Nom et prénom : **……**

Contrôle continu (2h00min)

Exercice 1 :

On considère l'interface suivante :

```
Public interface InetrOperation {
    public double unaire(float x) ;
    public int binaire(int x, int y) ; }
```

Indiquer pour chaque classe citée ci-dessous, si elle implémente correctement l'interface `InetrOperation`, en expliquant le cas échéant la (ou les) source(s) d'erreur(s).

```
class ImplInter1 implements InetrOperation {
    private String str ;
    public double unaire(float x) {return x*2 ;}
    public int binaire(int x, int y) {return x + y ;}
    public void affectation(String str) {this.str = str;} }
```

Votre réponse :

……
……
……
……

```
class ImplInter2 implements InetrOperation {
    private String str ;
    public double unaire(float x) {return x/2 ;}
    public double binaire(int x, int y) {return ((double)x)/y ;}
    public void affectation(String str) {} }
```

Votre réponse :

……
……
……
……

```
class ImplInter3 implements InetrOperation {
    private String str ;
    public double unaire(float x) {return Math.abs(x) ;}
    public int binaire(int y, int x) {return x/y ;}
    public void affectation(String str) {this.str = str;}
    public String getS() { return str ;} }
```

Votre réponse :

……
……
……
……

```
class ImplInter4 implements InetrOperation {
    public double unaire(int x, int y) {return Math.max(x, y) ;}
    public int binaire(int y, int x) {return x + y ;}
    public void affectation(String str) {System.out.println(str);} }
```

Votre réponse :

……
……
……
……

Nom et prénom : **……**

Exercice 2 :

On considère le programme suivant. Complétez le tableau ci-dessous, en justifiant votre réponse, par l'affichage produit par chaque instruction. Le cas échéant indiquez la (ou les) source(s) d'erreur(s) en mettant X dans la case correspondante (erreur au niveau de la compilation (**Comp**) ou au moment d'exécution (**Exéc**)).

```
class A {}
class B extends A {}        class C extends B {}        class D extends C {}
class W { void affiche(D d) { System.out.println("W.D");} }// fin W
class X extends W { void affiche(A a) { System.out.println("X.A"); }
                    void affiche(B b) { System.out.println("X.B"); } }// fin X
class Y extends X { void affiche(B b) { System.out.println("Y.B"); } }// fin Y
class Z extends Y { void affiche(C c) { System.out.println("Z.C"); } }// fin Z
public class TestHeritage {
     public static void main(String[] args) {
                Z z = new Z();     C c = new C();     D d = new D();
```

Instruction	Erreur		Donnez l'affichage et justifiez votre réponse ou indiquez la (les) source(s) d'erreur(s)
	Comp	Exéc	
((Y)z).affiche(c);			
((W)z).affiche(c);			
z.affiche((B)c);			
z.affiche((D)c);			
z.affiche((C)d);			
z.affiche(c);			
((Y)z).affiche((C)d);			
z.affiche(d);			
((X)z).affiche(d);			

```
} }// fin TestHeritage
```

Nom et prénom : **……**

Exercice 3 :

1. Ecrivez une classe nommée **Ouvrage**. Cette classe dispose des membres suivants :
   - id de type int (visibilité : protected).
   - titre de type String (visibilité : protected).
   - auteur de type String (visibilité : protected).
   - nbrePage de type int (visibilité : public).
   - Deux constructeurs prenant les arguments (id, auteur, titre, nbrePage) et (id, auteur, titre).
   - La méthode toString() qui retourne les informations d'un ouvrage.
2. Ecrivez une classe **Livre** qui hérite de la classe **Ouvrage**. Cette classe a comme membres :
   - qteStock de type int (visibilité : protected).
   - Un constructeur avec les arguments suivants (id, auteur, titre, qteStock).
   - La méthode toString() qui retourne les informations d'un livre.
3. Ecrivez une nouvelle classe **PFE** héritant de la classe **Ouvrage**. Cette classe a comme membres :
   - encadrant, filière de type String (visibilité : public).
   - Un constructeur avec les arguments suivants (id, auteur, titre, nbrePage, encadrant, filière).
   - La méthode toString() qui retourne les informations d'un PFE.
4. Créez une classe **Biblio**. Cette classe aura comme membres :
   - **tabOuvrage** un tableau de type **Ouvrage**, **taille** du tableau de type int (visibilité : public).
   - **nbreOuvrage** de type int statique qui compte le nombre le nombre d'ouvrages (visibilité : private).
   - Constructeur permet de créer le tableau.
   - La méthode **ajoutOuvrage()** permettant d'ajouter un ouvrage, livre ou pfe au tableau **tabOuvrage**.
   - La méthode **afficheOuvrage()** permet d'afficher tous les ouvrages du tableau.
5. Créez une application pour tester votre programme :
   - Créez un tableau de taille 5.
   - Créez des objets de type ouvrage, livre et PFE.
   - Ajoutez ces objets au tableau.
   - Affichez le contenu de votre tableau.

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Nom et prénom : ……