TD 3 - Héritage

Programmation Orientée Objet

Objectif

- Comprendre le principe de l'héritage
- Comprendre comment créer un constructeur à la classe dérivée.
- Comprendre comment créer une méthode avec des arguments optionnels.
- Comprendre comment faire appel à une méthode de la super classe.

1 Héritage

On vous demande d'écrire un programme plus complexe pour la DRH d'une entreprise. Vous devrez prendre en considération plusieurs types de métier. Il existe deux catégories de personne dans l'entreprise : les employés et les actionnaires. Les employés sont les personnes qui travaillent dans l'entreprise. Les actionnaires sont les personnes extérieures à l'entreprise qui donnent des ordres.

1) créez la classe `Employe` qui hérite de la classe `Personne`. Que remarquez-vous ?

La classe personne réutilisée possède un constructeur nécessitant des paramètres. Il faut donc redéfinir le constructeur de la classe `Employé` afin d'effectuer un appel au constructeur de la classe `Personne`.

2) Ajoutez à la classe `Employe` un attribut privé contenant le métier de l'employé, ansi qu'un numéro d'identifiant. Ce numéro sera égal à -1 si aucun identifiant n'a encore été affecté à l'employé. Vérifier que ces attributs puissent être donnés à la construction de manière optionnelle. Créez les accesseurs de ces attributs.

3) Créez une méthode publique `travailler` qui fait appel à la méthode `affiche()` de la classe `Personne` puis qui affiche le message suivant : "Mon identifiant est : <sMetier>. Je suis employé et mon métier est : <sMetier>".

4) Surchargez l'opérateur d'égalité et de différence pour la classe `Employe`.

```
#ifndef PERSONNE_H
#define PERSONNE_H
#include<iostream>
#include<string>

using namespace std;
////////////////////////////////////////////////////////////////////////////////////////////////////
// Definition de la classe Personne
////////////////////////////////////////////////////////////////////////////////////////////////////

class Personne{
        string sNom;
        string sPrenom;

        public:
                // Constructeur et destructeur
                Personne(const string &, const string &);
                ~Personne();

                //Accesseurs
                string getNom() const;
                void setNom(const string &);
                string getPrenom() const;
                void setPrenom(const string &);

                // fonctions utilitaires
                void affiche() const {
                                cout << "Personne : " << sNom << " " << sPrenom << endl;
                }
                void saisir(const string &, const string &);
};
```

```
#endif
```

```
#include "Personne.h"

// ******************************************
// Constructeur
// ******************************************

Personne::Personne(const string & sBuffer1, const string & sBuffer2){
        #ifdef DEBUG
        cout << " constructeur avec parametres\n";
        #endif
        // Saisie du nom et du prénom
        setNom(sBuffer1);
        setPrenom(sBuffer2);
}

// ******************************************
// Destructeur
// ******************************************
Personne::~Personne(){
        #ifdef DEBUG
        cout << " destructeur" << endl;
        #endif
}

// ******************************************
// Accesseurs
// ******************************************
string Personne::getNom() const{
        return(sNom);
}
void Personne::setNom(const string & Nom){
        sNom = Nom;
}
string Personne::getPrenom() const{
        return(sPrenom);
}
void Personne::setPrenom(const string & Prenom){
        sPrenom = Prenom;
}

// ******************************************
// Autres fonctions membres
// ******************************************
void Personne::saisir(const string & Nom, const string & Prenom){
        setNom(Nom);
        setPrenom(Prenom);
}
```

```
#ifndef EMPLOYE_H
#define EMPLOYE_H
#include<iostream>
#include<cstring>
#include "Personne.h"

using namespace std;
////////////////////////////////////////////////////////////////////////////////////////////////////
// Definition de la classe Employe
////////////////////////////////////////////////////////////////////////////////////////////////////


class Employe : public Personne {
                string sMetier;
                int iId;
        public:
                // Constructeur et destructeur
                Employe(const string & n, const string & p, const string & m="", const int id=-1);
                ~Employe(){};

                //Accesseurs
                void setMetier(const string &);
                string getMetier() const;
                void setId(const int );
                int getId();

                // operator
                bool operator==(Employe&);
                bool operator!=(Employe&);
```

```
                // fonctions utilitaires
                void travailler ();

/*              void affiche(){
                        cout << "Personne : " << sNom << " " << sPrenom << endl;
                }
                void saisir(const char *, const char *);
                void raz();
*/
};


#endif
```

```
#include "Employe.h"
Employe::Employe(const string & n, const string & p, const string & m, const int id):Personne(n, p) {
                        #ifdef DEBUG
                                cout << "constructeur Employe" << endl
                        #endif
                                setMetier(m);
                                iId=id;
                };
void Employe::setMetier(const string & e){
        sMetier = e;
}

string  Employe::getMetier() const{
        return (sMetier);
}

void Employe::setId(const int id){
        iId=id;
}

int Employe::getId(){
        return (iId);
}

void Employe::travailler(){
        Personne::affiche(); // appel de la fct affiche de la classe Personne
        cout << "Mon identifiant est : " << iId << ". Je suis employé et mon métier est : " << sMetier << endl
}

// surcharge de l'opérateur d'égalité
// Pour que l'operation retourne vrai
// il faut que tous les champs soient vrais
// ATTENTION : la fct strcmp retourne 0 si
// 2 chaines sont egales. Il faut donc en prendre la négation
// Pour donner la veriter (L1 du strcmp) ou bien tester l egalite
// avec zero comme en ligne 2
bool Employe::operator==(Employe& autre) {
        if ((iId==autre.iId) &&
                (sMetier == autre.sMetier) && //L1
                (Personne::getNom() == autre.Personne::getNom()) && //L2
                (Personne::getPrenom() == autre.Personne::getPrenom()))
                return true;
        else
                return false;
}
// surcharge de l'opérateur de différence
// Il suffit qu'un seul des champs soit différent pour que
// l'operation retourne faux
bool Employe::operator!=(Employe& autre) {
        if ((iId!=autre.iId) ||
                (sMetier != autre.sMetier) ||
                (Personne::getNom() == autre.Personne::getNom()) ||
                (Personne::getPrenom() == autre.Personne::getPrenom()))
                return true;
        else
                return false;
}
```

```
#include "Personne.h"
#include "Employe.h"

int main()
{
        // Buffer de saisie
        string sBuffer1;
        string sBuffer2;
```

```
        // Saisie du nom et prenom
        cout << "Entrez le nom de la personne : ";
        cin >> sBuffer1;

        cout << "Entrez le prenom de la personne : ";
        cin >> sBuffer2;

        // Premiere solution
        Employe e1(sBuffer1, sBuffer2);
        e1.setMetier("Ingenieur");
        e1.setId(1234);

        // Deuxieme solution
        Employe e2(sBuffer1, sBuffer2, "Ingenieur", 1234);

        // Appel de la methode travailler
        e1.travailler();
        e2.travailler();

        if (e1==e2)
                cout << "OK : e1 == e2" << endl;
        else
                cout << "Erreur : e1 != e2" << endl;

        // 3ieme employe different
        Employe e3(sBuffer1, sBuffer2,"Technicien",1234);
        e3.travailler();

        if (e1!=e3)
                cout << "OK : e1 != e3" << endl;
        else
                cout << "Erreur : e1 == e3" << endl;

}
```

2 Héritage multiple

Seule une personne peut être à la fois employé et actionnaire. Il s'agit du PDG de l'entreprise.

1) Dessinez le graphe de dépendance des différentes classes.

2) Créez la classe `Actionnaire`, ainsi que la classe `PDG`. La classe `Actionnaire` aura une méthode `ordonner` (semblable à la méthode `travailler` de la classe). Donnez un `main` illustratif de l'utilisation de ces classes.

```
#ifndef PERSONNE_H
#define PERSONNE_H
#include<iostream>
#include<string>

using namespace std;
////////////////////////////////////////////////////////////////////////////////////////////
// Definition de la classe Personne
////////////////////////////////////////////////////////////////////////////////////////////

class Personne{
        string sNom;
        string sPrenom;

        public:
                // Constructeur et destructeur
                Personne(const string &, const string &);
                ~Personne();

                //Accesseurs
                string getNom() const;
                void setNom(const string &);
                string getPrenom() const;
                void setPrenom(const string &);

                // fonctions utilitaires
                void affiche() const {
                                cout << "Personne : " << sNom << " " << sPrenom << endl;
                }
                void saisir(const string &, const string &);
};


#endif
```

```cpp
#include "Personne.h"

// ****************************************
// Constructeur
// ****************************************

Personne::Personne(const string & sBuffer1, const string & sBuffer2){
        #ifdef DEBUG
        cout << " constructeur avec parametres\n";
        #endif
        // Saisie du nom et du prénom
        setNom(sBuffer1);
        setPrenom(sBuffer2);
}

// ****************************************
// Destructeur
// ****************************************
Personne::~Personne(){
        #ifdef DEBUG
        cout << " destructeur" << endl;
        #endif
}

// ****************************************
// Accesseurs
// ****************************************
string Personne::getNom() const{
        return(sNom);
}
void Personne::setNom(const string & Nom){
        sNom = Nom;
}
string Personne::getPrenom() const{
        return(sPrenom);
}
void Personne::setPrenom(const string & Prenom){
        sPrenom = Prenom;
}

// ****************************************
// Autres fonctions membres
// ****************************************
void Personne::saisir(const string & Nom, const string & Prenom){
        setNom(Nom);
        setPrenom(Prenom);
}
```

```cpp
#ifndef EMPLOYE_H
#define EMPLOYE_H
#include<iostream>
#include<cstring>
#include "Personne.h"

using namespace std;
////////////////////////////////////////////////////////////////////////////////////////////////////
// Definition de la classe Employe
////////////////////////////////////////////////////////////////////////////////////////////////////


class Employe : public Personne {
                string sMetier;
                int iId;
        public:
                // Constructeur et destructeur
                Employe(const string & n, const string & p, const string & m="", const int id=-1);
                ~Employe(){};

                //Accesseurs
                void setMetier(const string &);
                string getMetier() const;
                void setId(const int );
                int getId() const;

                // operator
                bool operator==(Employe&);
                bool operator!=(Employe&);

                // fonctions utilitaires
                void travailler ();
```

```
/*              void affiche(){
                                cout << "Personne : " << sNom << " " << sPrenom << endl;
                }
                void saisir(const char *, const char *);
                void raz();
*/
};


#endif
```

```
#include "Employe.h"
Employe::Employe(const string & n, const string & p, const string & m, const int id):Personne(n, p) {
                        #ifdef DEBUG
                                cout << "constructeur Employe" << endl
                        #endif
                                setMetier(m);
                                iId=id;
                };
void Employe::setMetier(const string & e){
        sMetier = e;
}

string  Employe::getMetier() const{
        return (sMetier);
}

void Employe::setId(const int id){
        iId=id;
}

int Employe::getId() const{
        return (iId);
}

void Employe::travailler(){
        Personne::affiche(); // appel de la fct affiche de la classe Personne
        cout << "Mon identifiant est : " << iId << ". Je suis employé et mon métier est : " << sMetier << endl
}

// surcharge de l'opérateur d'égalité
// Pour que l'operation retourne vrai
// il faut que tous les champs soient vrais
// ATTENTION : la fct strcmp retourne 0 si
// 2 chaines sont egales. Il faut donc en prendre la négation
// Pour donner la veriter (L1 du strcmp) ou bien tester l egalite
// avec zero comme en ligne 2
bool Employe::operator==(Employe& autre) {
        if ((iId==autre.iId) &&
                (sMetier == autre.sMetier) && //L1
                (Personne::getNom() == autre.Personne::getNom()) && //L2
                (Personne::getPrenom() == autre.Personne::getPrenom()))
                return true;
        else
                return false;
}
// surcharge de l'opérateur de différence
// Il suffit qu'un seul des champs soit différent pour que
// l'operation retourne faux
bool Employe::operator!=(Employe& autre) {
        if ((iId!=autre.iId) ||
                (sMetier != autre.sMetier) ||
                (Personne::getNom() == autre.Personne::getNom()) ||
                (Personne::getPrenom() == autre.Personne::getPrenom()))
                return true;
        else
                return false;
}
```

```
#ifndef ACTIONNAIRE_H
#define ACTIONNAIRE_H
#include<iostream>
#include<cstring>
#include "Personne.h"

using namespace std;
////////////////////////////////////////////////////////////////////////////////////////////////////
// Definition de la classe Actionnaire
////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
class Actionnaire : public Personne {
                int iId;
        protected: // pour accÃ¨s au nombre de parts des classes derivees
                int iNbPart;
        public:
                // Constructeur et destructeur
                Actionnaire(const string & n, const  string &p, const int part=0, const int id=-1):Personne(n,
                        #ifdef DEBUG
                                cout << "constructeur Actionnaire" << endl
                        #endif
                                iNbPart=part;
                                iId=id;
                };
                ~Actionnaire(){};

                //Accesseurs
                void setPart(const int);
                int getPart() const;
                void setId(const int );
                int getId() const;

                // operator
                bool operator==(Actionnaire&);
                bool operator!=(Actionnaire&);

                // fonctions utilitaires
                void commander ();

};


#endif
```

```
#include "Actionnaire.h"

void Actionnaire::setPart(const int p){
        iNbPart=p;
}

int Actionnaire::getPart() const{
        return (iNbPart);
}

void Actionnaire::setId(const int id){
        iId=id;
}

int Actionnaire::getId() const {
        return (iId);
}

void Actionnaire::commander (){
        Personne::affiche(); // appel de la fct affiche de la classe Personne
        cout << "Mon identifiant est : " << iId << ". Je suis actionnaire j'ai " << iNbPart << " parts." << en
}

// surcharge de l'opérateur d'égalité
// Pour que l'operation retourne vrai
// il faut que tous les champs soient vrais
bool Actionnaire::operator==(Actionnaire& autre) {
        if ((iId==autre.iId) &&
                (iNbPart==autre.iNbPart) && //L1
                (Personne::getNom() == autre.Personne::getNom()) &&
                (Personne::getPrenom() == autre.Personne::getPrenom()))
                return true;
        else
                return false;
}
// surcharge de l'opérateur de différence
// Il suffit qu'un seul des champs soit différent pour que
// l'operation retourne faux
bool Actionnaire::operator!=(Actionnaire& autre) {
        if ((iId!=autre.iId) ||
                (iNbPart!=autre.iNbPart) ||
                (Personne::getNom() == autre.Personne::getNom()) ||
                (Personne::getPrenom() == autre.Personne::getPrenom()))
                return true;
        else
                return false;
}
```

```
#ifndef PDG_H
#define PDG_H
#include<iostream>
#include<cstring>
#include "Employe.h"
#include "Actionnaire.h"

using namespace std;
////////////////////////////////////////////////////////////////////////////////////////////////////
// Definition de la classe PDG
////////////////////////////////////////////////////////////////////////////////////////////////////


class PDG : public Employe, public Actionnaire {
                int iSalaire;
        public:
                // Constructeur et destructeur
                PDG(const string & n, const string & p, const int salaire, const string & m="", const int part
                ~PDG(){};

                //Accesseurs
                void setSalaire(const int);
                int getSalaire() const;

                // fonctions utilitaires
                void commander () const;

};


#endif
```

```
#include "PDG.h"

PDG::PDG(const string & n, const string & p, const int salaire, const string & m, const int part, const int id
                        #ifdef DEBUG
                                cout << "constructeur PDG" << endl
                        #endif
                                iNbPart=part;
                                Employe::setId(id); // Il y a un probleme ici car les classes employees et act
                                                    // Il faut choisisr par où on va passer : Employe, Actionnaire
                                                    // Pour cela on utilise l'operateur de portee
                };

void PDG::setSalaire(const int p){
        iSalaire=p;
}

int PDG::getSalaire() const{
        return (iSalaire);
}


void PDG::commander () const {
        Employe::affiche(); // appel de la fct affiche de la classe Personne

        cout << "Mon identifiant est : " << Employe::getId() << ". Je suis PDG j'ai " << iNbPart << " parts."
}
```

```
#include "Personne.h"
#include "Employe.h"
#include "Actionnaire.h"
#include "PDG.h"

void saisirQQ1(string & nom, string & prenom){
        // Saisie du nom et prenom
        cout << "Entrez le nom de la personne : ";
        cin >> nom;

        cout << "Entrez le prenom de la personne : ";
        cin >> prenom;

}

int main()
{
        // Buffer de saisie
        string sBuffer1;
        string sBuffer2;

        // Saisie d'un employe
```

```
        saisirQQ1(sBuffer1, sBuffer2);
        Employe e(sBuffer1, sBuffer2, "Ingenieur", 1234);

        // Saisie d'un actionnaire
        saisirQQ1(sBuffer1, sBuffer2);
        Actionnaire a (sBuffer1, sBuffer2, 49, 1235);

        // Saisie d'un PDG
        saisirQQ1(sBuffer1, sBuffer2);
        PDG p (sBuffer1, sBuffer2, 100000, "PDG", 51, 1236);
//      PDG(const string & n, const string & p, const int salaire, const string & m="", const int part=0, cons

        // Appel de la methode travailler et à la méthode ordonner
        e.travailler();
        a.commander ();


}
```