
Polaris
LA NUIT DE L'INFO
2021

Le défi : Movai code

Introduction:

Si le génie logiciel est l'art de produire de bons logiciels, il est nécessaire de fixer les critères de qualité d'un logiciel tel que l'optimisation et le respect des conventions .

Y'en a marre de ça !!!!!!

Nous avons choisi donc de sortir de l'ordinaire c'est-à-dire trouver la solution avec laquelle sa compléxcité est considérée élevée.

Elle se caractérise d'une faible fiablité et robustesse ,nécessite un énorme espace memoire et temps d'exécution assez important .

De sorte que notre code ne sera pas utiliser par d'autres développeurs au futur afin de donner un code qui est ni évolutif ni conviviale .

L'objectif :

Alors l'objectif de la nuit de l'info de cette année consiste à réaliser un défi sur le sujet du site web "Les Sauveteurs du Dunkerquois".

En fait, notre but principal c'est d'effectuer une recherche selon les critéres désirés d'un sauvé ,en mettant en relief l'utilisation d'un mauvais code en terme de critére.

L’explication de code :

En partant de cette idée , l'utilisateur doit donner un ensemble des sauvés qui vont être stockés dans des fichiers textes séparés (chaque sauvé posséde son propre fichier texte) .

Puis nous avons parcouri tous les fichiers un par un en créant pour chaque critére un tableau à part (noms,prenoms,jours,mois,années,sexes).

Pour la recherche d'un sauvé, l'utilisateur doit entrer tous les variables d'un sauvé .La recherche se faite par parcourir des tableaux des noms , des prénoms, des jours, des mois, des années, des sexes ,si tous ces derniers existent, le programme va afficher deux messages "on a trouvé la personne " et "la personne $nom" sinon il va afficher "il n'existe pas". Pour obtenir un mauvais code ,nous n'avons pas utiliser ni l'encapsulation ni le polymorphisme .En plus,pour augmenter le temps d'exécution, à chaque attribut nous avons fait un accés de lecture et d'écriture.De plus, nous n'avons pas fait un contrôle de saisie .Nous n'avons pas utiliser la méthode procédurale.

Nous n’avons pas implementé des commentaires explicatifs plutôt des commentaires inutiles.

De surcroît,nous avons utiliser plusieurs instructions conditionnelles (if-else).

En outre ,nous avons déclarer des noms des variables non significatifs .Ce processus de recherche consomme un large espace de mémoire à cause d'une utilisation multiple de fichier .

De plus nous avons utiliser uniquement des méthodes statiques de telle sorte que nous ne pouvons pas les utiliser dans d'autre classe.

Prenons l'exemple de cents sauvés ,pour cela nous devons créer 100 fichiers pour chaque sauvé puis les parcourir un par un afin de construire des tableaux

pour chaque critére (nom,prénom,jour,moi,année,sexe) de taille 100 et pour chercher un tel sauvé nous devons parcourir 1200 fois (6*2*100) ,s'il existe il va affiché deux messages "on a trouvé la personne " et "la personne $nom" sinon il va affiché "il n'existe pas".