# Classes et objets en C++

## I – Notion d'objet

**Un objet est une entité du monde réel . On associe à un objet une représentation informatique. On distingue 2 types d'objets : les objets concrets qu'on peut toucher du doigt (Personne, Livre, …) et les objets abstraits (Compte bancaire, un Point, …). Un objet possède les caractéristiques suivantes :**

- **Un ensemble de propriétés que l'on appelle des attributs**
- **Un ensemble d'opérations que l'objet sait réaliser ou que l 'on peut réaliser sur l'objet. Ces opérations sont organisées en fonctions appelées méthodes**
- **Un nom unique qui l'identifie dans un programme**

**Exemples :**
**Un point possède comme attributs une abscisse x et une ordonnée y. On peut définir une méthode translater(…) qui est une fonction qui permet de déplacer un point quelconque .**

**Un compte bancaire peut posséder les attributs suivants : un numéro, un libellé, un sens et un**

**solde. On peut définir les méthodes suivantes : crediter(..), debiter(…), transferer(…) sur des comptes bancaires .**

## II-Cycle de vie d'un objet

- **Un objet doit d'abord être créé : créer un objet consiste à définir les valeurs initiales de ses attributs. Par exemple un point P1 est créé si on fournit les valeurs de ses coordonnées x et y. Un compte C1 est créé lorsqu'on fixe les valeurs du numéro , du sens , du libellé et du solde.**
- **Après la création de l'objet , on fait appel aux méthodes qui permettent d'exploiter l'objet. Par exemple, on peut créditer ou débiter le compte C1 ou translater le Point P1.**
- **Destruction de l'objet automatique**

## III-Notion de Classe

**Les objets qui possèdent les mêmes attributs et auxquels on peut appliquer les mêmes opérations(méthodes) sont regroupés dans une famille appelée Classe.**

**Tous les langages de programmations possèdent des types prédéfinis : int , double, char, etc…**

**Lorsque l'utilisateur crée une classe, il crée alors un nouveau type qui étend donc les types existants. Il lui faut alors définir des variables de ce type appelées instances d'objets.**

**Ainsi, on a les correspondances suivantes :**

| Programmation structurée | Programmation objet |
|---|---|
| Types | Classes |
| Variables | instances d'objets |
| Fonctions | Méthodes |

**Une classe est une structure de programmation définie pour des objets issus d'une même famille et dans laquelle on décrit :**

- **Leurs attributs communs**
- **Leurs méthodes communes**

## IV- Notion d'encapsulation

**Un objet doit être protégé lors qu'on y accède. Pour pouvoir accéder aux valeurs des attributs en dehors de la classe où ils sont définis , on doit utiliser des méthodes. Ainsi pour accéder à la valeur d'un attribut en dehors de sa classe de définition, on utilise une méthode appelée accesseur ou getter. Pour fixer ou**

**modifier la valeur d'un attribut en dehors de sa classe de définition, on utilise une méthode appelée modificateur ou setter.**

## V-Notion de constructeur

**Avant de pouvoir faire appel à un getter, un setter ou autre méthode, on doit d'abord créer l'objet.**

**Créer un objet consiste à fixer les valeurs initiales de ses attributs.**

**Cette opération se fait via une méthode spécifique de la classe appelée un constructeur.**

**L'utilisateur doit spécifier au moins un ou plusieurs constructeurs.**

**S'il n'en définit pas, le système propose un constructeur par défaut qui crée des objets vides c.a.d. des objets sans valeurs initiales des attributs. Il faudra alors dans ce cas faire appel aux setters pour fixer les valeurs initiales des attributs de l'objet.**

**L'utilisateur doit éviter le constructeur par défaut et proposer son propre constructeur.**

**Le constructeur d'une classe C quelconque est une méthode qui porte le même nom que la classe qui ne possède pas de type de retour (même pas le type void) et qui peut être avec ou sans paramètres. Si le**

**constructeur possède des arguments, ces derniers représentent alors les valeurs initiales à fournir aux arguments.**

**C'est pourquoi l'on dit que les objets sont créés à partir de leurs classes ( rôle du constructeur). Les objets sont alors des instances de leurs classes.**

**En général, la règle est la suivante :**

**Si une classe possède n attributs, le constructeur doit alors posséder n arguments, un argument pour initialiser chaque attribut.**

**Ainsi dans une classe , l'ordre de définition des méthodes est le suivant :**

- **Définir un constructeur avec arguments**
- **Définir les getters et setters**
- **Définir les autres méthodes**

# VI – Objets statiques

## 6-1 Constructeur par défaut

**Exemple d'une classe point utilisant un constructeur par défaut. Les setters sont utilisés pour fixer les valeurs des attributs.**

**A - Ficher point.h**

```
#ifndef POINT_H
#define POINT_H
#include <iostream>
```

```
using namespace std;
class point
{
private:
   double x;
   double y;
public:
   double getx();
   double gety();
   void setx(double ax);
   void sety(double ay);

};
#endif // POINT_H
```

**B/ fichier point.cpp**

```
#include "point.h"
#include <iostream>
using namespace std;

double point::getx()
{
   return x;
}
```

```
double point::gety()
{
   return y;
}

void point::setx(double ax)
{
   x=ax;
}

void point::sety(double ay)
{
   y=ay;
}
```

C/ fichier application.cpp

```
#include "point.h"
#include <iostream>
using namespace std;
int main()
{
   point p1;
   p1.setx(5);
   p1.sety(6);
   cout<<"abscisse de p1:"<<p1.getx()<<endl;
```

```
   cout<<"ordonnee de p1:"<<p1.gety()<<endl;
}
```

**6-2 Constructeur avec paramétres**

**A/fichier point.h**

```
#ifndef POINT_H
#define POINT_H
#include <iostream>
using namespace std;
class point
{
private:
   double x;
   double y;
public:
   point(double ax,double ay);
   double getx();
   double gety();
   void setx(double ax);
   void sety(double ay);

};
#endif // POINT_H
```

**B/ fichier point.cpp**

```
#include "point.h"
```

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

point::point(double ax,double ay)
{
   x=ax;
   y=ay;
}
double point::getx()
{
   return x;
}

double point::gety()
{
   return y;
}

void point::setx(double ax)
{
   x=ax;
}

void point::sety(double ay)
```

```
{
   y=ay;
}
```

**C/ fichier application.cpp**

```
#include "point.h"
#include <iostream>
using namespace std;
int main()
{
   point p1(-6,-5);
   cout<<"abscisse de p1:"<<p1.getx()<<endl;
   cout<<"ordonnee de p1:"<<p1.gety()<<endl;
}
```

**6-3 Constructeur interactif**

**Un constructeur interactif est un constructeur sans paramètres dans lequel les valeurs des attributs sont saisis au clavier ( très utile pour les applications)**

**A/ fichier point.h**

```
#ifndef POINT_H
#define POINT_H
#include <iostream>
using namespace std;
class point
{
```

```
private:
   double x;
   double y;
public:
   point();
   double getx();
   double gety();
   void setx(double ax);
   void sety(double ay);

};
#endif // POINT_H
```

B/ fichier point.cpp

```
#include "point.h"
#include <iostream>
using namespace std;

point::point()
{
   cout<<"donnez l'abscisse"<<endl;
   cin>>x;
   cout<<"donnez l'ordonnée"<<endl;
   cin>>y;
}
```

```
double point::getx()
{
   return x;
}

double point::gety()
{
   return y;
}

void point::setx(double ax)
{
   x=ax;
}

void point::sety(double ay)
{
   y=ay;
}
```

**C/ fichier point.cpp**

```
#include "point.h"
#include <iostream>
#include <vector>
using namespace std;
```

```
int main()
{
   point p1;
    cout<<"abscisse du point p1 "<<p1.getx()<<endl;
    cout<<"ordonnée du point p1 "<<p1.gety()<<endl;
   return 0;
}
```

## 6-4 Vecteur d'objets statiques

**Lorsqu'on veut gérer un très grand nombre d'objets, on utilise un vecteur qui est un tableau de taille illimitée.**

### A/ fichier voiture.h

```
#ifndef VOITURE_H
#define VOITURE_H
#include <iostream>
#include <string>
using namespace std;

class voiture
{
   private:
    string numero;
    string marque;
    int prix;
   public:
```

```
    voiture();
    string getnumero();
    string getmarque();
    int getprix();
};
#endif // VOITURE_H
```

**B/ fichier voiture.cpp**

```
#include "voiture.h"
#include <iostream>
#include <string>
using namespace std;

voiture::voiture()
{
   cout<<"Donnez le numero"<<endl;
   getline(cin,numero);
   cout<<"Donnez la marque"<<endl;
   getline(cin,marque);
   cout<<"Donnez le prix"<<endl;
   cin>>prix;
   cin.get();



}
```

```
string voiture::getnumero()
{
   return numero;
}

string voiture::getmarque()
{
   return marque;
}

int voiture::getprix()
{
   return prix;
}
```

**C/ application.cpp**

```
#include "voiture.h"
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main()
{
  int  choix=0;
```

```
vector<voiture> lt;
string mq;

do
{
   cout<<"1.ajouter une voiture"<<endl;
   cout<<"2.afficher les voitures"<<endl;
   cout<<"3.afficher les voitures d'une marque donnée"<<endl;
   cout<<"4.Sortie"<<endl;
   cout<<"   Votre choix?"<<endl;
   cin>>choix;
   cin.get();
   switch(choix)
   {
      case 1:
           {
            voiture v;
           lt.push_back(v);


           }
           break;
      case 2:
          for (int i=0;i<lt.size();i++)
           {
```

```
      cout<<"_____________________________"<<endl;
      cout<<"numero: "<<lt[i].getnumero()<<endl;
      cout<<"marque: "<<lt[i].getmarque()<<endl;
      cout<<"prix: "<<lt[i].getprix()<<endl;
      }
     cout<<"_____________________________"<<endl;
      break;
case 3:
       cout<<"Donnez la marque"<<endl;
       getline(cin,mq);
       for (int i=0;i<lt.size();i++)
        {

       if (lt[i].getmarque().compare(mq)==0)
       {
        cout<<"_____________________________"<<endl;
        cout<<"numero: "<<lt[i].getnumero()<<endl;
        cout<<"marque: "<<lt[i].getmarque()<<endl;
        cout<<"prix: "<<lt[i].getprix()<<endl;
        cout<<"_____________________________"<<endl;

       }

      }
```

```
            break;
        case 4:
            break;
    }

  }
  while(choix!=4);
  cout <<"Fin du programme "<<endl;

  return 0;
}
```

# VI – Objets dynamiques

**Un objet dynamique est déclaré sous la forme d'un pointeur sur la classe**

## 7-1 Exemple d'objets dynamiques

**A/ fichier point.h**

```
#ifndef POINT_H
#define POINT_H
#include <iostream>
using namespace std;
class point
{
private:
  double x;
```

```
    double y;
public:
    point();
    point(double ax,double ay);
    double getx();
    double gety();

};
#endif // POINT_H
```

B/ fichier point.cpp

```
#include "point.h"
#include <iostream>
using namespace std;

point::point()
{
    cout<<"valeur de x?"<<endl;
    cin>>x;
    cout<<"valeur de y?"<<endl;
    cin>>y;
}

 point::point(double ax,double ay)
{
```

```
   x=ax;
   y=ay;
}
double point::getx()
{
   return x;
}

double point::gety()
{
   return y;
}
```

C/ fichier application.cpp

```
#include "point.h"
#include <iostream>
using namespace std;
int main()
{
   point * p1 = new point;
   cout<<"abscisse de p1:"<<p1->getx()<<endl;
   cout<<"ordonnée de p1:"<<p1->gety()<<endl;
   point * p2 = new point(5,6);
   cout<<"abscisse de p2:"<<p2->getx()<<endl;
   cout<<"ordonnée de p2:"<<p2->gety()<<endl;
```

```
return 0;
}
```

**7-2 Liste d'objets dynamiques**

**A/ fichier voiture.h**

```
#ifndef VOITURE_H
#define VOITURE_H
#include <iostream>
#include <string>
using namespace std;

class voiture
{
   private:
    string numero;
    string marque;
    int prix;
   public:
    voiture();
    string getnumero();
    string getmarque();
    int getprix();
};
#endif // VOITURE_H
```

**B/ fichier voiture.cpp**

```
#include "voiture.h"
#include <iostream>
#include <string>
using namespace std;

voiture::voiture()
{
   cout<<"Donnez le numero"<<endl;
   getline(cin,numero);
   cout<<"Donnez la marque"<<endl;
   getline(cin,marque);
   cout<<"Donnez le prix"<<endl;
   cin>>prix;
   cin.get();



}

string voiture::getnumero()
{
   return numero;
}
```

```
string voiture::getmarque()
{
   return marque;
}

int voiture::getprix()
{
   return prix;
}
```

C/fichier application.cpp

```
#include "voiture.h"
#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main()
{
  int  choix=0;
   vector<voiture *> lt;
   string mq;

   do
   {
      cout<<"1.ajouter une voiture"<<endl;
```

```
cout<<"2.afficher les voitures"<<endl;
cout<<"3.afficher les voitures d'une marque donnée"<<endl;
cout<<"4.Sortie"<<endl;
cout<<"   Votre choix?"<<endl;
cin>>choix;
cin.get();
switch(choix)
{
   case 1:
        {
         voiture * v = new voiture;
        lt.push_back(v);

        }
        break;
   case 2:
       for (int i=0;i<lt.size();i++)
        {
        cout<<"_____"<<endl;
        cout<<"numero: "<<lt[i]->getnumero()<<endl;
        cout<<"marque: "<<lt[i]->getmarque()<<endl;
        cout<<"prix: "<<lt[i]->getprix()<<endl;
        }
       cout<<"_____"<<endl;
```

```
            break;
        case 3:
             cout<<"Donnez la marque"<<endl;
             getline(cin,mq);
             for (int i=0;i<lt.size();i++)
              {

             if (lt[i]->getmarque().compare(mq)==0)
             {
              cout<<"_____"<<endl;
              cout<<"numero: "<<lt[i]->getnumero()<<endl;
              cout<<"marque: "<<lt[i]->getmarque()<<endl;
              cout<<"prix: "<<lt[i]->getprix()<<endl;
              cout<<"_____"<<endl;

             }

            }
             break;
        case 4:
           break;
    }

}
```

```
   while(choix!=4);
   cout <<"Fin du programme "<<endl;


   return 0;
}
```

**VIII –TP : Projet à réaliser (voir explications)**

**Créer une classe Etudiant comportant qui permet de stocker le numero, nom, prenom, un tableau de notes et un tableau de coefficients associés aux notes. En plus du constructeur interactif qui permettra de saisir ces données, cette classe devra proposer une méthode qui affiche la moyenne de chaque étudiant. On utilisera une liste d'objets dynamiques.**

**A/ fichier etudiant.h**

```
#ifndef ETUDIANT_H
#define ETUDIANT_H
#include <iostream>
#include <string>
using namespace std;


class etudiant
{
private:
   string numero;
   string nom;
   string prenom;
```

```
    double * notes;
    int * coef;
    int nombre;
    public:
    etudiant();
  void afficher();

};
#endif // POINT_H
```

B/ fichier etudiant.cpp

```
#include "etudiant.h"
#include <iostream>
#include <string>
using namespace std;

etudiant::etudiant()
{
    cout<<"Numéro Etudiant?"<<endl;
    getline(cin,numero);
    cout<<"Nom Etudiant?"<<endl;
    getline(cin,nom);
    cout<<"Prénom Etudiant?"<<endl;
    getline(cin,prenom);
    cout<<"Nombre de notes?"<<endl;
```

```
    cin>>nombre;
    notes = new double[nombre];
    coef = new int[nombre];
    for (int i=0;i<nombre;i++)
    {
     cout<<"Note numero "<<i+1<<" ?"<<endl;
     cin >>notes[i];
     cout<<"Coef numero "<<i+1<<" ?"<<endl;
     cin >>coef[i];

    }
    cin.get();

}

void etudiant::afficher()
{
    double somnotes=0;
    int somcoef=0;
    double moy;
    cout<<"Numéro Etudiant: "<<numero<<endl;
    cout<<"Nom Etudiant: "<<nom<<endl;
    cout<<"Prénom Etudiant: "<<prenom<<endl;
    cout<<"Numero     Notes     Coef "<<endl;
```

```
   for(int i=0;i<nombre;i++)
   {
      cout<<(i+1)<<"        "<<notes[i]<<"       "<<coef[i]<<endl;
      somnotes=somnotes+notes[i]*coef[i];
      somcoef=somcoef+coef[i];
   }
   moy=somnotes/somcoef;
   cout<<"moyenne:"<<moy<<endl;
}
```

C/ fichier application.cpp

```
#include "etudiant.h"
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main()
{
   int  choix=0;
   vector<etudiant*> lt;

   do
   {
      cout<<"1.ajouter un étudiant"<<endl;
```

```
cout<<"2.afficher les étudiants et leurs moyennes"<<endl;
cout<<"3.Sortie"<<endl;
cout<<"   Votre choix?"<<endl;
cin>>choix;
cin.get();
switch(choix)
{
   case 1:
      {
       etudiant * et = new etudiant;
       lt.push_back(et);
       break;
      }

   case 2:
       for (int i=0;i<lt.size();i++)
       {
       cout<<"_____"<<endl;
       lt[i]->afficher();
       }
      cout<<"_____"<<endl;
       break;
   case 3:
        break;
```

```
        }


    }
    while(choix!=3);
    cout <<"Fin du programme "<<endl;
}
```