
Rapport de TPs et Contrôle

Poo

Réaliser par : EL JARRAH Khaoula

TP1 :

Classe Matchfoot :

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  class Matchfoot{
6      //Attributs
7      string eqp1;
8      string eqp2;
9      int nbrBUT1=0;
10     int nbrBUT2=0;
11
12     public:
13         //constructeurs
14         Matchfoot(){
15             eqp1="Maroc";
16             eqp2="Brazil";
17         }
18         Matchfoot(string e1,string e2,int nbr1,int nbr2):eqp1(e1),eqp2(e2),nbrBUT1(nbr1),nbrBUT2(nbr2){
19         }
20
21         Matchfoot(const Matchfoot & a){
22             eqp1=a.eqp1;
23             eqp2= a.eqp2;
24             nbrBUT1=a.nbrBUT1;
25             nbrBUT2=a.nbrBUT2;
26         }
27 }
```

Dans cette classe, on crée les attributs (eq1,eq2,nbrbut1 et nbrbut2) et les constructeurs .

28	
29	<code>//Getters and Setters</code>
30	<code>string gete1(){</code>
31	<code> return eqp1;</code>
32	<code>}</code>
33	<code>string gete2(){</code>
34	<code> return eqp2;</code>
35	<code>}</code>
36	<code>int getb1(){</code>
37	<code> return nbrBUT1;</code>
38	<code>}</code>
39	<code>int getb2(){</code>
40	<code> return nbrBUT2;</code>
41	<code>}</code>
42	<code>void sete1(string e1){</code>
43	<code> eqp1=e1;</code>
44	<code>}</code>
45	<code>void sete2(string e2){</code>
46	<code> eqp2=e2;</code>
47	<code>}</code>
48	<code>void setnb1(int nb1){</code>
49	<code> nbrBUT1=nb1;</code>
50	<code>}</code>
51	<code>void setnb2(int nb2){</code>
52	<code> nbrBUT2=nb2;</code>
53	<code>}</code>
54	

Ensuite , on crée les getters et les setters de toutes les attributs.

56	<code>void affichMatch(){</code>
57	<code> cout<<"Le premier equipe est : "<< eqp1<<endl;</code>
58	<code> cout<<"le numero de but de premier equipe est : "<< nbrBUT1<<endl;</code>
59	<code> cout<<"Le deuxieme equipe est : "<< eqp2<<endl;</code>
60	<code> cout<<"le numero de but de deuxieme equipe est : "<< nbrBUT2<<endl;</code>
61	<code>}</code>
62	
63	<code>void marquerBut(string eq){</code>
64	<code> if(eq=="A"){</code>
65	<code> nbrBUT1+=1;</code>
66	<code> }else if(eq=="B"){</code>
67	<code> nbrBUT2+=1;</code>
68	<code> }</code>
69	<code>}</code>
70	
71	<code>int comparerresult(){</code>
72	<code> if(nbrBUT1==nbrBUT2)</code>
73	<code> return 0;</code>
74	<code> else if(nbrBUT1>nbrBUT2)</code>
75	<code> return 1;</code>
76	<code> else if(nbrBUT1<nbrBUT2)</code>
77	<code> return -1;</code>
78	<code>}</code>
79	
80	<code>};</code>

Pour les autre méthodes demandées, on trouve la fonction `afficheMatch()` qui afficher le match, aussi il y a la fonction

marquebut(string) qui marque le but de l'équipe demandé dans le paramètre, enfin la fonction compare résultat qui compare les résultats des équipes.

Classe Championnat :

```
1  #include <iostream>
2  #include "match.cpp"
3
4  class championnat{
5      //Attributs
6      Matchfoot tabMatch[];
7      int nbrmatch;
8      string tabeqpts[][];
9
10     public :
11         //Constructeurs
12         championnat(){
13             nbrmatch =1;
14             tabMatch=new Matchfoot[nbrmatch];
15             tabeqpts[1][1];
16         }
17         championnat(Matchfoot m[],int n,tabeq[][]){
18             tabMatch= new Matchfoot[n];
19             nbrmatch = n;
20             for(int i=0; i<40 ; i++){
21                 tabMatch[i]=m[i];
22             }
23             for(int i =0 ;i<10;i++){
24                 for(int j =0; j<2 ;j++){
25                     tabeqpts[i][j]=tabeq[i][j];
26                 }
27             }
28         }
29     }
```

Dans cette classe, on crée les attributs (un tableau des match, nbr match et tableaux des groupes) et les constructeurs .

```

31 void ajouterMatch(Matchfool ml){
32     tabMatch[nbrmatch]=ml;
33     nbrmatch++;
34 }
35
36 void listeEquipes(){
37     for(int i=0 ;i<10 ;i+=2){
38
39         tabeqpts[i][0]=tabMatch[i].gete1()
40     }
41 }

```

Pour les autre méthodes demandées, on trouve la fonction `afficheMatch()` qui afficher le tableau de matchs, aussi il y a la fonction `listeEqipe()` qui affiche les équipes des matchs .

```

87 int pointpareq(string eq){
88     int score=0;
89     for (int i=0;i<tabmatch.size();i++ ){
90
91
92         if (eq==tabmatch.at(i).geteqA()) {
93             if (tabmatch.at(i).compareresultat()==1){
94                 score=score+3;
95             }
96             if (tabmatch.at(i).compareresultat()==-1){
97                 score=score+0;
98             }
99             if (tabmatch.at(i).compareresultat()==0){
100                 score=score+1;
101             }
102         }
103
104         if (eq==tabmatch.at(i).geteqB()) {
105             if (tabmatch.at(i).compareresultat()==1){
106                 score=score+0;
107             }
108             if (tabmatch.at(i).compareresultat()==-1){
109                 score=score+3;
110             }
111             if (tabmatch.at(i).compareresultat()==0){
112                 score=score+1;
113             }
114         }
115     }
116 }

```

```

120 void calculepts(){
121     string E;
122     for (int i =0;i<tabeq.size();i++){
123         E=tabeq.at(i).at(0);
124         tabeq.at(i).at(1)=pointpareq(E);
125     }
126 }
127
128
129 //affichage
130
131 void afficherresultat(){
132     cout << "equipe      points" <<endl;
133     for (int i=0;i<tabeq.size();i++){
134         cout << "<< " << tabeq.at(i).at(0)<< " >> : " <<tabeq.at(i).at(1)<<endl;
135     }
136 }
137
138
139 };
140

```

Il y en a aussi la fonction `calculepts()` qui calcule les points des équipes, et la fonction `afficherresultat()` qui affiche les résultats des équipes.

Main :

```

6 int main(int argc, char** argv) {
7     Matchfoot m1;
8     Matchfoot m2("france","bresil",0,0);
9     Matchfoot m3("maroc","niger",4,3);
10    Matchfoot m4(m3);
11
12    cout<<"Le match numero 4 :"<<endl;
13    m4.affichMatch();
14    cout<<endl;
15
16    m2.marquerBut("A");
17    m2.marquerBut("B");
18    cout<<endl;
19
20    m2.affichMatch();
21    if(m2.compareresult()==0)
22        cout<<"Le match M2 est null :"<<endl;
23    cout<<endl;
24    m3.affichMatch();
25    if(m3.compareresult()==1){
26        cout<<"L'equipe gagnante est: "<<m3.gete1();
27        cout<<" qui ont le resultat :"<<m3.getb1();
28    }
29 }
30 return 0;
31 }

```



Résultat :

```
F:\Tp1 POO\main.exe
Le match numero 4 :
Le premier equipe est :maroc
le numero de but de premier equipe est :4
Le deuxieme equipe est :niger
le numero de but de deuxieme equipe est :3

Le premier equipe est :france
le numero de but de premier equipe est :1
Le deuxieme equipe est :bresil
le numero de but de deuxieme equipe est :1
Le match M2 est null :

Le premier equipe est :maroc
le numero de but de premier equipe est :4
Le deuxieme equipe est :niger
le numero de but de deuxieme equipe est :3
L'equipe gagnante est: maroc qui ont le resultat :4
-----
Process exited after 0.115 seconds with return value 0
Appuyez sur une touche pour continuer...
```

TP2 :

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  static int cp=0;
5  class document{
6      float prix ;
7      string titre ;
8      int code;
9      public:
10     document (){
11         cp++;
12         cout<<cp<<endl;
13     }
14     document (float a, string b, int c):prix(a),titre(b),code(c){
15         cp++;
16         cout<<cp<<endl;
17     }
18     document(const document &c){
19         prix=c.prix;
20         titre=c.titre;
21         code=c.code;
22         cp++;
23         cout<<cp<<endl;
24     }
25     int getCode(){
26         return code;
27     }
28     float getPrix(){
29         return prix;
30     }
31 }
```

Dans cette classe, on crée les attributs (prix, titre et code), les constructeurs et les getters/setters.

```
27 int getCode(){  
28     return code;  
29 }  
30 float getPrix(){  
31     return prix;  
32 }  
33 string getTitre(){  
34     return titre;  
35 }  
36 void setPrix(float p){  
37     prix=p;  
38 }  
39 bool operator ==(document c){  
40     if(c.code==code) return 0;  
41     else return 1;  
42 }  
43 bool operator <(document d1){  
44     if(d1.prix<prix)  
45         return 0;  
46     else if(d1.prix>prix)  
47         return 1;  
48 }  
49 void Solder(float c){  
50     prix =prix-(prix*(c/100));  
51     cout<<prix;  
52 }  
53 friend ostream& operator<<(ostream &o,const document &d){  
54     o<<"le code :"<< d.code<<" et le titre est : "<<d.titre<<endl;  
55     return o;  
56 }  
57 };
```

On ajoute des opérateurs : un == qui compare l'égalité entre deux documents et l'autre < qui compare la supériorité des deux et encore autre << qui affiche un document. On a méthode compte les prix de document qui on a les soldes.

```

3  class livre : public document{
4      string auteur;
5      int nbrP;
6      public:
7          livre():document(){
8              }
9          livre(float a, string b , int c,string d ,int e):document(a,b,c),auteur(d),nbrP(e){
10             }
11          livre(const livre &l):document(l){
12              auteur=l.auteur;
13              nbrP=l.nbrP;
14          }
15          friend ostream& operator<<(ostream &o,livre l){
16              o<<l.getCode()<<" " <<l.getTitre()<<" " <<l.auteur<<" " <<l.nbrP<<endl;
17              return o;
18          }
19      };
20
21  class dictionnaire : private document{
22      string langue;
23      int nbT;
24      public:
25          dictionnaire(){
26              }
27          dictionnaire(float a, string b , int c,string d ,int e):document(a,b,c),langue(d),nbT(e){
28              }
29          dictionnaire(const dictionnaire &d):document(d){
30              langue=d.langue;
31              nbT=d.nbT;
32          }
33
34          friend ostream& operator<<(ostream &o, dictionnaire d){
35              o<<d.getCode()<<" " <<d.getTitre()<<" " <<d.langue<<" " <<d.nbT<<endl;
36              return o;
37          }
38      };

```

Cette classe livre hérite de la classe précédente, on ajoute les attributs (auteur, nbrP), les constructeurs (on appelle les constructeur de père) et operateurs.

Après la classe livre, une autre classe qui hérite aussi de la même classe mère (document) qui s'appelle dictionnaire qui contient des constructeur de la même façon que livre et operateur

```

3  using namespace std;
4  class collectionlivre{
5      livre *l;
6      int taille;
7      int nl;
8      char *res;
9      public:
10         collectionlivre(){
11             }
12         collectionlivre(livre *l,int taille , char *res, int nl=0){
13             this->taille=taille;
14             l= new livre [taille];
15             for (int i =0; i< taille ;i++)
16                 this->l[i]=l[i];
17             this->nl=nl;
18
19             res= new char [taille];
20             for (int i =0; i< taille ;i++)
21                 this->res[i]=res[i];
22         }
23         void verifiercode(int cd){
24             for(int i=0;i<taille;i++)
25                 if(l[i].getCode()==cd){
26                     cout<<"Le livre existe"<<endl;
27                 }else
28                     cout<<"Le livre n'existe pas"<<endl;
29         }
30
31         void verifierLivre(livre *liv){
32             for(int i=0;i<taille;i++)
33                 if(l[i]==liv[i]){
34                     cout<<"Le livre existe"<<endl;
35                 }else
36                     cout<<"Le livre n'existe pas"<<endl;
37         }
38     };

```

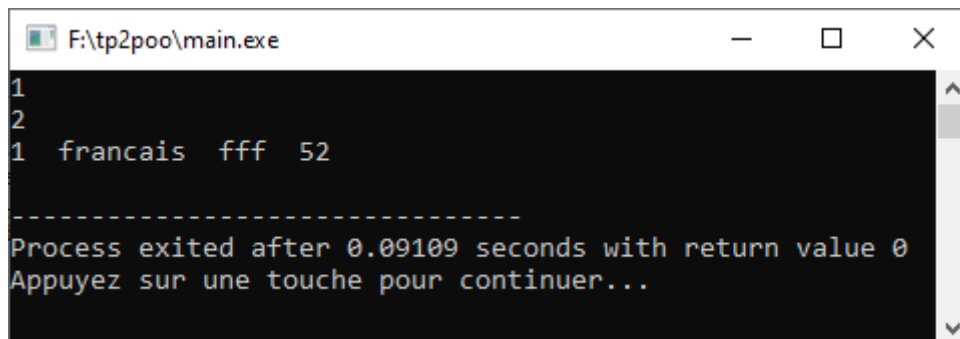

La quatrième classe collectionLivre fait appel au classe livre pour construire une tableau de livres.

```
39 void Ajouter(livre *liv, char *Res="L"){
40     l[taille]=liv;
41     taille +=1;
42     ln= ln+1;
43     res[taille]=Res;
44 }
45 livre rechercher(int cd){
46     for(int i=0;i<taille;i++)
47         if(l[i].getCode()==cd){
48             return l[i];
49         }else
50             cout<<"Le livre n'existe pas"<<endl;
51 }
52
53
54 };
```

```
1 #include <iostream>
2 #include "livre.cpp"
3
4 /* run this program using the console pau
5
6 int main(int argc, char** argv) {
7     livre l(200, "français", 1, "fff", 52);
8     cout<<l;
9 }
```



Résultat :



TP3 :

```
1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  using namespace std;
5  //-----classe point
6  class point {
7
8      private :
9          float x;
10         float y;
11
12     public:
13         point(){
14             x=0;
15             y=0;
16         }
17         void afficher(){
18             cout << "X: " << x << " Y: " << y << endl;
19         }
20         void changer_coord(float a, float b){
21             x=a;
22             y=b;
23         }
24     };
```

Dans cette classe, on crée les attributs (float x, float y) et les constructeurs et les méthodes afficher et changer_coord.

```
26 class figure{
27     private:
28         string color;
29         float e;
30
31     public:
32         figure(){
33             color ="black";
34             e=1;
35         }
36         void afficher(){
37             cout << "colour :" << color << " epaisseur : " << e << endl;
38         }
39         void changer_color(string a){
40             color = a;
41         }
42         void changer_epaisseur(float a){
43             e=a;
44         }
45     };
```

Dans cette classe figure, on crée les attributs (color, float e) et les constructeurs et les méthodes afficher et changer_paisseur et changer_color.

```

47 class cercle : public point, public figure{
48     private:
49         float rayon;
50     public:
51         cercle(){
52             rayon=0;
53             figure();
54             point();
55         }
56         virtual void afficher(){
57             figure::afficher() ;
58             point::afficher() ;
59             cout << "Rayon : "<<rayon<<endl;
60         }
61         void changer_rayon(float a){
62             rayon=a;
63         }
64     };

```

Dans cette classe qui hérite de classe point et figure, on ajoute l'attribut (rayon) et les constructeurs et les méthodes afficher (qui fait appel au afficher supérieur en utilisant la virtualisation pour éviter la confusion des méthodes de même nom) et changer_coord.

```

66 class cylindre : public cercle {
67     private:
68         float hauteur;
69     public:
70         cylindre(){
71             cercle();
72             hauteur =0;
73         }
74         virtual void afficher(){
75             cout << "cylindre : "<< endl;
76             cercle::afficher();
77             cout << "hauteur : "<<hauteur << endl;
78         }
79         void changer_hauteur(float a){
80             hauteur = a;
81         }
82     };

```

Dans cette classe qui hérite à son tour la classe cercle, on crée les attributs (hauteur) et les constructeurs et les méthodes afficher (qui fait appel au afficher supérieur en utilisant la virtualisation pour éviter la confusion des méthodes de même nom) et changer_hauteur.

```

84 int main(){
85     point p;      p.afficher();
86     figure f;     f.afficher();
87     cercle c;     c.afficher();
88     cylindre y;   y.afficher();
89 }

```



Résultat :

```

F:\P.O.O\TP-3.exe
X: 0 Y: 0
colour :black epaisseur : 1
colour :black epaisseur : 1
X: 0 Y: 0
Rayon : 0
cylindre :
colour :black epaisseur : 1
X: 0 Y: 0
Rayon : 0
hauteur : 0

-----
Process exited after 0.1413 seconds with return value 0
Appuyez sur une touche pour continuer...

```

TP4 :

```

1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  using namespace std;
5
6  class personne{
7
8      private:
9
10         int id;
11         string nom;
12         string prenom;
13         int age;
14
15     public:
16
17         personne(){id=0;    nom="cross"    ;prenom="oshino"    ;age=10;}
18         personne(int a,string b,string c,int d){id=a;        nom=b;        prenom=c;        age=d;}
19
20         void afficher(){
21             cout << "ID: "<<id<<" || FIRSTNAME: "<<prenom<<" || LASTNAME: "<<nom<<" || AGE: "<<age<<endl;
22         }
23     };

```

Dans cette classe personne, on crée les attributs (id, nom, prenom et âge) et les constructeurs et les méthodes afficher (qui affiche les attribut d'objet) .

```

27 class etudiant: virtual public personne{
28
29     private:
30
31         string cycle;
32         string classe;
33         int classement;
34
35     public :
36
37         etudiant(){
38             cycle="zero";
39             classe="math";
40             classement = 1;
41             personne();
42         }
43         etudiant(int a,string b,string c,int d,string e,string f,int g):personne(a,b,c,d){
44             cycle=e; classe=f; classement=g;
45         }
46
47         virtual void afficher(){
48             personne::afficher() ;
49             cout <<"cycle: "<<cycle<<" || classe: "<<classe<<" || classement: "<<classement<<endl;
50         }
51     };

```

Dans cette classe qui hérite de la classe étudiant, on ajoute les attributs (cycle, classe, classement) et les constructeurs et les méthodes afficher (qui fait appel au afficher supérieur en utilisant la virtualisation pour éviter la confusion des méthodes de même nom).

```

55 class enseignant : virtual public personne{
56
57     private:
58
59         float salaire;
60         int nbr_heures;
61
62     public:
63
64         enseignant(){
65             personne();
66             salaire=0;
67             nbr_heures=0;
68         }
69
70         enseignant(int a, string b,string c,int d,float e,int f):personne(a,b,c,d){
71             salaire=e; nbr_heures=f;
72         }
73
74         int geth(){return nbr_heures;}
75
76         virtual void afficher(){
77             personne::afficher() ;
78             cout << "SALARY: "<<salaire<<" || HOUR COUNT: "<<nbr_heures<<endl;
79         }
80

```

Dans cette classe enseignant qui hérite aussi de la classe personne, on ajoute les attributs (salaire, nbr_heures) et les constructeurs et les

méthodes afficher (qui fait appel au afficher supérieur en utilisant la virtualisation pour éviter la confusion des méthodes de même nom).

```
87 class doctorant :public etudiant,public enseignant,public personne{
88
89     private:
90         int prix_heure;
91
92     public:
93
94         doctorant(){
95             etudiant();enseignant();
96             prix_heure=0;
97         }
98
99         doctorant(int a,string b,string c,int d, string e,string f,int g,float h,int i,int j):etudiant(a,b,c,d,e,f,g),enseignant(a,b,c,d,h,i){
100             prix_heure=j;
101         }
102
103         void afficher(){
104             etudiant::afficher() ;
105             enseignant::afficher() ;
106             cout << "PRICE PER HOUR:"<<prix_heure<<endl;
107         }
108
109         void calculesalaire(){
110             cout << "CURRENT SALARY: " <<prix_heure * enseignant::geth() <<endl;
111         }
112     };
```

Dans cette classe qui hérite à son tour des classes enseignant et personne, on ajoute l'attribut (prix_heure) et les constructeurs et les méthodes afficher (qui fait appel au afficher(s) supérieur en utilisant la virtualisation pour éviter la confusion des méthodes de même nom) et calculesalaire().

```
116 int main(){
117
118     personne p;
119     etudiant e(10,"a","b",41,"de","m",12);
120     enseignant E(10,"a","b",41,1000,3);
121     doctorant d(10,"a","b",41,"de","m",12,1000,3,450);
122
123     p.afficher() ;
124     cout <<"-----" <<endl;
125     e.afficher() ;
126     cout <<"-----" <<endl;
127     E.afficher() ;
128     cout <<"-----" <<endl;
129     d.afficher() ;
130     cout <<"-----" <<endl;
131
132     d.calculesalaire() ;
133 }
```



Résultat :

```
F:\P.O.O\TP-4.exe
ID: 0 || FIRSTNAME: oshino || LASTNAME: cross || AGE: 10
-----
ID: 10 || FIRSTNAME: b || LASTNAME: a || AGE: 41
cycle: de || classe: m || classement: 12
-----
ID: 10 || FIRSTNAME: b || LASTNAME: a || AGE: 41
SALARY: 1000 || HOUR COUNT: 3
-----
ID: 0 || FIRSTNAME: oshino || LASTNAME: cross || AGE: 10
cycle: de || classe: m || classement: 12
ID: 0 || FIRSTNAME: oshino || LASTNAME: cross || AGE: 10
SALARY: 1000 || HOUR COUNT: 3
PRICE PER HOUR:450
-----
CURRENT SALARY: 1350
-----
Process exited after 0.1184 seconds with return value 0
Appuyez sur une touche pour continuer...
```

TP5 :

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  class batiment{
5      string adresse;
6      float superficie;
7      float prix;
8
9      public:
10         batiment(){
11         }
12         batiment(string a,float s,float p):adresse(a),superficie(s),prix(p){
13         }
14         string geta(){
15             return adresse;
16         }
17         float gets(){
18             return superficie;
19         }
20         float getp(){
21             return prix;
22         }
23         void seta(string a){
24             adresse= a;
25         }
26         void sets(float s){
27             superficie= s;
28         }
29         void setp(float p){
30             prix= p;
31         }
32     }
```

Cette classe a pour attributs (adresse, superficile et prix) et les constructeurs et les getters/setters.

```

27 void sets(float s){
28     superficie= s;
29 }
30 void setp(float p){
31     prix= p;
32 }
33 virtual void affichbatiment() const{
34     cout<<"l'adresse est"<<adresse<<endl;
35     cout<<"la superficie est"<<superficie<<endl;
36     cout<<"le prix est"<<prix<<endl;
37 }
38 virtual float calcprixbatiment()const{
39     float prixbat=prix*superficie;
40     return prixbat;
41 }
42 };

```

Puis on a les méthodes afficherbatiment et calculeprixbatiment().

```

3 class maison : public batiment{
4     int nbpiece;
5     bool existjardin;
6     int superficiejardin;
7     public
8     maison(){
9     }
10    maison(string adr,int nbp,    bool e,int sj):batiment(adr),nbpiece(nbp),existjardin(e),superficiejardin(sj){
11        if(existjardin==0)
12            superficiejardin=0;
13    }
14    int getnbp(){
15        return nbpiece;
16    }
17    bool gete(){
18        return existjardin;
19    }
20    int getsj(){
21        return superficiejardin;
22    }
23    void setnbp(int nbp){
24        nbpiece= nbp;
25    }
26    void sete(bool e){
27        existjardin= e;
28        if(existjardin==0)
29            superficiejardin=0;
30    }

```

Cette classe hérite du classe batiment,et qui ajoute les attributs (nbpiece, existjardin et superficijardin) et les constructeurs et les Getters/Setters.


```

31 void setsj(int sj){
32     superficiejardin= sj;
33 }
34
35 void affichbatimentantant() const{
36     cout<<"l'nbpiece est"<<nbpiece<<endl;
37     cout<<"la existjardin est"<<existjardin<<endl;
38     cout<<"le superficiejardin est"<<superficiejardin<<endl;
39 }
40
41 float calcprixbatiment()const{
42     float prixbat=(batiment::getp()/2)*superficiejardin;
43     return prixbat;
44 };

```

```

94 int main(){
95     cout <<"/-- batiment a:"<<endl;
96     batiment a;                                a.afficherbatiment() ;
97     cout<<"-----"<<endl;
98     cout <<"/-- batiment b:"<<endl;
99     batiment b("testing adresse",12,10);        b.afficherbatiment() ;
00     cout<<"-----"<<endl;
01     cout <<"/-- maison  c:"<<endl;
02     maison c("testing adresse maison",5,1,4,true,3); c.afficherbatiment() ;
03     cout<<"-----"<<endl;
04     cout<<"//-prix batiment a: "<<a.calcprixbatiment() <<endl;
05     cout<<"//-prix batiment b: "<<b.calcprixbatiment() <<endl;
06     cout<<"//-prix maison  c: "<<c.calcprixbatiment() <<endl;
07

```



Résultat :

```

F:\P.O.O\TP-5.exe
/-- batiment a:
adresse : Av. moulay ismeal,Rabat || superficie : 0 || prix per metre cube : 0
-----
/-- batiment b:
adresse : testing adresse || superficie : 12 || prix per metre cube : 10
-----
/-- maison  c:
adresse : testing adresse maison || superficie : 5 || prix per metre cube : 1
Nb pieces : 4 || existence jardin : 1 || superficie jardin : 3
-----
//-prix batiment a: 0
//-prix batiment b: 120
//-prix maison  c: 6.5
-----
Process exited after 0.1362 seconds with return value 0
Appuyez sur une touche pour continuer...

```

TP6 :

```

1  #include <iostream>
2  #include <string>
3  #include <vector>
4  using namespace std;
5
6  class livre{
7      protected:
8          string titre;
9          string auteur;
10         string editeur;
11         int annepublication ;
12
13     public:
14         livre();
15         livre(string t,string a,string e,int ap):titre(t),auteur(a),editeur(e),annepublication(ap){
16
17         }
18         livre(const livre& l){
19             titre=l.titre;
20             auteur=l.auteur;
21             editeur=l.editeur;
22             annepublication=l.annepublication;
23         }
24         void afficher(){
25             cout<<"le titre est : "<<titre<<endl;
26             cout<<"l'auteur est : "<<auteur<<endl;
27             cout<<"l'editeur est : "<<editeur<<endl;
28             cout<<"l annee de publication est : "<<annepublication<<endl;
29         }
30     };

```

Cette classe hérite du classe bâtiment,et qui ajoute les attributs (nbpiece, existjardin et superficijardin) et les constructeurs et les Getters/Setters.

```

32 class livreNumerique :virtual public livre{
33     int taille;
34     string format;
35
36     public:
37         livreNumerique();
38         livreNumerique(string t,string a,string e,int ap,int ta,string f):livre(t,a,e,ap),taille(ta),format(f){
39
40         }
41         livreNumerique(const livreNumerique& ln):livre(ln){
42             taille=ln.taille;
43             format=ln.format;
44         }
45         void afficher(){
46             livre::afficher();
47             cout<<"la taille est : "<<taille<<endl;
48             cout<<"le format est : "<<format<<endl;
49         }
50     };

```

Cette classe hérite du classe livre, et qui ajoute les attributs (taille et format) et les constructeurs et les méthodes afficher(qui fait appel au afficher(s) supérieur en utilisant la virtualisation pour éviter la confusion des méthodes de même nom).

```

53 class conteneur{
54     vector <livre*> livres;
55
56     public:
57         void ajouterlivre( livre* liv){
58             if(liv != NULL){
59                 livres.push_back(liv);
60             }
61         }
62         void afficher() const{
63             for(int i=0;i<sizeof(livres);i++){
64                 livres[i]->afficher();
65             }
66         }
67         void supprimer(livre* liv){
68             for(int i=0;i<livres.size();i++){
69                 if(livres[i] == liv){
70                     livres[i]->erase(liv);
71                     delete livres[i];
72                 }
73             }
74         }
75     }
76 };

```

Cette classe a pour attribut un vecteur de livres avec pointeur qui point sur les objets de types différents et les méthodes afficher() et ajouterlivre qui ajoute un livre par la methode **push_back**.

```

101 int main(){
102
103     livrenumerique A("A1", "A2", "A3", "1111", 11, "PDF");
104     livrenumerique B("B1", "B2", "B3", "2222", 22, "WRD");
105     livrenumerique C("C1", "C2", "C3", "3333", 33, "PPT");
106     livrenumerique D("D1", "D2", "D3", "4444", 44, "TXT");
107     livrenumerique E("E1", "E2", "E3", "5555", 55, "PYY");
108     conteneur<livrenumerique> N;
109     N.ajouter(A); N.ajouter(B); N.ajouter(C); N.ajouter(D); N.ajouter(E);
110     N.afficher() ;
111     N.supprimer("C");
112     N.afficher();

```



Résultat :

```

F:\P.O.O\TP-6.exe
le 1 eme livre :-----
A1 | A2 | A3 | 1111 | 11Mo | PDF
le 2 eme livre :-----
B1 | B2 | B3 | 2222 | 22Mo | WRD
le 3 eme livre :-----
C1 | C2 | C3 | 3333 | 33Mo | PPT
le 4 eme livre :-----
D1 | D2 | D3 | 4444 | 44Mo | TXT
le 5 eme livre :-----
E1 | E2 | E3 | 5555 | 55Mo | PYY
////////////////////////////////////
la liste des livres : //////////////////////////////////
le 1 eme livre :-----
A1 | A2 | A3 | 1111 | 11Mo | PDF
le 2 eme livre :-----
B1 | B2 | B3 | 2222 | 22Mo | WRD
le 3 eme livre :-----
C1 | C2 | C3 | 3333 | 33Mo | PPT
le 4 eme livre :-----
D1 | D2 | D3 | 4444 | 44Mo | TXT
le 5 eme livre :-----
E1 | E2 | E3 | 5555 | 55Mo | PYY
////////////////////////////////////

Process exited after 0.1454 seconds with return value 0
Appuyez sur une touche pour continuer...

```

TP7 :

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  using namespace std;
5
6
7  class Client {
8      protected:
9          string nom;
10         string cin;
11         int numCheque;
12         float montantTotal;
13
14     public:
15         Client(string n,string cn,int nc,float mt):nom(n),cin(cn),numCheque(nc),montantTotal(mt){
16         }
17
18         virtual float getMontant(){
19             return montantTotal;
20         }
21         string getnom(){
22             return nom;
23         }
24         virtual int getNCheque(){
25             return numCheque;
26         }
27     };
28
```

Cette classe a pour les attributs (nom, cin, numCheque, montantT) et les constructeurs et les Getters/Setters.

```
40 class ClientCredit : public Client{
41     float MontantMensual;
42     float LastMensual;
43     int nbrMensual;
44
45     public:
46         ClientCredit(string n,string cn,int nc,float mt,float mm,float lm,int nbr):
47             Client(n,cn,nc,mt),MontantMensual(mm),LastMensual(lm),nbrMensual(nbr){
48         }
49
50         float getMontant(){
51             if(nbrMensual==1){
52                 return LastMensual;
53             }else {
54                 return MontantMensual;
55             }
56         }
57     };

```

Cette classe hérite du classe Client, et qui ajoute les attributs (montantmensual, lastmensual et nbrmensual) et les constructeurs (qui fait appel au constructeur supérieur en utilisant).

```

29 class ClientComptant : public Client{
30     string codeBanque;
31     public:
32         ClientComptant(string n,string cn,int nc,float mt,string cb):Client(n,cn,nc,mt),codeBanque(cb){
33         }
34         float getMontant(){
35             Client::getMontant();
36         }
37     };
38 };
39
40 class ClientCredit : public Client{
41     float MontantMensual;
42     float LastMensual;
43     int nbrMensual;
44
45     public:
46         ClientCredit(string n,string cn,int nc,float mt,float mm,float lm,int nbr):
47         Client(n,cn,nc,mt),MontantMensual(mm),LastMensual(lm),nbrMensual(nbr){
48
49         }
50         float getMontant(){
51             if(nbrMensual==1){
52                 return LastMensual;
53             }else {
54                 return MontantMensual;
55             }
56         }
57 };

```

Les deux classes ClientComptant et ClientCrédit héritent du Client.

```

59 class Societe{
60     string nom;
61     vector <Client*> cls;
62     int nbrClt;
63
64     public:
65         Client& operator+=(Client* cl){
66             cls.push_back(cl);
67         }
68         string operator[](int num){
69             for(int i;i<cls.size();i++){
70                 if(num == cls[i]->getNCheque()){
71                     return cls[i]->getnom();
72                 }
73             }
74         }
75
76         float Totalite(){
77             float total=0;
78             for(int i;i<cls.size();i++){
79                 total += cls.at(i)->getMontant();
80             }
81         }
82
83 };

```

La classe Société qui contient en attributs un vecteur avec pointeur de client et les constructeurs et les opérateurs et méthode Totalité.

Contrôle Poo :

```
8 class compteBancaire{
9     private:
10         string numero;
11         double solde;
12     public:
13         compteBancaire(string a="0000111122223333",double b=0){
14             numero=a;
15             solde=b;
16         }
17         compteBancaire(const compteBancaire &a){
18             numero=a.numero ;
19             solde=a.solde ;
20         }
21         virtual void deposter(double montant){
22             solde+=montant;
23             cout << montant <<" DH est déposé au compte : " << numero <<endl;
24         }
25         virtual void retirer(double montant){
26             solde-=montant;
27             cout << montant <<" DH est retiré du compte : " << numero <<endl;
28         }
29         virtual void afficher(){
30             cout << "||// Info du compte : "<< endl;
31             cout << "||-> Numero compte: " << numero << endl;
32             cout << "||-> Solde : " << solde << endl;
33         }
34         void set_N(string a){numero=a;}
35         void set_S(double b){solde=b;}
36         string get_N(){return numero;}
37         double get_S(){return solde;}
38     };
```

La classe CompteBancaire a pour les attributs (numero et solde) et les constructeurs, les méthodes deposter et retirer en utilisant la virtualisation et méthode afficher() et aussi les getters et setters.

```
42 class compteCourant:virtual public compteBancaire{
43
44     private:
45         double decouvertAutorise;
46     public:
47         compteCourant(string a="1111222233334444",double b=0,double c=0):compteBancaire(a,b){
48             decouvertAutorise=c;
49         }
50         compteCourant(const compteCourant &a):compteBancaire(a){
51             decouvertAutorise=a.decouvertAutorise;
52         }
53         void retirer(double montant){
54             double S= compteBancaire::get_S();
55
56             if (montant <= S + decouvertAutorise){
57                 compteBancaire::retirer(montant);
58             }
59             else {
60                 cout << "- le montant dépasse la quantité max autorisé !" <<endl;
61             }
62         }
63         void afficher(){
64             compteBancaire::afficher() ;
65             cout <<"||-> Decouvert Autorisé : " << decouvertAutorise <<endl;
66         }
67         void set_D(double d){decouvertAutorise=d;}
68     };
```

La classe CompteCourant qui hérite de CompteCancaire a pour les attributs (decouvertautorise) et les constructeurs, les méthodes déposer et retirer en utilisant la virtualisation et méthode afficher() et aussi les getters et setters.

```

72 class compteEpargne:virtual public compteBancaire{
73
74     private :
75         double taxinteret;
76
77     public:
78         //constructeurs
79         compteEpargne(string a="2222333344445555",double b=0,double c=0):compteBancaire(a,b){
80             taxinteret=c;
81         }
82         compteEpargne(const compteEpargne &a):compteBancaire(a){
83             taxinteret=a.taxinteret;
84         }
85         //methode ""deposer""
86         virtual void deposer(double montant){
87             montant += montant*taxinteret;
88             compteBancaire::deposer(montant);
89         }
90         virtual void afficher(){
91             compteBancaire::afficher() ;
92             cout<< "|-> Tax Interet : "<< taxinteret <<endl;
93         }
94         double get_T(){return taxinteret;}
95         double set_T(double T){taxinteret=T;}
96
97     };

```

La même chose pour cette classe.

```

101 class comptecourantEpargne:virtual public comptecourant,virtual public compteEpargne{
102
103     private:
104
105     public:
106         comptecourantEpargne(string a="5555666677778888",double b=0,double c=0,double d=0){
107             comptecourant::compteBancaire::set_N(a);
108             comptecourant::compteBancaire::set_S(b);
109             comptecourant::set_D(c);
110             compteEpargne::set_T(d);
111         }
112         comptecourantEpargne(const comptecourantEpargne &a):comptecourant(a),compteEpargne(a){
113         }
114         void deposer(double montant){
115             compteEpargne::deposer(montant);
116             comptecourant::set_S(compteEpargne::get_S());
117         }
118         void retirer(double montant){
119             comptecourant::retirer(montant);
120             compteEpargne::set_S(comptecourant::get_S());
121         }
122         void afficher(){
123             comptecourant::afficher();
124             cout<< "|-> Tax Interet : "<< compteEpargne::get_T() <<endl;
125         }
126     };

```

La classe ci-dessus qui hérite de CompteCourant et classe compteEpargne a pour les méthodes les constructeurs, les méthodes

deposer et retirer en utilisant la virtualisation et méthode afficher() et aussi méthode afficher.

```

130 class banque{
131
132     private :
133         vector <comptebancaire* > comptesbancaires;
134     public :
135         void ajoutercompte(comptebancaire* compte){
136             for (int i=0;i<comptesbancaires.size();i++){
137                 if(comptesbancaires.at(i)->get_N() == compte->get_N()){
138                     cout <<"compte deja existant !! " <<endl;
139                     break;
140                 }
141             }
142             comptesbancaires.push_back(compte);
143             cout << "le compte : "<<compte->get_N()<<" est ajoute !!" <<endl;
144         }
145         int supprimercompte(string numero){
146             for(int i=0;i<comptesbancaires.size();i++){
147                 if(numero==comptesbancaires.at(i)->get_N()){
148                     vector <comptebancaire*> ::iterator it= comptesbancaires.begin()+i;
149                     comptesbancaires.erase(it);
150                     cout << "le compte : "<< numero << " est supprimé !!"<<endl;
151                     return 0;
152                 }
153             }
154             cout << "le compte " <<numero<<" n'existe pas !" <<endl;
155         }
156         void affichercomptes(){
157             for (int i=0;i<comptesbancaires.size();i++){
158                 cout <<" "<<endl;
159                 cout <<"++++++<<endl;
160                 cout <<"|||"<<i+1<<" eme compte : "<<endl;
161                 comptesbancaires.at(i)->afficher();
162             }
163         }
164     }
165
166     //methode ""getTotalSoldes""
167     double gettotalsoldes(){
168         double total=0;
169         for (int i=0;i<comptesbancaires.size();i++){
170             total += comptesbancaires.at(i)->get_S();
171         }
172         return total;
173     }
174 };

```

Cette classe a pour paramètre un vecteur avec pointeur de comptebancaire.


```

177 int main(){
178
179     compteBancaire A("1111 1111 1111 1111",10000);
180     compteCourant B("2222 2222 2222 2222",20000,10000);
181     compteEpargne C("3333 3333 3333 3333",30000,0.5);
182     compteCourantEpargne D("4444 4444 4444 4444",40000,5000,0.1);
183
184     D.afficher();
185     cout<<"-----"<<endl;
186     D.deposer (10000);
187     cout<<"-----"<<endl;
188     D.retirer(5000);
189     cout<<"-----"<<endl;
190     D.retirer(1000000);
191     cout<<"-----"<<endl;
192     D.afficher();
193
194     banque bank;
195     bank.ajoutercompte(&A);
196     bank.ajoutercompte(&B);
197     bank.ajoutercompte(&C);
198     bank.ajoutercompte(&D);
199

```



Résultat

F:\P.O.O\projet c++.exe

```

// Info du compte :
-> Numero compte: 4444 4444 4444 4444
-> Solde : 40000
-> Decouvert Autorisé : 5000
-> Tax Interet : 0.1
-----
11000 DH est deposé au compte : 4444 4444 4444 4444
-----
5000 DH est retiré du compte : 4444 4444 4444 4444
-----
- le montant dépasse la quantité max autorisée !!
-----
// Info du compte :
-> Numero compte: 4444 4444 4444 4444
-> Solde : 46000
-> Decouvert Autorisé : 5000
-> Tax Interet : 0.1
le compte : 1111 1111 1111 1111 est ajoute !!
le compte : 2222 2222 2222 2222 est ajoute !!
le compte : 3333 3333 3333 3333 est ajoute !!
le compte : 4444 4444 4444 4444 est ajoute !!

```

```

202     cout<<"//afficher les banques"<<endl;
203     bank.affichercomptes();
204     cout <<"/////////////////////////////////////"<<endl;
205
206     //supprimer un compte existant et non-existant
207     cout <<"//supprimer un compte existant et non-existant"<<endl;
208     bank.supprimercompte("2222 2222 2222 2222");
209     bank.supprimercompte("4444 4444 5555 4444");
210     cout <<"/////////////////////////////////////"<<endl;
211
212     //verifier le changement
213     cout << "// la list apres suppression : "<<endl;
214     bank.affichercomptes();
215

```

Résultat

```

//supprimer un compte existant et non-existant
le compte : 2222 2222 2222 2222 est supprimé !!
le compte 4444 4444 5555 4444 n'existe pas !!
////////////////////////////////////
// la list apres suppression :

+++++
|||1 eme compte :
// Info du compte :
|-> Numero compte: 1111 1111 1111 1111
|-> Solde : 10000
+++++

+++++
|||2 eme compte :
// Info du compte :
|-> Numero compte: 3333 3333 3333 3333
|-> Solde : 30000
|-> Tax Interet : 0.5
+++++

+++++
|||3 eme compte :
// Info du compte :
|-> Numero compte: 4444 4444 4444 4444
|-> Solde : 46000
|-> Decouvert Autorisé : 5000
|-> Tax Interet : 0.1

```