# BizTalk File Transfer

**Function**
Receive a file of at most 2 mb from a location and post it to another location.
**Description**
Upon receiving the source location, system shall assign action an id.
No two actions have same id.
Log shall record <id+descriptinon of task+user id>.
System shall periodically check the source location.
At every check system shall log this action, i.e., write ("check" +<time of check>).
If file is dropped to the location, system shall check the Rules (NULL for this case).
System shall report this action on the log. After this, system shall send the file to destination.
Either system accomplish the task or fail, it should record "success"+<time of job id> or "fail"<time of job id> message to log.
**Inputs**
File source location, File destination location, rule.
**Source**
Input parameters on GUI.
**Outputs**
Success or Fail.
**Destination**
Main control loop.

**SAT Solvers**

Sat is the problem of deciding whether a particular Boolean formula in some normal form has an assignment to its terms that makes the formula true. SAT can be summarized as follows: given a list of clauses, determine if there exists an assignment that satisfies all of them simultaneously. SAT instance is a set of clauses, and each clause is a set of literals. A literal is a variable that is either negated or not.

The problem with this approach is that we will not be able to quickly look up variables, and checking to see if a literal is negated or not, and negating it if not, would be rather slow string operations. SAT is NP-complete. In a way SAT provides a generic ‘language’ which can be used to express NP complete problems in verification, scheduling.

A SATsolver is used to search for solutions to the problem. The output of a SAT solver only needs to answer true (SAT) or false (UNSAT and therefore proved unsatisfiable) depending on the satisfiability of the formula.

Brute force approach is not good, it provide slower solution. Brute Force is truth table approach . Assign all possible values to the different literals and see whether there is one assignment which is SAT. We want to perform a more intelligent search. So It was not used Brute Force Solution, David Putnam resolution based SATsolver was implemented.

**SAT Solvers Input and Output**
The input to the SAT solver is a Boolean Propositional formula in some normal form (ex. Conjunctive Normal Form)
Ex. $(a \vee \neg c) \wedge (\neg a) \wedge (b \vee c \vee \neg a)$
a, b and c are variables
A literal is either a +ve or a –ve instance of a variable
A CNF formula is a conjunction of one or more clauses, each of which is a disjunction of one or more literals.

The output of a SAT solver only needs to answer true (SAT) or false (UNSAT and therefore proved unsatisfiable) depending on the satisfiability of the formula.

**Description of the Code**

The program consists of four classes:

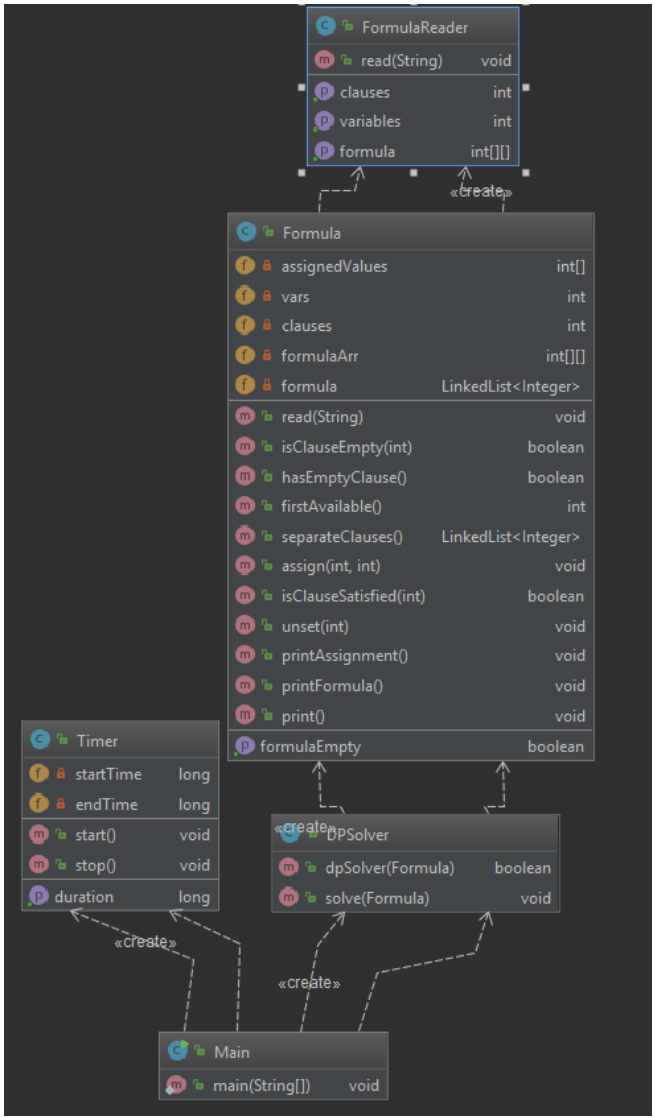
**FormulaReader** reads the file, maintains the number of variables, the number of clauses and the formula. Finaly and passes it to the Formula class.

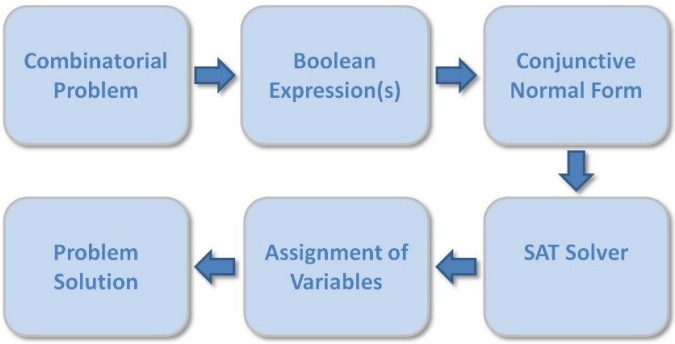
**Formula** reads data according to structures such as variable, clauses and checks for empty or element status. it is also the class that controls the clause format.

**DPSolver** includes the Davis-Putnam algorithm implementation for SAT solving. The "solve" method solves the given formula with the mentioned algorithm and prints the variables that solves the CNF expression given in the formula, or states that the formula is unsolvable.

**Timer** calculates the runtime of the program in seconds

**Main** produce a positive or negative result using other classes and write the working time on the screen




Example

```
c   DIMACS representation:
p cnf   3   2
    1   2   0
    -2  -3  0
```

Example

```
The formula is satisfiable with the following variables!
Assigned Variables Array: [0, 1, 1, -1]
Total SAT Execution Time = 0 seconds!

Process finished with exit code 0
```