Clarity - Backend code challenge

This challenge is meant to test your software development skills.

You have 24 hours to reply with your answer, but we expect these challenges to take no more than 3 hours. We don't take into account how long it takes for you to respond, ie 2 hours or 22 hours are considered equal. Your answer will be evaluated based on:

- Correctness, does it work?
- Is it tested?
- Is it well-designed?
- Would this code be easy to extend or maintain?
- Is the code easy to comprehend?
- Performs well?
- Is the code efficient resource-wise?

We want to see your programming skills, avoid the use of frameworks like Spring Boot, Spring, Spark, etc, it is not needed to solve the problem. You should avoid using any libraries or frameworks except for the language standard library. Also, please feel free to consult any relevant documentation.

Think in Clean Code, Clean Architecture, Testing and simple solutions that works!!

Also, be sure you send something that we can compile and test easily.

Challenge 1: Equation Solver

A file contains newline-terminated simple equations formatted like:

```
varname = [ number1 | varname1 ] { # [ number2 | varname2 ] }  { # [ number3 |
varname3 ] } ...
```

For example:

```
var = var1 # var2
res = x # 7796 # y # 7486 # 5924
temp = 1977
cons = 169 # 7667 # 3195
```

Each line represents an assignment (=) that sets a value to the var *varname*. The sign # represents a commutative and associative operation (ex: addition, multiplication, ...) that will be resolved in run time. The lines appearing in the file, do not follow any particular order.

Goal to Achieve:

Based on the code provided, develop a program that finds the value of a specific variable. The program must also find out when the solution is impossible to be reached.

Some examples:

Below you can find several examples considering that # is the addition operator. (# meaning +)

example 1: solve `a` in this equation

```
a = 1 # 2
```

solution: a = 3

example 2: solve `forwardResult` in these equations

```
myvar = 1 # 2
forwardResult = myvar # 2
```

solution: forwardResult = 5

example 3: solve `backwardResult` in these equations

```
fun = last # 2
backwardResult = fun # last # 3
last = 1
```

solution: backwardResult = 7

example 4: solve `z` in these equations

```
x = y # 2
z = x # y # 3
```

solution: z = ? (no solution)

example 5: solve `look1` in these equations

```
look1 = 2 # look2
look2 = look3 # 99 # 12
look3 = 1 # look1
```

solution: look1 = ? (no solution)

About the Program Template

This is the template for your to use to build your program. It provides you with the application structure in order that you can focus on the core of the equation solver.

Additional, the included jar `operator.jar` is used by the class `SolverCommand` to run the code. The jar contains a method wrapping the XOR operator, but your code must abstain from making direct references to any specific operator and instead making use of the interface `Operator`.

How to get the help about the program

```
$ ./run.sh --help
Usage: run.sh [-hV] -f=<fileName> -v=<targetVariable>
Variable solver
  -h, --help              Show this help message and exit.
  -V, --version           Print version information and exit.
  -f, --file=<fileName>   The name of the file.
  -v, --target-variable=<targetVariable>
                          The variable to solve.
```

How to parse a given file

example:

```
$ cat input-example.txt
varA = 55 # varB # 2
varB = varC # 99 # 12 # 256 # 1000
varC = 5

$ ./run.sh -f input-example.txt -v varA
09:40:06.375 [main] INFO   ai.clarity.challenge.SolverCommand -

Clarity Code Challenge
    file name: input-example.txt
    target variable: varA

09:40:06.389 [main] INFO  ai.clarity.challenge.SolverCommand - SOLUTION:
09:40:06.390 [main] INFO  ai.clarity.challenge.SolverCommand - varA = 646
```

Instructions on how to implement your solution

Below you can find the things that you are expected to do:

- Change the method `Solver.solve` to implement your solution. You can add more classes but always under the package `ai.clarity.challenge.solver`
- You should not need to modify any class outside the package `ai.clarity.challenge.solver` unless you find a mistake. In such a case, let us know explicitly.

- A basic test class `SolverTest` is provided but consider to include more tests if you find it convenient.
- The folder **examples** includes a wide range of input files to help you test your code.
- In addition, provide any explanation that you believe is required to understand your code.
- One recommendation: Do not forget to verify the edge cases like long files, some mal-formatted lines or corner-cases.

Challenge 2: Object Model with Java Classes

It is required to develop an application for a restaurant. In particular, we need you to **create the set of DTO java classes** of the application.

Requirements

The application should take care of the available meals, their price and taxes. The restaurant can have different types of offers like a daily menu, weekend menu or happy hour. The manager of the restaurant should be able to see detailed information about the items sold, like item cost, tax, quantity, etc.

Design an Object Model with Java Classes for this application that:

1. Allows managing orders: order meals, drinks, menus, mark orders as paid, generate bills, etc.
2. Provides real time control of the stock: when the restaurant runs out of a specific meal, refill stock, etc.
3. Provides aggregated data (i.e. weekly) of the restaurant accountability: aggregated revenue, costs, Taxes, offers ratio, etc.

Try to be as detailed as possible with each requirement. Requirements are presented by priority. Make sure you've got a good design for one of the requirements before starting with the next one. We would prefer 2 points well-designed over having a soluition for all of them but unfinished.

Please focus on the software design rather than the high level architecture, but if you feel like (and have time), a small description of your architecture proposal will always be welcomed.

Please feel free to make assumptions as necessary with the proper documentation.

Good luck !!!