

**Rapport du ProJet Java**

#### Sujet

**Application de gestion d’erreurs**

**De compilations & conventions**

**(IDE)**

**Réalisé par : Encadré par :**

AARAB Jamal M.Bouzoubaa

SOUL Redouane

LEGHRISSI Ayoub

Sommaire

1. Introduction :
2. Cahier de charges :
3. Vue globale sur l’application :
4. Code Java :
5. Outils :
6. Conclusion :
7. Introduction :

En début d'année, on nous a demandé de créer une application de gestion de données en utilisant Java et le langage C, alors que nous avions peu de connaissances préalables en C. Nous avons donc opté pour la programmation fonctionnelle, mais avec l'augmentation du nombre de lignes de code, nous avons constaté que notre progression diminuait, car nous passions la plupart de notre temps à trouver l'emplacement et la cause des erreurs. Cette méthode de programmation s'est donc révélée chronophage et frustrante, d'autant plus qu'il nous fallait également corriger les fautes d'orthographe.

Cependant, la programmation orientée objet, qui est connue pour ses quatre piliers (encapsulation, abstraction, polymorphisme et héritage), peut nous permettre d'éviter ces problèmes rencontrés dans la programmation fonctionnelle. Elle présente également de nombreux autres avantages tels que la minimisation des coûts de développement et la garantie d'une structure de code facile à maintenir.

En utilisant la programmation orientée objet, nous pourrions organiser notre code en classes, qui encapsulent les données et les fonctions qui les manipulent, nous permettant ainsi d'éviter les erreurs causées par des conflits de noms de variables ou de fonctions. De plus, l'abstraction nous permettrait de cacher les détails de l'implémentation et de ne montrer que ce qui est pertinent pour l'utilisateur de l'application. Le polymorphisme nous permettrait de créer des fonctions qui peuvent agir différemment selon le type d'objet qu'elles reçoivent en entrée, ce qui est très utile pour gérer différentes situations. Enfin, l'héritage nous permettrait de créer des classes qui héritent des propriétés et des fonctions d'une classe parente, nous permettant ainsi de réutiliser du code déjà écrit et testé.

En guise d'exercice pour mieux comprendre les concepts de la POO, nous allons créer un éditeur de texte très basique en nous basant sur une modélisation UML. Cela nous permettra de mettre en pratique les principes de l'encapsulation, de l'abstraction, du polymorphisme et de l'héritage que nous avons appris, tout en nous aidant à mieux comprendre leur utilisation dans un projet concret.

**II. Cahier de charges :**

**III. Vue globale sur l’application :**

**IV. Code JAVA :**

**V. OUTILS :**

Dans cette partie, on va présenter les différents outils qui nous ont aidé pour réaliser notre projet :

1. **GitHub**

****

L’outil parfait qui permet la coordination entre les différents membres du groupe et qui facilite la lecture du code et la détection des modifications et la récupération des différentes versions de notre application.

**2.Stackoverflow**



C’est le forum qu’on a utilisé pour poser nos différentes questions en relation avec le code.

**3.Figma**

 Permet la modélisation des différentes interfaces graphiques.

**4.Zoom :**



Logiciel de communication que l’on a utilisé pour organiser des réunions à distance .

**5.IDE Eclipse :**

Logiciel dont on s’est inspiré pour la réalisation de notre propre IDE, et que l’on a même utilisé pour taper notre code JAVA.

1. **Conclusion :**