# TP Révision– Notions de classe et héritage

Le but du TP est de développer une classe pour décrire une personne. Les informations à gérer seront le nom, le prénom de la personne ainsi que son numéro de téléphone et son adresse électronique (une chaîne de caractères).

Pour ce faire, nous utiliserons deux classes déjà écrites : la classe ***Téléphone*** et la classe ***Individu***. Cette dernière permet de gérer le nom et le prénom d'un individu (voir diagramme joint et fichiers disponibles sur Moodle). Considérez que vous n'avez pas le droit de modifier ces 2 classes.

On prévoira dans la classe ***Personne*** les opérations ou méthodes suivantes :

- ✓ un **constructeur sans argument** qui initialisera tous les attributs avec des valeurs par défaut
- ✓ un **constructeur avec en argument le nom et le prénom** de la personne. Les autres informations seront initialisées par défaut
- ✓ un **constructeur avec en argument les 4 informations** décrivant une personne : nom, prénom, téléphone et adresse mail. Ces informations seront des chaînes de caractères.
- ✓ une méthode pour **afficher** les informations connues sur une personne. Il s'agit d'afficher sur la console.
- ✓ une méthode pour **saisir** au clavier les 4 informations décrivant une personne
- ✓ une méthode nommée ***information*** qui renverra sous la forme d'une chaîne de caractères toutes les informations connues sur la personne, présentées de la manière suivante (les retours à la ligne seront respectés) :

  *Nom Prénom*
  *numéro de téléphone*
  *adresse électronique*

## Questions

1. Compléter le diagramme UML donné à la fin du sujet pour montrer les relations entre les classes ***Téléphone***, ***Individu*** et ***Personne***, et mettre en évidence les attributs et opérations de la classe ***Personne***.

2. Coder la classe ***Personne***. Dans un premier temps, on ne vérifiera pas la validité d'une adresse électronique.

3. Ecrire une fonction ***main*** pour tester la classe ainsi écrite. Vous pourriez par exemple :

   - déclarer et créer 3 instances de la classe ***Personne***, chacune initialisée avec un constructeur différent (donc l'une avec les valeurs par défaut, l'autre avec un nom et un prénom explicite, et la troisième avec 4 informations explicites)

- afficher sur la console les 3 instances
- saisir au clavier les valeurs explicites de l'instance initialisée par défaut et afficher l'instance ainsi modifiée
- faire appel à la méthode ***information*** sur les 3 instances et afficher la chaîne renvoyée par cette méthode

**4.** Est-il possible dans la fonction ***main*** d'afficher seulement le nom et le prénom d'un objet de type ***Personne*** (sans modifier les classes développées) ? Si oui, tester cette possibilité.

**5.** Compléter la classe de manière à s'assurer que les adresses électroniques sont valides. Lors de la saisie, l'adresse sera redemandée en cas d'erreur. Dans le constructeur, c'est l'adresse par défaut qui sera affectée si celle donnée en paramètre est invalide.
On pourra considérer qu'une adresse est valide, si elle respecte, le format suivant :

a. des lettres, des chiffres, l'un des caractères '-' '_' '.'
b. le symbole '@'
c. des lettres, des chiffres, l'un des caractères '-' '_'
d. un point '.'
e. puis 2 ou 3 lettres

**6.** Tester à nouveau la classe ***Personne.***

**7. Révision du polymorphisme**

A. Le but de cette question est d'écrire une fonction ***main*** qui utilise les classes précédentes dans le but de réviser le concept du polymorphisme. Dans cette fonction :

- ✓ déclarer un tableau de 5 éléments qui pourra contenir aussi bien des instances de la classe ***Individu*** que des instances de la classe ***Personne***.
- ✓ pour chaque élément du tableau, demander à l'utilisateur s'il souhaite créer une instance d'***Individu*** ou de ***Personne***. Créer l'instance selon son souhait et effectuer la saisie au clavier des informations décrivant l'individu ou la personne.
- ✓ afficher ensuite les éléments du tableau pour vérifier le bon fonctionnement

B. Modifier le programme précédent de manière à interdire la présence d'homonymes dans le tableau. Si un individu ou une personne existe déjà dans le tableau avec le même nom et prénom, la saisie de celui-ci ou de celle-ci sera recommencée.

## Diagramme UML à compléter