

Analyse des troubles sanguins

Heme Biotech

Présentation pour le projet n°2

Description

Plan

- 1) Présentation de l'application
- 2) Localisation des problèmes et leurs solutions
- 3) Difficultés rencontrées et méthode de résolution
- 4) Alternatives dans la conception du programme

1. Présentation de l'application

Comment est conçu le programme d'analyse des troubles sanguins ?

- Le programme fonctionne en complexité temps et espace linéaire: le temps d'exécution et l'espace de stockage occupée augmentent proportionnellement à la taille du fichier texte reçu par le programme. Le nombre d'insertion de chaînes pour chaque clé dans le fichier result.out est le type de données abstrait important.
- L'architecture SOLID est appliqué sur le programme: une classe pour chaque responsabilité, augmentation possible du nombre de fonctionnalité sans casser le programme, ségrégation des interfaces (programme divisible en entité séparé, pas de pollution des interfaces) et injection des dépendances.
- Styles de programmation utilisés: orienté-objet, fonctionnelle et impératif.

1. Présentation de l'application

Comment est conçu le programme d'analyse des troubles sanguins ? Il s'agit d'un problème type « Document distance problem ».

Types de données abstraites (ADT)	Structures des données
Classe ReadFile: Méthode getSymptom() implémentée contre son interface: <ul style="list-style-type: none">- lire le fichier symptoms.txt.- retourner chaque ligne dans une liste.	<ul style="list-style-type: none">- chaîne de caractère en entrée de la méthode path1;- tableau de chaînes de caractères data;
Classe SymptomCounter: Méthode SymptomCounter() implémentée contre son interface: <ul style="list-style-type: none">- parcourir le tableau de chaîne précédent en comptant les occurrences de chaque chaîne dans ce tableau.- insérer les symptômes et leurs occurrences dans une table de hachage par ordre alphabétique.	<ul style="list-style-type: none">- tableau de chaînes de caractères symptomList;- table de hachage result;
Classe WriteFile: Méthode symptomFileWriter() implémentée contre son interface: <ul style="list-style-type: none">- Transformer la table de hachage des symptômes en fichier result.out contenant les symptômes et leurs occurrences par ordre alphabétique.	<ul style="list-style-type: none">- table de hachage symptoms convertit en fichier result.out. Fichier disponible dans le répertoire du programme et exploitable pour l'utilisateur.

2. Localisation des problèmes et solutions

Pourquoi le programme d'Alex n'est pas fonctionnel ?

- Code de lecture du fichier, analyse et écriture intégrée dans la classe main: les paradigmes de la programmation orientée-objet en Java ne sont pas respectés. La classe main regroupe toutes les fonctionnalités du programme. Il n'est pas possible de modifier, ajouter et enlever une fonctionnalité sans casser le programme.
- Hormis les 3 symptômes indiqués, les autres symptômes ne sont pas pris en compte.
- Aucune des 3 parties du code n'est réutilisable: les principes SOLID ne sont pas respectés.
- Le travail de Caroline consistant à créer la classe et son interface est la bonne marche à suivre pour faire fonctionner le programme.

3. Difficultés rencontrées et méthodes de résolution

Quels problèmes ai-je rencontré et comment je les ai résolus ?

- Difficultés de choix et compréhension des classes Java disponibles pour lire et écrire dans des fichiers (BufferedReader, Java NIO, PrintWriter, Writer etc.).
- Rédaction des diagrammes des 3 classes en UML ne permet pas de trouver une solution, juste de s'assurer que les principes SOLID sont bien respectés.
- Vérification des résultats en utilisant d'autres technologies avant de revenir et terminer le programme en Java: améliorer la compréhension et vérifier si on a bien les mêmes résultats entre Java, Python, C et C++.
- Autres alternatives possibles en C et C++ sous CLion: affichage des données.
- En Python sous PyCharm:

3. Difficultés rencontrées et méthodes de résolution

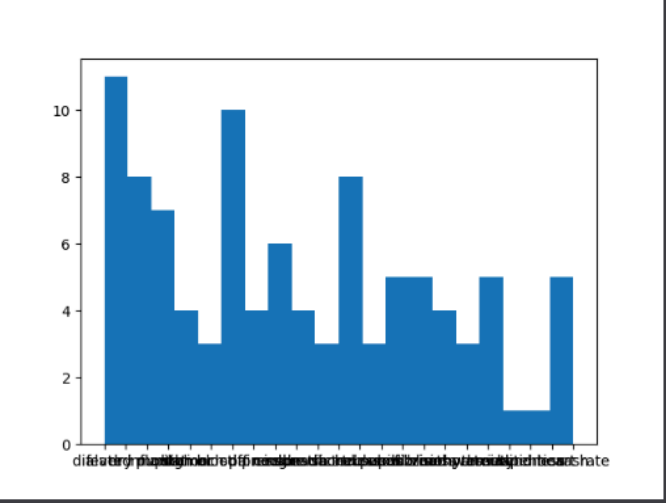
pythonProject > main.py

Project: pythonProject ~\Pycharm

- venv
 - clustering.py
 - main.py
- External Libraries
- Scratches and Consoles

```
1 from typing import List, Any, Union
2
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 from pandas import Series
6
7 df = open("/Users/GoldenEagle/IdeaProjects/Projet-2-bis/PHEME_01_code/symptoms.txt")
8 lines = df.readlines()
9 df.close()
10
11 # remove /n at the end of each line
12 for index, line in enumerate(lines):
13     lines[index] = line.strip()
14
15 df_symptoms = pd.DataFrame(lines, columns=['Symptoms'])
16 print(df_symptoms)
17
18 # count the occurrences of each symptom in df_symptoms
19 df_occurrences = [df_symptoms[index].value_counts() for index in df_symptoms]
20 print(df_occurrences)
21
22 # display symptoms
23 plt.hist(df_symptoms, bins=20)
```

SciView: Data Plots 20,2 kB



Run: main

[100 rows x 1 columns]

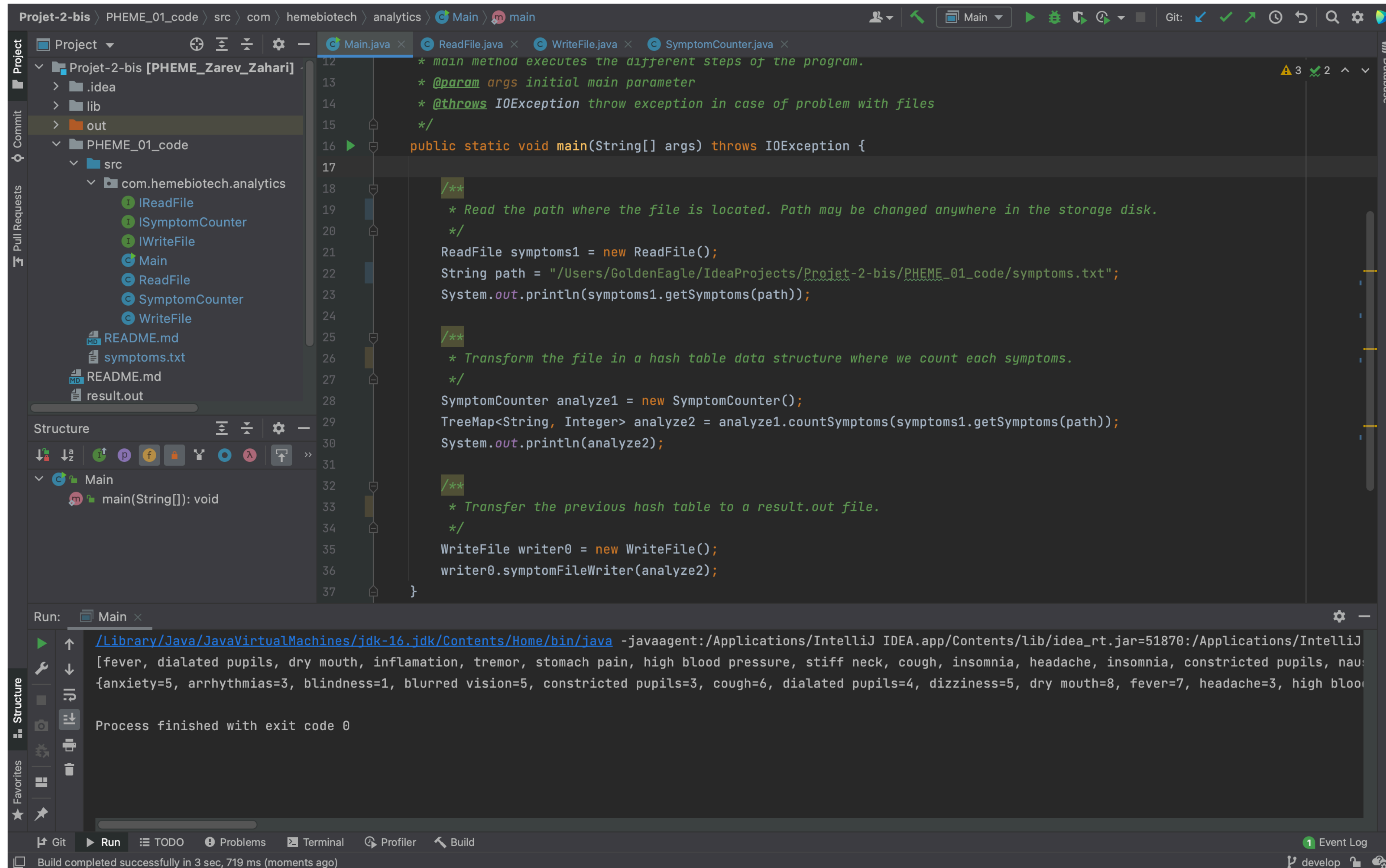
high blood pressure	10
dry mouth	8
fever	7
inflammation	7
cough	6
anxiety	5
blurred vision	5
nausea	5
dizziness	5

Welcome to the future of programming.
Kite is now integrated with your IDE.
[Learn how to use Kite](#) [Don't show this at startup](#)

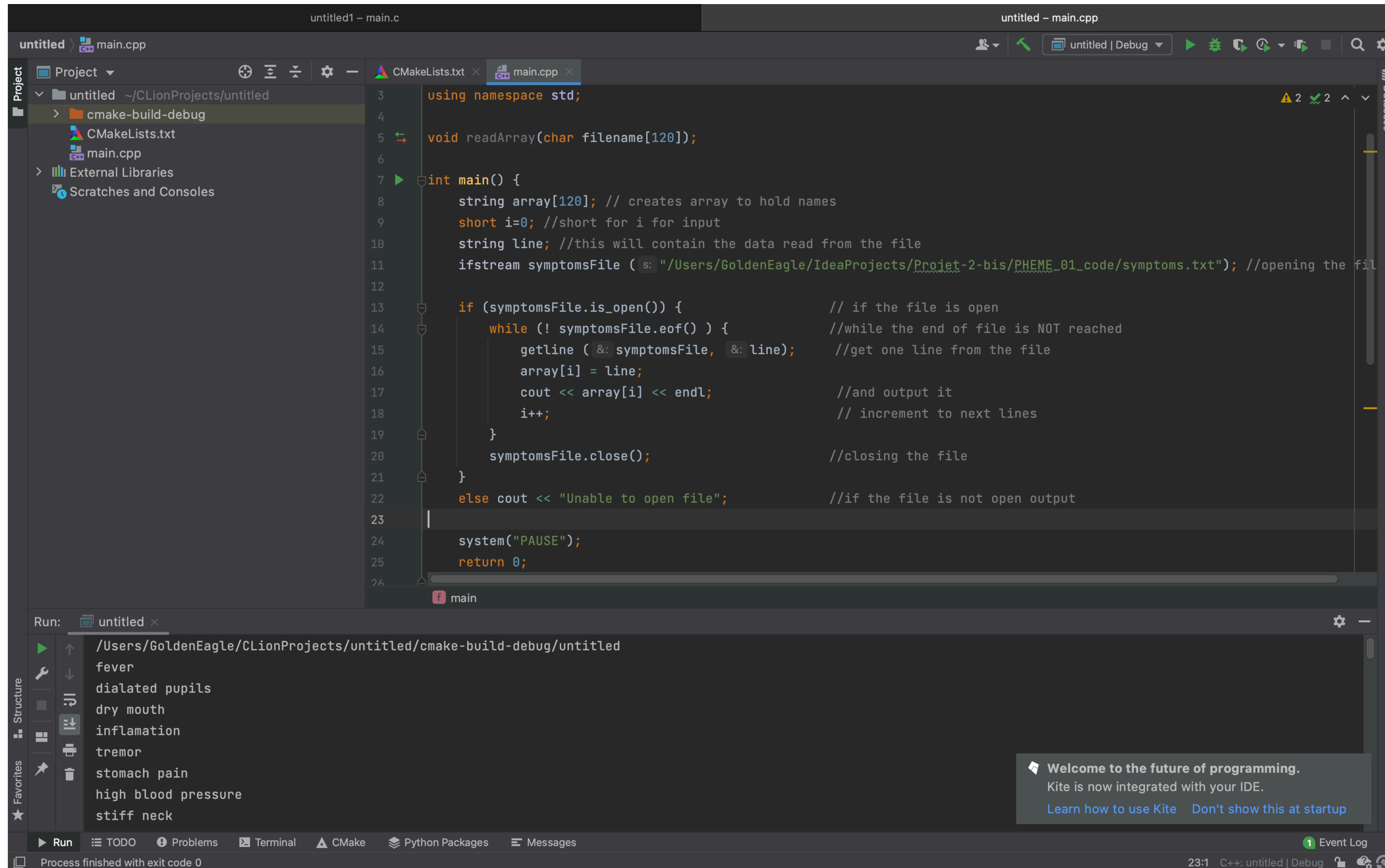
Run TODO Problems Terminal Python Packages Python Console Event Log

Welcome to the future of programming.: Kite is now integrated with your IDE. // Learn how to use Kite // Don't show this at startup (28 minutes ago) 10:1 Python 3.9 (pythonProject)

3. Difficultés rencontrées et méthodes de résolution



3. Difficultés rencontrées et méthodes de résolution



4) Alternatives dans la conception du programme

Quelles sont les alternatives de conception de ce programme ?

- Style de programmation déclaratif aurait pu être utilisé pour compter les occurrences des symptômes dans le fichier: classes streams et ses méthodes. Il s'agit de combiner streams qui fait office de pipeline avec la table de hachage TreeMap et le tableau de chaînes de caractère.
- Penser différemment en temps logarithmique le problème de distance entre chaînes de caractère et éviter le temps linéaire $O(n)$: modifier la classe SymptomCounter. C'est une alternative si nous devons traiter un fichier contenant 10 millions de symptômes dans le désordre par exemple.
- Ajout d'une fonction de visualisation graphique des symptômes et leurs occurrences.