# RAPPORT TP3 2022

**DECEMBER 13**

**Université de Montréal**

*Présenté à Mohamed Lokbani*

*Présenté par Jorge Olivera*

# Description et utilisation des fonctions

1. **Description du programme.**

Le programme s'ouvre et présente une interface graphique où l'utilisateur doit choisir un fichier contenant 442 enregistrements avec des informations sur la maladie des patients diabétiques. Plus d'informations peuvent être trouvées en cliquant sur le bouton **Info** puis sur le sous-menu **Aide.**
Si vous exécutez le programme, vous allez ouvrir le fichier **diabetes.tab.txt** pour accéder au menu principal et voir les 9 options proposées par le programme (Plus de détails dans la section **Aide**).

2. **Utilisation des fonctions.**

Le programme est divisé en 2 parties, le data_controller et la partie view.
Dans le contrôle des données, nous trouverons les fonctions suivantes :

- ✓ **void load(file_name) :** Reçoit le nom du fichier en arguments et charge les 442 enregistrements en tant qu'objets dans la variable **all_data**.
- ✓ **list_all_data() :** Retourne un vecteur avec les 442 enregistrements trouvés dans la variable **all_data**.
- ✓ **mean() :** Effectue le calcul de la moyenne et renvoie le résultat sous forme d'objet.
- ✓ **stddeviation() :** Effectue le calcul de la déviation standard et renvoie le résultat sous forme d'objet.
- ✓ **getHeader() :** Retourne l'en-tête avec le nom de chacune des caractéristiques de l'étude clinique.
- ✓ **getDescription() :** Retourne une chaîne avec toutes les informations de l'étude clinique.
- ✓ **list_head() :** Retourne un vecteur avec les cinq premières lignes de la lecture.
- ✓ **coefficients_droite_data_a() :** Retourne la variable a avec son information.
- ✓ **coefficients_droite_data_b() :** Retourne la variable b avec son information.
- ✓ **r_square_data() :** Retourne la variable r avec son information.

- ✓ **prediction_data() :** Retourne un double avec l'information sur la prédiction.

Dans la partie view, nous trouverons les fonctions suivantes :

- ✓ void OnFichierOuvrir : qui effectue la lecture du fichier
- ✓ void OnChoix : qui gère les 9 options qui choisira l'utilisateur
- ✓ void OnFichierSauvegarderSous : qui effectue la écriture du fichier
- ✓ void OnFichierQuitter : qui met fin au programme
- ✓ void OnInfoApropos : qui signale la version, la date et la personne qui a développer l'application.
- ✓ void OnAide : un bref résumé de ce que le programme peut faire.

# Structure de données employées et justification

## 1. Avantages de la méthode choisie.

J'ai décidé de gérer les données en créant une classe appelée data, cette classe est responsable de la surtaxe des opérateurs suivants :

- ✓ Surcharge de l'opérateur de soustraction **« - »** : effectue la soustraction de chacun des attributs.
- ✓ Surcharge de l'opérateur d'addition/égal **« += »** : effectue l'addition de chacun des attributs.
- ✓ Surcharge de l'opérateur de division **« / »** : effectue la division de chacun des attributs.
- ✓ Surcharge de l'opérateur exposant **« ^ »** : effectue la puissance de chacun des attributs.
- ✓ Surcharge de la fonction opérateur **« () »** : effectue la racine carrée de chacun des attributs.

Ce modèle permet une meilleure organisation puisqu'il traite les opérations directement liées à l'information, permettant aux autres méthodes de gérer efficacement le contrôle des données et leur affichage.

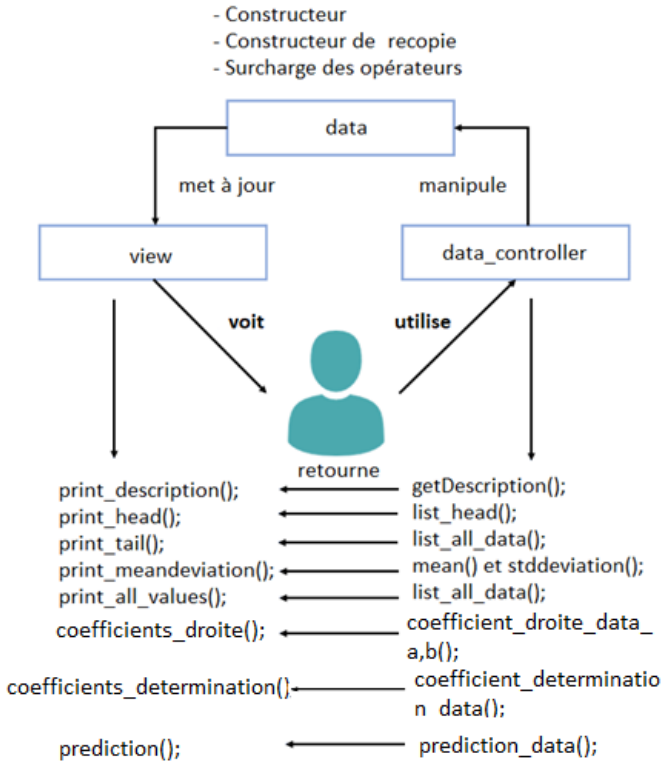
# Algorithme y justification

## 1. Description de l'algorithme.

Le programme commence par créer un objet **view** qui reçoit le nom du fichier et exécute une méthode **choices**.
La méthode **onChoix** reçoit l'option choisie par l'utilisateur et affiche les informations souhaitées. Les parties les plus importantes du programme se concentrent sur les méthodes suivantes :

- ✓ **void load()** : lit le fichier diabète en créant un vecteur avec 442 instances de l'objet de données.
- ✓ **void onChoix() :** cette méthode appelle gère la information envoyée par la classe data_controller **mean() et stddeviation()** qui utilisent la surcharge de la classe data pour effectuer les opérations de soustraction, addition, division, entre autres.

**Voici la présentation du modèle :**



# Discussion et améliorations

## 1. Discussion.

- ✓ **La conception orientée objet :** la manière d'organiser les informations en objets me donne la possibilité de regrouper les données de manière modulaire, ce qui m'a aidé à manipuler efficacement le traitement de chaque ligne lors de la lecture du fichier texte.
- ✓ **La surcharge des opérateurs :** la classe de données s'est occupée du traitement des opérations arithmétiques en donnant une meilleure visibilité sur les tâches des fonctions qui seront exécutées avec un code plus clair.
- ✓ **Gestion des flux :** la méthode **view** et **controller** permet une meilleure circulation des données. Lors de la lecture du fichier texte, les informations sont stockées dans un vecteur d'objets, ce qui permet un accès facile.
- ✓ **L'utilisation d'un makefile :** la création de ce fichier m'a permis de mieux gérer lors de la compilation les classes et leurs dépendances, en veillant à ce que chacune d'elles soit mise à jour avant de créer l'exécutable.

## 2. Améliorations possibles.

L'interface graphique offre une meilleure expérience utilisateur, mais il lui manque encore un design moderne qui soit cohérent avec d'autres programmes souvent utilisés par les utilisateurs.