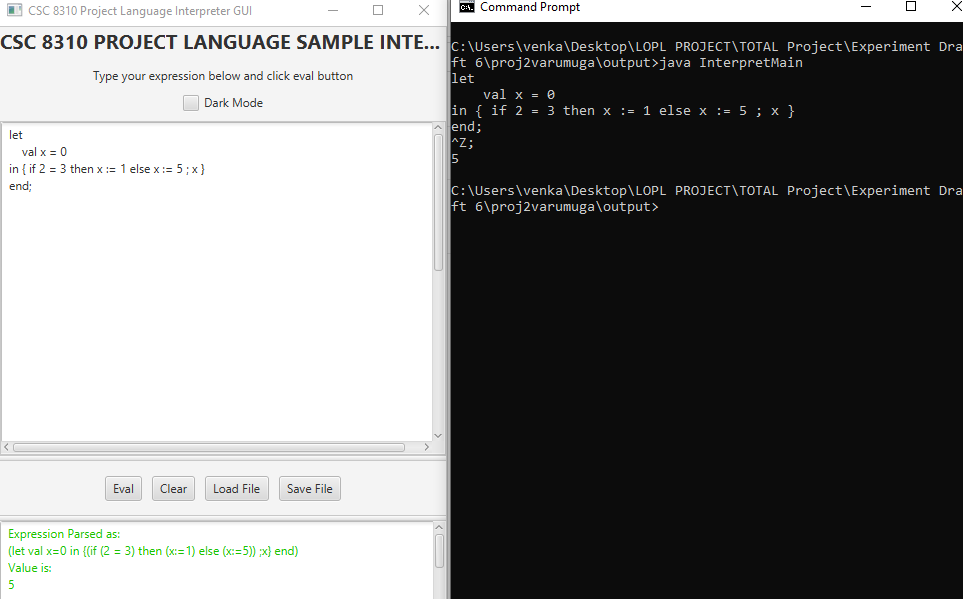
Example 1:



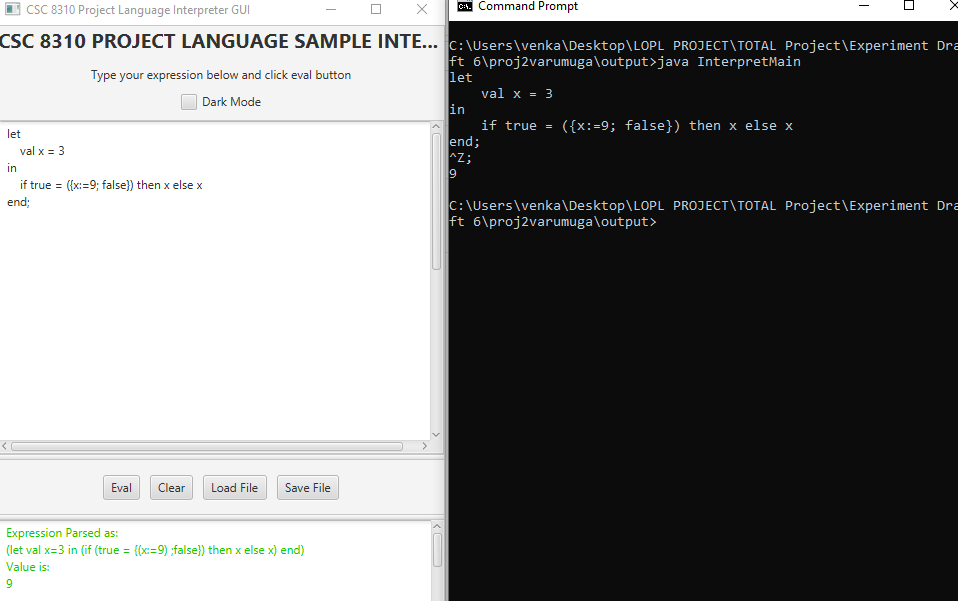
Example 2:



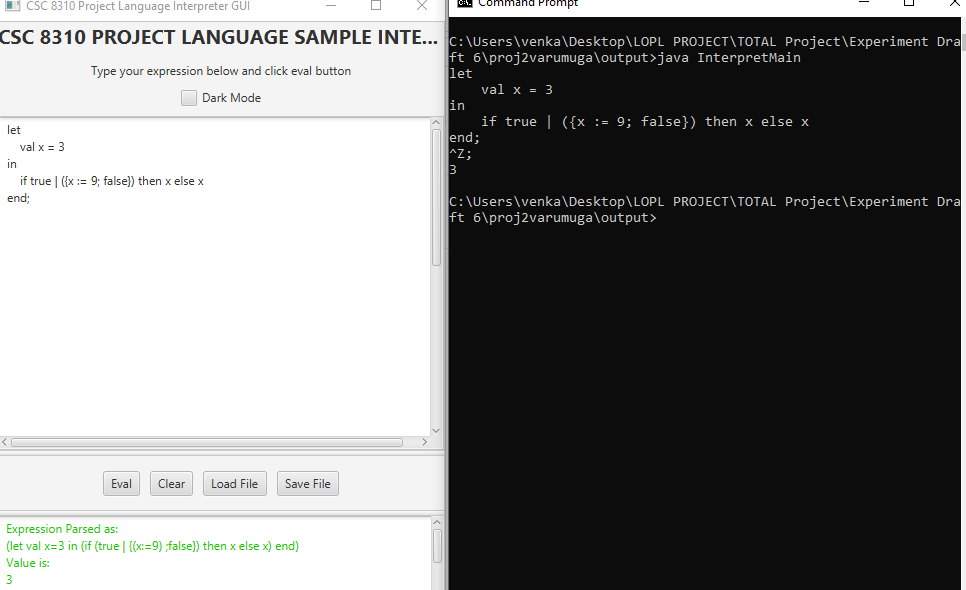
Example 3:



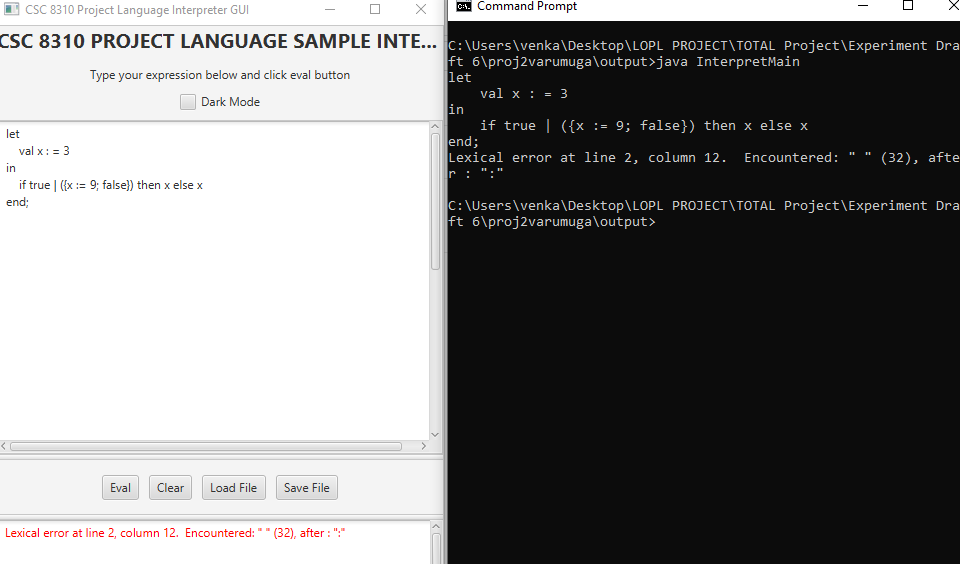
Example 4:



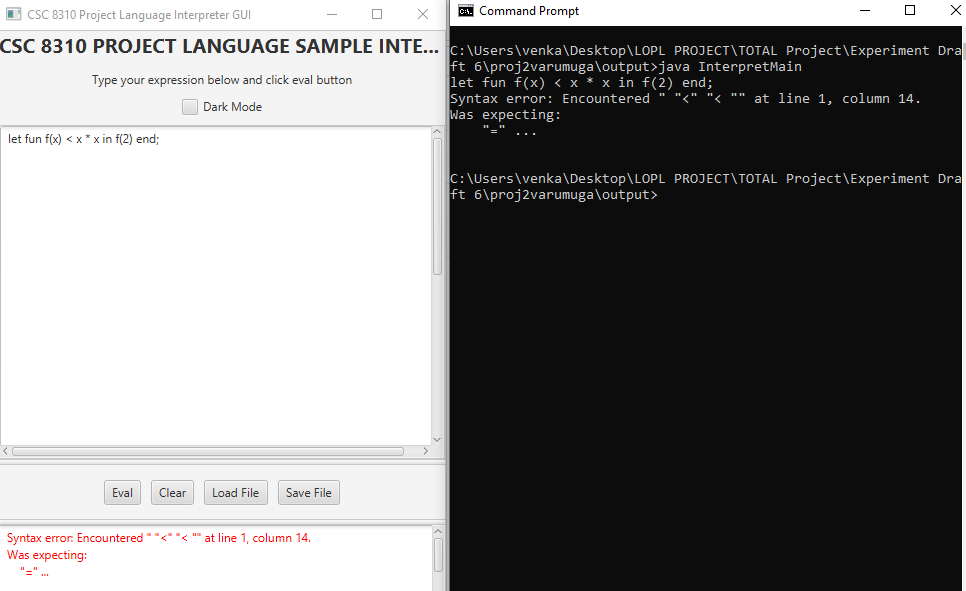
Example 5:



Example 6:



Example 7:



Example 8:

