Explication des classes

I - Administration

classe appelée **Administration** dans un programme C#. Cette classe contient des méthodes spécifiques pour la gestion des étudiants, des enseignants, des calendriers et des cours. Voici une explication détaillée de chaque méthode :

**Méthode ManageStudents()**

Cette méthode permet à l'utilisateur de gérer les étudiants.

Elle affiche un menu proposant différentes actions pour les étudiants : ajouter un étudiant, modifier ses informations, afficher ses détails, supprimer un étudiant, ou retourner au menu principal.

Chaque action est réalisée en appelant les méthodes correspondantes de l'objet **Student**.

**Méthode ManageTeachers()**

Cette méthode permet à l'utilisateur de gérer les enseignants.

Elle affiche un menu similaire avec des options pour ajouter un enseignant, modifier ses informations, afficher ses détails, supprimer un enseignant, ou retourner au menu principal.

Chaque action est réalisée en appelant les méthodes correspondantes de l'objet **Teacher**.

**Méthode CalendarMenu(Calendar calendar, Administration admin)**

Cette méthode gère les opérations liées au calendrier, telles que la modification de la date d'un examen ou de la date limite d'un cours.

Elle affiche un menu proposant des actions pour changer la date d'un examen, modifier la date limite d'un cours, ou retourner au menu principal.

En fonction du choix de l'utilisateur, elle appelle les méthodes appropriées de l'objet **Calendar** pour effectuer les modifications nécessaires.

**Méthode CoursesMenu(CourseManager courseManager)**

Cette méthode gère les opérations liées à la gestion des cours.

Elle affiche un menu proposant d'ajouter un cours, de mettre à jour un cours, de supprimer un cours, d'afficher les détails d'un cours, ou de retourner au menu principal.

Chaque action est réalisée en appelant les méthodes correspondantes de l'objet **CourseManager**.

En résumé, la classe **Administration** agit comme un contrôleur pour gérer les différents aspects liés aux étudiants, enseignants, calendriers et cours en utilisant des objets spécialisés (**Student**, **Teacher**, **Calendar**, **CourseManager**). Elle offre une interface utilisateur via la console pour interagir avec ces différents éléments en fonction des choix de l'utilisateur.

II- calendar

Cette classe est conçue pour gérer les dates d'examens et les dates limites des cours en utilisant des dictionnaires pour stocker ces informations. Voici une explication détaillée des différentes parties de la classe :

Attributs de la classe `Calendar`

- `examDates` : Un dictionnaire utilisé pour stocker les noms des examens en tant que clés et les dates correspondantes en tant que valeurs de type `DateTime`.

- `courseDeadlines` : Un dictionnaire utilisé pour stocker les noms des cours en tant que clés et les dates limites correspondantes en tant que valeurs de type `DateTime`.

### Constructeur `Calendar()`

- Initialise les dictionnaires `examDates` et `courseDeadlines` en créant de nouvelles instances de `Dictionary<string, DateTime>()`.

### Méthode `DisplayExamDates()`

- Affiche les dates des examens stockées dans le dictionnaire `examDates` en parcourant chaque élément et en affichant le nom de l'examen avec sa date associée sous forme de texte.

### Méthode `DisplayCourseDeadlines()`

- Affiche les dates limites des cours stockées dans le dictionnaire `courseDeadlines` en parcourant chaque élément et en affichant le nom du cours avec sa date limite associée sous forme de texte.

### Méthode `ChangeExamDate(string examName, DateTime newDate, Administration admin)`

- Permet de modifier la date d'un examen spécifique.

- Si l'objet `Administration` passé en paramètre n'est pas null (c'est-à-dire qu'il est différent de null), cela signifie qu'un administrateur est authentifié et autorisé à effectuer des modifications.

- Si l'administrateur est autorisé, la méthode met à jour la date de l'examen correspondant au nom de l'examen donné avec la nouvelle date spécifiée.

### Méthode `ChangeCourseDeadline(string courseName, DateTime newDeadline, Administration admin)`

- Permet de modifier la date limite d'un cours spécifique.

- Comme pour la méthode de changement de date d'examen, cette méthode vérifie également si l'objet `Administration` passé en paramètre n'est pas null pour déterminer l'autorisation de modification.

- Si l'administrateur est autorisé, la méthode met à jour la date limite du cours correspondant au nom de cours donné avec la nouvelle date limite spécifiée.

En résumé, la classe `Calendar` permet de stocker et de gérer les dates d'examens et les dates limites des cours en utilisant des dictionnaires. Elle offre des fonctionnalités pour afficher ces dates et pour permettre à un administrateur authentifié de les modifier.

III- Coursemanager :

`CourseManager` . Cette classe est conçue pour gérer des opérations de base sur des cours, telles que l'ajout, la mise à jour, la suppression et l'affichage de leur contenu. Voici une explication détaillée des différentes parties de la classe :

### Attribut de classe `CoursesDirectory`

- `CoursesDirectory` est une constante de type string qui contient le nom du répertoire où les fichiers des cours seront stockés. Dans cet exemple, le nom du répertoire est "Cours".

### Méthode `AddCourse(string courseName, string courseDetails)`

- Cette méthode prend en paramètres le nom du cours (`courseName`) et les détails du cours (`courseDetails`).

- Elle crée un chemin de fichier en combinant le répertoire des cours avec le nom du fichier correspondant au cours à l'aide de `CoursesDirectory`.

- Vérifie d'abord si le répertoire des cours existe. Si ce n'est pas le cas, il est créé.

- Ensuite, vérifie si le fichier du cours existe déjà. S'il n'existe pas, crée un nouveau fichier texte avec les détails du cours. Sinon, affiche un message indiquant que le cours existe déjà.

### Méthode `UpdateCourse(string courseName, string newCourseDetails)`

- Cette méthode met à jour le contenu d'un cours existant avec de nouveaux détails.

- Elle vérifie si le fichier du cours existe. Si oui, elle remplace le contenu existant du fichier par les nouveaux détails du cours en utilisant `File.WriteAllText`. Sinon, elle affiche un message indiquant que le cours n'existe pas.

### Méthode `DeleteCourse(string courseName)`

- Cette méthode supprime un cours existant.

- Elle vérifie si le fichier du cours existe. Si oui, elle supprime le fichier du cours en utilisant `File.Delete`. Sinon, elle affiche un message indiquant que le cours n'existe pas.

### Méthode `ViewCourseContent(string courseName)`

- Cette méthode affiche le contenu d'un cours spécifique.

- Elle vérifie si le fichier du cours existe. Si oui, elle lit le contenu du fichier à l'aide de `File.ReadAllText` et l'affiche à la console. Sinon, elle affiche un message indiquant que le cours n'existe pas.

En résumé, la classe `CourseManager` offre des fonctionnalités pour gérer les cours en les ajoutant, les mettant à jour, les supprimant et en affichant leur contenu à partir de fichiers texte stockés dans un répertoire spécifique. Elle utilise les classes du namespace `System.IO` pour interagir avec le système de fichiers et les opérations de lecture et d'écriture de fichiers.

IV-programme

Ce code représente un programme principal dans un environnement de gestion scolaire en C#. Il utilise différentes classes (Student, Teacher, CourseManager, Calendar, Administration) pour gérer les étudiants, les enseignants, les cours, le calendrier, etc.

### Explication du code :

#### `Program` (classe principale)

- \*\*Attributs statiques :\*\*

- `courseManager`: Une instance statique de la classe `CourseManager`, responsable de la gestion des cours.

- \*\*Méthodes statiques :\*\*

- `StudentMenu(Student student)`: Gère les opérations sur les étudiants en affichant un menu avec des options telles que l'ajout, la modification, la visualisation et la suppression des étudiants.

- `TeacherMenu(Teacher teacher)`: Gère les opérations sur les enseignants en affichant un menu avec des options similaires à celles des étudiants.

- `AdminMenu(Administration admin, Calendar calendar)`: Gère les opérations administratives en affichant un menu pour gérer les étudiants, les enseignants, le calendrier et les cours.

- `Main()`: Méthode principale du programme. Elle crée des instances de différentes classes nécessaires, puis affiche un menu principal pour sélectionner des options telles que la gestion des étudiants, des enseignants, l'administration ou quitter le programme.

### Déroulement du programme :

1. Le programme démarre en affichant un menu principal avec des options pour gérer les étudiants, les enseignants, l'administration ou quitter le programme.

2. En fonction du choix de l'utilisateur, il accède à différentes sous-méthodes de gestion : gestion des étudiants, gestion des enseignants, menu d'administration pour gérer le calendrier et les cours.

3. Chaque menu spécifique offre des options pour effectuer des opérations comme l'ajout, la modification, la visualisation et la suppression d'étudiants, d'enseignants, de dates de calendrier et de cours, en fonction du contexte spécifique.

En somme, ce programme est conçu pour offrir une interface utilisateur en ligne de commande pour gérer les différentes entités (étudiants, enseignants, calendrier, cours) dans un système de gestion scolaire en utilisant les différentes classes spécialisées définies dans le code.

V-Student

Ce code définit une classe nommée `Student` dans le cadre d'un programme de gestion d'étudiants. Cette classe possède des propriétés représentant les détails d'un étudiant tels que son ID, son nom, son prénom, sa classe actuelle, son niveau et sa spécialité. Voici une explication détaillée du code :

- \*\*Propriétés de la classe `Student`\*\* :

- `StudentID` : Identifiant de l'étudiant.

- `Password` : Mot de passe de l'étudiant pour effectuer des opérations.

- `Name` : Nom de l'étudiant.

- `Prenom` : Prénom de l'étudiant.

- `CurrentClass` : Classe actuelle de l'étudiant.

- `Niveau` : Niveau d'études de l'étudiant.

- `Special` : Spécialité ou domaine d'études de l'étudiant.

- \*\*Attribut statique et Liste de Students\*\* :

- `lastAssignedId` : Un attribut statique utilisé pour attribuer automatiquement des identifiants aux étudiants créés. Il commence à partir de 1000.

- `StudentsList` : Une liste statique qui stocke les instances de la classe `Student`. Elle est utilisée pour gérer tous les étudiants créés.

- \*\*Méthodes de la classe `Student`\*\* :

- `Add()` : Permet d'ajouter un nouvel étudiant en demandant à l'utilisateur de saisir les détails comme le nom, le prénom, la classe, le niveau, la spécialité et en générant automatiquement un identifiant unique.

- `Edit()` : Permet de modifier les détails d'un étudiant en demandant son ID et son mot de passe, puis en permettant la modification des informations si les identifiants sont valides.

- `Delete()` : Permet de supprimer un étudiant de la liste en demandant son ID et son mot de passe, puis en retirant l'étudiant correspondant si les identifiants sont valides.

- `ViewDetails()` : Affiche les détails d'un étudiant en demandant son ID et son mot de passe, puis en affichant les informations de l'étudiant correspondant si les identifiants sont valides.

### Fonctionnement global :

La classe `Student` permet de manipuler des objets étudiant en offrant des fonctionnalités pour ajouter, modifier, supprimer et afficher les détails des étudiants. Les informations des étudiants sont stockées dans une liste statique `StudentsList`. Les différentes méthodes offrent une interaction avec ces données en répondant aux entrées de l'utilisateur pour effectuer les opérations appropriées sur les objets `Student`.

VI- Teacher

Ce code définit une classe nommée `Teacher` dans le cadre d'un programme de gestion des enseignants. Voici une explication détaillée du code :

- \*\*Propriétés de la classe `Teacher`\*\* :

- `TeacherID` : Identifiant unique de l'enseignant.

- `Password` : Mot de passe de l'enseignant pour effectuer des opérations.

- `Name` : Nom de l'enseignant.

- `Prenom` : Prénom de l'enseignant.

- `Prof` : Domaine ou sujet enseigné par l'enseignant.

- \*\*Attribut statique et Liste des Enseignants\*\* :

- `lastAssignedId` : Un attribut statique utilisé pour attribuer automatiquement des identifiants uniques aux enseignants créés. Il commence à partir de 2000.

- `TeachersList` : Une liste statique qui stocke les instances de la classe `Teacher`. Elle est utilisée pour gérer tous les enseignants créés.

- \*\*Méthodes de la classe `Teacher`\*\* :

- `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 et en générant automatiquement un identifiant unique.

- `Edit()` : Permet de modifier les détails d'un enseignant en demandant son ID et son mot de passe, puis en permettant la modification des informations si les identifiants sont valides.

- `Delete()` : Permet de supprimer un enseignant de la liste en demandant son ID, puis en retirant l'enseignant correspondant si l'ID est trouvé dans la liste.

- `ViewDetails()` : Affiche les détails d'un enseignant en demandant son ID, puis en affichant les informations de l'enseignant correspondant si l'ID est trouvé dans la liste.

### Fonctionnement global :

La classe `Teacher` permet de manipuler des objets enseignant en offrant des fonctionnalités pour ajouter, modifier, supprimer et afficher les détails des enseignants. Les informations des enseignants sont stockées dans une liste statique `TeachersList`. Les différentes méthodes offrent une interaction avec ces données en répondant aux entrées de l'utilisateur pour effectuer les opérations appropriées sur les objets `Teacher`.