Université de Skikda
Faculté de science
Département d'informatique

SOLUTION TP N04

1-.

```
public class Nombre {
      private double valeur;

      public Nombre(double valeur) {
            this.valeur = valeur;   }

      public double getValeur() {
            return valeur;    }

      public void setValeur(double valeur) {
            this.valeur = valeur;   }
      public void afficher()  {
      System.out.println("la valeur de ce nombre est : " + this.valeur);     }
      public void doubler(double n) {
            n= n*2;     }   }
```

2- la méthode setValeur() est surcharger comme suit :

```
public void setValeur (Nombre n)    {
      this.valeur = n.valeur; }
```

3- la méthode doubler() est surcharger comme suit :

```
public void doubler (Nombre n){
            n.valeur = n.valeur * 2 ;     }
```

4-

```
b = 4;
```

5-

```
Nombre n1 = new Nombre(0);
Nombre n2 = new Nombre(4);
```

6-

```
n1.setValeur(b);
```

7-

```
n1.setValeur(n2);
```

8-

```
n1.doubler(n2);
n2.afficher();
```

on peut également utiliser l'objet n2 lui même (n2.doubler(n2);)

9-

```
n1.doubler(b);
System.out.println("la valeur de b est : " + b );
```

On remarque que La valeur de b n'est pas changé (n'est pas doublé) elle reste la même (4)

Car le passage de paramètre dans le cas d'un paramètre de type primitif s'effectué par valeur. En d'autre terme toutes les modifications au sein de la méthode sont effectuées que sur une copie de ce paramètre dont la valeur réel de ce paramètre en dehors de cette méthode n'est pas touchée.

10-

```
public void doublerNombre(Nombre n)
      {     Nombre p = new Nombre (n.valeur);
            p.valeur = p.valeur * 2;
            n = p;      }
```

11-

```
   n2.doublerNombre(n1); // on peut utiliser n1 lui-même à la place de n2
   n1.afficher();
```

On remarque que La valeur de n1 n'est pas changé (n'est pas doublé) elle reste la même (4)

Car le passage de paramètre dans le cas d'un paramètre de type class (objet) s'effectué par référence.

Donc toutes modifications effectuées sur l'objet pointé par cette référence au sein de la méthode sont prises en compte (c'est le cas de la question 8 où ont a donné n2 comme paramètre et on a doublé la valeur de l'objet pointé par cette référence)

Cependant toute modification au sein de la méthode apportée sur la référence elle-même (n = p) n'est pas prise en compte en dehors de cette méthode. En d'autre terme la valeur de la référence est passée par valeur.

Ça veut dire que la référence n1 pointe toujours sur l'objet créer dans la question 5 et elle ne pointe pas sur l'objet crée au sein de la méthode (l'objet pointé par p).

à la fin de ce tp on obtient les deux classes suivantes :

```
public class Nombre {
      private double valeur;

      public Nombre(double valeur) {
            this.valeur = valeur;   }

      public double getValeur() {
            return valeur;    }

      public void setValeur(double valeur) {
            this.valeur = valeur;   }
      public void afficher()
      {System.out.println("la valeur de ce nombre est : " + this.valeur);    }
      public void doubler(double n)
      {     n= n*2;     }
      public void setValeur (Nombre n)
      {     this.valeur = n.valeur; }
      public void doubler (Nombre n)
      {     n.valeur = n.valeur * 2 ;     }
      public void doublerNombre(Nombre n)
      {     Nombre p = new Nombre (n.valeur);
            p.valeur = p.valeur * 2;
            n = p;}  }

public class ProgNombre {

            public static void main(String[] args) {
            double b = 4;
            Nombre n1 = new Nombre(0);
            Nombre n2 = new Nombre(4);
            n1.setValeur(b);
            n1.setValeur(n2);
            n1.doubler(n2);
            n2.afficher(); // la valeur est 8
            n1.doubler(b);
            System.out.println("la valeur de b est : " + b );// la valeur rete 4
            n2.doublerNombre(n1);
            n1.afficher(); //la valeur reste 4 }
}
```