**« Résumé chapitre8 »**

‘‘ LA TECHNIQUE D'HERITAGE ’’

Réalisé par :

ELYOUSFI Mohamed (BDCC1)

Supervisé par :

M. Khalifa MANSOURI

*Année universitaire 2020/2021*

1. **Définition**

* L’héritage (ou spécialisation, ou dérivation) permet d’ajouter des propriétés à une classe existante pour en obtenir une nouvelle plus précise.
* L’idée est : "un B est un A avec des choses en plus".

Exemple : Un étudiant est une personne, et a donc un nom (et un prénom, ...). De plus, il a un numéro CNE.

1. **Mise en œuvre de la technique de l'héritage en c++**

Exemple :

class point{

int x,y;

public :

void initialiser(int abs,int ord){

x=abs;y=ord;

}

void deplacer(int dx,int dy){

x+=dx;

y+=dy;

}

void affiche(){

cout <<" Le Point est en "<<x<<" "<<y <<" \n";

}

};

class pointcol :public point{

short color;

public:

void colore(short c){

color=c;

}

};

Les objets de la classe **pointcol** peut accédés maintenant au membres public de la classe point.

1. **Utilisation dans une classe dérivée, des membres de la classe de base**

**Solution 1 :**

class pointcol :public point{

short color;

public:

void colore(short c){

color=c;

}

**void afficheC(){**

**cout <<x << " "<<y <<" \n";**

**cout <<" Couleur : "<<color<<" \n";}**

};

Cette solution n'est pas valable car elle ne respecte pas l'encapsulation

**Solution 2 :**

void afficheC(){

affiche();

cout <<" Couleur : "<<color<<" \n";}

1. **Redéfinition des fonctions membre**

Il est possible en C++ de donner au méthodes le même nom des méthodes de la classe mère, mais il faut faire appel à l'opérateur de portée (::).

Exemple :

void pointcol::affiche(){

point::affiche();

cout <<" Couleur : "<<color<<" \n";

}

1. **Appel des constructeurs et des destructeurs**

Exemple :

point(int abs=0,int ord=0){

cout<<" ++ Const.point "<<abs<< " "<<ord <<endl;

x=abs;

y=ord;

}

~point(){

cout<<"-- Destr.point "<<x<< " "<<y <<endl;

}

};

class pointcol :public point{

short color;

public:

pointcol(int ,int ,short);

~pointcol(){

cout<<"-- Destr.pointcol "<<color<<endl;

}

};

pointcol::pointcol(int abs=0,int ord=0,short c=1):point(abs,ord){

cout<<"++ Cons.pointcol "<<abs<< " "<<ord <<" "<<c<<endl;

color=c;

}

1. **Contrôle des accès**

Considèrent le cas :

Class A {

XXXX :

Int x;

};

Class B : YYYY A {

….;

….;

};

|  |  |  |
| --- | --- | --- |
| **XXXX** | **YYYY** | **Résultat** |
| Public | Public | Accès sans restriction à x de A |
| Private | Public | Aucun accés à x de A |
| Protected | Public | B peut accéder à x de A, l'accés est également accordé aux classes dérivées à partir de B |
| Public | Private | B peut accéder à x de A, l'accés est également accordé aux classes dérivées à partir de B |
| Private | Private | Aucun accés à x de A |
| Protected | Private | B peut accéder à x de A, l'accés est également accordé aux classes dérivées à partir de B |