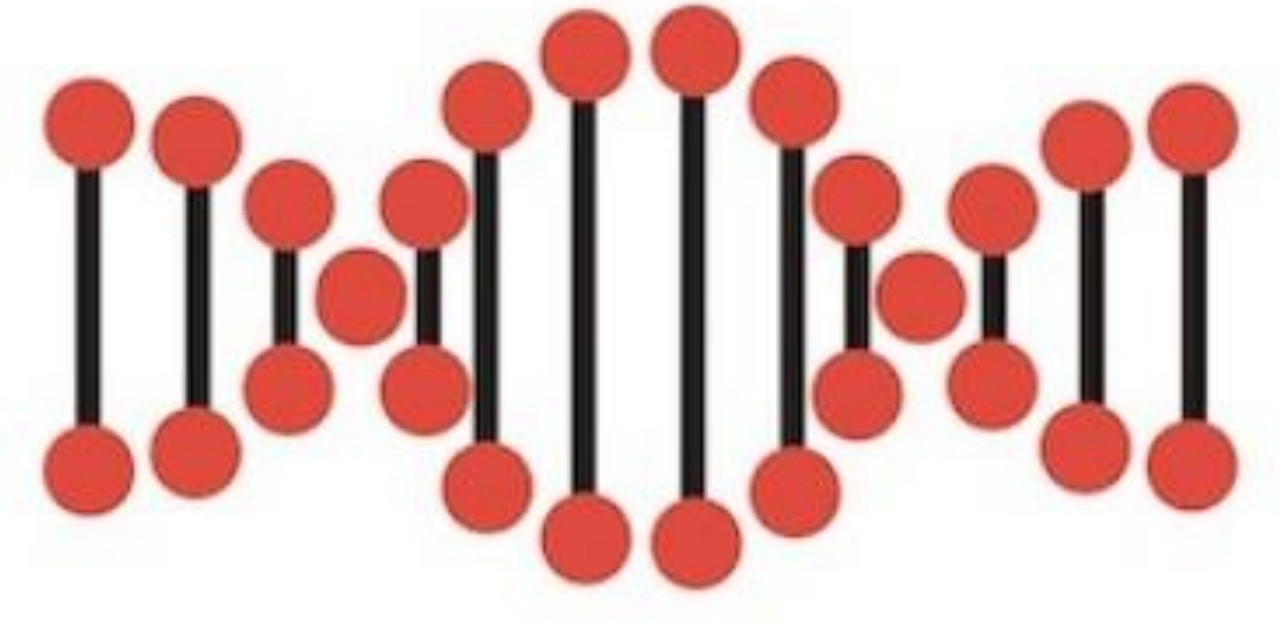
Projet 2 : Debugger une application JAVA

DÉVELOPPEMENT D'UN LOGICIEL D'ANALYSE DES DONNÉES,

Heme Biotech



ANALYSE DU CODE EXISTANT

```java
package com.hemebiotech.analytics;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;

public class AnalyticsCounter {
    private static int headacheCount = 0;    // initialize to 0
    private static int rashCount = 0;        // initialize to 0
    private static int pupilCount = 0;       // initialize to 0
```

Premier problème :

Il déclare les compteurs des symptômes les uns après les autres en tant que variable globale :

- cela induit que l'on connait forcément chaque symptômes ce qui est une contrainte pour l'utilisateur.
- le programme devient de suite obsolète dès lors que l'on a un nouveau symptôme.

ANALYSE DU CODE EXISTANT

```
public static void main(String args[]) throws Exception {
    // first get input
    BufferedReader reader = new BufferedReader (new FileReader("symptoms.txt"));
    String line = reader.readLine();
```

Deuxième problème :

Dans la fonction main on annonce qu'il va y avoir une exception "throws Exception " ce qui est normal car on ira lire fichier puis générer un fichier.

- Le problème est qu'il n'y a aucune gestion de ces exception d'entrée-sortie.

ANALYSE DU CODE EXISTANT

```
        System.out.println("symptom from file: " + line);
        if (line.equals("headache")) {
            headCount++;
            System.out.println("number of headaches: " + headCount);
        }
        else if (line.equals("rush")) {
            rashCount++;
        }
        else if (line.contains("pupils")) {
            pupilCount++;
        }

        line = reader.readLine();    // get another symptom
    }

    // next generate output
    FileWriter writer = new FileWriter ("result.out");
    writer.write("headache: " + headacheCount + "\n");
    writer.write("rash: " + rashCount + "\n");
    writer.write("dialated pupils: " + pupilCount + "\n");
```

Troisième problème :

Il saisit le nom des symptômes pour les comparer aux chaines de caractères lues dans le fichier. Cela peut engendrer des erreurs de frappe et l'occurrence cela a été le cas ici :

- En effet les symptômes rush et pupils n'existe pas dans le fichier et il les associe respectivement lors de l'écriture sur le fichier à générer aux symptômes rash et dialated pupils ce qui donne forcément des résultats erronés.

- Ajouter à ces point ci-dessus, il n'a pas saisie tout les symptômes le fichier résultat sera donc incomplet.

ANALYSE DU CODE EXISTANT

Autres remarques :

1- Il s'agit là d'un programme simple à comprendre, un programme plus compliqué serait difficile à comprendre car il est mal commenté.

- En effet il est aisé de comprendre qu'une variable est mis à zéro, un commentaire est inutile.

2- Il n'externalise aucune fonction or que JAVA est un programmation orienté objet.

CORRECTIONS APPORTEES

Dans un premier j'ai mis à plat la problèmatique et j'en ai extrait les contrainte :

La fonction principal attendue du programme est : "à partir du fichier symtptomes.txt, de déterminer le nombre de patient ayant présenté un type de symptôme.«

La contrainte associée à cette fonction est le fichier est évolutif et qu'il pourrait avoir d'autres symptômes que l'on ne connait pas encore.

Il en découle que l'on devra :

1- Ouvrir le fichier, le lire pour en extraire le nombre de symptômes ainsi que le nombre de patient traité. (Fonction secondaire n°1)

2- Dès que l'on a eu ces informations on génère un dictionnaire avec le couple de (String Key, Int Value) = (String symptome, Int nombreDeSymtomes)(Fonction secondaire n°2)

3- Enfin on retranscrit ces éléments dans un fichier de sortie. (Fonction secondaire n°3)

CORRECTIONS APPORTEES

On peut constater que :

1-Les 3 fonctions secondaire peuvent être dissociées donc faire l'objet de 3 class distinctes;

2-Seule la première fonction secondaire sera structuré de 2 méthodes :

a- La première est celle qui va nous permettre de lire le fichier pour extraire les donnée et produire une List des données issues du fichier.
METHODE SymptomReader.ReadFiles(String pathRead)

b- La deuxième pour produire un ensemble composé de données non répétitifs. METHODE SymptomReader.symptomes(symptomesLst)

3- La seconde fonction secondaire sera composé d'une seule méthode qui devra produire un dictionnaire de données (String,Integer) à partir d'un ensemble et d'une liste. METHODE SymptomCalcul.Calcul(symptomesLst, symptomes). On classera par ordre alphabétique les symptômes à l'aide de `TreeMap`

4- La troisième fonction secondaire utilisera la METHODE SymptomWriter.CreatFiles(pathWrite, symptomeNbre).