TP 2 - Premières classes

Programmation Orientée Objet

1 Classe Produit

1) On souhaite développer une classe *Produit* permettant de traiter des produits divers. Chaque produit est caractérisé par un nom et un identifiant de type entier. Développez la classe en la dotant d'accesseurs.

2)Surchargez l'opérateur d'affichage de cette classe ainsi que son opérateur d'affectation.

3)Implémentez un constructeur et un destructeur. Ils afficheront un message lors de leur exécution afin de visualiser leur fonctionnement. Testez les avec un programme principal (on utilisera un programme séparé et on réalisera un *makefile*).

4) Instanciez dans le programme principal un tableau statique de Produits. Remplissez-le et affichez-le.

5) Faite comme avec la question 4 mais en utilisant un tableau dynamique.

6) Pour les deux tableaux, tester les destructeurs et constructeurs par affichage.

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


class Produit
{
        int iId;
        string sNom;
public:
        Produit ();
        ~Produit ();

        // getteurs et setteurs
        int getId () const;
        void setId (const int);
        string getNom () const;
        void setNom (const string &);

        // surcharge des operateurs
        friend ostream & operator<< (ostream & out, Produit & p);
        Produit& operator= (const Produit& m);
};
#endif
```

```
#include "Produit.h"

////////////////////////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////////////////////////

        Produit::Produit (){
#ifdef DEBUG
                cout << "\t** Constructeur **"<<endl;
#endif
        }
        Produit::~Produit (){
#ifdef DEBUG
                cout << "\t** Destructeur **"<<endl;
#endif
        }
        int Produit::getId () const{
                return iId;
        }
        void Produit::setId (const int i){
                iId=i;
        }
        string Produit::getNom () const{
                return sNom;
```

```
        }
        void Produit::setNom (const string &s){
                sNom=s;
        }

        ostream & operator<< (ostream & out, Produit & p){
#ifdef DEBUG
                cout << "Je suis dans l operateur d affichage" << endl;
#endif
                out << p.iId << " "  << p.sNom << endl;
                return out;
        }

        Produit& Produit::operator= (const Produit& p){
#ifdef DEBUG
                cout << "Je suis dans l operateur d affectation" << endl;
#endif
                iId = p.iId;
                sNom = p.sNom;
                return *this;
        }
```

```
#include "Produit.h"
#include "Stock.h"
////////////////////////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////////////////////////

const int TAILLE=3;

int main()
{
        cout << "\t******************" << endl;
        cout << "\tDebut du Programme" << endl;
        cout << "\t******************" << endl;

        // Declaration de P1
        cout << "\tDeclaration et instanciation de P1" << endl;
        Produit P1;

        P1.setNom("Corde simple");
        P1.setId(123456);

        cout << P1 << endl;

        // Declaration de P2
        cout << "\tDeclaration de P2" << endl;
        Produit *P2;

        cout << "\tInstanciation de P2" << endl;
        P2 = new Produit();;

        *P2 = P1;

        cout << *P2 << endl;

        delete P2;

        //***********
        // Question 4
        //***********

        Produit stockStatique [TAILLE];
        int iCode=0;
        string sNom;

        // Saisie du tableau statique
        cout << "\t*******************************" << endl;
        cout << "\t** Saisie du tableau statique **" << endl;
        cout << "\t*******************************" << endl;
        for(int i=0; i< TAILLE; i++){
                cout << "Entrez le nom du produit : ";
                cin >> sNom;
                stockStatique[i].setNom(sNom);
                stockStatique[i].setId(++iCode);
        }

        // Affichage du tableau statique
        cout << "\t*******************************" << endl;
        cout << "\t** Affichage du tableau statique **" << endl;
```

```
        cout << "\t*******************************" << endl;
        for(int i=0; i< TAILLE; i++){
                cout << stockStatique[i] << endl;
        }


        //***********
        // Question 5
        //***********

        Produit *stockDynamique = new Produit [TAILLE];

        // Saisie du tableau statique
        cout << "\t*******************************" << endl;
        cout << "\t** Saisie du tableau dynamique **" << endl;
        cout << "\t*******************************" << endl;
        for(int i=0; i< TAILLE; i++){
                cout << "Entrez le nom du produit : ";
                cin >> sNom;
                stockDynamique[i].setNom(sNom);
                stockDynamique[i].setId(++iCode);
        }

        // Affichage du tableau statique
        cout << "\t*******************************" << endl;
        cout << "\t** Affichage du tableau dynamique **" << endl;
        cout << "\t*******************************" << endl;
        for(int i=0; i< TAILLE; i++){
                cout << stockDynamique[i] << endl;
        }

        delete [] stockDynamique;

        cout << "\t******************" << endl;
        cout << "\t Fin du Programme " << endl;
        cout << "\t******************" << endl;
        return (EXIT_SUCCESS);
}
```

```
OPTS=-DDEBUG
all: main
main: Main.cpp Produit.o Stock.o
        g++ ${OPTS} Produit.o Stock.o Main.cpp -o main
Produit.o: Produit.cpp
        g++ ${OPTS} -c Produit.cpp
Stock.o: Stock.cpp
        g++ ${OPTS} -c Stock.cpp

clean:
        rm *.o
```

2 Classe Stock

Implémentez une classe *Stock* gérant un stock de produits. Pour cela, vous définirez les constructeurs, destructeurs et opérateurs que vous jugerez nécessaires. Le stock sera géré par simplicité en utilisant un tableau statique de produits.

```
#ifndef STOCK_H
#define STOCK_H
#include<iostream>
#include "Produit.h"
using namespace std;

const int TAILLESTOCK=10;

class Stock
{
        int iNbElt;
        Produit tab[TAILLESTOCK];

        void saisirProduit(Produit&);
public:
        Stock ();
        ~Stock ();

        // getteurs et setteurs
        int getNbElt () const;
```

```
        void setNbElt (const int);

        //Fonction utilitaire
        void RemplirStock();

        // surcharge des operateurs
        friend ostream & operator<< (ostream & out, Stock & p);
        //Stock& operator= (const Stock& m);
};

#endif
```

```
#include "Stock.h"

////////////////////////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////

        Stock::Stock (){
#ifdef DEBUG
                cout << "\t** Constructeur **"<<endl;
#endif
        iNbElt=0;
        }
////////////////////////////////////////////////////////////////////////////////////////////////////

        Stock::~Stock (){
#ifdef DEBUG
                cout << "\t** Destructeur **"<<endl;
#endif
        }
////////////////////////////////////////////////////////////////////////////////////////////////////

        int Stock::getNbElt () const{
                return iNbElt;
        }
////////////////////////////////////////////////////////////////////////////////////////////////////

        void Stock::setNbElt (const int i){
                iNbElt=i;
        }
////////////////////////////////////////////////////////////////////////////////////////////////////
        void Stock::saisirProduit(Produit & P){
                int i;
                string s;
                cout << "Saisie d'un produit :" << endl ;
                cout << "**********************" << endl;
                cout << "Entrer le code du produit :";
                cin >> i;
                cout << "Entrer le nom du produit :";
                cin >> s;

                P.setNom(s);
                P.setId(i);
        }

////////////////////////////////////////////////////////////////////////////////////////////////////
        void Stock::RemplirStock(){
                Produit P;
                char cRep='o';
                do{
                        if (iNbElt < TAILLESTOCK){
                                cout << endl << " Il reste " << TAILLESTOCK - iNbElt << " emplacements de libr

                                saisirProduit(tab[iNbElt++]);
                        }else{
                                cout << "Le stock est complet!!!!" << endl;
                        }
                        cout << endl << "Voulez-vous ajouter un element au stock? (o/n) : ";
                        cin >> cRep;
                }while ((cRep != 'n')&&(cRep !='N'));
        }

////////////////////////////////////////////////////////////////////////////////////////////////////
        ostream & operator<< (ostream & out, Stock & p){
#ifdef DEBUG
                cout << "Je suis dans l operateur d affichage" << endl;
#endif
                out << "Stock de : " << p.iNbElt << " Ã©lÃ©ments" << endl;
```

```
                for (int i=0; i < p.iNbElt; i++)
                        out << "** : " << p.tab[i] << endl;
                return out;
        }
/*
        Produit& Produit::operator= (const Produit& p){
#ifdef DEBUG
                cout << "Je suis dans l operateur d affectation" << endl;
#endif
                iId = p.iId;
                sNom = p.sNom;
                return *this;
        }
*/
```

```
#include "Produit.h"
#include "Stock.h"
////////////////////////////////////////////////////////////////////////////////////////////////////
// Programme principal
////////////////////////////////////////////////////////////////////////////////////////////////////

int main()
{
        cout << "\t******************" << endl;
        cout << "\tDebut du Programme" << endl;
        cout << "\t******************" << endl;

Stock stock;

stock.RemplirStock();

cout << stock << endl;

        cout << "\t******************" << endl;
        cout << "\t Fin du Programme " << endl;
        cout << "\t******************" << endl;
        return (EXIT_SUCCESS);
}
```

```
OPTS=-DDEBUG
all: main
main: Main.cpp Produit.o Stock.o
        g++ ${OPTS} Produit.o Stock.o Main.cpp -o main
Produit.o: Produit.cpp
        g++ ${OPTS} -c Produit.cpp
Stock.o: Stock.cpp
        g++ ${OPTS} -c Stock.cpp

clean:
        rm *.o
```