**Les classes abstraites**

abstract class Produit

{

protected string reference;

protected string designation;

protected int prixVente;

protected Produit pdt;

protected List<Produit> produits = new List<Produit>();

public Produit(string reference, string designation, int prixVente)

{

this.reference = reference;

this.designation = designation;

this.prixVente = prixVente;

}

public abstract void Afficher();

public void proc(Produit pdt)

{

this.pdt = pdt;

pdt.Afficher();

}

public void addListPdt(Produit pdt)

{

this.produits.Add(pdt);

}

public void afficherListPdt()

{

for (int i = 0; i < produits.Count; i++)

{

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Console.WriteLine(produits.ElementAt(i).GetType());

produits.ElementAt(i).Afficher();

}

}

}

La classe produit est la classe mère. C’est une classe abstraite, elle ne va donc pas instancier d’objet, ce seront ses sous-classes qui le feront.

class PdtFab : Produit

{

private int numFab;

private int nbHFab;

private int coutMat;

public PdtFab(string reference, string designation, int prixVente, int numFab, int nbHFab, int coutMat) : base(reference, designation, prixVente)

{

this.numFab = numFab;

this.nbHFab = nbHFab;

this.coutMat = coutMat;

}

public override void Afficher()

{

Console.WriteLine("référence : " + reference);

Console.WriteLine("désignation : " + designation);

Console.WriteLine("prix de vente : " + prixVente);

Console.WriteLine("numero fabriqué : " + numFab);

Console.WriteLine("nombre d'heure nécessaire : " + nbHFab);

Console.WriteLine("cout de la matière première : " + coutMat);

}

}

Override de la procédure Afficher() qui porte la même signature (le même nom) que celle de la super classe Produit.

class PdtAch : Produit

{

private int numFournisseur;

private int prixAchat;

public PdtAch(string reference, string designation, int prixVente, int numFournisseur, int prixAchat) : base(reference, designation, prixVente)

{

this.numFournisseur = numFournisseur;

this.prixAchat = prixAchat;

}

public override void Afficher()

{

Console.WriteLine("référence : " + reference);

Console.WriteLine("désignation : " + designation);

Console.WriteLine("prix de vente : " + prixVente);

Console.WriteLine("numero fournisseur : " + numFournisseur);

Console.WriteLine("prix d'achat : " + prixAchat);

}

}

Il n’y a pas de copie dans la méthode Affiche de Produit, on est obligé de recopier la référence, la désignation et le prix de vente dans les méthodes filles.

class Program

{

static void Main(string[] args)

{

Produit pdt;

PdtFab pf;

PdtAch pa;

// pdt = new Produit("ref", "", 0);

pf = new PdtFab("f01", "réveil matin", 150, 1, 10, 50);

pa = new PdtAch("a01", "montre bracelet", 75, 1, 45);

pdt = pa;

Console.WriteLine("Affichage d'un produit fabriqué : ");

pdt.proc(pf);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ");

Console.WriteLine("Affichage d'un produit acheté : ");

pdt.proc(pa);

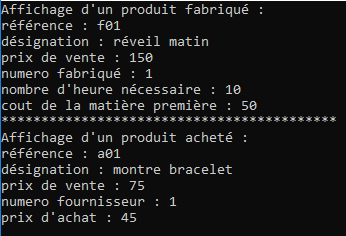
Console.ReadLine();

}

}

Si on enlève le commentaire qui définit un nouveau Produit, cela affiche une erreur car une classe abstraite ne peut pas être instancié.

## Screenshot



<?php

abstract class Cours{

protected $id;

protected $ageMini;

protected $lesJours; // collection des jours où a lieu le cours

public function \_\_construct($unId,$unAgeMini){ // constructeur

$this->id=$unId;

$this->ageMini=$unAgeMini;

$this->lesJours=array();

}

abstract protected function Afficher();

}

class CoursIndividuel extends Cours{

private $lInstrument; //instrument concerné par le cours individuel

public function \_\_construct($unId,$unAgeMini,$unInstrument){ // constructeur

parent::\_\_construct($unId,$unAgeMini);

$this->lInstrument=$unInstrument;

}

public function GetLInstrument(){

return $this->lInstrument ;

}

public function GetLibelle(){

// retourne le libellé du cours individuel qui est en fait l’intitulé de l’instrument

return $this->lInstrument;

}

public function Afficher()

{

echo "<br>Cours individuel : <br>";

echo "id : ".$this->id;

echo "<br>Age minimum : ".$this->ageMini;

echo "<br>Instrument : ".$this->GetLibelle()."<br>";

}

}

//extends permet l’héritage des classes en php

class CoursCollectif extends Cours{

private $libelle; // libellé du cours collectif

private $ageMaxi;

private $nbPlacesMaxi;

// constructeur

public function \_\_construct($unId,$unAgeMini,$unLibelle,$unAgeMaxi,$unNbPlacesMaxi){

parent::\_\_construct($unId,$unAgeMini);

$this->libelle=$unLibelle;

$this->ageMaxi=$unAgeMaxi;

$this->nbPlacesMaxi=$unNbPlacesMaxi;

}

public function GetLibelle(){

// retourne le libellé du cours collectif

return $this->libelle;

}

public function GetAgeMaxi(){

return $this->ageMaxi;

}

public function GetNbPlacesMaxi(){

return $this->nbPlacesMaxi;

}

public function Afficher()

{

echo "<br>Cours collectif : <br>";

echo "id : ".$this->id;

echo "<br>Age minimum : ".$this->ageMini;

echo "<br>Libellé : ".$this->libelle;

echo "<br>Age maximum".$this->ageMaxi;

echo "<br>Nombre de places maximum : ".$this->nbPlacesMaxi."<br>";

}

}

$ci[] = new CoursIndividuel(1,12,"violon");

$ci[1] = new CoursIndividuel(2,6,"piano");

$cc[] = new CoursCollectif(1,8,"apprentissage du solfège",18,"20");

$cc[1] = new CoursCollectif(2,6,"guitare",18,"10");

?>

</!DOCTYPE html>

<html>

<head>

<title>Music</title>

</head>

<body>

<?php

foreach ($ci as $unCi) {

$unCi->Afficher();

}

foreach ($cc as $unCc) {

$unCc->Afficher();

}

?>

</body>

</html>

**Ce code php affiche la page suivante :**

