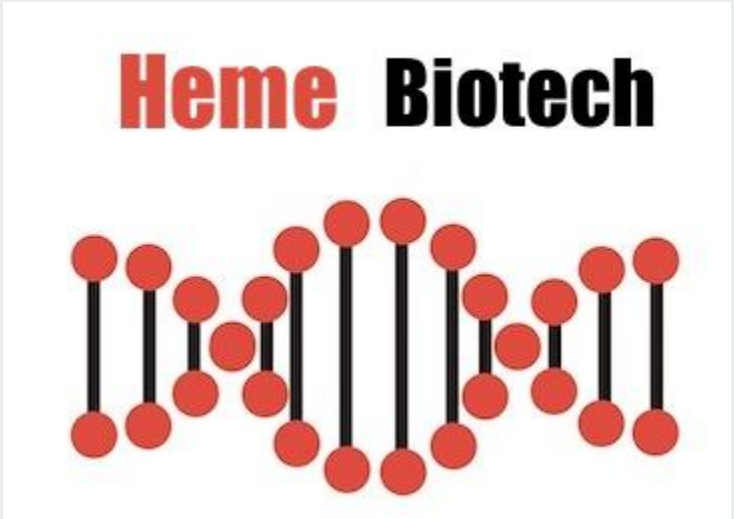
Présentation application java hemebiotech



Résumé des besoins

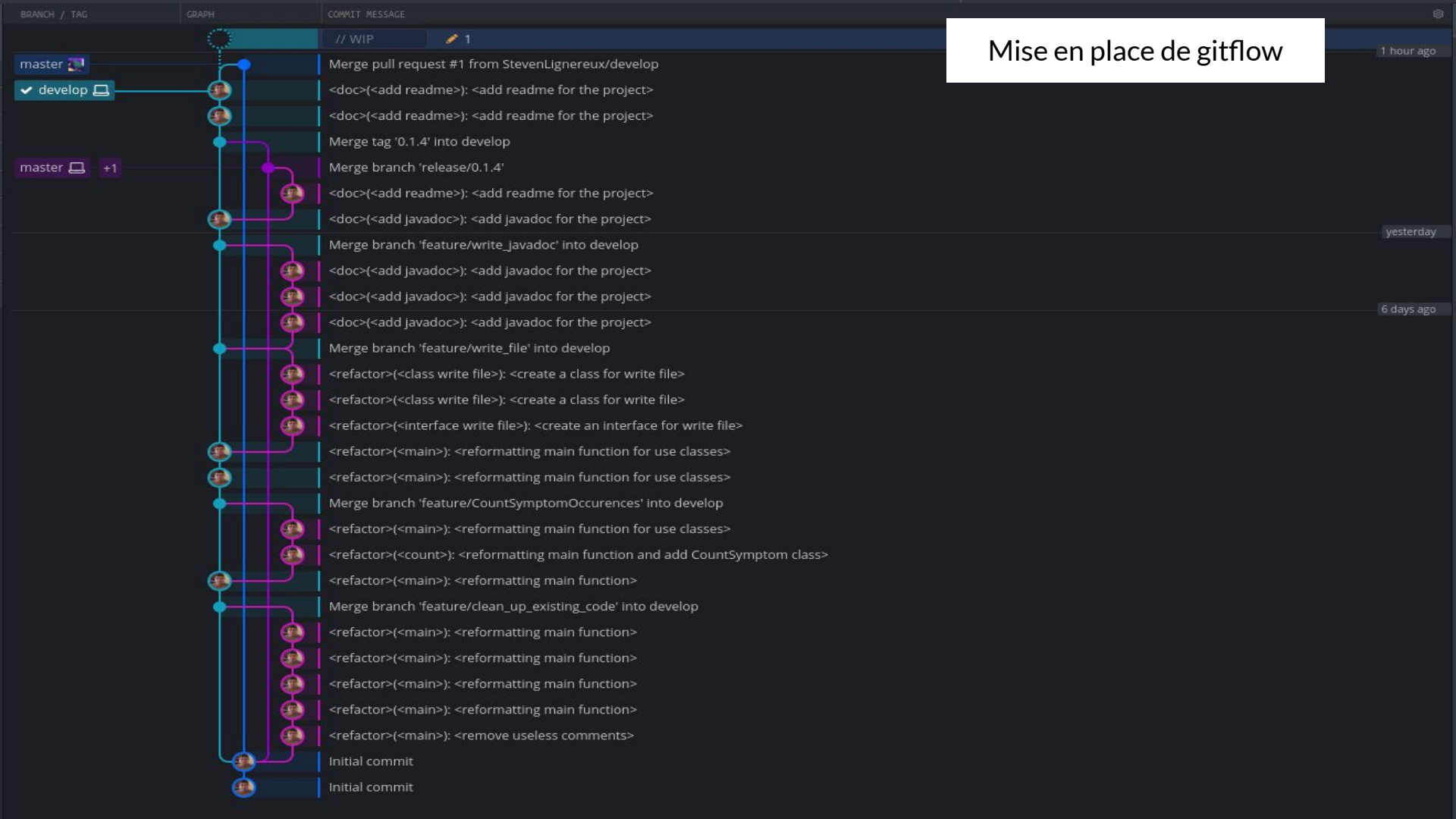
- Le but de l'application est de partir d'un fichier contenant une liste de symptômes avec des doublons pour arriver à un fichier de résultats avec le nom du symptôme et son nombre d'occurences
- Le projet est déjà commencé et installé mais ne fonctionne pas comme voulu
- Plusieurs problèmes de codes

Code d'Alex

- tout est dans une seule grosse fonction
- le code ne fonctionne pas comme il faut
- commentaires inutiles
- ne respect pas les bonnes pratiques
- variables qui ne sont pas explicitent et qui se chevauchent
- utilisation de la mauvaise fonction
- pas de gestion des erreurs
- pas de fermeture des ressources

```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

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

        int i = 0;    // set i to 0
        int headCount = 0;    // counts headaches
        while (line != null) {
            i++;    // increment i
            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");
        writer.close();
    }
}
```

Mise en place de gitflow



Rendre le code fonctionnel

- utilisation d'une map pour extraire les symptoms
- suppression des commentaires

```java
package com.hemebiotech.analytics;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.util.Map;
import java.util.TreeMap;

public class AnalyticsCounter {

    public static void main(String[] args) throws Exception {
        // Use the maps to extracting the symptoms of the file
        File symptomList = new File("symptoms.txt");
        Map<String, Integer> mapSymptomWithNumberOccurence = new TreeMap<>();
        FileReader symptomListReader = new FileReader(symptomList);
        BufferedReader bufferedSymptomReader = new BufferedReader(symptomListReader);

        // one symptom in the list
        String symptomStr;

        while ((symptomStr = bufferedSymptomReader.readLine()) != null) {
            if (mapSymptomWithNumberOccurence.containsKey(symptomStr)) {
                int countOccurence = mapSymptomWithNumberOccurence.get(symptomStr);
                countOccurence++;
                mapSymptomWithNumberOccurence.replace(symptomStr, countOccurence);
            } else {
                mapSymptomWithNumberOccurence.put(symptomStr, 1);
            }
        }

        // next generate output
        FileWriter writerSymptom = new FileWriter("result.out");
        for (String symptom : mapSymptomWithNumberOccurence.keySet()) {
            System.out.println(symptom + "=" + mapSymptomWithNumberOccurence.get(symptom));
            writerSymptom.write(symptom + "=" + mapSymptomWithNumberOccurence.get(symptom) + "\n");
        }
        symptomListReader.close();
        writerSymptom.close();
    }
}
```

Refactoring du code

- Création d'une classe CountSymptom qui sera chargée de compter les symptômes :
  - elle contient une méthode permettant d'obtenir une map de symptômes avec le nombre d'occurences

- Refactorer la classe AnaliticCounter pour utiliser les classes :
  - Utiliser la classe ReadFile pour lire le fichier
  - Utiliser la classe Countsymptom pour obtenir une liste de symptomes triés avec le nombre d'occurences

```java
package com.hemebiotech.analytics;

import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public final class CountSymptom {
    private final List<String> listOfSymptoms;

    public CountSymptom(List<String> listOfSymptoms) {
        this.listOfSymptoms = listOfSymptoms;
    }

    public Map<String, Integer> getMapOfSymptomsWithNumberOccurences(){
        // Using a TreeMap for alphabetic order
        Map<String, Integer> mapSymptomsWithNumberOccurences = new TreeMap<>();

        for (String symptom: listOfSymptoms){
            if (mapSymptomsWithNumberOccurences.containsKey(symptom)){
                int countOccurence = mapSymptomsWithNumberOccurences.get(symptom);
                countOccurence++;
                mapSymptomsWithNumberOccurences.replace(symptom, countOccurence);
            } else {
                mapSymptomsWithNumberOccurences.put(symptom, 1);
            }
        }

        return mapSymptomsWithNumberOccurences;
    }
}
```

```java
package com.hemebiotech.analytics;

import java.io.FileWriter;
import java.util.List;
import java.util.Map;

public class AnalyticsCounter {

    public static void main(String[] args) throws Exception {
        ReadSymptomDataFromFile symptomReader = new ReadSymptomDataFromFile("symptoms.txt");
        List<String>listOfSymptoms = symptomReader.GetSymptoms();

        CountSymptom countSymptomInstance = new CountSymptom(listOfSymptoms);
        Map<String, Integer> resultSymptom = countSymptomInstance.getMapOfSymptomsWithNumberOccurences();

        // next generate output
        FileWriter writerSymptom = new FileWriter("result.out");
        for (String symptom : resultSymptom.keySet()) {
            System.out.println(symptom + "=" + resultSymptom.get(symptom));
            writerSymptom.write(symptom + "=" + resultSymptom.get(symptom) + "\n");
        }
        writerSymptom.close();
    }
}}
  return go(f, seed, [])
}
```

Refactoring du code

- Création d'une interface pour gérer la création et l'écriture du fichier de résultats
- Création d'une classe qui implémente cet interface
- Mise à jour la classe AnalyticCounter pour utiliser ces classes

```java
package com.hemebiotech.analytics;

import java.io.IOException;
import java.util.Map;

public interface IWriteDataFile {

    void writeData(Map<String, Integer>map) throws IOException;
}
```

```java
package com.hemebiotech.analytics;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Map;

public class WriteDataFile implements IWriteDataFile {
    private final String filePath;

    public WriteDataFile(String filePath) {
        this.filePath = filePath;
    }


    @Override
    public void writeData(Map<String, Integer> map) {

        try (BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(filePath))) {
            for (Map.Entry<String, Integer> entry : map.entrySet()) {
                bufferedWriter.write(entry.getKey() + "=" + entry.getValue());
                bufferedWriter.newLine();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Code final et javadoc

- Mise à jour de toutes les classes
- Génération de la javadoc

```java
package com.hemebiotech.analytics;

import java.util.List;
import java.util.Map;

/**
 * The main class allows you to extract/read symptoms of a external file and count and alphabetical
order the list of symptoms with number of occurrences.
 * Finally, write the list in a result file.
 *
 * @author steven
 * @since 30/08/2022
 */

public class AnalyticsCounter {
    /**
     * Constant for path of symptoms files
     *
     */
    public static String INPUTFILE = "symptoms.txt";

    /**
     * Constant for path of result file
     */
    public static String OUTPUTFILE = "result.out";

    /**
     * The main class method called methods of classes
     *
     * @param args String array object
     * @throws Exception exception object for catch errors
     */
    public static void main(String[] args) throws Exception {

        ReadSymptomDataFromFile symptomReader = new ReadSymptomDataFromFile(INPUTFILE);
        List<String> listOfSymptoms = symptomReader.GetSymptoms();

        CountSymptom countSymptomInstance = new CountSymptom(listOfSymptoms);
        Map<String, Integer> resultSymptom =
countSymptomInstance.getMapOfSymptomsWithNumberOccurences();

        IWriteDataFile writeDataFile = new WriteDataFile(OUTPUTFILE);
        writeDataFile.writeData(resultSymptom);
    }
}
```

Difficultés et améliorations possibles

Les difficultés majeures rencontrées sur le projet et des pistes d'amélioration du code

1. Apprentissage du langage java
    a. courbe d'apprentissage élevée
    b. syntaxe peu familière

2. Utilisation de gitflow
    a. petit projet donc peu commode à utiliser pour un développeur

- Améliorations à apporter au code
    - pas d'utilisation de tests
    - interface graphique dans un autre temps
    - code probablement améliorable avec l'aide d'un senior