Documentation ProjetC#

Expliquation des de chaque Classes crée :

Classes Student :

Cette classe représente un étudiant dans le système de gestion scolaire.

Chaque étudiant a des caractéristiques telles que son nom, sa classe actuelle, son niveau d'études, et un identifiant unique (ID).

Les méthodes de cette classe permettent d'ajouter, mettre à jour, supprimer et afficher les détails d'un étudiant.

Attributs de la classe :

StudentID : Un identifiant unique attribué à chaque étudiant.

Password : Le mot de passe associé à l'étudiant pour l'authentification.

Name : Le nom de l'étudiant.

Prenom : Le prénom de l'étudiant.

CurrentClass : La classe actuelle de l'étudiant.

Niveau : Le niveau d'études de l'étudiant.

Special : La spécialité ou filière d'études de l'étudiant.

StudentsList : Une liste statique contenant tous les objets de la classe Student.

Méthode Add :

Permet d'ajouter un nouvel étudiant en demandant à l'utilisateur de saisir les détails tels que le nom, le prénom, la classe, le niveau d'études, la spécialité,

et le mot de passe.

Génère automatiquement un identifiant unique pour l'étudiant.

Ajoute l'étudiant à la liste statique StudentsList.

Méthode Edit :

Permet de modifier les informations d'un étudiant existant.

L'utilisateur doit fournir l'ID de l'étudiant et le mot de passe pour accéder à la modification.

Si les informations fournies sont correctes, l'utilisateur peut saisir de nouvelles informations telles que le nom, le prénom, la classe, le niveau d'études et la spécialité.

Méthode Delete :

Permet de supprimer un étudiant de la liste.

L'utilisateur doit fournir l'ID de l'étudiant et le mot de passe pour accéder à la suppression.

Si les informations fournies correspondent à un étudiant existant, cet étudiant est retiré de la liste.

Méthode ViewDetails :

Permet de visualiser les détails d'un étudiant.

L'utilisateur doit fournir l'ID de l'étudiant et le mot de passe pour accéder aux informations.

Si les informations fournies sont correctes, les détails de l'étudiant (ID, nom, prénom, classe, niveau d'études, spécialité) sont affichés à l'écran.

```
using System;
using System.Collections.Generic;

namespace Project
{
class Student
{
    private static int lastAssignedId = 1000;
    public int StudentID { get; set; }
    public string Password { get; set; }
    public string Name { get; set; }
    public string Prenom { get; set; }
    public string CurrentClass { get; set; }

    public string Niveau { get; set; }

    public string Special { get; set; }


    public static List<Student> StudentsList = new List<Student>();

    public void Add()
    {
        Console.WriteLine("Enter student details:");
        Console.Write("Nom : ");
        Name = Console.ReadLine();
        Console.Write("Prenom: ");
        Prenom = Console.ReadLine();
        Console.Write("ckasse: ");
        CurrentClass = Console.ReadLine();
        Console.Write("niveau d'etude: ");
        Niveau = Console.ReadLine();
```

```
        Console.Write("niveau d'etude: ");
        Niveau = Console.ReadLine();
        Console.Write("etude: ");
        Special = Console.ReadLine();
        StudentID = ++lastAssignedId;
        Console.WriteLine($"ID: {StudentID}");
        Console.Write("Mot de passe: ");
        Password = Console.ReadLine();


        StudentsList.Add(this);

        Console.WriteLine("etudiant ajouté");
    }

    public void Edit()
    {
        Console.Write("entrer l'ID: ");
        int studentIdToEdit = int.Parse(Console.ReadLine());

        Console.Write("le mot de passse: ");
        string passwordToEdit = Console.ReadLine();

        Student studentToEdit = StudentsList.Find(s => s.StudentID == studentIdToEdit && s.Password == passwordToEdit

        if (studentToEdit != null)
        {
            Console.WriteLine("Entrer les nouvelles information:");
            Console.Write("Nom : ");
            Name = Console.ReadLine();
            Console.Write("Prenom: ");
            Prenom = Console.ReadLine();
```

```
            Console.Write("niveau d'etude: ");
            Niveau = Console.ReadLine();
            Console.Write("etude: ");

            Console.WriteLine("etudiant modifié");
        }
        else
        {
            Console.WriteLine("Impossible");
        }
    }

    public void Delete()
    {
        Console.Write("entrer ID: ");
        int studentIdToDelete = int.Parse(Console.ReadLine());

        Console.Write("Entrer mot de passe : ");
        string passwordToDelete = Console.ReadLine();

        Student studentToDelete = StudentsList.Find(s => s.StudentID == studentIdToDelete && s.Password == passwordTc

        if (studentToDelete != null)
        {
            StudentsList.Remove(studentToDelete);
            Console.WriteLine("etudiant effacé");
        }
        else
        {
            Console.WriteLine("Impossible");
        }
    }
```

```
    public void ViewDetails()
    {
        Console.Write("Entrer ID: ");
        int studentIdToView = int.Parse(Console.ReadLine());

        Console.Write("Mot de passe : ");
        string passwordToView = Console.ReadLine();

        Student studentToView = StudentsList.Find(s => s.StudentID == studentIdToView && s.Password == passwordToView

        if (studentToView != null)
        {
            Console.WriteLine("Info Etudiant:");
            Console.WriteLine($"ID: {studentToView.StudentID}");
            Console.WriteLine($"Nom: {studentToView.Name}");
            Console.WriteLine($"Prenom: {studentToView.Prenom}");
            Console.WriteLine($"Classe: {studentToView.CurrentClass}");
            Console.WriteLine($"Niveau d'etude: {studentToView.Niveau}");
            Console.WriteLine($"etude : {studentToView.Special}");
        }
        else
        {
            Console.WriteLine("Impossible");
        }
    }
}

}
```

Classes Teacher :

Cette classe représente un enseignant dans le système.

Chaque enseignant a un nom, un identifiant unique, et une liste de cours qu'il enseigne.

Les méthodes de cette classe permettent d'ajouter,

mettre à jour, supprimer et afficher les détails d'un enseignant.

Le code a une structure similaire a la classe student

Attributs de la classe :

TeacherID : Un identifiant unique attribué à chaque enseignant.

Password : Le mot de passe associé à l'enseignant pour l'authentification.

Name : Le nom de l'enseignant.

Prenom : Le prénom de l'enseignant.

Prof : Le domaine dans lequel l'enseignant donne cours.

TeachersList : Une liste statique contenant tous les objets de la classe Teacher.

Méthode Add :

- Permet d'ajouter un nouvel enseignant en demandant à l'utilisateur de saisir les détails tels que le nom, le prénom, le domaine d'enseignement (Prof), et le mot de passe.
- Génère automatiquement un identifiant unique pour l'enseignant.
- Ajoute l'enseignant à la liste statique TeachersList.

Méthode Edit :

Permet de modifier les informations d'un enseignant existant.

L'utilisateur doit fournir l'ID de l'enseignant et le mot de passe pour accéder à la modification.

Si les informations fournies sont correctes, l'utilisateur peut saisir de nouvelles informations telles que le nom et le

domaine d'enseignement.

Méthode Delete :

Permet de supprimer un enseignant de la liste.

L'utilisateur doit fournir l'ID de l'enseignant pour accéder à la suppression.

Si les informations fournies correspondent à un enseignant existant, cet enseignant est retiré de la liste.

Méthode ViewDetails :

Permet de visualiser les détails d'un enseignant.

L'utilisateur doit fournir l'ID de l'enseignant pour accéder aux informations.

Si les informations fournies sont correctes, les détails de l'enseignant (ID, nom, prénom, domaine d'enseignement) sont affichés à l'écran.

-

```
using System;
using System.Collections.Generic;

namespace Project
{
class Teacher
{
    private static int lastAssignedId = 2000;
    public int TeacherID { get; set; }
    public string Password { get; set; }
    public string Name { get; set; }
    public string Prenom { get; set; }
    public string Prof { get; set; }

    public static List<Teacher> TeachersList = new List<Teacher>();

    public void Add()
    {
        Console.WriteLine("Enter teacher details:");
        Console.Write("Name: ");
        Name = Console.ReadLine();
        Console.Write("Prenom: ");
        Prenom = Console.ReadLine();
        Console.Write("Prof de: ");
        Prof = Console.ReadLine();
        TeacherID = ++lastAssignedId;
        Console.WriteLine($"ID: {TeacherID}");
        Console.Write("mot de passe: ");
        Password = Console.ReadLine();


        TeachersList.Add(this);
```

```
        TeachersList.Add(this);

        Console.WriteLine("proffeseur ajouté");
    }

    public void Edit()
    {
        Console.Write("ID du Prof: ");
        int teacherIdToEdit = int.Parse(Console.ReadLine());

        Console.Write("Entrer mot de passe: ");
        string passwordToEdit = Console.ReadLine();

        Teacher teacherToEdit = TeachersList.Find(t => t.TeacherID == teacherIdToEdit && t.Password == passwordToEdit

        if (teacherToEdit != null)
        {
            Console.WriteLine("nouvelle info:");
            Console.Write("Nom: ");
            teacherToEdit.Name = Console.ReadLine();
            Console.Write("Prof de: ");
            teacherToEdit.Prof = Console.ReadLine();


            Console.WriteLine("Prof modifié");
        }
        else
        {
            Console.WriteLine("Introuvable");
        }
    }
```

```
    public void Delete()
    {
        Console.Write("ID: ");
        int teacherIdToDelete = int.Parse(Console.ReadLine());

        Teacher teacherToDelete = TeachersList.Find(t => t.TeacherID == teacherIdToDelete);

        if (teacherToDelete != null)
        {
            TeachersList.Remove(teacherToDelete);
            Console.WriteLine("effacé !");
        }
        else
        {
            Console.WriteLine("Introuvable");
        }
    }

    public void ViewDetails()
    {
        Console.Write("ID: ");
        int teacherIdToView = int.Parse(Console.ReadLine());

        Teacher teacherToView = TeachersList.Find(t => t.TeacherID == teacherIdToView);

        if (teacherToView != null)
        {
            Console.WriteLine("Info du prof:");
            Console.WriteLine($"ID: {teacherToView.TeacherID}");
            Console.WriteLine($"Nom: {teacherToView.Name}");
            Console.WriteLine($"Prof de: {teacherToView.Prof}");
        }
```

Classes Course :

Cette permet la Gestion des cours du classe c’est a dire l’ajout d’un cours la modification et la suppression d’un cours

Attribut de la classe :

CoursesDirectory : Une constante représentant le répertoire où seront stockés les fichiers des cours.

Méthode AddCourse :

Permet d'ajouter un nouveau cours en créant un fichier texte pour stocker les détails du cours.

Le nom du cours est utilisé comme nom de fichier, et les détails du cours sont écrits dans ce fichier.

Vérifie d'abord si le répertoire des cours existe, et le crée s'il n'existe pas.

Vérifie si le fichier du cours existe avant de créer un nouveau fichier.

Méthode UpdateCourse :

Permet de mettre à jour les détails d'un cours existant en remplaçant le contenu du fichier correspondant.

Vérifie si le fichier du cours existe avant de le mettre à jour.

Méthode DeleteCourse :

Permet de supprimer un cours en supprimant le fichier correspondant.

Vérifie si le fichier du cours existe avant de le supprimer.

Méthode ViewCourseContent :

Permet de visualiser le contenu d'un cours en lisant le fichier correspondant.

Affiche le contenu du fichier à la console.

Vérifie si le fichier du cours existe avant de lire son contenu.

Le code utilise un répertoire spécifique (CoursesDirectory) pour stocker les fichiers des cours. Si ce répertoire n'existe pas, il est créé.

Les noms des fichiers des cours sont générés en utilisant le nom du cours fourni, avec l'extension .txt.

Les opérations de gestion des cours sont effectuées directement sur le système de fichiers, sans utiliser de base de données

```
using System;
using System.Collections.Generic;
using System.IO;

namespace Project
{
class CourseManager
{
    private const string CoursesDirectory = "Cours";

    public void AddCourse(string courseName, string courseDetails)
    {
        string courseFileName = $"{CoursesDirectory}\\{courseName}.txt";

        if (!Directory.Exists(CoursesDirectory))
        {
            Directory.CreateDirectory(CoursesDirectory);
        }

        if (!File.Exists(courseFileName))
        {
            using (StreamWriter sw = File.CreateText(courseFileName))
            {
                sw.Write(courseDetails);
            }

            Console.WriteLine($"Cours de '{courseName}' jouté");
        }
        else
        {
            Console.WriteLine($"Cours de '{courseName}' existe deja");
        }
```

```
public void UpdateCourse(string courseName, string newCourseDetails)
{
    string courseFileName = $"{CoursesDirectory}\\{courseName}.txt";

    if (File.Exists(courseFileName))
    {
        File.WriteAllText(courseFileName, newCourseDetails);
        Console.WriteLine($"Cours '{courseName}' modifié.");
    }
    else
    {
        Console.WriteLine($"Cours '{courseName}' n'existe pas");
    }
}

public void DeleteCourse(string courseName)
{
    string courseFileName = $"{CoursesDirectory}\\{courseName}.txt";

    if (File.Exists(courseFileName))
    {
        File.Delete(courseFileName);
        Console.WriteLine($"Cours '{courseName}' effacé");
    }
    else
    {
        Console.WriteLine($"Cours '{courseName}' n'existe pas.");
    }
}
    public void ViewCourseContent(string courseName)
{
    string courseFileName = $"{CoursesDirectory}\\{courseName}.txt";
```

```
    public void ViewCourseContent(string courseName)
    {
        string courseFileName = $"{CoursesDirectory}\\{courseName}.txt";

        if (File.Exists(courseFileName))
        {
            string content = File.ReadAllText(courseFileName);
            Console.WriteLine($"contenu du cours de '{courseName}':");
            Console.WriteLine(content);
        }
        else
        {
            Console.WriteLine($"Cours '{courseName}' n'existe pas.");
        }
    }
}
}
```

Classes Administration :

Cette represente admin qui resenteraint en fait le directeur ou un haut placé de l'établissement ayant tout les droits

Méthode ManageStudents :

Permet à l'administrateur de gérer les étudiants en fournissant un menu interactif.

L'utilisateur peut ajouter, modifier, voir les détails ou supprimer un étudiant.

La gestion des étudiants se fait en utilisant les méthodes de la classe Student que vous avez précédemment partagée.

Méthode ManageTeachers :

Permet à l'administrateur de gérer les enseignants en fournissant un menu interactif.

L'utilisateur peut ajouter, modifier, voir les détails ou supprimer un enseignant.

La gestion des enseignants se fait en utilisant les méthodes de la classe Teacher que vous avez précédemment partagée.

Méthode CalendarMenu :

Permet à l'administrateur de gérer le calendrier en fournissant un menu interactif.

L'utilisateur peut changer la date d'un examen, la limite d'un cours, ou revenir au menu précédent.

Les opérations sur le calendrier sont effectuées en utilisant les méthodes de la classe Calendar.

Méthode CoursesMenu :

Permet à l'administrateur de gérer les cours en fournissant un menu interactif.

L'utilisateur peut ajouter, modifier, mettre à jour, supprimer un cours ou voir le contenu d'un cours.

Les opérations sur les cours sont effectuées en utilisant les méthodes de la classe CourseManager.

Chaque menu est présenté dans une boucle qui se répète jusqu'à ce que l'utilisateur choisisse l'option de sortie.

L'objet Student, Teacher, Calendar, et CourseManager est instancié à l'intérieur de chaque méthode pour permettre l'utilisation des méthodes associées.

Les saisies de l'utilisateur sont vérifiées pour s'assurer qu'elles sont valides.

Les méthodes des classes associées (Student, Teacher, Calendar, et CourseManager) sont utilisées pour effectuer les opérations appropriées en fonction du choix de l'utilisateur.

```
using System;
using System.Collections.Generic;

namespace Project
{
class Administration
{

    public void ManageStudents()
    {
        Student student = new Student();

        bool exit = false;
        while (!exit)
        {
            Console.WriteLine("Ajouter un eleve ");
            Console.WriteLine("Modifier un eleve");
            Console.WriteLine("Information sur un eleve");
            Console.WriteLine("Effacer un Eleve");

            Console.WriteLine("Retour");
            Console.Write("choissisez un nombre: ");

            int choice;
            if (int.TryParse(Console.ReadLine(), out choice))
            {
                switch (choice)
                {
                    case 1:
                        student.Add();
                        break;
                    case 2:
```

```
                    student.Add();
                    break;
                case 2:
                    student.Edit();
                    break;
                case 3:
                    student.ViewDetails();
                    break;
                case 4:
                    student.Delete();
                    break;
                case 5:
                    exit = true;
                    break;
                default:
                    Console.WriteLine("Nombre impossible");
                    break;
            }
        }
        else
        {
            Console.WriteLine("choisis un nombre de 1-5");
        }

        Console.WriteLine();
    }
}

public void ManageTeachers()
{
    Teacher teacher = new Teacher();
```

```
Teacher teacher = new Teacher();

bool exit = false;
while (!exit)
{
    Console.WriteLine("Ajouter un Prof");
    Console.WriteLine("Modifier un Prof");
    Console.WriteLine("Voir un Prof");
    Console.WriteLine("Effacer un Prof");
    Console.WriteLine("Retour");
    Console.Write("Choissisez un Nombre: ");

    int choice;
    if (int.TryParse(Console.ReadLine(), out choice))
    {
        switch (choice)
        {
            case 1:
                teacher.Add();
                break;
            case 2:
                teacher.Edit();
                break;
            case 3:
                teacher.ViewDetails();
                break;
            case 4:
                teacher.Delete();
                break;
            case 5:
                exit = true;
                break;
```

```
                    Console.WriteLine("Nombre impossible");
                    break;
                }
            }
            else
            {
                Console.WriteLine("Nombre impossible");
            }

            Console.WriteLine();
        }
    }
    public void CalendarMenu(Calendar calendar, Administration admin)
    {
        bool exit = false;
        while (!exit)
        {
            Console.WriteLine("-1 Gérer le Calendrier");
            Console.WriteLine("-2 Changer un date d'examin");
            Console.WriteLine("-3 Changer la limite d'un cours");
            Console.WriteLine("-4 Retour");
            Console.Write("Choix: ");

            int choice;
            if (int.TryParse(Console.ReadLine(), out choice))
            {
                switch (choice)
                {
                    case 1:
                        Console.Write("Entrer le nom Examin: ");
                        string examName = Console.ReadLine();
                        Console.Write("Entrer une nouvelle date (dd/mm/yyyy): ");
                        if (DateTime.TryParse(Console.ReadLine(), out DateTime newExamDate))
```

```
                    calendar.ChangeExamDate(examName, newExamDate, admin);
                }
                else
                {
                    Console.WriteLine("format Invalide");
                }
                break;
            case 2:
                Console.Write("Nom du cours: ");
                string courseName = Console.ReadLine();
                Console.Write("limite cours (dd/mm/yyyy): ");
                if (DateTime.TryParse(Console.ReadLine(), out DateTime newDeadline))
                {
                    calendar.ChangeCourseDeadline(courseName, newDeadline, admin);
                }
                else
                {
                    Console.WriteLine("impossible");
                }
                break;
            case 3:
                exit = true;
                break;
            default:
                Console.WriteLine("impossible");
                break;
        }
    }
    else
    {
        Console.WriteLine("impossible");
    }
```

```
            {
                Console.WriteLine("impossible");
            }

            Console.WriteLine();
        }
    }

    public void CoursesMenu(CourseManager courseManager)
    {
        bool exit = false;
        while (!exit)
        {
            Console.WriteLine("gerer les cours");
            Console.WriteLine("Modifier Cours");
            Console.WriteLine("update des Cours");
            Console.WriteLine("Effacer Cours");
            Console.WriteLine("voir un cours");
            Console.WriteLine("Retour");
            Console.Write("Choix :");

            int choice;
            if (int.TryParse(Console.ReadLine(), out choice))
            {
                switch (choice)
                {
                    case 1:
                        Console.Write("Nom du cours : ");
                        string courseName = Console.ReadLine();
                        Console.Write("details du cours: ");
                        string courseDetails = Console.ReadLine();
                        courseManager.AddCourse(courseName, courseDetails);
                        break;
```

```
                    Console.Write("Nom du cours : ");
                    string courseName = Console.ReadLine();
                    Console.Write("details du cours: ");
                    string courseDetails = Console.ReadLine();
                    courseManager.AddCourse(courseName, courseDetails);
                    break;
                case 2:
                    Console.Write("Nom du cours: ");
                    string updateCourseName = Console.ReadLine();
                    Console.Write("entrer le cours updated: ");
                    string updatedCourseDetails = Console.ReadLine();
                    courseManager.UpdateCourse(updateCourseName, updatedCourseDetails);
                    break;
                case 3:
                    Console.Write("Nom du cours a effacer: ");
                    string deleteCourseName = Console.ReadLine();
                    courseManager.DeleteCourse(deleteCourseName);
                    break;
                case 4:
                    Console.Write("choissisez le cours: ");
                    string viewContentCourseName = Console.ReadLine();
                    courseManager.ViewCourseContent(viewContentCourseName);
                    break;
                case 5:
                    exit = true;
                    break;
                default:
                    Console.WriteLine("Impossible");
                    break;
            }
        }
        else
        {
```

```
        }
        else
        {
            Console.WriteLine("Impossible.");
        }

        Console.WriteLine();
    }
}


}
}
```