TP 4 : Encapsulation et polymorphisme

Objectif du TP

L'objectif de ce TP est d'apprendre à utiliser l'encapsulation et polymorphisme

Exercice 1 :

Créez une classe **Employe** pour représenter les informations d'un employé. Cette classe doit inclure :
Des attributs privés pour stocker le nom de l'employé (string), son identifiant (int) et son salaire (double).
Un constructeur par défaut initialisant ces valeurs à des valeurs par défaut (par exemple, nom vide, identifiant à 0 et salaire à 0.0).
Un autre constructeur pour initialiser les données avec des valeurs spécifiques.
Des méthodes publiques pour accéder et modifier ces attributs (utilisez des setters et des getters).
Assurez-vous que les valeurs attribuées aux attributs ne violent pas les règles métier, par exemple, le salaire ne peut pas être négatif.
Implémentez cette classe **Employe** avec les méthodes nécessaires pour gérer l'encapsulation des attributs. Testez ensuite cette classe en créant des objets **Employe**, en définissant différents employés avec des noms, identifiants et salaires variés, en modifiant ces attributs à l'aide des setters, et en récupérant ces informations à l'aide des getters

Exercice 2

Créez une classe de base FormeGeometrique avec une méthode aire() qui calcule et renvoie l'aire de la forme géométrique (initialisez cette méthode avec 0.0).
Ensuite, créez trois classes dérivées : Rectangle, Cercle et Triangle, chacune héritant de FormeGeometrique.
La classe Rectangle devrait avoir des attributs pour la longueur et la largeur du rectangle, ainsi qu'une méthode aire() qui calcule et renvoie l'aire du rectangle.
La classe Cercle devrait avoir un attribut pour le rayon du cercle, ainsi qu'une méthode aire() qui calcule et renvoie l'aire du cercle.
La classe Triangle devrait avoir des attributs pour la base et la hauteur du triangle, ainsi qu'une méthode aire() qui calcule et renvoie l'aire du triangle.
Créez des objets de chaque classe, calculez leur aire à l'aide de la méthode aire() et affichez le résultat.

La solution TP 6(1)

Exercice 1

```cpp
#include <iostream>
#include <string>
using namespace std;

class Employe {
private:
    string nom;
    int identifiant;
    double salaire;

public:
    // Constructeur par défaut
    Employe() {};//: nom(""), identifiant(0), salaire(0.0)

    // Constructeur avec des valeurs spécifiques
    Employe( string Nom, int ident, double sal) {
        nom = Nom;
        if (ident >= 0) {
            identifiant = ident;
        } else {
            identifiant = 0;
            cout << "Identifiant incorrect. Initialisé à 0." << endl;
        }
        if (sal >= 0) {
            salaire = sal;
        } else {
            salaire = 0.0;
            cout << "Salaire incorrect. Initialisé à 0.0." << endl;
        }
    }
    // Méthodes pour accéder et modifier les attributs
    void setNom(string Nom) {
        nom = Nom;
    }
    string getNom()  {
        return nom;
    }

    void setIdentifiant(int ident) {
        if (ident >= 0) {
            identifiant = ident;
        } else {
            cout << "Identifiant incorrect. Aucun changement effectué." << endl;
        }
    }
    int getIdentifiant()  {
        return identifiant;
    }

    void setSalaire(double sal) {
        if (sal >= 0) {
            salaire = sal;
        } else {
            cout << "Salaire incorrect. Aucun changement effectué." << endl;
        }
    }
    double getSalaire() {
        return salaire;
    }
```

```cpp
int main() {
    Employe employe1; // Utilisation du constructeur par défaut

    Employe employe2("Alice", 101, 3500.0); // Utilisation du constructeur avec des valeurs spécifiques

    // Utilisation des setters pour modifier les attributs
    employe1.setNom("Bob");
    employe1.setIdentifiant(102);
    employe1.setSalaire(4000.0);

    // Affichage des informations des employés à l'aide des getters
    cout << "Informations de l'employe1 :" << endl;
    cout << "Nom : " << employe1.getNom() << endl;
    cout << "Identifiant : " << employe1.getIdentifiant() << endl;
    cout << "Salaire : " << employe1.getSalaire() << endl;

    cout << "\nInformations de l'employe2 :" << endl;
    cout << "Nom : " << employe2.getNom() << endl;
    cout << "Identifiant : " << employe2.getIdentifiant() << endl;
    cout << "Salaire : " << employe2.getSalaire() << endl;

    return 0;
}
```

Exercice 2

```
////////EX2
#include <iostream>
#include <cmath>
using namespace std;

// Classe de base FormeGeometrique
class FormeGeometrique {
public:
    // Méthode virtuelle pour calculer l'aire
    virtual double aire() {
        return 0.0;
    }
};

// Classe Rectangle dérivée de FormeGeometrique
class Rectangle : public FormeGeometrique {
private:
    double longueur;
    double largeur;

public:
    Rectangle(double _longueur, double _largeur) : longueur(_longueur), largeur(_largeur) {}

    // Redéfinition de la méthode aire pour le rectangle
    double aire() {
        return longueur * largeur;
    }
};
class Cercle : public FormeGeometrique {
private:
    double rayon;

public:
    Cercle(double _rayon) : rayon(_rayon) {}

    // Redéfinition de la méthode aire pour le cercle
    double aire()  {
        return M_PI * rayon * rayon;
    }
};

// Classe Triangle dérivée de FormeGeometrique
class Triangle : public FormeGeometrique {
private:
    double base;
    double hauteur;

public:
    Triangle(double _base, double _hauteur) : base(_base), hauteur(_hauteur) {}

    // Redéfinition de la méthode aire pour le triangle
    double aire() {
        return 0.5 * base * hauteur;
    }
};
int main() {
    Rectangle rect(5.0, 3.0);
    Cercle cercle(4.0);
    Triangle triangle(6.0, 8.0);

    cout << "Aire du rectangle : " << rect.aire() << endl;
    cout << "Aire du cercle : " << cercle.aire() << endl;
    cout << "Aire du triangle : " << triangle.aire() << endl;

    return 0;
}
```

TP 6 : Encapsulation et polymorphisme (2)

Exercice 1

Créez une classe **Livre** pour représenter les informations d'un livre. Cette classe doit inclure :
Des attributs privés pour stocker le titre du livre (string), son auteur (string), son nombre de page (int) et son prix (double).
Un constructeur par défaut initialisant ces valeurs à des valeurs par défaut (par exemple, titre vide, auteur vide, identifiant à 0 et prix à 0.0).
Un autre constructeur pour initialiser les données avec des valeurs spécifiques.
Des méthodes publiques pour accéder et modifier ces attributs (utilisez des setters et des getters).
Assurez-vous que les valeurs attribuées aux attributs ne violent pas les règles métier, par exemple, le prix ne peut pas être négatif.
Implémentez cette classe Livre avec les méthodes nécessaires pour gérer l'encapsulation des attributs. Testez ensuite cette classe en créant des objets Livre, en définissant différents livres avec des titres, auteurs, identifiants et prix variés, en modifiant ces attributs à l'aide des setters, et en récupérant ces informations à l'aide des getters.

Exercice 2

Créez une classe de base **Media** avec une méthode afficher() pour afficher les détails du média.
Ensuite, créez trois classes dérivées : Livre, CD et Film, chacune héritant de Media.
La classe Livre devrait avoir des attributs pour le titre du livre, l'auteur et le nombre de pages, ainsi qu'une méthode afficher() pour afficher les détails du livre.
La classe Film devrait avoir des attributs pour le titre du film, le réalisateur et la durée en minutes, ainsi qu'une méthode afficher() pour afficher les détails du film.
La classe CD devrait avoir des attributs pour le titre du CD, l'auteur et la taille en octects, ainsi qu'une méthode afficher() pour afficher les détails du CD.
Créez des objets de chaque classe, initialiser chaque objets et appeler la méthode afficher() pour afficher les détails de chaque objet.

La solution TP 6(2)

Exercice 1

```
#include <iostream>
#include <string>
using namespace std;
class Livre {
private:
    string titre;
    string auteur;
    int nombrePages;
    double prix;
public:
    // Constructeur par défaut
    //Livre() : titre(""), auteur(""), nombrePages(0), prix(0.0) {}
Livre(){};
    // Constructeur avec des valeurs spécifiques
    Livre(string  _titre,  string _auteur, int _nombrePages, double _prix) {
        titre = _titre;
        auteur = _auteur;
        if (_nombrePages >= 0) {
            nombrePages = _nombrePages;
        } else {
            nombrePages = 0;
            cout << "Nombre de pages incorrect. Initialisé à 0." << endl;
        }
        if (_prix >= 0) {
            prix = _prix;
        } else {
            prix = 0.0;
            cout << "Prix incorrect. Initialisé à 0.0." << endl;
        }
    }
```

```
// Méthodes pour accéder et modifier les attributs
void setTitre( string _titre) {
    titre = _titre;
}

string getTitre() const {
    return titre;
}

void setAuteur( string _auteur) {
    auteur = _auteur;
}
string getAuteur()  {
    return auteur;
}

void setNombrePages(int _nombrePages) {
    if (_nombrePages >= 0) {
        nombrePages = _nombrePages;
    } else {
        cout << "Nombre de pages incorrect. Aucun changement effectué." << endl;
    }
}
int getNombrePages()  {
    return nombrePages;
}
```

```
  void setPrix(double _prix) {
      if (_prix >= 0) {
          prix = _prix;
      } else {
          cout << "Prix incorrect. Aucun changement effectué." << endl;
      }
  }
  double getPrix()  {
      return prix;
  }};
int main() {    Livre livre1; // Utilisation du constructeur par défaut
    Livre livre2("Harry Potter", "J.K. Rowling", 350, 25.99); // Utilisation du constructeur avec des valeurs spécifiques
    // Utilisation des setters pour modifier les attributs
    livre1.setTitre("Le Seigneur des Anneaux");
    livre1.setAuteur("J.R.R. Tolkien");
    livre1.setNombrePages(500);
    livre1.setPrix(30.5);
    // Affichage des informations des livres à l'aide des getters
    cout << "Informations du livre 1 :" << endl;
    cout << "Titre : " << livre1.getTitre() << endl;
    cout << "Auteur : " << livre1.getAuteur() << endl;
    cout << "Nombre de pages : " << livre1.getNombrePages() << endl;
    cout << "Prix : " << livre1.getPrix() << endl;
    cout << "\nInformations du livre 2 :" << endl;
    cout << "Titre : " << livre2.getTitre() << endl;
    cout << "Auteur : " << livre2.getAuteur() << endl;
    cout << "Nombre de pages : " << livre2.getNombrePages() << endl;
    cout << "Prix : " << livre2.getPrix() << endl;
    return 0;
}
```

Exercice 2

```cpp
#include <iostream>
#include <string>
using namespace std;

// Classe de base Media
class Media {
public:
    virtual void afficher() = 0; // Méthode virtuelle pure
};

// Classe Livre dérivée de Media
class Livre : public Media {
private:
    string titre;
    string auteur;
    int nombrePages;

public:
    Livre(const string& _titre, const string& _auteur, int _nombrePages)
        : titre(_titre), auteur(_auteur), nombrePages(_nombrePages) {}

    // Redéfinition de la méthode afficher pour Livre
    void afficher() {
        cout << "Livre - Titre: " << titre << ", Auteur: " << auteur << ", Pages: " << nombrePages << endl;
    }
};
// Classe Film dérivée de Media
class Film : public Media {
private:
    string titre;
    string realisateur;
    int dureeMinutes;
public:
    Film(const string& _titre, const string& _realisateur, int _dureeMinutes)
        : titre(_titre), realisateur(_realisateur), dureeMinutes(_dureeMinutes) {}

    // Redéfinition de la méthode afficher pour Film
    void afficher()  {
        cout << "Film - Titre: " << titre << ", Realisateur: " << realisateur << ", Duree (minutes): " << dureeMinutes << endl;
    }
};
// Classe CD dérivée de Media
class CD : public Media {
private:
    string titre;
    string auteur;
    int tailleOctets;
public:
    CD(const string& _titre, const string& _auteur, int _tailleOctets)
        : titre(_titre), auteur(_auteur), tailleOctets(_tailleOctets) {}

    // Redéfinition de la méthode afficher pour CD
    void afficher() {
        cout << "CD - Titre: " << titre << ", Auteur: " << auteur << ", Taille (octets): " << tailleOctets << endl;
    }
};
```

```cpp
int main() {
    Livre livre("Le Petit Prince", "Antoine de Saint-Exupéry", 96);
    Film film("Inception", "Christopher Nolan", 148);
    CD cd("Thriller", "Michael Jackson", 750);

    livre.afficher();
    film.afficher();
    cd.afficher();

    return 0;
}
```