COMPTE RENDU TP 4

1) RENDRE LE CODE DU PROGRAMME PRINCIPAL. QUELLE EST LA VALEUR AFFICHEE SUR LA CONSOLE ?

```
namespace TP._04;

public class Program
{
    static void Main()
    {
        Test t1 = new Test();
        Test t2 = new Test();
        Console.WriteLine(Test.Combien());
    }
}
```

La console affiche un résultat de 2, ce qui reflète le compte des instances créées de la classe Test.

2) RENDRE LE CODE DE LA PROPRIETE

```
private int _val;

17 usages  More...
public int Val
{
    get => _val;
    set => _val = value;
}
```

```
public Test()
{
    ++_compteur;
    Val = 0;
}
```

Voici les ajustements réalisés au sein du constructeur :

3) QUEL EST LE NOMBRE D'INSTANCES EN MEMOIRE ? POUVEZ-VOUS EXPLIQUER CE NOMBRE ?

```
Console.WriteLine(Test.Combien() + " | " +t1.Val + " | " +t2.Val + " | " +t.Val + " | ");
```

```
2 | 10 | 0 | 10 |
```

Ce phénomène s'explique par le fait que t et t1 font référence au même objet en mémoire, impliquant qu'ils partagent les mêmes données plutôt que d'en posséder des copies distinctes.

4) QUELLES SONT LES VALEURS DE T1.VAL, T2.VAL ET T.VAL ? POUVEZ VOUS EXPLIQUER LA VALEUR DE T.VAL ?

L'explication repose sur le fait que t et t1 partagent la même adresse mémoire, signifiant que t ne duplique pas les informations de t1 mais y accède directement.

5) RENDRE LE CODE DU CONSTRUCTEUR PAR COPIE

```
1 usage
public Test(Test elem)
{
    ++_compteur;
    Val = elem.Val;

}
```

6) QUELLES SONT LES VALEURS DE CES PROPRIETES ? POUVEZ-VOUS EXPLIQUER CES DERNIERES ? COMBIEN D'INSTANCES DE TEST SONT ALORS PRESENTES EN MEMOIRE ?

Code utilisé :

```
static void Main()
{

    Test t1 = new Test();
    Test t2 = new Test();
    t1.Val = 10;
    Test t = new Test(t1);

    Test t3 = new Test(t1);
    Console.WriteLine($"la valeur contenue dans t3 : {t3.Val}");

    t1.Val = -1;
    Console.WriteLine("nombre d'instances : " + Test.Combien() + " | t1 : " +t1.Val + " |  t2 : " +t2.Val + " | t3 : " +t3.Val + " | t : " + t.Val);
```

Résultat : `nombre d'instances : 4 | t1 : -1 |  t2 : 0 | t3 : 10 | t : 10`

Les modifications effectuées ne modifient pas les valeurs de t et t3 lorsque t1 est modifié, grâce à l'implémentation du nouveau constructeur.

Le total s'élève à quatre instances en mémoire.

7) QUELLES SONT LES VALEURS AFFICHEES ? POUVEZ-VOUS EXPLIQUER POURQUOI ? D'APRES-VOUS L'OPERATION == COMPARE-T-IL LES VALEURS OU LES REFERENCES ?

```
la valeur contenue dans t3 : 10
nombre d'instances : 4 | t1 : -1 |  t2 : 0 | t3 : 10 | t : 10
nombre d'instances : 4 | t1 : 0 |  t2 : 0 | t3 : 0 | t : 0
False
False
False
```

Les trois comparaisons retournent faux, indiquant que t1 diffère de t2 et t3 malgré des valeurs identiques, car l'opérateur == évalue les références plutôt que les valeurs elles-mêmes.

8) JOINDRE LE CODE DE EQUALS A VOTRE COMPTE-RENDU

```
#region equals

1 usage
protected bool Equals(Test other)
{
    return _val == other._val;
}

public override bool Equals(object? obj)
{
    if (ReferenceEquals(null, obj)) return false;
    if (ReferenceEquals(this, obj)) return true;
    if (obj.GetType() != this.GetType()) return false;
    return Equals((Test)obj);
}

public override int GetHashCode()
{
    return _val;
}

public static bool operator ==(Test? left, Test? right)
{
    return Equals(left, right);
}

public static bool operator !=(Test? left, Test? right)
{
    return !Equals(left, right);
}

#endregion
```

9) JOINDRE LE CODE DE VOTRE TEST A VOTRE COMPTE-RENDU

```
Console.WriteLine();
Console.WriteLine("Comparaison par valeur avec la méthode Equals : ");
Console.WriteLine(t1.Equals(t2));
Console.WriteLine(t1.Equals(t3));
Console.WriteLine(t1.Equals(t));
```

10) QUELLE EST LA DIFFERENCE ENTRE T1==T3 ET T1.EQUALS(T3) ?

L'opérateur == évalue l'égalité des références, tandis que la méthode Equals compare les valeurs spécifiques des attributs entre les instances.

11) QUEL EST, D'APRES VOUS, LE ROLE DU DESTRUCTEUR ?

Le destructeur est destiné à libérer une instance de la classe de la mémoire, réduisant également le compteur de la classe par un. Il sert également à signaler la suppression de l'instance.

12) QUEL EST LE NOMBRE D'OCCURRENCES AVANT ET APRES L'ATTENTE ? COMMENT EXPLIQUEZ-VOUS LA DIFFERENCE ?

```
Nombre D'instances 438527
Nombre D'instances 118381
```

L'ajout de plusieurs instances et un délai d'attente permettent d'observer l'effet du destructeur, libérant de la mémoire en détruisant des instances de manière programmée.