Héritage

Exercice 1 :

1. Ce code compile-t-il ? Si la réponse est non, expliquer les erreurs. Si la réponse est oui, qu'affiche-t-il ?

```java
public class Livre{
        private String titre;
        private int nbEmprunt;
        private static int nbEmpruntTotal;
        private boolean disponible = true;

        public Livre(String s){
                this.titre = s;
        }

        public emprunt(){
                disponible = false;
        }

        public static void main(String[] args){
                Livre a = new Livre();
                Livre b = Livre("Tintin et le vaccin interdit");

                a.emprunt();
                Livre.nbEmprunt++;
                Livre.nbEmpruntTotal++;

                b.emprunt();
                Livre.nbEmprunt++;
                Livre.nbEmpruntTotal++;

                System.out.println(a.nbEmpruntTotal);
        }
}
```

2. Ce code compile-t-il ? Si la réponse est non, expliquer les erreurs. Si la réponse est oui, qu'affiche-t-il ?

```java
public class Livre{
        protected String titre;
        protected int dateParution;

        public Livre(String s){
                this.titre = s;
        }

        public void affiche(){
                System.out.println("Je suis un livre de " + dateParution
                + " ! Mon titre est " + this.titre + ".");
```

```
        }
}
```

```
public class BD extends Livre{

        public BD(String s){
                super(s);
        }

        public void affiche(){
                System.out.println("Je suis une BD de " + dateParution
                + " ! Mon titre est " + this.titre + ".");
        }
}
```

```
public class Main{
        public static void main(String[] args){
                Livre a = new Livre("Alice au pays du Covid");
                BD b = new BD("Tintin et le vaccin interdit");

                a.affiche();
                b.affiche();
        }
}
```

Correction Exercice 1 : Pour la première question, le code ne compile pas.

```
public class Livre{
        ...

        public emprunt(){ //manque le type de retour
                disponible = false;
        }

        public static void main(String[] args){
                Livre a = new Livre(); // Le constructeur par défaut n'est pas généré..
                Livre b = Livre("Tintin et le vaccin interdit"); //Il manque le new

                a.emprunt();
                Livre.nbEmprunt++; //nbEmprunt n'est pas une variable static
                Livre.nbEmpruntTotal++;

                b.emprunt();
                Livre.nbEmprunt++; //nbEmprunt n'est pas une variable static
                Livre.nbEmpruntTotal++;

                System.out.println(a.nbEmpruntTotal);
        }
}
```

Pour la deuxième question, le code compile. Il affiche :
"Je suis un livre de 0 ! Mon titre est Alice au pays du Covid.
Je suis une BD de 0 ! Mon titre est Tintin et le vaccin interdit."

Exercice 2 : Où sont les erreurs dans ce code ?

```java
public class Test {

        public static void main(String[] args) {
                final Livre l = new Livre();
                l.setTitre("Mobidic");
        }
}
```

```java
public final class Livre {
        private String titre;
        private String auteur;
        final private int cout;

        public Livre() {
                this("","");
        }

        public Livre(String p_titre, String p_auteur) {
                titre  = p_titre ;
                auteur = p_auteur ;
                cout = 10;

                final int i;
                i = 1;

                final int i2 = 2;
                i2 = 5;
        }

        public void setTitre(String s) {
                titre = s;
        }

        public final void afficher() {
                System.out.println("Titre : " + titre + "\n Auteur : " + auteur );
        }
}
```

```java
public class BD extends Livre{
        private String dessinateur;

        public BD() {
                super();
                dessinateur = "";
```

```
7           }
8
9           public BD(String p_titre, String p_auteur, String p_dessinateur) {
10                  super(p_titre, p_auteur);
11                  dessinateur = dessinateur;
12          }
13
14          public void afficher() {
15                  super.afficher();
16                  System.out.println("\n Dessinateur : " + dessinateur );
17          }
18
19          public void afficher(String s) {
20                  this.afficher();
21                  System.out.println(s);
22          }
23
24
25  }
```

```
1   public class Test {
2
3           public static void main(String[] args) {
4                   final Livre l = new Livre();
5                   l.setTitre("Mobidic"); //Pas d'erreur ici
6                   // Livre est un type adresse, modifier l'objet est permis
7                   // ce qui n'est pas permis c'est de changer l'adresse.
8           }
9   }
```

Correction Exercice 2 :

```
1   public final class Livre {
2           private String titre;
3           private String auteur;
4           final private int cout;
5
6           public Livre() {
7                   this("","");
8           }
9
10          public Livre(String p_titre, String p_auteur) {
11                  titre  = p_titre ;
12                  auteur = p_auteur ;
13                  cout = 10; //Pas d'erreur ici
14                  //cout est initialisé implicitement à 0
15                  //mais final impose seulement de ne
16                  //l'expliciter explicitement qu'une fois.
```

```
17
18                  final int i;
19                  i = 1; // Ok
20
21                  final int i2 = 2;
22                  i2 = 5; // erreur i2 est final
23          }
24
25          public void setTitre(String s) {
26                  titre = s;
27          }
28
29          public final void afficher() {
30                  System.out.println("Titre : " + titre + "\n Auteur : " + auteur );
31          }
32  }
```

```
1   public class BD extends Livre{ //erreur, la classe Livre est final
2           private String dessinateur;
3
4           public BD() {
5                   super();
6                   dessinateur = "";
7           }
8
9           public BD(String p_titre, String p_auteur, String p_dessinateur) {
10                  super(p_titre, p_auteur);
11                  dessinateur = dessinateur;
12          }
13
14          public void afficher() { //erreur, afficher est final
15                  super.afficher();
16                  System.out.println("\n Dessinateur : " + dessinateur );
17          }
18
19          public void afficher(String s) { //pas d'erreur ici
20              //on ne fait pas une redéfinition mais une surcharge
21                  this.afficher();
22                  System.out.println(s);
23          }
24
25
26  }
```

Exercice 3 : On va continuer à utiliser la classe `Box` de la séance précédente. Dans une interface graphique, on peut noter qu'il y a des rectangles particuliers avec du texte, et qu'il y a des rectangles encore plus particuliers avec du texte et qui déclenche une action quand on clique dessus... Il semble que l'héritage soit bien utile ici !

1. Créez une classe `TextBox` qui n'est rien de plus qu'une `Box` à laquelle on ajoute du texte. Implémentez un constructeur qui prend en argument quatre **int** représentant les coordonnées des

deux coins du `TextBox` et une chaîne de caractères qui représente le texte.

```java
public class TextBox extends Box{
        private String text;

        public TextBox() {
                super();
                this.text = null;
        }

        public TextBox(int absHG, int ordHG, int absBD, int ordBD, String text) {
                super(absHG, ordHG, absBD, ordBD);
                this.text = text;
        }
}
```

2 Implémentez une méthode **public** `String toString()`.

```java
@Override
public String toString() {
        return super.toString() + "| Text : " + text;
}
```

3 Qu'avez-vous choisi pour la visibilité de vos variables d'instance de la classe `Box` (par exemple pour les coordonnées des coins) ?

4 Mettez les variables d'instance en `private` (sans rien changer d'autres). Si vous avez une erreur de compilation, quelle est-elle ?
Expliquez votre erreur et corrigez la.

Si les variables sont en private on ne peut pas utiliser leurs coordonnées dans une autre classe, même une classe fille. Il faut alors passer par le mot clé super ou un getter selon le context. Le plus possible, il faut garder tout ce qui peut être déclaré private comme tel.

5 Expliquez ci-dessous votre choix entre `private` et `protected`. Même si ce n'est pas la solution que vous privilégiez, on vous impose maintenant d'utiliser `private`. Modifiez votre code si besoin est.

```java
public class Box {
    private Pixel pHG;
    private Pixel pBD;
    private final int ID_BOX;
    public static int nbBox;

    // ...

    public Pixel getCoinHG() {
        return pHG;
    }

    public Pixel getCoinBD() {
```

```
14            return pBD;
15        }
16    }
```

6 Redéfinissez la méthode `equals` de la classe `Object`. Vérifiez que vous codez bien une redéfinition en utilisant l'annotation `@Override`. Vous pouvez tester votre méthode avec le code suivant.

```
1  Box b = new Box(0,10,10,0);
2  TextBox tb = new TextBox(0,10,10,0,"hello");
3  System.out.println(b.equals(tb) + " ?? " + tb.equals(b));
4  Box fb = new TextBox(0,10,10,0,"hello");
5  System.out.println(fb.equals(tb) + " ?? " + tb.equals(fb));
```

Que devrait être le résultat de l'exécution de ce code ? Testez le !

Voici le code de la méthode equals.

```
1  //Dans TextBox.java
2  @Override
3  public boolean equals(Object o) {
4          if(o instanceof TextBox){
5                  TextBox t = (TextBox) o;
6                  if(super.equals(t) && text.equals(t.text)) {
7                          return true;
8                  }
9          }
10         return false;
11 }
12
13 //Dans Box.java en utilisant des getters dans Pixel.java
14 //Je fais le choix de ne pas considerer id dans le test
15 @Override
16 public boolean equals(Object o) {
17         if(o instanceof Box){
18                 Box b = (Box) o;
19                 if((b.pBD.getAbs() == pBD.getAbs()) &&
20                    (b.pBD.getOrd() == pBD.getOrd()) &&
21                    (b.pHG.getAbs() == pHG.getAbs()) &&
22                    (b.pHG.getOrd() == pHG.getOrd())){
23                         return true;
24                 }
25         }
26         return false;
27 }
```

Le test doit retourner :

```
true ?? false
true ?? true
```

7 Implémentez une méthode `clone()` pour dupliquer un élément.

Voici un premier essai volontairement érroné.

```java
//Dans TextBox.java
public class TextBox extends Box  implements Cloneable{
    ...

        @Override
        public Textbox clone() throws CloneNotSupportedException{
                TextBox t = (TextBox) super.clone();
                return t;
        }
}

//Dans Box.java
public class Box implements Cloneable{
    ...
    @Override
        public Box clone() throws CloneNotSupportedException{
                Box b = (Box) super.clone();
                return b;
        }
}
```

8 On va placer ces lignes de code dans la méthode `main`.

```java
Box a = new Box(100, 200, 300,100);
Box b = (Box) a.clone();
System.out.print(a==b);
System.out.print("|"+a.equals(b));
Pixel p = b.getCoinHG();
p.setCoordonnees(0,200);
System.out.print(a);
```

Modifiez si besoin votre code pour que l'exécution affiche `false|true` (ce ne sont pas les mêmes `Box`, mais deux `Box` avec les mêmes coordonnées sont égales).
Le résultat de la ligne 7 est-il celui que vous attendiez ? Si non, corrigez votre code.

On remarque que quand on modifie la `Box b`, la `Box a` est également modifiée. En effet si `b` est une copie de `a` il partage les références vers les mêmes pixels. On a copié les références pas les pixels. Il fallait écrire cela à la question précédante pour vraiment copier les informations. L'utilisation de l'interface `Cloneable` et des exceptions comme `CloneNotSupportedException` seront vues plus loin dans le cours.

```java
//Dans TextBox.java
public class TextBox extends Box  implements Cloneable{
    ...

        @Override
        public TextBox clone() throws CloneNotSupportedException{
                TextBox t = (TextBox) super.clone();
                t.text = new String(text);
                return t;
```

```
        }
}

//Dans Box.java
public class Box implements Cloneable{
    ...
    @Override
    public Box clone() throws CloneNotSupportedException{
                Box b = (Box) super.clone();
                b.idBox = nbBox++;
                b.pBD = new Pixel(pBD.getAbs(),pBD.getOrd());
                b.pHG = new Pixel(pHG.getAbs(),pHG.getOrd());
                return b;
        }
}
```