Créer un ***Dynamic web Project***

Exercice 1

1. Créez et implémentez la classe `Adherent` avec les attributs suivants : nom, prénom, date de naissance, adresse mail, adresse postale.

   Pour représenter les dates vous pouvez utiliser la classe statique `LocalDate` du package `java.time`.

   * Ajouter 2 constructeurs à votre classe : 1 sans paramètres et un autre avec paramètres

2. Créez la méthode `toString()` (qui renvoie un objet de type `String`) générant un texte de présentation des informations concernant l'objet `Adherent`.
3. Ajoutez une méthode `setNom(...)` qui permet de changer le nom d'un objet de type `Adherent`.
4. Vérifiez votre solution dans le programme principal (la classe `GestionAdherents`) en instanciant dans la méthode `main(...)` une variable `lolo` de type `Adherent` et en affichant ses informations.
5. Créez une nouvelle variable `toto` de type `Adherent` construite avec exactement les mêmes paramètres que `lolo`. Comparez les deux variables avec l'opérateur `==`. Que constatez-vous ?
6. Modifiez votre programme pour faire en sorte que `toto` fasse référence à `lolo`. Vérifiez qu'une modification des attributs de `toto` se répercute bien sur `lolo`.
7. Créez une classe `Bibliotheque` qui aura comme attributs un nom, une adresse et un tableau d'adhérents inscrits gérés sous forme de liste. Pour déclarer une telle liste vous pouvez utiliser la classe `ArrayList` du package `java.util`.

   Ajouter à cette classe un constructeur, qui prend comme paramètre un nom et une adresse. Définissez la méthode `toString()` dans `Bibliotheque` avec un texte qui liste l'ensemble des adhérents de cette bibliothèque

8. Ajoutez une méthode `inscrire(...)` dans la classe `Bibliotheque` qui prend en paramètre un adhérent et l'ajoute aux adhérents inscrits à la Bibliothèque
9. Ajoutez une méthode `desinscrire(...)` qui supprime un adhérent passé en paramètre de la liste des adhérents inscrits.
10. Vérifiez le bon fonctionnement de votre application dans la classe principale en créant une bibliothèque `BiblioKids` et en y inscrivant quatre adhérents (dont `toto` et `lolo` définis précédemment). Désinscrivez ensuite `toto`. Que constatez-vous ?

Exercice 2

1. Une `Livre` est définie par un identifiant, un titre et un `auteur` (une classe avec des attributs id_auteur, nom et prénom). Créez ces 2 classes avec des constructeurs adéquats ainsi que les accesseurs pour chacun des attributs.
2. Un `exemplaire` de livre est définie par un éditeur et par une année d'édition.

   Implémentez la classe et ajoutez des méthodes accesseurs pour chacun des attributs.

3. Ajoutez à la classe `Adherents` un attribut correspondant à la liste de ses exemplaires empruntés et une méthode `emprunter(...)` qui prend en paramètre un exemplaire et une date de prêt, (crée un objet de type `exemplaire` et l'ajouter à la liste des emprunts de l'adhérent).
4. Vérifiez que votre programme fonctionne bien en l'exécutant depuis la classe principale.
5. Construisez le diagramme de classes correspondant au programme *Java* que vous avez écrit. Vous pouvez le faire sur papier ou en utilisant un logiciel de modélisation.

Exercice 3

- Observez que la classe `Adherent` a un constructeur avec 5 paramètres. Le nombre de paramètres risque de croître car beaucoup d'autres attributs sont susceptibles d'être ajoutés à cette classe

- Observez aussi que lorsqu'on construit une instance d'`Adherent` il est facile de se tromper dans l'ordre des paramètres du constructeur. Heureusement que l'IDE vous aide en vous suggérant cet ordre lorsque vous programmez...De plus, les valeurs de certains attributs peuvent être inconnues au moment de la construction de l'objet

Plusieurs solutions peuvent être envisagées :

1. Une solution est de définir plusieurs constructeurs avec différents paramètres et de les faire collaborer
2. Créez une nouvelle classe `Adherent1` (en copiant les attributs de la classe `Adherent` de l'exercice 2) et modifiez-la afin de pouvoir instancier les Adhérents de différentes manières : - en indiquant uniquement le nom et le prénom - en indiquant uniquement le nom, le prénom et la date de naissance - en indiquant uniquement le nom, le prénom et le mail Vérifiez votre programme dans la classe principale (avec votre main()).
3. Avec cette approche, est-il possible d'ajouter un constructeur supplémentaire afin d'indiquer uniquement le nom, le prénom et l'adresse ?
4. Une autre solution serait de suivre le modèle *Java Beans*, en proposant un seul constructeur minimal et en fournissant des méthodes modifieurs (*setters*) pour chaque attribut de la classe. Créez une telle classe appelée `AdherentJavaBeans`.

Aller plus loin :

On vous demande de développer une solution en combinant les bonnes idées des deux versions précédentes. Voici comment on voudrait pouvoir créer des adhérents dans la classe principale :

```
class Gestion Adherents {
        public static void main(String args[]) {
                Adherent lolo = new Adherent Builder()
                .ajouterNom("Dupont")
                .ajouterPrenom("Philippe")
                .ajouterDateNaissance(LocalDate.of(2002, Month.JANUARY, 12))
                .ajouterMail("dupont@gmail.fr")
                .ajouterAdresse("1, av. des Champs-Elysées, 75008 Paris")
                .build();
        }
}
```

1. Ajoutez donc la classe `AdherentBuilder` à votre application pour que l'instruction ci-dessus fonctionne.
2. la classe `AdherentBuilder` sert uniquement à instancier des objets `Adherent` de manière "organisée" et lisible. Le problème est qu'il est toujours possible d'instancier un objet de type `Adherent` sans utiliser le builder que vous avez écrit…

   C'est pour cela qu'il est possible d'améliorer la dernière solution en déclarant la classe `AdherentBuilder` comme classe interne statique de la classe `Adherent` et de rendre `private` le constructeur de la classe `Adherent`.