WORKSHOP

Tout afficher Tout réduire

SUJET

Énoncé

EXERCICE : RAPPELS POO - CRÉATION D'UN GESTIONNAIRE DE BIBLIOTHÈQUE NUMÉRIQUE EN C# .NET

Objectif : Créer une application console en C# pour gérer une bibliothèque numérique en suivant les principes de la programmation orientée objet (POO). Ce workshop vous guidera étape par étape, avec des explications détaillées, des commandes spécifiques à exécuter, et des questions pour vérifier la compréhension.

Introduction (15 minutes)

Introduction (15 minutes)

Présentation des objectifs :

1. **Créer une application console** qui permet d'ajouter, supprimer, rechercher, emprunter et retourner des livres.
2. **Appliquer les principes POO** : classes, objets, encapsulation, héritage et polymorphisme.
3. **Utiliser la navigation via les flèches** dans la console pour minimiser les saisies au clavier, en se concentrant sur l'expérience utilisateur (UX).
4. **Favoriser l'autonomie des participants** en leur posant des questions pour vérifier leur compréhension et en les guidant progressivement.

Rappel des concepts POO :

- **Classes et Objets :** Une classe est une structure qui définit un modèle pour créer des objets. Un objet est une instance d'une classe qui possède des caractéristiques (propriétés) et des comportements (méthodes).
- **Encapsulation :** Principe qui consiste à protéger les données en limitant leur accès direct depuis l'extérieur de la classe. On utilise souvent des propriétés (`get` et `set`) pour contrôler cet accès.
- **Héritage :** Mécanisme qui permet à une classe de dériver d'une autre classe, héritant ainsi de ses propriétés et méthodes. L'héritage permet la réutilisation du code.
- **Polymorphisme :** Capacité d'une méthode à se comporter différemment en fonction du contexte ou des objets sur lesquels elle est appliquée. C'est un concept clé pour rendre le code flexible et maintenable.

Structure du Projet

Structure du Projet

Le projet C# sera composé de trois fichiers principaux :

1. **`Livre.cs`** : Définition de la classe `Livre`.
2. **`Bibliotheque.cs`** : Définition de la classe `Bibliotheque` qui gère les livres.
3. **`Program.cs`** : Point d'entrée de l'application, gère la navigation du menu.

Étape 1 : Création du projet (10 minutes)

Étape 1 : Création du projet (10 minutes)

1.**Ouvrez un terminal ou une console.**

2.**Créez un nouveau projet console :**

1. - **Commande à exécuter :**

```
dotnet new console -n GestionBibliotheque
```

3.**Naviguez dans le répertoire du projet :**

1. **Commande à exécuter :**

```
cd GestionBibliotheque
```

Étape 2 : Création de la classe `Livre` (20 minutes)

Étape 2 : Création de la classe `Livre` (20 minutes)

1.**Créez un fichier nommé`Livre.cs`dans le répertoire du projet.** Ce fichier contiendra la définition de la classe `Livre`.

2.**Dans ce fichier`Livre.cs`,définissez la classe`Livre`avec les propriétés suivantes :**

1. **Propriétés :**
   - `Titre` (type `string`) : Le titre du livre.
   - `Auteur` (type `string`) : L'auteur du livre.
   - `EstDisponible` (type `bool`, initialisé à `true`) : Indique si le livre est disponible ou emprunté.
   - `NomEmprunteur` (type `string`) : Le nom de la personne qui a emprunté le livre.
   - `DateEmprunt` (type `DateTime?`) : La date à laquelle le livre a été emprunté.
   - `DateRetour` (type `DateTime?`) : La date prévue de retour du livre.

3.**Ajoutez un constructeur à la classe`Livre`qui accepte deux paramètres :`titre`et`auteur`.** Ce constructeur doit initialiser les propriétés `Titre` et `Auteur`.

4.**Redéfinissez la méthode`ToString`pour afficher les informations du livre sous une forme lisible.** Cette méthode doit renvoyer une chaîne de caractères décrivant le titre, l'auteur, et l'état du livre (disponible ou emprunté, avec les détails de l'emprunt).

Questions :

- Pourquoi est-il important d'avoir un constructeur dans cette classe ?
- Comment la méthode `ToString()` améliore-t-elle la lisibilité des objets `Livre` ?

Test

Test :

Voici le code que vous pouvez ajouter dans le fichier `Program.cs` pour tester la classe `Livre` que vous venez de définir. Ce code vous permettra de créer un objet `Livre`, puis d'afficher ses informations en utilisant la méthode `ToString()`.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        // Création d'un objet Livre
        Livre livreTest = new Livre("The Dark Forest", "Cixin Liu");

        // Affichage des informations du livre en utilisant la méthode ToString()
        Console.WriteLine(livreTest.ToString());

        // Attente d'une entrée utilisateur pour garder la console ouverte
        Console.ReadLine();
    }
}
```

Résultat attendu :

Lorsque vous exécutez ce code, vous devriez voir quelque chose comme ceci s'afficher dans la console :

```
The Dark Forest par Cixin Liu - Disponible
```

Cela confirme que la classe `Livre` a été correctement définie et que la méthode `ToString()` fonctionne comme prévu.

Étape 3 : Création de la classe `Bibliotheque` (30 minutes)

Étape 3 : Création de la classe `Bibliotheque` (30 minutes)

1.**Créez un fichier nommé`Bibliotheque.cs`dans le répertoire du projet.** Ce fichier contiendra la définition de la classe `Bibliotheque`.

2.**Dans`Bibliotheque.cs`,définissez la classe`Bibliotheque`qui contiendra une collection de livres.** Utilisez une `List<Livre>` pour stocker ces livres.

3.**Ajoutez les méthodes suivantes à la classe`Bibliotheque` :**

1. - `AjouterLivre(Livre livre)` : Cette méthode doit ajouter un livre à la collection et afficher un message confirmant l'ajout.
   - `SupprimerLivre(string titre)` : Cette méthode doit supprimer un livre par son titre et afficher un message confirmant la suppression. Si le livre n'est pas trouvé, affichez un message approprié.
   - `RechercherLivre(string titre)` : Cette méthode doit rechercher un livre par son titre et renvoyer l'objet `Livre` correspondant.
   - `EmprunterLivre(string titre, string nomEmprunteur)` : Cette méthode doit permettre d'emprunter un livre (changer son état à "emprunté", enregistrer le nom de l'emprunteur et les dates).
   - `RetournerLivre(string titre)` : Cette méthode doit permettre de retourner un livre (le marquer comme disponible et réinitialiser les informations d'emprunt).
   - `AfficherLivres()` : Cette méthode doit afficher tous les livres présents dans la collection.

Questions :

- Pourquoi est-il préférable d'utiliser une `List<Livre>` pour stocker les livres dans la bibliothèque ?
- Comment gérez-vous les erreurs, comme lorsqu'un livre n'est pas trouvé dans la collection ?

Test

Test :

- Dans le fichier `Program.cs`, testez chaque méthode de la classe `Bibliotheque` en ajoutant, supprimant, recherchant, empruntant et retournant des livres. Assurez-vous que chaque méthode fonctionne comme prévu. Vous pouvez utiliser ce code pour vérifier rapidement que les méthodes de la classe Bibliotheque fonctionnent correctement.

```
using System;

class Program

{

    static void Main(string[] args)

   {

        // Création d'une instance de la classe Bibliotheque

        Bibliotheque bibliotheque = new Bibliotheque();

         // Test de la méthode AjouterLivre

        bibliotheque.AjouterLivre(new Livre("The Dark Forest", "Cixin Liu"));

        bibliotheque.AjouterLivre(new Livre("Peste", "Chuck Palahniuk"));

         // Affichage des livres après ajout

        Console.WriteLine("Livres après ajout :");

        bibliotheque.AfficherLivres();

         // Test de la méthode RechercherLivre

        Console.WriteLine("\nRecherche du livre 'Peste' :");

        Livre livreTrouve = bibliotheque.RechercherLivre("1984");

        if (livreTrouve != null) Console.WriteLine(livreTrouve);

         // Test de la méthode EmprunterLivre

        Console.WriteLine("\nEmprunt du livre 'The Dark Forest' :");

        bibliotheque.EmprunterLivre("The Dark Forest", "Amir");

         // Affichage des livres après emprunt

        Console.WriteLine("\nLivres après emprunt :");
```

```
        bibliotheque.AfficherLivres();
        // Test de la méthode RetournerLivre
        Console.WriteLine("\nRetour du livre 'The Dark Forest' :");
        bibliotheque.RetournerLivre("The Dark Forest ");
        // Affichage des livres après retour
        Console.WriteLine("\nLivres après retour :");
        bibliotheque.AfficherLivres();
        // Test de la méthode SupprimerLivre
        Console.WriteLine("\nSuppression du livre 'The Dark Forest' :");
        bibliotheque.SupprimerLivre("The Dark Forest ");
        // Affichage des livres après suppression
        Console.WriteLine("\nLivres après suppression :");
        bibliotheque.AfficherLivres();
    }}
```

Étape 4 : Création du point d'entrée `Program.cs` (40 minutes)

Étape 4 : Création du point d'entrée `Program.cs` (40 minutes)

1.**Dans le fichier`Program.cs`,configurez la méthode`Main`pour gérer la boucle principale de l'application.**

2.**Créez un menu interactif qui permet de choisir parmi les options suivantes :**

1.
   - Ajouter un livre
   - Supprimer un livre
   - Rechercher un livre
   - Emprunter un livre
   - Retourner un livre
   - Afficher tous les livres
   - Quitter l'application

3.**Pour chaque option du menu, appelez la méthode appropriée de la classe`Bibliotheque`.** Assurez-vous que les saisies utilisateur (comme le titre du livre ou le nom de l'emprunteur) sont correctement traitées.

Questions :

- Comment la boucle `while` permet-elle de maintenir l'application en cours d'exécution ?
- Comment pourriez-vous gérer les entrées invalides de l'utilisateur (par exemple, un choix de menu incorrect) ?

Test :

- Exécutez l'application et testez chaque fonctionnalité du menu. Assurez-vous que toutes les opérations sur les livres fonctionnent correctement, de l'ajout à l'emprunt en passant par l'affichage.

Étape 5 : Compilation et exécution (20 minutes)

Étape 5 : Compilation et exécution (20 minutes)

1. **Compilez le projet en utilisant la commande suivante :**
   - **Commande à exécuter :**

```
dotnet build
```

Questions :

- Quelles erreurs pourriez-vous rencontrer lors de la compilation, et comment les corriger ?

1. **Exécutez l'application avec la commande suivante :**
2.
   - **Commande à exécuter :**

```
dotnet run
```

Tests à effectuer :

- Ajoutez plusieurs livres à la bibliothèque.

- Testez la recherche d'un livre.
- Empruntez et retournez un livre.
- Supprimez un livre et vérifiez qu'il n'apparaît plus dans la liste.

Étape 6 : Discussion finale et révision (30 minutes)

Étape 6 : Discussion finale et révision (30 minutes)

1.**Récapitulatif des concepts POO utilisés :**

1. - **Classes et Objets :** Représentation des livres et de la bibliothèque.
   - **Encapsulation :** Protection des données des objets via des propriétés.
   - **Constructeurs :** Initialisation des objets avec des valeurs par défaut.

2.**Revoyez le code final avec les participants et comparez-le à leurs propres implémentations.** Vérifiez s'ils ont bien compris les concepts et s'ils les ont appliqués correctement.

Livrables attendus

Code final attendu :

`Livre.cs` :

```
using System;

 public class Livre{

    public string Titre { get; set; }

    public string Auteur { get; set; }

    public bool EstDisponible { get; set; } = true;

    public string NomEmprunteur { get; set; }

    public DateTime? DateEmprunt { get; set; }

    public DateTime? DateRetour { get; set; }

     public Livre(string titre, string auteur)

    {

        Titre = titre;

        Auteur = auteur;

    }

     public override string ToString()

    {

        return  $"{Titre} par {Auteur} - {(EstDisponible ? "Disponible" : $"Emprunté par {NomEmprunteur} jusqu'au {DateRetour.Value.ToShortDateString()}")}";

    }}
```

`Bibliotheque.cs`

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

 public class Bibliotheque{

    private List<Livre> livres = new List<Livre>();

     public void AjouterLivre(Livre livre)

    {

        livres.Add(livre);

        Console.WriteLine($"Le livre '{livre.Titre}' a été ajouté à la bibliothèque.");

    }

     public void SupprimerLivre(string titre)

    {

        Livre livreASupprimer = livres.FirstOrDefault(l => l.Titre.Equals(titre, StringComparison.OrdinalIgnoreCase));
```

```
        if (livreASupprimer != null)
        {
            livres.Remove(livreASupprimer);
            Console.WriteLine($"Le livre '{titre}' a été supprimé.");
        }
        else
        {
            Console.WriteLine($"Le livre '{titre}' n'a pas été trouvé.");
        }
    }
    public Livre RechercherLivre(string titre)
    {
        return livres.FirstOrDefault(l => l.Titre.Equals(titre, StringComparison.OrdinalIgnoreCase));
    }
    public void EmprunterLivre(string titre, string nomEmprunteur)
    {
        Livre livreAEmprunter = RechercherLivre(titre);
        if (livreAEmprunter != null && livreAEmprunter.EstDisponible)
        {
            livreAEmprunter.EstDisponible = false;
            livreAEmprunter.NomEmprunteur = nomEmprunteur;
            livreAEmprunter.DateEmprunt = DateTime.Now;
            livreAEmprunter.DateRetour = DateTime.Now.AddDays(14);
            Console.WriteLine($"Le livre '{titre}' a été emprunté par {nomEmprunteur}.");
        }
        else if (livreAEmprunter != null && !livreAEmprunter.EstDisponible)
        {
            Console.WriteLine($"Le livre '{titre}' est déjà emprunté.");
        }
        else
        {
            Console.WriteLine($"Le livre '{titre}' n'a pas été trouvé.");
        }
    }
    public void RetournerLivre(string titre)
    {
        Livre livreARetourner = RechercherLivre(titre);
        if (livreARetourner != null && !livreARetourner.EstDisponible)
        {
            livreARetourner.EstDisponible = true;
            livreARetourner.NomEmprunteur = null;
            livreARetourner.DateEmprunt = null;
            livreARetourner.DateRetour = null;
            Console.WriteLine($"Le livre '{titre}' a été retourné à la bibliothèque.");
        }
```

```
            else if (livreARetourner != null && livreARetourner.EstDisponible)
            {
                Console.WriteLine($"Le livre '{titre}' n'était pas emprunté.");
            }
            else
            {
                Console.WriteLine($"Le livre '{titre}' n'a pas été trouvé.");
            }
        }
         public void AfficherLivres()
        {
            if (livres.Count > 0)
            {
                Console.WriteLine("Liste des livres dans la bibliothèque :");
                foreach (var livre in livres)
                {
                    Console.WriteLine(livre.ToString());
                }
            }
            else
            {
                Console.WriteLine("La bibliothèque est vide.");
            }
        }}
```

Program.cs :

```
using System;
 class  Program{
     static void Main(string[] args)
    {
            Bibliotheque bibliotheque = new Bibliotheque();
            bool quitter = false;
             while (!quitter)
            {
                AfficherMenu();
                ConsoleKeyInfo choix = Console.ReadKey();
                Console.Clear();
                 switch (choix.KeyChar)
                {
                    case '1':
                        AjouterLivre(bibliotheque);
                        break;
                    case '2':
                        SupprimerLivre(bibliotheque);
                        break;
                    case '3':
```

```
                    RechercherLivre(bibliotheque);
                    break;
                case '4':
                    EmprunterLivre(bibliotheque);
                    break;
                case '5':
                    RetournerLivre(bibliotheque);
                    break;
                case '6':
                    bibliotheque.AfficherLivres();
                    break;
                case '7':
                    quitter = true;
                    Console.WriteLine("Merci d'avoir utilisé le gestionnaire de bibliothèque.");
                    break;
                default:
                    Console.WriteLine("Choix non valide, veuillez réessayer.");
                    break;
            }
             Console.WriteLine("\nAppuyez sur une touche pour continuer...");
            Console.ReadKey();
            Console.Clear();
        }
    }
     static void AfficherMenu()
    {
        Console.WriteLine("===== Menu Bibliothèque =====");
        Console.WriteLine("1. Ajouter un livre");
        Console.WriteLine("2. Supprimer un livre");
        Console.WriteLine("3. Rechercher un livre");
        Console.WriteLine("4. Emprunter un livre");
        Console.WriteLine("5. Retourner un livre");
        Console.WriteLine("6. Afficher tous les livres");
        Console.WriteLine("7. Quitter");
        Console.WriteLine("============================");
        Console.Write("Votre choix : ");
    }
     static void AjouterLivre(Bibliotheque bibliotheque)
    {
        Console.Write("Entrez le titre du livre : ");
        string titre = Console.ReadLine();
        Console.Write("Entrez l'auteur du livre : ");
        string auteur = Console.ReadLine();
        Livre nouveauLivre = new Livre(titre, auteur);
        bibliotheque.AjouterLivre(nouveauLivre);
```

```
}
 static void SupprimerLivre(Bibliotheque bibliotheque)
{
    Console.Write("Entrez le titre du livre à supprimer : ");
    string titre = Console.ReadLine();
    bibliotheque.SupprimerLivre(titre);
}
 static void RechercherLivre(Bibliotheque bibliotheque)
{
    Console.Write("Entrez le titre du livre à rechercher : ");
    string titre = Console.ReadLine();
    Livre livre = bibliotheque.RechercherLivre(titre);
    if (livre != null)
    {
        Console.WriteLine(livre.ToString());
    }
    else
    {
        Console.WriteLine($"Le livre '{titre}' n'a pas été trouvé.");
    }
}
 static void EmprunterLivre(Bibliotheque bibliotheque)
{
    Console.Write("Entrez le titre du livre à emprunter : ");
    string titre = Console.ReadLine();
    Console.Write("Entrez votre nom : ");
    string nomEmprunteur = Console.ReadLine();
    bibliotheque.EmprunterLivre(titre, nomEmprunteur);
}
 static void RetournerLivre(Bibliotheque bibliotheque)
{
    Console.Write("Entrez le titre du livre à retourner : ");
    string titre = Console.ReadLine();
    bibliotheque.RetournerLivre(titre);
}}
```