

# Compte rendu TP - Génie logiciel

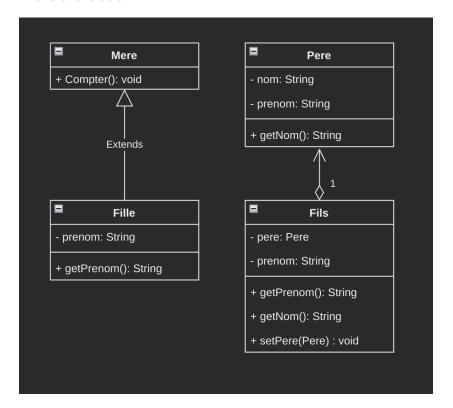
# TP 1: Design pattern et POO (Java)

#### Quelques précisions concernant le code :

- une utilisation des codes de chaque partie se trouve dans la classe "Main" de chaque package
- le code correspondant à chaque question est séparé dans chaque classe "Main" par des commentaires
- les deux questions bonus ont été implémentées directement dans les classes "Main"
- la logique de code a été réalisée en partie en collaboration avec Kévin KESTELOOT (notamment pour les questions bonus)

# A. <u>Première partie : Héritage ou composition</u>

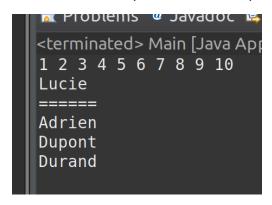
i. Diagramme de classe







iii. Retour de la classe de test correspondant à la question 3



- B. <u>Deuxième partie : Le design pattern Stratégie</u>
- ii. Quels sont les inconvénients à utiliser l'héritage pour définir le comportement de Canard ?

### Réponses:

- 2. Les changements de comportement au moment de l'exécution sont difficiles.
- 6. Les modifications peuvent affecter involontairement d'autres canards.
- vi. Retour de l'implémentation du design pattern

```
<terminated> Main (1) [Java Applic
  Je suis un colvert.
  Cancan !
  Je vole !
  =====
  Je suis un leurre.
  ...
  Je ne vole pas.
  =====
  Je suis un prototype.
  ...
  Je vole !
  Coin coin !
  Je vole par propulsion !
```



bonus. Retour du programme de choix dynamique des objets et options

```
Bonjour et bienvenue dans le programme de choix de canard ! *

Pour commencer, veuillez choisir votre type de canard :

[1] : Colvert
[2] : Prototype
[3] : Leurre
[4] : Mandarin

Votre choix (1, 2, 3 ou 4) :

2

Vous avez choisi le canard : prototype

Veuillez maintenant choisir le comportement de vol désiré :

[1] : Pas de vol
[2] : Vol classique
[3] : Vol par propulsion

Votre choix (1, 2 ou 3) :

3

Vous avez choisi le type de vol : vol par propulsion
```

```
Veuillez maintenant choisir le comportement de cancan désiré :

[1] : Muet
[2] : Cancan
[3] : Couinement

Votre choix (1, 2 ou 3) :

1

Vous avez choisi le type de cancan : muet

Voici le canard que vous avez choisi !
Je suis un prototype.
...
Je vole par propulsion !

Merci d'avoir utilisé notre programme !
```

#### Samuel-Charles



- C. <u>Troisième partie : Le design pattern Décorateur</u>
- ii. Retour de l'implémentation du design pattern lors du test de la commande de boissons

```
<terminated> Main (2) [Java Application] /root/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_19.0.1.v202211
Je suis un sumatra de moyenne taille _:_ 9.0 euros
Je suis un expresso de petite taille _:_ 5.0 euros
Je suis un expresso de grande taille _:_ 7.0 euros
Je suis un sumatra de moyenne taille avec du chocolat avec du chocolat avec de la chantilly _:_ 15.5 euros
```

### bonus. Retour du programme de choix dynamique des objets et options

```
terminated> Main (2) [Java Application] /root/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.linux.x86_64_19/
🧟 Bonjour et bienvenue dans notre café ! 🎅
 🕤 Pour commencer, veuillez choisir votre boisson :
 [1] : Expresso
[2] : Déca
[3] : Colombia
[4] : Sumatra
Votre choix (1, 2, 3 ou 4) :
Vous avez choisi la boisson : déca
🤏 Veuillez maintenant choisir la taille de votre boisson :
[1] : Petite
    : Moyenne
[3] : Grande
Votre choix (1, 2 ou 3) :
Vous avez choisi une boisson de grande taille.
 [1] : Lait
[2] : Chocolat
[3] : Caramel
[4] : Chantilly
[0] : Aucun
Votre choix d'ingrédient(s) :
Veuillez séparer les ingrédients par des espaces en précisant la quantité pour chaque ingrédient
Ex. '1:3 2:4' si vous souhaitez 1 supplément lait et 2 suppléments chantilly
Si vous ne souhaitez aucun ingrédient, tapez 0:0 :
```



## **DITTE-DESTRÉE**

Samuel-Charles



18 nov. 2022

2:3

Vous avez choisi les ingrédients : 2 supplément(s) caramel

Votre commande finale est donc : Je suis un déca de grande taille avec du caramel avec du caramel

Le montant de votre commande s'élève à : 15.0 €.

