Practical work on the simplex method

The work consists in modifying a class named `Dictionnary.java` which models a dictionary. The attributes of this class used to code the problem are:

```
int nbBasicVar;
int nbNonBasicVar;
int[] arrayBasicVar;
int[] arrayNonBasicVar;
double[][] D;
```

In the arrays `arrayBasicVar` et `arrayNonBasicVar`, **we do not use the boxes of index 0**.
The array `arrayBasicVar` has dimension `nbBasicVar + 1`.
The array `arrayNonBasicVar`has dimension `nbNonBasicVar + 1`.
The matrix `D` has dimension `(nbBasicVar + 1)` × `(nbNonBasicVar + 1)`.

The problem data is read from a file and corresponds to a problem in standard form. In order to work, it is necessary to understand the coding used.
If the problem is (whose encoding is in pb1.txt):

maximize $z = 4x_1 + 3\ x_2$

with

$2x_1 + 3x_2 \leq 24$

$5x_1 + 3x_2 \leq 30$

$x_1 + 3x_2 \leq 18$

$x_1 \geq 0, x_2 \geq 0$

then the file contains (first line: the number of variables then the number of constraints other than the sign constraints, following lines: the coefficients then the constants of the constraints, last line: the coefficients of the variables in z; at the beginning, if the variables in z are null, z is also null):

```
2 3
2 3 24
5 3 30
1 3 18
4 3
```

The first feasible dictionary is:

$x_3 = 24 - 2x_1 - 3x_2$

$x_4 = 30 - 5x_1 - 3x_2$

$x_5 = 18 - x_1 - 3x_2$

$z = 0 + 4x_1 + 3x_2$

and, after initializing the program variables (initialization from the file is already programmed):

`nbBasicVar = 3`
`nbNonBasicVar = 2`

`arrayNonBasicVar` has dimension 3 and contains at indices 1 et 2 the numbers 1, 2 (for the variables x_1 et x_2, which are initially the non-basic variables)

`arrayBasicVar` has dimension 4 and contains at indices 1, 2 and 3 les numbers 3, 4, 5 (for the variables x_3, x_4, x_5, which are initially the basic variables)

The matrix `D` has dimension 4 × 3 and contains :

```
 0  4  3
24 -2 -3
30 -5 -3
18 -1 -3
```

After pivoting by entering the variable x1 and leaving the variable x4, that is, entering the variable of index 1 in `tabNonBasicVar` and leaving the variable of index 2 in `tabBasicVar`, the new dictionary must be:

$x_3 = 12 + 0{,}4\ x_4 - 1{,}8\ x_2$
$x_1 = 6 - 0{,}2\ x_4 - 0{,}6\ x_2$
$x_5 = 12 + 0{,}2\ x_4 - 2{,}4\ x_2$
$z = 24 - 0{,}8\ x_4 + 0{,}6\ x_2$

The coding must then be:
`arrayNonBasicVar` contains at indices 1 et 2 the numbers 4 et 2
`arrayBasicVar` contains at indices 1, 2 and 3 les numbers 3, 1 et 5
The matrix `D` contains:

```
24 -0,8  0,6
12  0,4 -1,8
 6 -0,2 -0,6
12  0,2 -2,4
```

With this coding, one step is to look in `arrayNonBasicVar` the index of an entering variable, in `arrayVarBase` the index of an leaving variable, then to build the new dictionary obtained by pivoting with these two choices of indices; for that, it is necessary to compute for the new dictionary the matrix D, the arrays `arrayBasicVar` and `arrayNonBasicVar`.

REMARKS

- **WARNING : In the comments of the program, we say that the variable x1 is the variable of number 1, the variable x2 is the variable of number 2, ... When we talk about index in programming, it is always an index in arrays and not an index of a variable.**
- WARNING: do not use the attribute `incomplete` of the class `Dictionary`
- Decison variables have numbers between 1 and `nbNonBasicVar`; slack variables have numbers between `nbNonBasicVar` + 1 et `nbBasicVar + nbNonBasicVar`.
- For a dictionary of the first phase, when we search a feasible solution, a variable of number 0 is added.

You have to start by downloading the file simplexToComplete.jar as well as the file pbs.jar containing the dictionaries.

After launching Eclipse, to create the project:

- in "File", do New then Java Project: this opens a window;
- in the "Project name" framework, put the name of your choice;
- in the JRE framework, select "Use a project specific JRE"; on the right, click on "Configure JREs..." (under the two small frames): this opens a window called "Preferences"
- in the "Preferences" window, click on the "Add" button: this opens a window called "Add JRE";
- in the "Add JRE" window, select "Standard VM" then click on "Next": this opens a new window called "Add JRE";
- in "JRE home", write /cal/softs/java/jdk-11.0.19/ then click on Finish: this returns to the "Preferences" window; then click on "Apply and close"; this returns to the "New Java Project" window;
- in the JRE framework, at the (selected) line "Use a project specific JRE", choose "jdk-11.0.19" using the black triangle pointed down;
- click on the "Finish" button at the bottom; this opens a "New module-info.java" window; in this window, choose **"Don't create" in order NOT TO CREATE A MODULE**.

Then, for each of the two downloaded files

- click on the name of the project with the button on the right and choose "import";
- in the obtained window, in "General", choose "Archive File";
- after doing "Next", browse to choose the file to dowhload;
- click on finish.

You are then ready to work. You can already run the program; the main method is in the file `Main.java` of the `simplex` package. You can choose any of the data files provided in the directory named `pbs`.

WARNING: during all the work, do not modify anything outside the Dictionary class.

Details on the different files:

The dictionaries coded in pb1.txt up to pb15.txt are feasible.
The problems corresponding to pb1.txt up to pb13.txt are bounded.
The problems corresponding to pb14.txt and pb15.txt are unbounded.
If an iteration is performed from the dictionary encoded by pb12.txt using the entering variable of greatest coefficient, a degenerate dictionary is obtained.
The dictionary encoded by pb13.txt is degenerate. If we use the entering variable of greatest coefficient, without the rule of Bland, there is cycling.
The basic solutions associated with the dictionaries coded in pb16.txt up to pb20.txt are not feasible but the corresponding problems admit a feasible solution: the first phase of the simplex method gives a feasible dictionary for the initial problem.
The problems corresponding to pb21.txt and pb22.txt are not feasible.

You can find a documentation on the project code written in Java.

Work to be done

You have to implement (and test as you go along) some of the methods of the class `Dictionary`.

1. Implement methods in the following order:

- `isFeasible`
- `searchFirstIndexEnteringVariable`
- `searchIndexLeavingVariable`
- `pivote`

You can already try your program with all the data files giving an initial feasible dictionary with the choice of the entering variable according to the criterion of the variable of smallest index in the array `tabNonBasicVar`. We advise you to try at least pb1.txt and pb14.txt (which is unbounded).

2. Implément:

- `searchIndexEnteringVariableGreatestCoeff()`: you can test on pb3.txt; you should also notice that there is cycling with this method and the file pb13.txt
- `searchEnteringAdvantageousVariableIndex` : vous pouvez tester sur pb4.txt.

in order to test Bland rule:

- `searchIndexEnteringVariableSmallestNumber`,
- `searchIndexLeavingVariableSmallestNumber` (be careful to take among the outgoing variables the one with the lowest number).

You can test the Bland rule on pb13.txt to find that there is no longer any cycling with the choice of the entering variable of greatest coefficient by selecting "Apply Bland Rule".