**Comment transformer le programme pour en faire :**

* une classe impôt
* un programme de test pour le calcul d'une instance d'impôt particulière

**Quelles seraient les propriétés de la classe ? Quelles seraient les méthodes ?**

Quelles données sont fournies en entrée ? et sont particulière au calcul de chaque impôt ?

* la **structure familiale** : couple (marié ou pacsé) ou parent isolé (célibataire, séparé, divorcé ou veuf)
* le **nombre d'enfants à charge**
* les **revenus de l'année antérieure**

Quelles variables sont calculées ?

* le **revenu imposable**
* le **nombre de parts**
* le **quotient familial**
* le **montant de l'impôt**

Le principe premier d'une classe c'est sa **réutilisabilité** ==> que faire avec les tableaux dont les valeurs vont évoluer dans le temps ?

* **dissocier** les valeurs des tableaux, quels que soient les revenus du foyer fiscal
* des valeurs utiles au calcul de cette instance d'impôt particulier

Ce qui conduit à connaître pour la classe :

* la **tranche** à laquelle est soumise le foyer fiscal
* le **taux d'imposition**
* le **montant de l'abattement** auquel le foyer fiscal peut prétendre pour chacune de ses parts

**Résumons-nous**

**Propriétés** privées de la classe impôt

* **structure familiale** : booléen
* **nombre d'enfants à charge** : entier
* **revenus année antérieure** : double
* **tranche** : décimal
* **taux d'imposition** : décimal
* **montant abattement** : décimal
* **nbParts** : decimal car utile pour les différents calcul d'abattement et recherche de tranche, taux, ..

**Méthodes publiques** de la classe impôt

* le **nombre de parts**
* le **montant de l'impôt**

Remarque le constructeur sera un constructeur générique permettant la saisie de toutes les valeurs des propriétés à l'extérieur de la classe.

using System; using System.Collections.Generic; using System.Linq; using System.Text;

**namespace libImpot**

{

public class **Impot**

{

// propriétés privées de la classe

private bool couple;

private decimal nbEnfants ;

private decimal nbParts;

private decimal salairePercu;

private decimal plafond;

private decimal coefficient;

private decimal abattement;

// constructeur générique de la classe

public Impot() {}

// accesseurs de la classe

public bool **Couple**

{

get { return couple; } set { couple = value; }

}

public decimal **NbEnfants**

{

get { return nbEnfants; } set { nbEnfants = value; }

}

public decimal **NbParts**

{

get { return nbParts; } }

}

public decimal **SalairePercu**

{

get { return salairePercu; } set { salairePercu = value; }

}

public decimal **Plafond**

{

get { return plafond; } set { plafond = value; }

}

public decimal **Coefficient**

{

get { return coefficient; } set { coefficient = value; }

}

public decimal **Abattement**

{

get { return abattement; } set { abattement = value; }

}

***// Méthode publique de la classe permettant de calculer le nombre de parts***

**public void CalculerNbParts(bool Couple, decimal NbEnfants)**

{

decimal parts;

if (couple == true) // couple avec ou sans enfants

{

parts = 2 + (nbEnfants / 2);

}

else if (nbEnfants > 0)

{

parts = 1 + (nbEnfants / 2) + 0.5M; // personne isolée avec enfants

}

else parts = 1; // personne isolée sans enfant

if

(nbEnfants >= 3) parts = parts + 0.5M; // à partir de 3 enfants

**this.nbParts = parts;**

}

***// Calcul de l'impôt et affichage des résultats***

**public void CalculerImpot()**

{

Console.WriteLine("En couple : {0}", this.couple);

Console.WriteLine("Nombre d'enfants : {0}", this.nbEnfants);

Console.WriteLine("Nombre de parts : {0}", this.nbParts);

Console.WriteLine("--------------------------------------------");

***// Salaire imposable et quotient familial***

decimal salaireImposable = this.salairePercu \* 90 / 100;

decimal quotientFamilial = salaireImposable / this.nbParts;

Console.WriteLine("Salaire imposable : {0}", salaireImposable);

Console.WriteLine("Quotient familial : {0}", quotientFamilial);

Console.WriteLine("Tranche d'imposition : {0}", this.plafond);

Console.WriteLine("Taux d'imposition : {0}", this.coefficient);

Console.WriteLine("Abattement pour chaque part : {0}", this.abattement);

Console.WriteLine("--------------------------------------------");

***// Calcul de l'impôt***

decimal impot = Math.Floor((salaireImposable \* this.coefficient) - (this.abattement \* this.nbParts));

Console.WriteLine("Impôt à payer au titre de 2012 : {0}", impot);

}

}

}

}

**Le programme en mode console qui exploite la classe IMPOT**

using System; using System.Collections.Generic; using System.Linq; using System.Text;

**using libImpot;**

namespace IRPP2013

{

class Program

{

static void Main(string[] args)

{

*// Déclaration des constantes et des variables*

decimal coefAbattement = 0.90M;

decimal[] plafond = { 5964.00M, 11897.00M, 26421.00M, 70831.00M, 150001, 00M };

decimal[] taux = { 0M, 0.055M, 0.14M, 0.30M, 0.41M, 0.45M };

decimal[] abattement = { 0M, 327.97M, 1339.13M, 5566.33M, 13357.63M, 19357.63M };

bool couple = false;

decimal nbEnfants = 0.0M, nbParts = 0.0M;

decimal salairePercu = 0, revenuImposable = 0, quotientFamilial = 0;

string reponse = "";

bool OK = false;

*// Instanciation d'un objet de la classe*

Impot IRPPTest = new Impot();

*// Saisies des données pour un calcul d'impôt*

*// -----------------------------------------------------------*

reponse = null; // Couple ou célibataire

while (!OK)

{

Console.Write("Le foyer fiscal est-il composé d'un couple : oui ou non ? : ");

reponse = Console.ReadLine().ToLower(); // converti la chaine en caratères minuscules

if (reponse != "oui" && reponse != "non")

Console.Error.WriteLine("Réponse incorrecte. Recommencez");

else OK = true;

}

couple = reponse == "oui";

OK = false; // Saisie du nombre d'enfants

while (!OK)

{

Console.Write("Nombre d'enfants à charge : ");

reponse = Console.ReadLine();

try

{

nbEnfants = int.Parse(reponse);

if (nbEnfants >= 0) OK = true;

else Console.WriteLine("Réponse incorrecte. Recommencez");

}

catch (Exception)

{ Console.WriteLine("Réponse incorrecte. Recommencez"); }

}

OK = false; // Saisie du salaire

while (!OK)

{

Console.Write("Salaire perçu en 2012 : ");

reponse = Console.ReadLine();

try

{

salairePercu = decimal.Parse(reponse);

if (salairePercu >= 0) OK = true;

else Console.Error.WriteLine("Réponse incorrecte. Recommencez");

}

catch (Exception)

{ Console.Error.WriteLine("Réponse incorrecte. Recommencez"); }

}

// Affectation des valeurs des propriétés privées de l'objet

**IRPPTest.Couple = couple;**

**IRPPTest.NbEnfants = nbEnfants;**

**IRPPTest.CalculerNbParts(couple, nbEnfants);**

**IRPPTest.SalairePercu = salairePercu;**

// Calcul du revenu imposable

//-----------------------------------------------------------------------------

revenuImposable = (salairePercu \* coefAbattement);

// Calcul du quotient familial

//-----------------------------------------------------------------------------

**nbParts = IRPPTest.NbParts;**

quotientFamilial = revenuImposable / nbParts;

// recherche de la tranche d'impôts correspondant au quotient familial

//-----------------------------------------------------------------------------

int i = 0;

while ((quotientFamilial > plafond[i]) & (i<5))

{

i++;

}

// Affectation des valeurs des propriétés privées

**IRPPTest.Plafond = plafond[i];**

**IRPPTest.Coefficient = taux[i];**

**IRPPTest.Abattement = abattement[i];**

Console.WriteLine();

Console.WriteLine("Calcul de l'impôt");

Console.WriteLine("-------------------------------------------");

**IRPPTest.CalculerImpot();**

Console.WriteLine();

}

}

}