Correction  dernier TP Héritage

Remarque : pour que votre programme fonctionne (typeid donne le résultat voulu) , vous devez déclarer au niveau de la classe Equipement au moins une fonction virtuelle

```
#include "iostream"
#include "string"
#include "vector"
using namespace std;
```

## /*  Classe Equipement  */

```
class Equipement
{protected:
      int SN;
      string marque;
      double prix;
public:
      Equipement(int SN=0,string marque="",double prix=0)
      {this->SN=SN;
      this-> marque=marque;
      this-> prix=prix;
      }

      virtual void affiche(){}
      void setSN(int s){SN=s;}
      void setMarque(string m){marque=m;}
      void setPrix(double p){prix=p;}

      int getSN(){return SN;}
      string getMarque(){return marque;}
      double getPrix() {return prix;}

       friend ostream& operator<< (ostream& o , Equipement i)
       {o<<"SN : "<<i.SN<<"  Marque: "<<i.marque<<" Prix: "<<i.prix<<endl;
       return o;}


};
```

## /*  Classe Imprimante  */

```
class Imprimante :
      public Equipement
{
private:
      int vitesse;
public:
      Imprimante(int SN=0,string marque="",double prix=0, string p="", int
v=0): Equipement(SN,marque, prix)
      {vitesse=v;}

      int getVitesse(){return vitesse;}
      void setVitesse(int v){vitesse=v;}
       bool operator == (Imprimante i)
```

```
        {if(prix==i.prix)
        return true;
        else
        return false;}

        friend ostream& operator<< (ostream& , Imprimante );
         friend istream& operator>> (istream& ,  Imprimante &);
};
```

## /*  Classe Ordinateur  */

```
class Ordinateur: public Equipement
{
private:
      string processeur;
      int memoire;
public:

      Ordinateur(int SN=0,string marque="",double prix=0, string p="", int
m=0): Equipement(SN,marque, prix)
      {processeur=p; memoire=m;}

      bool operator<= (const Ordinateur & o)
            {if (prix<=o.prix)
            return true;
            return false;
            }
      friend ostream& operator<<(ostream& o, Ordinateur or);


      string getProcesseur(){return processeur;}
      int getMemoire() {return memoire;}

      void setProcesseur(string p){processeur=p;}
      void setMemoire(int m){memoire=m;}

      Ordinateur& operator- (double p)
      {
      prix-=p;
      return (*this);
      }
};
```

## /*  Classe Stock  */

```
class Stock
{
      vector <Equipement*> tab;
      int taille;

public:
      Stock(int t=10){taille=t;}
      ~Stock();
      Stock(const Stock &);
      Stock& operator+ (Equipement *);
      friend ostream& operator<< (ostream& , const Stock &);
};
```

## /*  Fichier Stock.cpp  */

```
Stock::~Stock()
{for(int i=0 ;i <tab.size(); i++)
  delete tab[i];
}

Stock::Stock(const Stock & S)
{
taille= S.taille;
Equipement *e;

for(int i=0; i<S.tab.size(); i++)
{
if(typeid(*S.tab[i]).name()== typeid(Ordinateur).name())
      e= new Ordinateur (*(static_cast<Ordinateur*>(S.tab[i])));
else
      if(typeid(*S.tab[i]).name()== typeid(Imprimante).name())
            e= new Imprimante (*(static_cast<Imprimante*>(S.tab[i])));
tab.push_back(e);
}
}


Stock& Stock::operator+( Equipement * eq)
{


      Equipement *e;
      if(tab.size()<taille)
      {if(typeid(*eq).name()== typeid(Ordinateur).name())
      e= new Ordinateur (*(static_cast<Ordinateur*>(eq)));
else
      if(typeid(*eq).name()== typeid(Imprimante).name())
            e= new Imprimante (*(static_cast<Imprimante*>(eq)));
tab.push_back(e);
      }
return (*this);
}
```

## /*  Les fonctions amies  */

```
ostream& operator<< (ostream& o , Imprimante i)
{o<<"SN : "<<i.SN<<"  Marque: "<<i.marque<<" Prix: "<<i.prix;
o<<"Vitesse"<<i.vitesse<<endl;
      return o;
}

istream& operator>> (istream& i ,  Imprimante & im)
{
cout<<"Donner SN  ";
i>>im.SN;
cout<<"Donner Marque  ";
i>>im.marque;
cout<<"Donner Prix: ";
```

```
i>>im.prix;
cout<<"Donner Vitesse  ";
i>>im.vitesse;
return i;
}



ostream& operator<<(ostream& o, Ordinateur or)
{
      o<<"SN : "<<or.SN<<"  Marque: "<<or.marque<<" Prix: "<<or.prix;
      o<<"Processeur: "<<or.processeur<<"Memoire "<<or.memoire<<endl;
      return o;
}



ostream& operator<< (ostream& o , const Stock & S)
{
for(int i=0; i<S.tab.size(); i++)
{

      if(typeid(*S.tab[i]).name()== typeid(Ordinateur).name())
o<< (*(static_cast<Ordinateur*>(S.tab[i])));
else
      if(typeid(*S.tab[i]).name()== typeid(Imprimante).name())
            o<<(*(static_cast<Imprimante*>(S.tab[i])));
}

return o;
 }
```

## /*  Programme Principal  */

```
void main()
{

      Equipement *e=new Ordinateur (1);
Equipement *e1=new Ordinateur(2);
Equipement *e2= new Imprimante (3);
Equipement *e3=new Imprimante (4);

Stock S;
S+(e);
S+(e1);
S+(e2);
S+(e3);

cout<<S;
int x;
cin>>x;

}
```