|  |
| --- |
| Documentation Développeur  De l’application JAVA pour la gestion des classes |

|  |
| --- |
| Par Amine NAKHIL, Loïc GUO et Quentin LIGNANI |



**Sommaire :**

**1. Page Admin\_Classe.java**

**2.Page Admin\_Menu.java**

**3. Page Admin\_viesco.java**

**4. Page Connexion.java**

**5. Page Eleves.java**

**6. Page Liste\_Eleve.java**

**7. Page Main.java**

**8. Page Modifier\_eleve.java**

**9. Page Modifier\_prof.java**

**10. Page motif.java**

**11. Page Planning\_prof.java**

**12. Page SelectTable.java**

**13. Page test.java**

**14. Page Testdb.java**

**15. Page Controller\_connexion.java**

**16. Page Controller\_eleve.java**

**17. Page Global.java**

**18. Page lycee\_java.sql**

1. Page Admin\_Classe.java

**Cette page est celle de la vue de l’administrateur sur les élèves. Elle apparait quand on clique dessus dans le menu initial de l’administrateur.**



**Nous importons les librairies de java nécessaires au code. En l’occurrence les librairies qui permettent de faire la jonction avec la base de données. Ensuite, nous importons les packages de l’IDE Eclipse nécessaires comme pour les listes déroulantes, les widgets, les boutons et l’implantation de texte.**



**Déclaration de la méthode « Admin\_Classe » qui hérite de la méthode « Global » dans le cadre du langage JAVA. On déclare ensuite les attributs shlClasse, Database, Connexion et Class. L’attribut shlClasse est accessible en protégé tandis que Classe est en privé. ……………………………………………………………………………………………**



**On déclare une classe de type void accessible en public. …………………………………………**



**On déclare une méthode void en accès protégé. Puis dans un premier temps, on place le texte « Classe », puis un formulaire sous forme de label « Classe : ». Ensuite, on affiche un menu déroulant qui va interroger la base de données pour obtenir la liste des classes au sein du menu déroulant.**



**On fait une boucle dans l’objectif de capturer ce que l’utilisateur a choisi dans le menu déroulant (une absence, une sanction ou un retard). Il faut que l’utilisateur valide son choix.**



**On ajoute un formulaire sous forme de label, dans lequel on écrit « libellé » tout en encodant de la manière nécessaire.**



**On ajoute un bouton de suppression dans lequel, il est logiquement écrit « Supprimer ». On ajoute également un bouton da validation avec écrit « Valider ». On ajoute un formulaire sous forme de label avec écrit dessus « Libellé : » en appliquant l’encodage. On instaure ensuite évidement des boutons qui permettent à l’utilisateur de nous faire part de son choix. On ajoute ensuite un bouton pour ajouter une classe avec écrit logiquement « Ajouter une classe ».**

2.Page Admin\_Menu.java



**Nous importons les librairies de java nécessaires au code. En l’occurrence les librairies de l’IDE Eclipse nécessaires. Par la suite, nous importons les librairies nécessaires au concepteur d’Eclipse Windows Builder. Enfin, nous importons les packages nécessaires comme pour les labels, les widgets, les boutons et l’implantation de texte.**



**On déclare une classe qui porte le nom du fichier à savoir « Admin\_Menu ». Elle hérité de « Global ». Elle est en accès public. On déclare ensuite l’attribut « Menu Admin » de type Shell en accès protégé.**



**Déclaration d’une nouvelle méthode en accès public. ………………..**



**Déclaration d’une nouvelle méthode en accès protégé. On déclare un nouvel attribut de type objet de nom MenuAdmin. Cela permet la déclaration d’un bouton, on y affiche le texte « Menu Admin ». On fait la même chose avec le nom de l’objet « Menu » et en y affichant le texte « Menu ».**



**On met un bouton qui affiche « Gestion utilisateurs ». Ce bouton doit rediriger vers la page « Liste\_Eleve » dans une nouvelle fenêtre. On met tout cela dans une procédure, et on y met la gestion d’erreur pour que si cela échoue, il y ait un message d’erreur.**



**On met un bouton qui affiche « Gestion Eleve ». Ce bouton doit rediriger vers la page « Liste\_Eleve » dans une nouvelle fenêtre. On met tout cela dans une procédure, et on y met la gestion d’erreur pour que si cela échoue, il y ait un message d’erreur.**



**On insère un bouton avec écrit « Modifier le planning »**

1. Page Admin\_viesco.java



**Nous importons les librairies de java nécessaires au code. En l’occurrence les librairies qui permettent de faire la jonction avec la base de données qui permettent l’interaction et l’interrogation des données contenues dans la base. Ensuite, nous importons les packages de l’IDE Eclipse nécessaires comme pour les listes déroulantes, les widgets, les boutons et l’implantation de texte, l’implantation de tableaux, de colonnes et d’éléments de tableaux.**



**On déclare la classe « Admin\_viesco » qui porte également le nom du fichier. Cette méthode est celle du fichier entier. Elle hérite de la méthode « Global ». On déclare également l’attribut ListeDesElves de type Shell et en accès protégé. On déclare également l’attribut « Table » de type tableau et en accès uniquement dans la méthode.**



**On déclare une méthode en accès public. ……………………………………………….**



**On déclare une méthode à l’intérieur du fichier, c’est une méthode void en accès protégé. On instaure un bouton sur lequel est écrit « Liste des Eleves ».**



**On met un widget composite et on indique son emplacement.**



**On déclare un formulaire de type Label, on indique les caractéristiques du texte comme la police d’écriture ainsi que sa couleur. On indique également l’emplacement du label. Le Label se nomme « Nom\_Eleve » et il affiche « Nom » à l’utilisateur.**



**On déclare un formulaire de type Label, on indique les caractéristiques du texte comme la police d’écriture ainsi que sa couleur. On indique également l’emplacement du label. Le Label se nomme « Prenom\_Eleve » et il affiche « Prenom » à l’utilisateur.**





**On déclare un tableau avec SWT. On y insère une colonne « Nom » et un élément « test ».**



**On ajoute un bouton pour modifier le profil, naturellement, on affiche « Modifier votre profil » en son sein.**



**On se connecte à notre base de données. On envoie une requête SQL pour récupérer l’ensemble du contenu des colonnes dans la table Eleve. En clair, on demande l’accès à une liste d’élèves. On demande spécifiquement les colonnes « nom » et « prenom », car ce sont les informations recherchées. On met une gestion d’erreur en cas d’erreur de connexion à la base de données.**



**On demande, si l’utilisateur clique sur le bouton pour ajouter une sanction ou une absence, de le rediriger vers la page « form ». On n’oublie pas la gestion d’erreur.**

1. Page Connexion.java



**Nous importons les librairies de java nécessaires au code. En l’occurrence les librairies qui permettent la gestion des exception SQL. Ensuite, nous importons les packages de l’IDE Eclipse nécessaires pour les widgets tel que les boutons et l’implantation de texte. Mais aussi avec les librairies de l’IDE Eclipse nécessaires pour la gestion des évènements**



**Cette méthode s’exécute comme partie du bloc d’instruction de la page. Cette méthode sert à y incorporer la gestion d’erreur pour la redirection vers la page de connexion. Dans le Try, on insère le code qui permet de rediriger vers la page de connexion et puis si cela échoue, dans le catch, on insère le code qui permet le message d’erreur.**



**Ouverture de l’interface de connexion.**



**On déclare une méthode void en accès protégé. On nomme l’en tête de la fenêtre ouverte.**



**En vert et en haut de la fenêtre, on affiche comme titre « Ma Gestion ». On affiche à l’utilisateur « Identifiant » et en dessous « Mot de passe » sous forme de texte. On prépare le Gestion d’erreur en déclarant l’affichage du message d’erreur en cas de mauvaise saisie. Ce message n’est pas visible par défaut.**



**Le bouton de validation contrôle la validité des informations de connexions fournies par l’utilisateurs, son identifiant et son mot de passe. On y met aussi une gestion d’erreur.**

1. Page Eleves.java









6. Page Liste\_Eleve.java

7. Page Main.java

8. Page Modifier\_eleve.java

9. Page Modifier\_prof.java

10. Page motif.java

11. Page Planning\_prof.java

12. Page SelectTable.java

13. Page test.java

14. Page Testdb.java

15. Page Controller\_connexion.java

16. Page Controller\_eleve.java

17. Page lycee\_java.sql