Titre du cours	Techniques de programmation
Code du projet	PRTF v1-0 Projet 1 Partie 1 2004-1123
Titre du projet	Système de galerie informatisée (SGI)
Pages	18 pages (incluant page couverture)
Date de publication	
Date de révision	

Date début : 2014-06-05
Date de fin : 2014-06-07

Techniques de programmation

Projet 1 Partie 1: Système de galerie informatisée

INTRODUCTION

Ce projet est le premier de deux projets de ce cours. Il s'agit d'une continuation des projets que vous avez réalisés pendant les trois cours précédents. Dans ce scénario vous êtes toujours membre d'une équipe de développement chargé de la conception et du développement d'une solution pour le Système de galerie informatisée (SGI).

Le Système de Galerie Informatisée est un système informatique proposé pour la gestion des oeuvres d'art en exposition et en vente dans une galerie d'art privée. Ce système offre la possibilité d'effectuer le suivi de chaque oeuvre d'art depuis son acquisition par la galerie. Ce système est conçu pour tourner sur un seul ordinateur.

L'application actuelle à été conçue et développée à l'aide de techniques de programmation structurée. La compagnie a décidé d'effectuer une mise à jour de l'application actuelle en incorporant des concepts de programmation orientée objet à la programmation structurée.

VOTRE RÔLE

Votre tâche consiste en la création du prototype de l'application en mode console à l'aide d'un langage orienté objet.

OBJECTIFS

Les objectifs de ce projet sont:

- Écrire du code source selon les concepts orientés objet.
- Utiliser les trois structures de contrôle.
- Créer une bibliothèque de classes réutilisables pour démontrer le concept d'encapsulation.
- Créer et utiliser une classe de base abstraite pour démontrer le principe d'héritage.
- Utiliser la technique de surcharge pour démontrer le concept de polymorphisme.
- Créer et utiliser un gestionnaire d'événements
- Déboguer et traiter les exceptions pour produire une application sans erreurs

TEMPS NÉCESSAIRE

Vous disposez de 15 heures pour compléter ce projet.

RESSOURCES

Pour compléter ce projet vous avez besoins des ressources suivantes:

Équipement matériel

- Un ordinateur avec une connexion Internet et accès à une imprimante
- Processeur Pentium II ou mieux
- 64 Mo de mémoire vive pour les postes de travail NT4; 96Mo de mémoire vive pour 2000 Pro; 160 pour le serveur NT4; 160 Mo pour XP Pro; 192 Mo de mémoire vive pour le serveur 2000
- Un disque dur local ou accès à une partition de disque dur de réseau ;
- CD-ROM
- Écran SVGA
- 1 disquette vierge

Logiciels

- Système d'exploitation Windows 2000, XP Professio nnel ou NT4
- Microsoft Office 2000 Professionnel
- Microsoft Visual Studio .NET
- AVG Antivirus (ou autre anti-virus)
- Microsoft Internet Explorer version 5 ou ultérieur

SPÉCIFICATIONS

Description générale

Le Système de Galerie Informatisée est un système informatique proposé pour la gestion des oeuvres d'art en exposition et en vente dans une galerie d'art privée. Ce système offre la possibilité d'effectuer le suivi de chaque oeuvre d'art depuis son acquisition par la galerie. Ce système est conçu pour tourner sur un seul ordinateur.

Portée

Le système traite strictement des oeuvres d'art qui ont été soumises par des artistes et non celles qui ont été empruntées d'autres institutions. L'information à propos l'œuvre d'art est limitée à l'artiste qui a créé l'œuvre ainsi que le conservateur responsable pour chaque oeuvre d'art.

Exigences de traitement

Le SGI suivra chaque **oeuvre** dès son entrée dans la galerie. L'information stockée comprend les éléments suivants :

- Un code d'œuvre – un code unique de cinq caractères sera attribué à chaque oeuvre.
- le titre de l'œuvre en 40 caractères.
- L'identificateur de l'artiste qui a réalisé l'œuvre
- l'année (quatre chiffres) où la galerie a acquis l'œuvre.
- La valeur estimée de l'œuvre
- Le prix de vente de l'œuvre

Le système assure également le suivi de l'état de chaque oeuvre en tout temps en réglant un drapeau d'**état** aux valeurs suivantes, selon le cas :

E : Exposée

V: Vendu

N : Entreposée

L'information suivante est enregistrée sur chaque artiste :

- un identificateur – code unique de 5 caractères attribué à chaque artiste.
- le nom de l'artiste (maximum de 40 caractères).
- Un identificateur du conservateur assigné à chaque artiste

Le SGI doit stocker l'information suivante sur chaque **conservateur** travaillant à la galerie :

- Un identificateur de cinq caractères identifiant chaque conservateur de manière unique;
- le nom du conservateur (30 caractères);
- total de commission

Chaque conservateur est responsable pour un artiste. Un conservateur peut être assigné à un ou plusieurs artistes. Il est possible qu'un conservateur n'ait pas d'assignation à un moment donné.

Fonctions

Les fonctions suivantes sont les principales fonctions retenues pour le système SGI.

Ajouter oeuvre d'art

Le système doit permettre l'entrée de toute l'information à propos d'un oeuvre d'art. L'état de l'œuvre est assigné à ce moment pour indiquer si l'œuvre est en exposition ou entreposée.

Vendre oeuvre d'art

Le système doit permettre la modification de l'état de l'œuvre à "Vendue". Suite à cette modification de l'état, le système doit identifier le conservateur associé avec l'artiste et lui attribuer la commission appropriée.

Ajouter artiste

Le système doit permettre l'ajout d'un nouvel artiste et lui assigner un conservateur

Ajouter conservateur

Le système doit permettre d'assigner un conservateur à un artiste (un conservateur peut être assigné à plusieurs artistes).

ANALYSE INITIALE DU PROBLÈME

Processus

Le système doit permettre à l'utilisateur d'effectuer les tâches suivantes :

1. Ajouter les conservateurs
2. Tenir à jour l'information conservée sur chaque conservateur
3. Ajouter artistes
4. Recevoir une oeuvre d'art dans la collection
5. Vendre une oeuvre d'art, modifier l'état de l'œuvre d'art (Exposé – vendu) mettre à jour les commissions du conservateur
6. Afficher des listes d'artistes, conservateurs et oeuvres d'art

INSTRUCTIONS

La phase d'analyse et conception est terminée. En tant que membre de l'équipe assigné au projet SGI votre tâche consiste en la création du prototype fonctionnel de l'application.

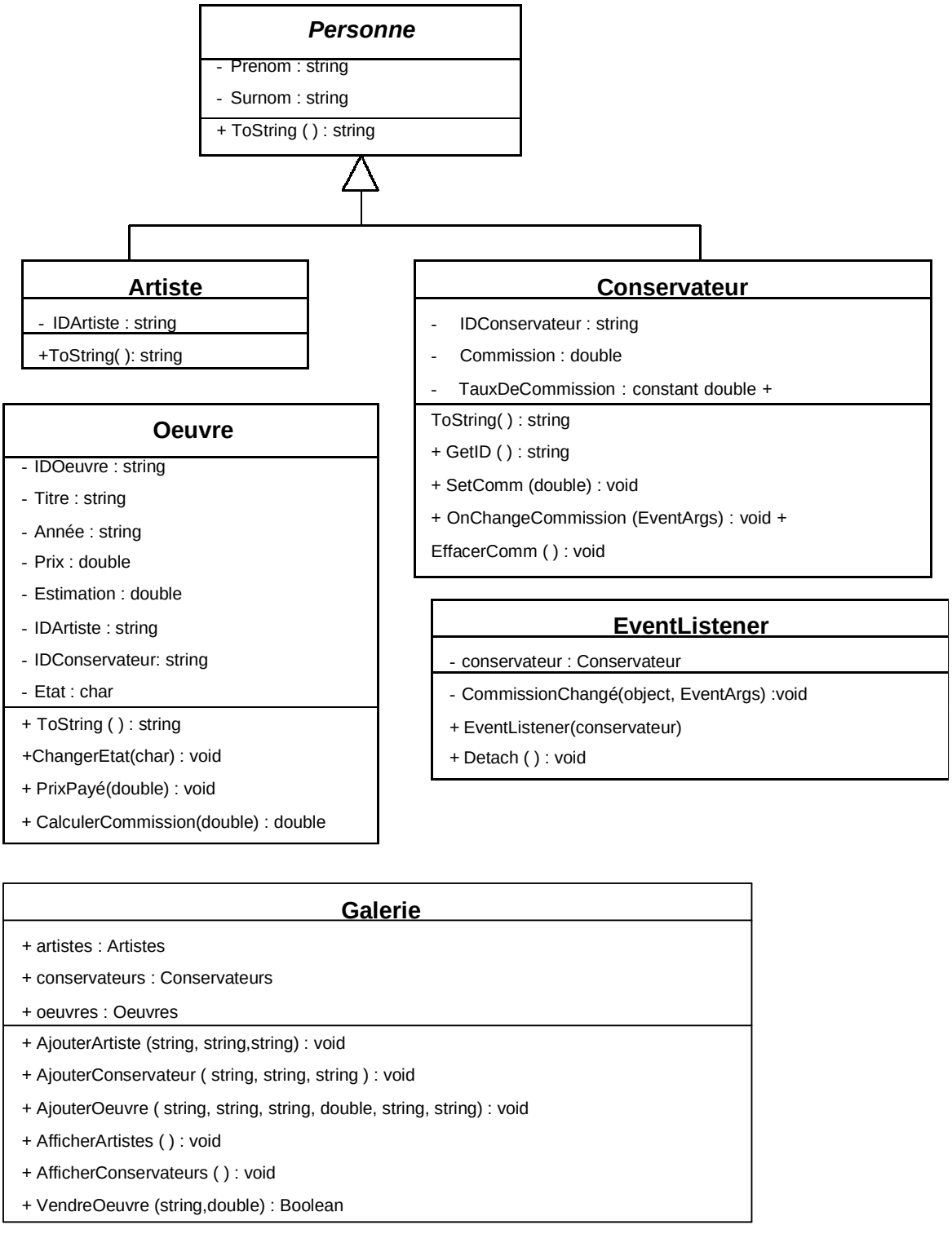
Un autre membre de l'équipe a complété les différents diagrammes illustrant les classes et les relations nécessaires à la création de la bibliothèque de classes.

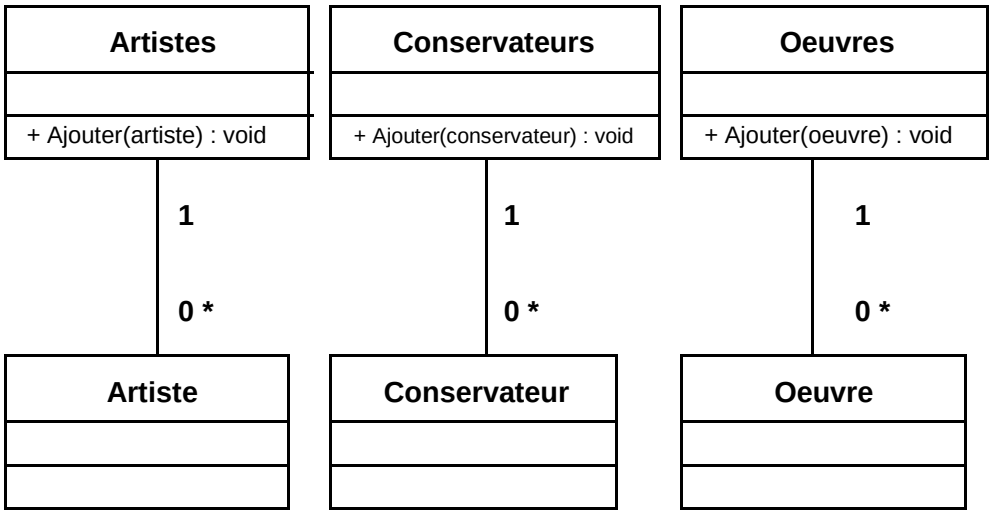
La bibliothèque de classes SGI contiendra toutes les classes et méthodes nécessaires à la création de l'application. La bibliothèque encapsulera toutes les classes et méthodes pour permettre aux autres programmeurs de les utiliser sans accéder au code source. La bibliothèque devra contenir les classes suivantes.

- Une classe de base abstraite *Personne*
- Deux classes dérivées: *Artiste, Conservateur*
- Une classe *Oeuvre*
- Trois classes collections: *Artistes*, *Conservateurs*, et *Oeuvre*s
- Une classe *Galerie* responsable pour l'ensemble des traitements
- Une classe *EventListener* pour gérer les événements

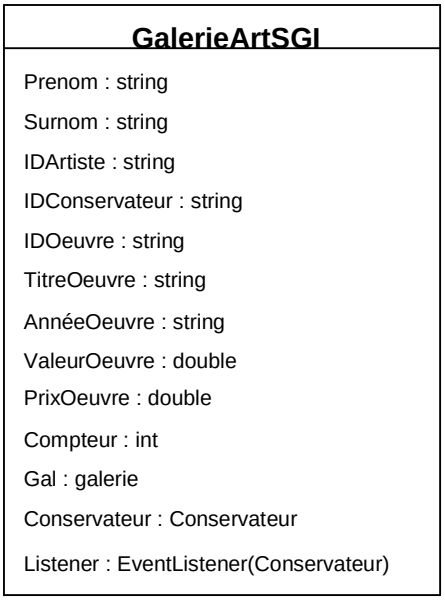
Le client utilisera une seule classe *GalerieArtSGI* qui est l'application console visible à l'utilisateur. Le client est utilisé à des fins d'essais et de prototypage. Votre responsabilité principale est de produire la bibliothèque de classe qui pourra être utilisée par les autres programmeurs dans le cadre de leurs programmes du SGI.

Étudiez les diagrammes de classes suivants. Remarquez les changements qui ont été apportés aux spécifications originales: L'identificateur du conservateur est maintenant associé à l'œuvre d'art et non à l'artiste. De plus l'état "N" (Entreposage n'est plus employé) pour les oeuvres d'art.





Le diagramme suivant illustre la classe client:



Les membres de la classe sont le minimum requis pour compléter ce projet. Vous pouvez ajouter d'autres membres au besoin.

Vous allez procéder par étapes. Dans un premier temps, vous allez créer la bibliothèque de classes avec les classes associées à l'artiste. Ensuite vous allez créer l'application client pour vérifier le fonctionnement des classes artiste. Une fois satisfait de votre travail vous allez créer les classes associées aux conservateurs dans la bibliothèque et tester ces

dernières dans l'application client. Vous allez continuer de cette façon jusqu'à l'obtention d'une application complète et fonctionnelle.

PROCÉDURE

Vous devez créer des organigrammes et /ou pseudocode pour tous les éléments de logique structurée. Vous devez remettre ces organigrammes ou pseudocodes à la fin de votre projet.

Étape 1:

1. Créez une bibliothèque de classes nommée SGI.
2. Ajoutez une classe nommée Personne avec les propriétés et méthodes indiquées dans le diagramme de classe à cette bibliothèque de classes.
    a. Ajoutez des méthodes d'accès pour les membres privés.
    b. Vous pouvez garder le constructeur par défaut.
    c. La méthode ToString doit retourner une seule chaîne de caractères contenant le prénom et nom.
    d. Compiler et monter la bibliothèque SGI. Corrigez les erreurs s'il y a lieu.

       La durée de compilation et génération peut varier selon l'équipement utilisé. Effectuez une révision de la syntaxe avant de générer la bibliothèque pour éviter des pertes de temps.
3. Ajoutez la classe Artiste à la bibliothèque SGI.
    a. Ajoutez la classe en tant que fichier séparé. Assurez-vous que l'espace de noms est SGI. Tous les fichiers de bibliothèque de classe doivent être dans l'espace de noms SGI.
    b. Assurez-vous que la classe Artiste hérite de la classe Personne.
    c. Vous pouvez garder le constructeur par défaut si vous voulez.
    d. Ajoutez des méthodes d'accès pour les membres privés de la classe.
    e. La méthode ToString redéfinit la méthode de la classe de base et doit retourner une seule chaîne de caractères contenant le prénom, le nom et l'identificateur de l'artiste.
    f. Compilez et générez la bibliothèque de classes. Corriger les erreurs.
4. Ajoutez la classe Artistes à la bibliothèque SGI.
    a. La méthode Ajouter doit appeler la méthode List.Add ( ) pour ajouter un artiste à la collection Artistes.
    b. Inclure une implémentation d'indexeur à accès par clé.
    c. Générez la bibliothèque SGI. Corrigez les erreurs s'il y a lieu.
5. Avant de continuer à ajouter des classes à la bibliothèque il est recommandé de vérifier le fonctionnement des classes que vous avez ajouté jusqu'à

maintenant. Il faut commencer la création de la classe Galerie. Vous allez ajouter plus d'éléments à cette classe à mesure que le projet avance.

a. Ajoutez la classe Galerie à la bibliothèque SGI.
b. Ajoutez une nouvelle instance de Artistes à la classe Galerie.
c. Ajoutez la méthode AjouterArtiste. Cette méthode recevra le prénom, le surnom et l'identificateur de l'artiste de l'application client et envoi ces informations au constructeur de la classe Artiste. AjouterArtiste utilise la méthode Add de la classe Artistes pour créer une nouvelle instance de Artiste et l'ajouter à la collection Artistes.
d. Ajoutez la méthode AfficherArtistes. Cette méthode reçoit les chaînes de caractères qui contiennent le nom de l'artiste et son identificateur de la collection Artistes. Cette méthode doit examiner la liste des artistes et (si la liste contient des artistes) ajoute l'information à une chaîne de caractères qui est retournée à l'application client.
e. Générez la bibliothèque SGI. Corrigez les erreurs s'il y a lieu.

6. La prochaine étape consiste en la création de l'application client console pour vérifier le travail que vous avez réalisé jusqu'à maintenant.
   a. Sauvegardez et fermez la bibliothèque SGI.

Étape 2:

1. Créez une nouvelle application console nommée GalerieArtSGI. Cette application est l'interface que l'utilisateur du système verra.
   a. Ajoutez la référence au projet SGI.dll à la solution. Le fichier .dll doit se trouver dans le répertoire SGI\bin\debug.
   b. Ajoutez les trois premières variables indiquées dans le diagramme de classe ci-dessus à la fonction Main.
   c. Créez la nouvelle instance de Galerie nommée gal.
   d. Invitez l'utilisateur à entrer le prénom de l'artiste et stocké le dans la variable prévue à cet effet. Faites la même chose pour le surnom et l'identificateur de l'artiste.
   e. Appelez la méthode AjouterArtiste de la classe Galerie et envoyer lui les trois valeurs saisies par l'utilisateur.
   f. Appelez la méthode AfficherArtiste pour afficher les artistes à l'écran.
   g. Compilez et générez l'application console GalerieArtSGI. Corrigez les erreurs.
   h. Sauvegardez votre travail.
2. Exécutez GalerieArtSGI sans débogage pour vérifier le fonctionnement de la bibliothèque SGI.

a. Si une erreur est détectée dans la bibliothèque de classes SGI, vous ne pouvez pas la corriger pendant que vous êtes dans l'application console GalerieArtSGI.
    i. Vous devez fermer GalerieArtSGI et ensuite ouvrir la bibliothèque SGI pour corriger les erreurs.
    ii. Après avoir corrigé les erreurs dans la bibliothèque, vous devez régénérer la bibliothèque, sauvegarder les changements et fermer la bibliothèque.
    iii. Ouvrez la console client GalerieArtSGI et régénérez l'application. Exécutez l'application à nouveau pour voir si les erreurs de la bibliothèque ont été corrigées. Si les erreurs persistent vous devez fermer à nouveau l'application client et ouvrir la bibliothèque. (Il est parfois difficile de savoir quelle application est ouverte. Référez-vous au titre dans le haut de la fenêtre active et le titre dans l'explorateur de solutions pour confirmer)

3. Lorsque vous êtes satisfaits que le code que vous avez produit jusqu'à maintenant, fermez GalerieArtSGI et ouvrez la bibliothèque SGI pour ajouter d'autres classes.

Étape 3:

1. Ajoutez une nouvelle classe à la bibliothèque SGI. Nommez la classe Conservateur. Assurez-vous que l'espace de noms est SGI.
    a. Après l'ouverture du fichier de classe Conservateur, vérifiez si l'espace de noms est SGI. Le cas échéant, modifiez l'espace de noms.
    b. Assurez-vous que la classe Conservateur hérite de la classe Personne.
    c. Ajoutez les membres privés de la classe ainsi que les méthodes d'accès. Le taux de commission est de 0.10.
    d. Dans le constructeur, la valeur par défaut de la commission est 0. Seulement le nom et l'identificateur sont passés au constructeur.
    e. Ajoutez la méthode ToString. Cette méthode redéfinit la méthode de la classe de base et retourne le prénom et nom du conservateur, son identificateur, et sa commission totale. (ASTUCE: Assurez-vous que cette méthode retourne seulement des valeurs de type string.)
    f. Ajoutez la méthode GetID. L'identificateur est retourné à la méthode VendreOeuvre dans Galerie pour identifier le conservateur pour lequel la commission doit être calculée.
    g. Ajoutez la méthode SetComm. Cette méthode reçoit le montant éligible pour la commission par rapport à une oeuvre d'art (ce montant est déterminé par la méthode CalculerCommission dans Oeuvre, que

vous allez traiter lors de la prochaine étape). SetComm utilise le TauxDeCommission pour calculer la commission et assigne ce montant au conservateur approprié selon l'œuvre d'art.

h. Pour l'instant, la classe Conservateur possède tous les éléments nécessaires pour vous permettre de continuer. Vous allez revenir à cette classe pour y ajouter un événement.

i. Sauvegarder votre travail et régénérez la bibliothèque. Corrigez les erreurs au besoin.

2. Créez la classe Conservateurs. Ajoutez les méthodes Ajouter et l'indexeur. Compilez, générez et corrigez les erreurs au besoin.
3. Édition de la classe Galerie:
    a. Inclure l'instance Conservateurs.
    b. Créez la méthode AjouterConservateur pour envoyer le prénom, le nom et l'identificateur du conservateur au constructeur Conservateur.
    c. Créez la méthode AfficherConservateurs.
    d. Compilez et régénérez la bibliothèque. Corrigez les erreurs.
    e. Sauvegardez la bibliothèque SGI.
4. Fermez la bibliothèque SGI et ouvrez GalerieArtSGI.
    a. Ajoutez la variable IDConservateur. Il n'est pas nécessaire d'ajouter des variables pour le prénom et nom du conservateur. Vous pouvez utiliser les variables de noms qui existent déjà.
    b. Ajouter le code nécessaire pour inviter l'utilisateur à saisir l'information du conservateur. Envoyez cette information dans la méthode AjouterConservateur de la classe Galerie.
    c. Ajoutez le code pour appeler la méthode AfficherConservateurs de Galerie pour afficher les conservateurs à l'écran.
    d. Compilez générez et exécutez l'application. Dans le cas où il y aurait des erreurs, suivez les instructions indiquées dans l'étape 2 pour corriger les erreurs.

Étape 4:

1. Fermez GalerieArtSGI et ouvrez la bibliothèque SGI. Ajoutez la classe Oeuvre à la bibliothèque.
    a. Le constructeur ne reçoit pas le prix ou l'état de l'œuvre. Assignez 0 comme prix par défaut et "E" à l'état.
    b. La méthode ToString doit retourner toute l'information à propos l'œuvre dans une seule chaîne de caractères. (ASTUCE: Assurez-vous que toutes les valeurs sont de types string.)

c. La méthode ChangerEtat change l'état de l'œuvre à "V" lorsqu'une oeuvre est vendue.

d. PrixPayé reçoit le prix payé pour l'œuvre et assigne cette valeur à la variable prévue à cet effet.

e. La méthode CalulerCommission reçoit le prix payé pour l'œuvre. La méthode retourne 25% de la différence entre la valeur original et le prix payé. Ce montant est ensuite envoyé à SetComm dans l'instance Conservateur.

f. Compilez et générez le code. Corrigez les erreurs au besoin.

2. Ajoutez la collection Oeuvres. Ajoutez les méthodes Ajouter et l'indexeur. Compilez et générez le code. Corrigez les erreurs.
3. Édition de la classe Galerie.

    a. Ajoutez une nouvelle instance de la collection Oeuvres.

    b. Ajoutez les méthodes AjouterOeuvres et AfficherOeuvres.

    c. Ajoutez la méthode VendreOeuvre, qui reçoit l'identificateur de l'œuvre vendue et le prix payé. Cette méthode emploi une boucle pour traverser la collection des oeuvres et identifier l'œuvre qui correspond à l'identificateur reçu. Lorsque l'œuvre correspondante est identifiée, la méthode vérifie son état. Si l'œuvre est vendue, la méthode retourne la valeur booléenne *false*. Si l'état indique "E" (en exposition) l'œuvre peut être vendue et la méthode extrait l'identificateur du conservateur assigné à l'œuvre à l'aide de la méthode GetID, et appel les méthodes ChangerEtat, PrixPayé et CalculerCommission de Oeuvre.

        Cette méthode utilise IDConservateur retourné par GetID pour identifier le conservateur dans la collection Conservateurs. Si le conservateur est trouvé, le prix est envoyé à la méthode SetComm de Conservateur.

    d. Compilez et générez le code. Corrigez les erreurs et sauvegardez la bibliothèque SGI.

4. Fermez la bibliothèque SGI et ouvrez GalerieArtSGI.

    a. Ajoutez le code invitant l'utilisateur à entrer l'information à propos l'œuvre d'art. Le constructeur pour Oeuvre assigne des valeurs par défaut à l'état et prix donc vous n'avez pas besoin de demander la saisie de ces valeurs.

    b. Créez une oeuvre d'art en envoyant ces valeurs à la méthode AjouterOeuvres de Galerie.

    c. Appelez AfficherOeuvres pour afficher les oeuvres d'art à l'écran.

    d. Invitez l'utilisateur à entrer l'identificateur de l'œuvre et le prix payé pour l'œuvre.

e. Envoyez ces valeurs via VendreOeuvre (Le prix doit être de type double). VendreOeuvre retourne une valeur booléenne qui peut être utilisée pour vérifier si la vente à été complétée. Si la valeur retournée est false, la vente n'a pas été complétée.

f. Appelez la méthode AfficherOeuvres et AfficherConservateurs pour vérifier que les modifications ont été effectuées correctement.

5. Compilez, générez et exécutez l'application. Corrigez les erreurs au besoin.
6. Fermez GalerieArtSGI, et ouvrez la bibliothèque SGI.

Étape 5:

1. La bibliothèque de classes est correcte dans sa forme actuelle. Mais il pourrait être intéressant d'ajouter une fonction qui vérifie si la commission a été payée au conservateur. Vous allez ajouter un événement qui déclenche un message qui sera affiché dans l'application client.
    a. Dans Conservateur dans l'espace de noms SGI, déclarez un gestionnaire d'événement délégué CommissionPayeHandler avec des paramètres de type object et EventArgs.
    b. Dans la classe Conservateur, déclarez l'événement Changed de type delegate.
    c. EventListener crée une instance de conservateur. Ajoutez un constructeur surchargé pour recevoir le prénom, le nom, l'identificateur et la commission du conservateur.
    d. Ajoutez une méthode virtuelle OnChangeCommission avec un paramètre de type EventArgs. Cette méthode doit vérifier si Changed est nul. Si Changed n'est pas nul, la méthode envoie l'instance du conservateur et un événement comme arguments.
    e. Ajoutez une méthode EffacerCommission pour remettre la commission à 0. Cette méthode doit aussi appeler OnChangeCommission et l'envoyer un paramètre EventArgs vide.
    f. Vous devez modifier la méthode SetComm. La méthode SetComm doit appeler OnChangeCommission et l'envoyer un paramètre EventArgs vide immédiatement après la dernière ligne de code qui met à jour la commission du conservateur.
2. Dans la bibliothèque SGI, ajoutez la classe EventListener, et ajoutez à celle-ci une nouvelle instance de Conservateur.
    a. Le constructeur EventListener doit recevoir une instance Conservateur. L'instance reçue sera assignée à l'instance Conservateur locale. Attachez l'événement Changed de Conservateur dans le constructeur.
    b. Ajoutez la méthode CommissionChangé avec des paramètres de type object et EventArgs. Cette méthode doit afficher un message à l'écran

pour informer l'utilisateur que le conservateur à été payé sa commission

c. Ajoutez une méthode pour détacher l'événement et fixer le conservateur à nul.

d. Générez la bibliothèque et corrigez les erreurs au besoin.

3. Fermez la bibliothèque SGI et ouvrez le client GalerieArtSGI.

   a. Ajoutez une nouvelle instance de Conservateur et EventListener.

   b. Après l'appel à VendreOeuvre, ajoutez un appel à la méthode EffacerCommission. À la fin du code ajoutez un appel à la méthode Detach de l'écouteur d'événements.

4. Compilez, générer et exécutez l'application. Déboguer et corriger l'application au besoin. Sauvegarder votre travail.

Étape 6:

1. L'interface GalerieArtSGI n'est pas très conviviale. Vous allez ajouter un menu pour faciliter son utilisation. Créez un algorithme pour vous aider à développer cette fonctionnalité.
2. Ajoutez un menu, invitez l'utilisateur à indiquer son choix et à l'aide d'une structure switch case dans une boucle, exécutez le choix de l'utilisateur. Assurez-vous d'inclure un choix pour quitter l'application. Modifiez Main pour retourner une valeur de type integer.
3. Créez des méthodes pour exécuter les tâches associées au menu:

   a. Créez une méthode NouvelArtiste contenant le code invitant l'utilisateur à entrer les informations de l'artiste et l'appel à AjouterArtiste.

   b. Créez une méthode NouveauConservateur contenant le code invitant l'utilisateur à entrer les informations du conservateur et l'appel à AjouterConservateur.

   c. Créez une méthode NouvelleOeuvre contenant le code invitant l'utilisateur à entrer les informations sur l'œuvre et l'appel à AjouterOeuvre.

   d. Créez une méthode VendreUneOeuvre contenant le code invitant l'utilisateur à entrer l'information pour la vente d'une oeuvre d'art et l'appel à VendreOeuvre.

   e. Vous pouvez créer une méthode pour chaque appel à une méthode Afficher…. Ou vous pouvez les appeler directement depuis le bloc de code *switch… case* approprié..

4. Régénérez corrigez et exécutez l'application. Déboguer l'application au besoin Lorsque l'application produit le résultat voulu, sauvegardé votre travail.

Étape 7:

1. Ajoutez les éléments de gestion d'erreurs et de traitement d'exception dans la bibliothèque SGI et dans GalerieArtSGI.
2. Sauvegardez votre travail pour le soumettre.
3. Gardez une copie de la bibliothèque SGI pour utilisation dans le deuxième projet de ce cours.

Étape 8:

1. Dans votre conclusion écrivez un paragraphe qui explique comment les concepts OO de l'encapsulation, héritage et polymorphisme sont incorporé dans ce projet. Décrivez la différence entre un programme structuré et un programme orienté objet.

Félicitations!

Vous avez réalisé votre première application orientée objet avec une bibliothèque de classes réutilisables.

SI VOUS AVEZ DU TEMPS

Vous pouvez ajouter d'autres fonctions à votre application. Assurez-vous de garder une copie de votre application actuelle avant de procéder avec des ajouts facultatifs.

1. Dans GalerieArtSGI, ajoutez la logique nécessaire à la validation des entrées. (Assurer que des valeurs des types appropriés sont saisies)
2. Ajoutez une ou plusieurs des méthodes suivantes à la bibliothèque.

TrouverConservateur(string) : string

Cette méthode appartient à la classe Galerie. Elle reçoit l'identificateur du conservateur et ensuite compare cette valeur aux identificateurs dans la collection Conservateurs. Si un identificateur correspondant se trouve dans la collection, la méthode appel ToString dans Conservateur. Le cas échéant, la méthode affiche simplement un message informant l'utilisateur que le conservateur recherché n'existe pas.

Ajoutez le code nécessaire à la saisie de l'identificateur dans GalerieArtSGI. Envoyez cette valeur à TrouverConservateur.

TrouverArtiste(string) : string

Cette méthode appartient à la classe Galerie. Elle reçoit l'identificateur de l'artiste et ensuite compare cette valeur aux identificateurs dans la collection Artistes. Si un identificateur correspondant se trouve dans la collection, la méthode appel ToString dans Artiste. Le cas échéant, la méthode affiche simplement un message informant l'utilisateur que l'artiste recherché n'existe pas.

Ajoutez le code nécessaire à la saisie de l'identificateur dans GalerieArtSGI. Envoyez cette valeur à TrouverArtiste.

SupprimerConservateur(string) : string

Cette méthode appartient à la classe Galerie. Elle reçoit l'identificateur du conservateur et ensuite compare cette valeur aux identificateurs dans la collection Conservateurs. Si un identificateur correspondant se trouve dans la collection, la méthode appel Supprimer dans Conservateurs. Le cas échéant, la méthode affiche simplement un message informant l'utilisateur que le conservateur recherché n'existe pas.

Ajoutez le code nécessaire à la saisie de l'identificateur dans GalerieArtSGI. Envoyez cette valeur à SupprimerConservateur.

Assurez-vous d'ajouter la méthode Supprimer à la collection Conservateurs.

SupprimerArtiste(string) : string

Cette méthode appartient à la classe Galerie. Elle reçoit l'identificateur de l'artiste et ensuite compare cette valeur aux identificateurs dans la collection Artistes. Si un identificateur correspondant se trouve dans la collection, la méthode appel

Supprimer dans Artistes. Le cas échéant, la méthode affiche simplement un message informant l'utilisateur que l'artiste recherché n'existe pas.

Ajoutez le code nécessaire à la saisie de l'identificateur dans GalerieArtSGI. Envoyez cette valeur à SupprimerArtiste.

Assurez-vous d'ajouter la méthode Supprimer à la collection Artistes.

GRILLE DE CORRECTION

Les éléments suivants seront évalués:

Élément du projet	Points
Production d'organigrammes et/ou pseudocode	**5**
Utilisation d'encapsulation, héritage et polymorphisme	**10**
Utilisation de classes collection	**5**
Utilisation de membres privés et publics, fonctions et fonctions d'accès.	**10**
Utilisation de constructeurs et constructeurs surchargés	**10**
Utilisation de gestionnaires d'événements et événements	**10**
Format de code approprié avec les commentaires pertinents	**5**
Le programme fonctionne correctement avec les sorties désirées	**20**
L'interface client fonctionne correctement avec la fonctionnalité requise	**8**
Débogage et traitement des erreurs complet produisant une application libre d'erreurs	**7**
Documentation et présentation incluant la conclusion	**10**
Total :	**100**

QUOI SOUMMETRE

Votre projet doit contenir:

- Page titre
- Description du projet
- Le code source C# démontrant que vous avez suivi vos algorithmes et les diagrammes de classes.
- Disquette avec la bibliothèque SGI et le client GalerieArtSGI développé en C#
- Une liste indiquant les entrés qui ont été validées. Ceci permet à votre instructeur d'évaluer votre projet correctement.
- Les algorithmes et / ou pseudocode
- Conclusion

PÉNALITÉS

- Pour chaque jour de retard, 5% de la note sera déduit.
- Les projets qui accusent un retard de plus de trois jours recevront un maximum de 60%.
- Les projets contenant un virus devront être remis à nouveau et recevront un maximum de 60%.