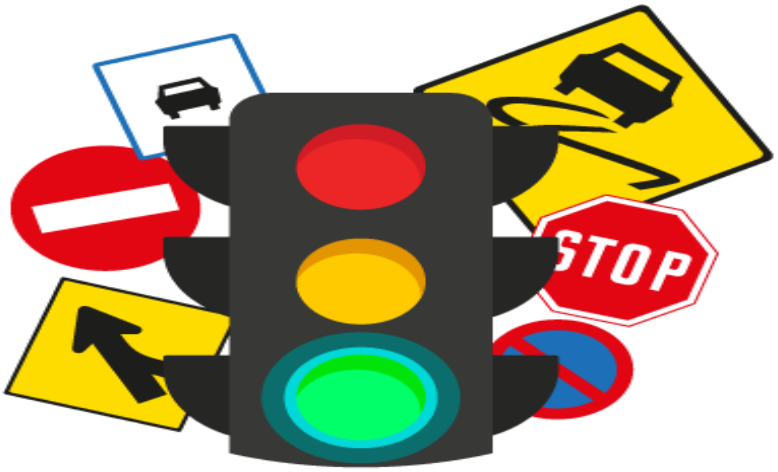
Université de la Manouba

Ecole Nationale des Sciences de l'Informatique

Mini-rapport du Stage d'été II1

-Stage de Programmation-

Gestion d'auto-école



Réalisé par

Takwa Cherif

Yasmine Boussaadoun

Maram Raddaoui

Année Universitaire 2019/2020

1. Présentation générale du sujet

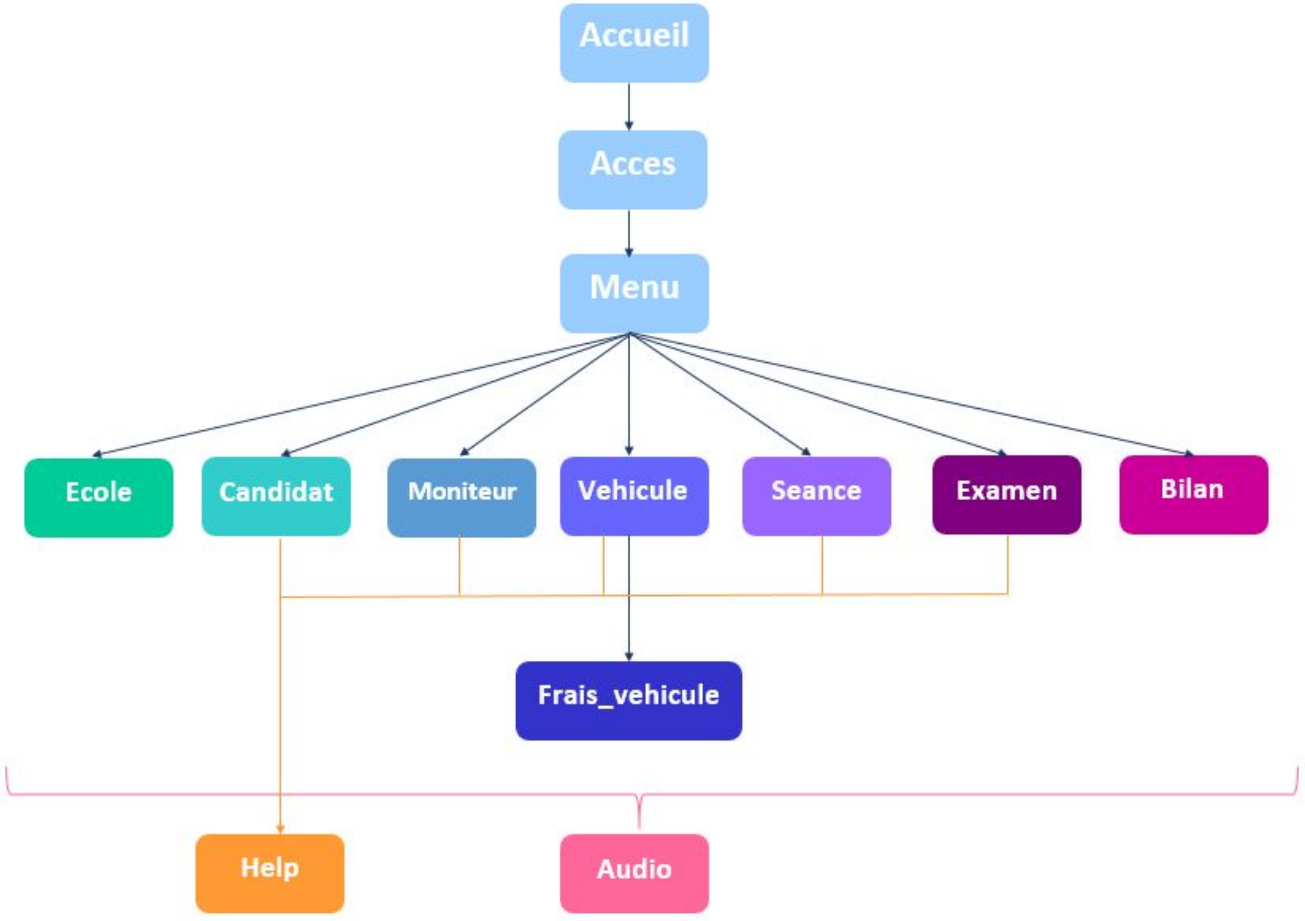
Une auto-école représente une des organisations ayant des ressources et des activités dont la gestion nécessite une application informatique. Ainsi, l'objectif de notre projet est de réaliser une application de gestion d'auto-école sous java propose de nombreuses fonctionnalités à l'administrateur pour gérer les informations de son établissement.

2. Présentation des fonctionnalités demandées de l'application

- La gestion des candidats
- La gestion des moniteurs
- La gestion des véhicules
- La gestion des examens
- La gestion des séances
- La gestion financière

3. Présentation de la conception de la solution proposée

3.1. Architecture du programme :



3.2. Détails de la solution

3.2.1. Candidat :

Cette classe permet l'ajout, la modification et la suppression d'un candidat ainsi que l'affichage des tous les candidats de l'école en spécifiant la catégorie du permis, le nombre des séances et le paiement.

- Les attributs:

- cin (String):représente le numéro de carte CIN.
- nom_prenom(String):représente le nom et le prénom du candidat.
- tel(long) :représente le numéro de téléphone du candidat.
- adresse(String) :représente l'adresse du candidat.
- datei(String):représente la date d'inscription.
- dateN (String):représente la date de naissance.
- type (long):représente le type d'inscription code, conduite, code et conduite ou conduite sans examen.
- categorie(String):représente la catégorie de permis A1, A, B, etc.
- paiement (String):représente le choix de paiement(forfait ou sans forfait).
- prixcode (float):représente le prix total de code.
- prixcond(float):représente le prix d'une séance conduite.
- nbcode (int):représente le nombre des séances de code.
- nbcond(int) :représente le nombre des séances de conduite.
- code(float):représente le coût d'examen code.
- cond(float):représente le coût d'examen conduite.
- montant(float) :représente le montant du forfait.
- frais(float): représente le montant total qui sera payé par le candidat.
- somme(float):représente le montant déjà payé.
- reste(float):représente le reste du montant total.
- paye(float):représente le nouveau montant payé.
- etatC(String):représente l'état sonore de l'interface

- Les méthodes :

- ajouter(): permet l'ajout d'un nouveau candidat .
- modifier(): permet la modification des informations d'un candidat.
- supprimer(): permet la suppression d'un candidat à travers son numéro de CIN et ses séances planifiées qu'il n'a pas encore étudier grâce à la fonction supprimer_seance(String cin).
- verifier(): vérifie la validation des champs obligatoires.
- cinExiste():vérifie l'existence du candidat.
- actualiser():met à jour la table des candidats.
- nombre_Seance(String cin,String nom_prenom,String type):vérifie s'il peut modifier le nombre des séances ou non selon le cas.
- ajouter_paiment():enregistre le montant payé par le candidat.
- reinitialiser(): remet les champs dans leur état initial.
- calculer(): permet d'avoir les informations sur le paiement, le reste toute au long le montant déjà payé et le frais total.
- supprimerMoniteur(String cin,String date):permet la suppression d'une séance du planning moniteur,utilisé dans la méthode nombre_Seance(String cin,String type).

- actualiserMoniteur(String cinM,String date,String nb): met à jour le nombre des élèves dans une séance de code d'un moniteur,utilisé dans la méthode nombre_Seance(String cin,String type).
- recuperer_champs_du_tableau(): permet la récupération des informations du candidat choisi à partir du table.
- recuperer_champs(): permet la récupération des informations du candidat lors de la saisie du numéro de CIN dans le champ.
- etat_son() : gère l'état sonore de l'interface.
- message(String,boolean) : affiche un message sonore .

3.2.2. Moniteur:

Cette classe permet l'ajout, la modification et la suppression d'un moniteur ainsi que l'affichage des tous les moniteurs de l'école en spécifiant sa spécialité et son salaire et l'affichage de la fiche de paiement de chaque moniteur.

- Les attributs :

- cin(long) : représente le numéro de la carte d'identité du moniteur .
- nom_prenom(String) : représente le nom et le prénom du moniteur.
- naissance(String) : représente la date de naissance du moniteur.
- telephone(long) : représente le numéro de téléphone du moniteur.
- adresse(String) : représente l'adresse du moniteur.
- recrutement(String) : représente la date de recrutement du moniteur.
- specialite(String) : représente la spécialité du moniteur.
- categorie(String) : représente la catégorie du permis du moniteur.
- prix_seance(float) : représente le prix de la séance du moniteur qui possède la spécialité conduite.
- salaire(String) : représente le salaire du moniteur qui possède la spécialité code.
- max_seances(int) : représente le nombre des séances maximale à travailler par jour pour un moniteur.
- etat_son(String) : représente l'etat sonore de l'interface.

Les méthodes :

- void ajouter() : permet d'ajouter un moniteur.
- void modifier() : permet la modification des informations d'un moniteur.
- void supprimer() : permet la suppression d'un moniteur.
- float salaire(Sting) : permet de calculer le salaire d'un moniteur qui possède la spécialité coduite.
- void recuperer_champs_du_tableau() : permet de récupérer les informations d'un moniteur choisi à partir du tableau.
- void actualiser() : actualiser le tableau qui affiche tous les moniteurs.
- void reinitialiser() : permet d'initialiser les champs de l'interface.
- void recuperer_champs() : permet de récupérer les informations d'un moniteur enregistré en écrivant le numéro de la carte cin dans le champs de ce dernier.
- void message(String,boolean) : permet d'afficher un message sonore.
- boolean cinExiste() : permet de vérifier l'existence d'une carte cin c'est à dire l'existence d'un moniteur.
- boolean correction_date(String) : permet de vérifier si la date saisie dans le champ de recherche est sous cette forme yyyy:mm ou non.

- void rechercher() : permet d'afficher tous les fiches des paiements de la date saisie dans le champ de recherche .
- void afficher_paiement() : permet d'afficher tous les fiches de paiement d'un moniteur spécifié en écrivant le numéro de sa carte d'identité .
- void message(String,boolean) : affiche un message sonore.
- void etat_son() : gère l'état sonore de l'interface.

3.2.3. Vehicule :

Cette classe permet la gestion des véhicules.

- Les attributs :
  - matricule(String) : représente la matricule du véhicule.
  - marque(String) : représente la marque du véhicule.
  - type(String) : représente le type du permis qui correspond au véhicule.
  - etat(String) : représente l'état du véhicule si en panne ou non.
  - date_achat(String) : représente la date d'achat du véhicule.
  - carburant(float) : indique le frais de consommation du carburant par heure.
  - etat_son(String) : indique l'état sonore de l'interface.
- Les méthodes :
  - void ajouter() : permet d'ajouter un véhicule.
  - void modifier() : permet la modification des informations d'un véhicule.
  - void supprimer() : permet la suppression d'un véhicule.
  - boolean ajout_frais(String): vérifie l'ajout des frais pour un nouveau véhicule ajouté.
  - void actualiser() : actualiser le tableau qui affiche tous les véhicules.
  - void reinitialiser() : initialiser les champs de l'interface.
  - boolean matricule_existe() : vérifie si le véhicule est déjà enregistré ou non.
  - void recuperer_champs() : récupérer les informations d'un véhicule enregistré en écrivant le matricule dans le champs de ce dernier.
  - void recuperer_champs_du_tableau() : récupérer les informations d'un véhicule choisi à partir du tableau.
  - void message(String,boolean) : affiche un message sonore.
  - void etat_son() : gère l'état sonore de l'interface.

3.2.4. Frais_vehicule :

Cette classe permet l'ajout des frais pour chaque véhicule et leurs affichages.

- Les attributs :
  - montant(float) : représente le montant du frais.
  - date(String) : représente la date du paiement du montant.
  - etat(String) : indique l'état sonore de l'interface.
- Les méthodes :
  - void enregistrer() : permet l'enregistrement du frais.
  - void message(String,boolean) : affiche un message sonore.
  - void actualiser() : actualiser le tableau qui affiche tous les véhicules.

3.2.5. Seance :

Cette classe permet l'ajout et la modification d'une séance ainsi que l'affichage des tous les séances enregistrées des candidats et des moniteurs et la recherche des séances d'un candidat ou d'un moniteur spécifique à partir du numéro de CIN ou du nom.

L'ajout des séances se fait directement après l'ajout d'un candidat selon les données de ce dernier.

- Les attributs:
  - cin (long):représente numéro de carte CIN.
  - nom_prenom(String):représente le nom et le prénom du candidat.
  - type(String):représente le type de la séance code ou conduite.
  - dateS(String):représente le date de la séance,date de jour + heure.
  - dat(String):représente la date précédente de la séance.
  - moniteur(String):représente le moniteur responsable de la séance.
  - vehicule(String):représente la véhicule choisi en cas de la séance conduite.
  - nsalle(String):représente le numéro de salle en cas de la seance code.
  - vec(Vector):représente les informations avant le changement ,utilisé dans la méthode modifier.
  - n1(int):représente le nombre des seances de code ajoutées.
  - n2(int):représente le nombre des séances de conduite ajoutées.
  - nbcond (int):représente le nombre des séances de conduite.
  - recherche(String):représente le moniteur ou le candidat recherché(par le nom et le prénom ou par le numéro de CIN.
- Les méthodes:

  -ajouter(): permet l'ajout d'une séance.

  -modifier(v): permet la modification des données d'une séance.

  -rechercher():permet la recherche des séances planifiées du moniteur ou du candidat, soit par le numéro de CIN soit par le nom et prénom.

  -afficheSeance():permet l'affichage de toute les séances du candidat à travers son numéro de CIN.

  -verif():vérifie la validation des champs obligatoires.

-cin(v):permet la conservation du numéro de CIN et le nom et prénom du candidat lors de l'ajout des séances.

-seanceCandidat():permet l'affichage de tout le planning candidat.

-seanceMoniteur():permet l'affichage de tout le planning moniteur.

-insertionMoniteur(cinM,nomM, dateS,specialite,veh,nb,numsalle): permet l'ajout d'une séance dans le planning moniteur.

-insertionCandidat(cin,np, type,dateS,moni,veh, nsalle): permet l'ajout d'une séance dans le planning candidat.

-actualiserMoniteur(cinM,date,nb):mettre à jour le nombre des élèves dans une séance de code d'un moniteur.

-actualiserCandidat(cin,np,date1,String date2,typ,moni,veh,nb):Cette fonction met à jour les informations d'un candidat.

-supprimerMoniteur(cin,date): permet la suppression d'une séance à une date spécifique du planning moniteur.

-date_change():permet de connaître si la date de la séance a changé ou pas.

-reinitialiser(Vector v):remet les champs dans leur état initial.

-recuperer_champs_du_table(Vector v):permet la récupération des informations du candidat choisi à partir du table.

-recuperer_champs():permet la récupération des informations du candidat lors de la saisie du numéro de CIN dans le champ.

-date_existe(): permet de vérifier si le candidat a déjà une séance a cette date ou non.

-remplirVehicule(): remplit la liste des véhicules disponibles à l'aide de la méthode date_vehicule(vehicule,date) qui permet d'avoir la liste des véhicules qui ne sont pas réservées à la date choisie.

-remplirMoniteur():Cette fonction remplie la liste des moniteurs disponibles à l'aide des méthodes:date_code(cin,nom,date) qui permet d'avoir la liste des moniteurs spécialité code qui sont disponibles à la date choisie et date_conduite(cin,nom,date)qui permet d'avoir la liste des moniteurs spécialité conduite qui sont disponibles à la date choisie.

-remplirsalle():remplit la liste des salles disponibles à l'aide de la méthode date_salle(salle , date) qui permet d'avoir la liste des salles qui ne sont pas réservées à la date choisie .

-etat_son(v) : gère l'état sonore de l'interface.

3.2.6. Examen:

Cette classe permet l'ajout, la modification et la suppression d'un examen ainsi que l'affichage des tous les examens et la recherche des examens selon la date.

- Les attributs :

- cin_candidat(long) : représente le numéro de la carte d'identité du candidat qui doit passer cet examen.
- nom_prenom(String) : représente le nom et le prénom du candidat qui doit passer cet examen.
- type(String) : représente le type d'examen (code ou bien conduite).
- date(String) : représente la date d'examen.
- prix(String) : représente le montant d'examen à payer.
- etat_son(String) : indique l'état sonore de l'interface.

● Les méthodes:

- void ajouter() : permet d'ajouter un examen.
- void modifier() : permet la modification des informations d'un examen.
- void supprimer() : permet la suppression d'un examen.
- void recuperer_champs_du_tableau() : permet de récupérer les informations d'un moniteur choisi à partir du tableau.
- void actualiser() : actualiser le tableau qui affiche tous les moniteurs.
- void reinitialiser() : permet d'initialiser les champs de l'interface.
- void message(String,boolean) : affiche un message sonore.
- boolean verifier() : permet de vérifier la validation de tous les champs obligatoire.
- void recherche() : permet d'afficher tous les fiches de paiement de la date saisie dans le champ de recherche.
- void etat_son() : gère l'état sonore de l'interface.

3.2.7. Bilan:

Cette classe permet l'affichage de la totalité des revenus pour chaque mois et chaque année dès l'ouverture de l'école jusqu'à la date actuelle et la recherche de revenu selon l'année ou le mois.

● Les attributs :
  - etat(String) : indique l'état sonore de l'interface.

● Les méthodes :
  - float calculer_candidat(String,String) : permet de calculer les revenus des candidats.
  - float calculer_moniteur(String,String) : permet de calculer les salaires des moniteurs.
  - float calculer_vehicule(String,String) : permet de calculer les frais des candidats.
  - void ajout_mensuel(String,String) : permet l'enregistrement le revenu d'un mois.
  - void ajout_annuel(String) : permet l'enregistrement le revenu d'une année.
  - void bilan_mensuel() : permet l'enregistrement les revenus des mois depuis l'ouverture de l'école.
  - void bilan_annuel() : permet l'enregistrement les revenus des années depuis l'ouverture de l'école.
  - String get_date_ouverture() : récupérer la date d'ouverture de l'école de la base de données.
  - void actualiser_mensuel() : afficher les revenus mensuels.
  - void actualiser_annuel() : affiche les revenus annuels.
  - Void rechercher() : permet la recherche du revenu d'un mois ou une année spécifique.
  - void audio(boolean) : génère le son à l'affichage des revenus.
  - void etat_son() : gère l'état sonore de l'interface.

3.2.8. Ecole:

Cette classe permet l'enregistrement des informations de l'école ainsi que les informations d'accès à l'application.

- Les attributs :
- nom_ecole(String): représente le nom de l'école.
- adresse(String) : représente l'adresse de l'école
- ouverture(String) : représente la date d'ouverture de l'école.
- nb_salles(int) : représente le nombre des salles dans l'école..
- capacite(int) : représente la capacité des élèves par salle.
- nom_prenom_directeur(String) : représente le nom et le prénom du directeur de l'école.
- utilisateur(String) : représente l'utilisateur de l'application.
- mot_de_passe(String) : représente le mot de passe de l'application pour accéder aux informations.
- question_secert(String) : représente la question de sécurité de l'application au cas ou il a oublié le mot de passe .
- reponse(String) : représente la réponse du question secret.
- Les méthodes :
- void Enregistrer() : permet d'enregistrer tous les informations de l'école.
- boolean verifier() : permet de vérifier la validation de tous les champs obligatoire.
- void recuperer_champs() : permet de récupérer les informations de l'école dans les champs
- void message(String,boolean) : affiche un message sonore..
- void etat_son() : gère l'état sonore de l'interface.

3.2.9. Menu:

Cette classe permet d'accéder aux classes Candidat, Moniteur, Vehicule, Seance, Examen, Bilan et Ecole à partir des boutons.

3.2.10. Acces:

Cette classe permet aux utilisateurs de s'identifier pour pouvoir accéder aux autres interfaces de l'application.

3.2.11. Audio:

Cette classe est utilisée pour ajouter les effets sonores à l'application.

- Les attributs :
  - type(String) : représente le type de l'objet s'il est un message ou un son.
  - message(String) : représente le message à afficher.
  - sound(String) : représente le chemin du son.
  - type_sound(boolean) : indique si le message est d'erreur ou de validation.
- Les méthodes :
  - void set_sound(String,boolean) : permet la modification du son.
  -

3.2.12. Help:

Cette classe est utilisée pour expliquer le mode de fonctionnement de certaines classes.

3.2.13. ConnexionMysql:

Cette classe permet la liaison entre les classes et la base de données.

3.2.14. Accueil:

Cette classe représente la 1ere page qui apparaît à l'ouverture de l'application. Son rôle est esthétique.

4. Manuel d'utilisation

4.1. Environnement logiciel

Pour l'implémentation nous avons utilisé le langage de programmation JAVA avec la bibliothèque SWING et l'environnement de gestion de base de données MYSQL.

4.2. Présentation du travail réalisé

4.2.1. Page d'accueil:

voir image 1 dans l'annexe.

4.2.2. Page d'accès:

voir image 2 dans l'annexe.
En cliquant sur mot de passe oublié dans la page d'accès, une fenêtre contenant le question de sécurité apparaît (voir image 3 dans l'annexe). Si la réponse est vraie, la page de l'école s'ouvre sinon rien ne se passe.

4.2.3. Page du menu:

voir image 4 dans l'annexe.

4.2.4. Page de l'école:

voir image 5 dans l'annexe.

4.2.5. Page du candidat:

voir image 6 dans l'annexe.

4.2.6. Page d'aide:

En cliquant sur le bouton d'aide dans la page du candidat, une page sera affichée (voir image 7 dans l'annexe)

4.2.7. Page du moniteur:

voir image 8 dans l'annexe.

4.2.8. Page du véhicule:

voir image 9 dans l'annexe.

4.2.9. Page des frais du véhicule:

voir image 10 dans l'annexe.

4.2.10. Page de séance:
voir image 11 dans l'annexe.

4.2.11. Page d'examen:
voir image 12 dans l'annexe.

4.2.12. Page du bilan:
voir image 13 dans l'annexe.

5. Les problèmes rencontrés et leurs solutions

Problèmes rencontrés	Solutions proposées
Manque d'informations sur les auto-écoles.	**contacter un directeur d'auto-école.**
Quelques erreurs au programme	**Trouver des solutions sur internet**

6. Références

https://stackoverflow.com/ : 25/08/2020

https://www.developpez.net/ : 23/08/2020

https://www.youtube.com/ : 29/08/2020

https://www.geeksforgeeks.org/ : 05/08/2020