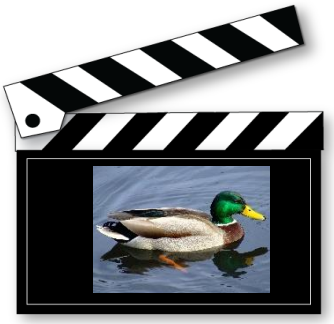
TP R2.01 : Extension du projet canard et Diagramme de séquence

… Finir le TP de la semaine dernière sur les canards …
… Avant de passer de commencer ce TP…

Séquence : retour vers les canards !



La semaine dernière vous avez implémenté le diagramme de classes du projet **canard** (que vous pouvez retrouver en annexe 1 et en faisant une rapide rétro-conception de votre code).

Le client de la mare aux canard souhaite désormais faire évoluer son jeu en proposant un nouveau type de canard (**PrototypeCanard**), ainsi qu'un nouveau comportement de vol (**PropulsionAReaction**)

1. Modifiez le diagramme de classes donné en annexe pour faire apparaître ces deux nouvelles classes sur ce modèle. Est-ce facile à modifier ?

   Afin de tester l'ajout de ces nouvelles classes (et de vérifier la non-régression sur les anciennes), votre collègue testeur vous demande d'implémenter le ***scénario de test*** suivant.

> ***Commencer par créer un colvert et demander au colvert de s'afficher, de nager, d'effectuer un cancan, puis un vol***
>
> ***Puis, créer un prototype de canard et lui demander de s'afficher et d'effecteur un vol***
>
> ***Modifier ensuite le comportement de vol ce prototype de canard qui doit pouvoir désormais voler avec une propulsion à réaction.***

Remarque : Le prototype de canard vient au monde sans aucun moyen de voler, ni aucun moyen de cancaner.

2. Implémentez les classes `PrototypeCanard` et `PropulsionAReaction` sur le modèle des classes implémentées sur votre projet.
   Faites en sorte que le code que vous ajoutez soit tester dans **CanardTest** et que la couverture de du code métier par les tests se maintienne à 100%

3. Implémentez le scénario de test précédent dans le package **`canard.application`** dans le **`main`** d'une classe que vous appellerez **`Client`**
   Exécutez ce **`main`** pour vérifier et valider sur la console le « bon » comportement du code implémenté qui doit être similaire à l'affichage suivant :

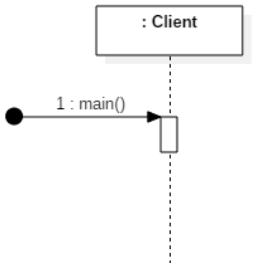
```
Je suis un vrai colvert
Tous les canards flottent, même les leurres!
Can-can
Je vole !
Je suis un prototype de canard
Je ne sais pas voler
Je vole avec un réacteur !
```

4. Comme le projet est maintenant fini, vous pouvez à la fois *commiter* et *pousser* (sur le *remote*) avec le message suivant :
   `ajout PrototypeCanard, PropulsionAReaction et Client`

   ⇨ cette fois-ci, essayez de faire les deux actions en une directement depuis Eclipse en appuyant sur le **bouton Commit and Push** 😉 …
   Vérifiez ensuite sur le *remote* que le code a bien été poussé correctement.

5. **Sur une feuille de papier :**

   Représentez sur un diagramme de séquences ***toutes*** les interactions (messages échangés entre les objets) lors de l'exécution de cette méthode `main`. Ce diagramme commencera de la manière suivante.
   A vous de le compléter avec les objets et messages appropriés.

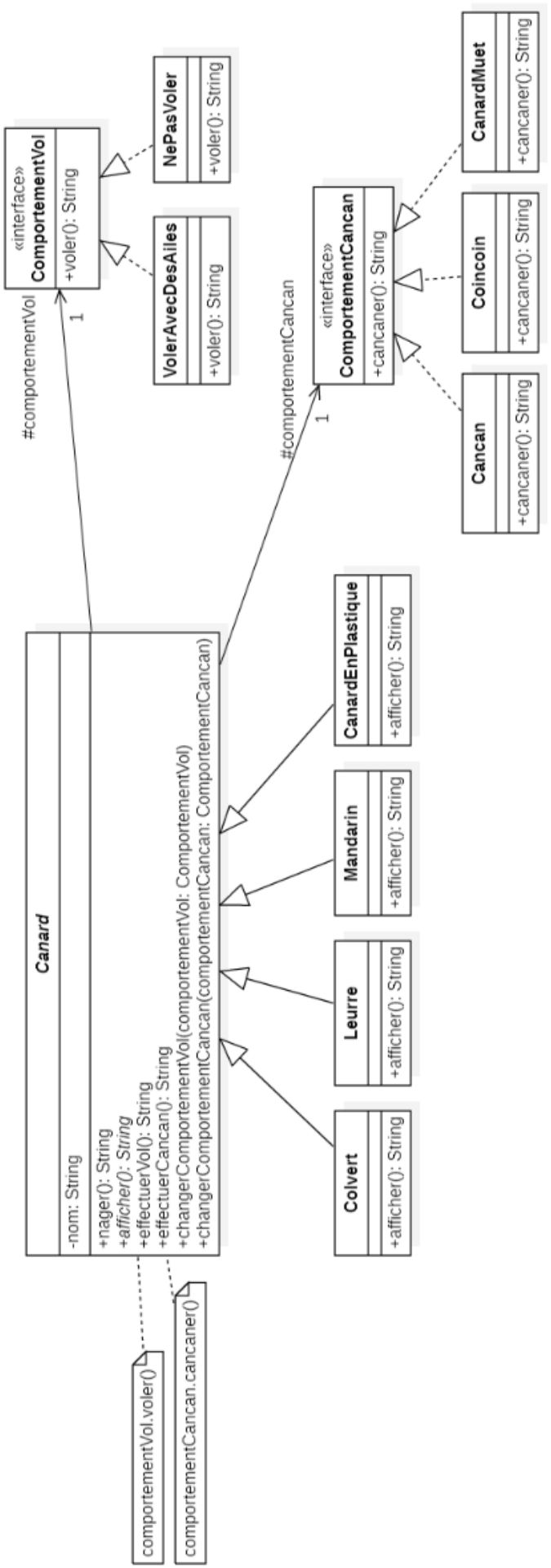


   Montrez votre diagramme de classes à votre enseignant de TP avant de continuer !!!

   … et suivez avec lui votre diagramme de séquences avec votre IDE.
   Par exemple, placez-vous dans sur la méthode **`effectuerVol`** dans la méthode main de la classe Client, faites un **CTRL+clic** … et voyez ce qui se passe 😉

S'il vous reste du temps sur cette séance, profitez-en pour finir des TPs précédents, notamment l'implémentation de l'exercice Rendez-vous au Far West du TP Implémenter des interfaces à partir d'un diagramme de classes simple dont l'énoncé vous est rappelé dans l'annexe 2.

Annexe 1 : Diagramme de classes : mare aux canards

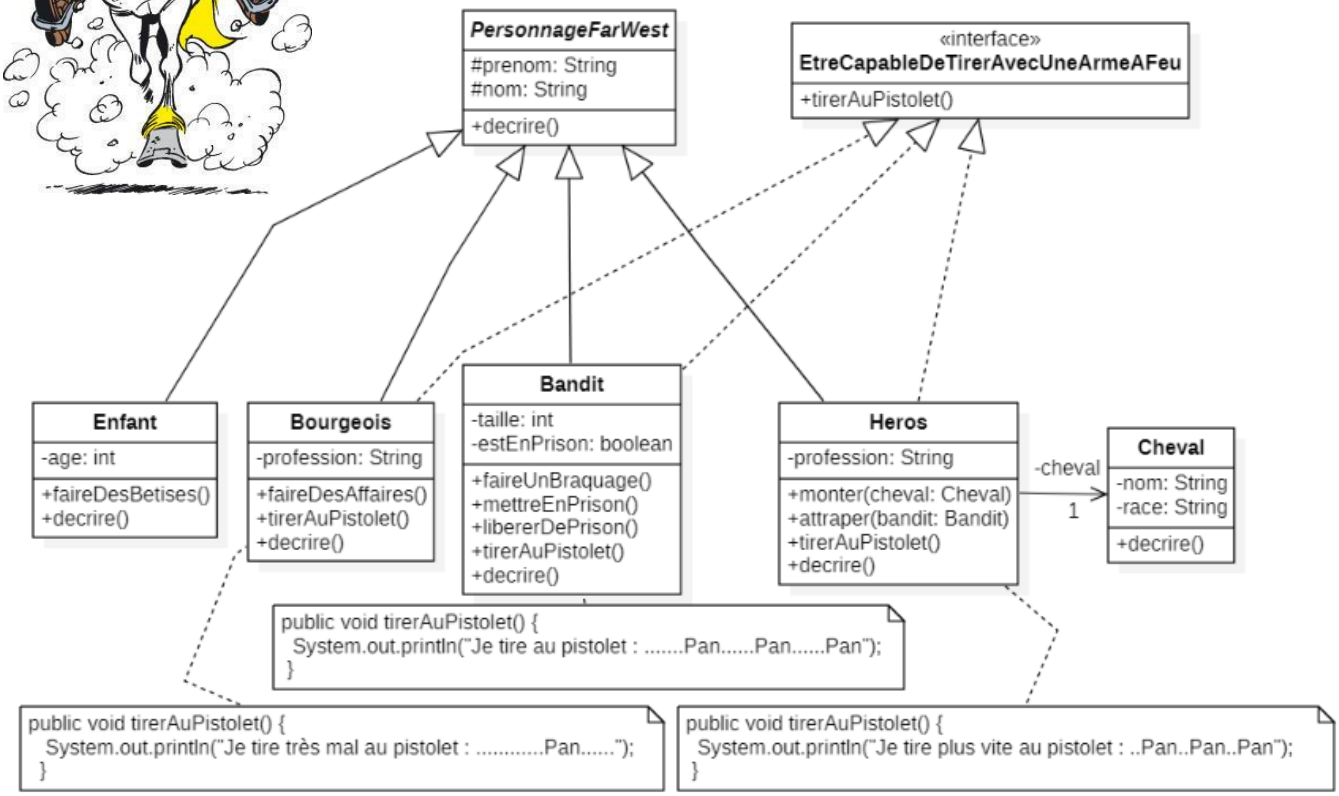


Annexe 2 : Rendez-vous au Far West …

❑ Rendez-vous dans votre projet **`farwest`** ou créez le, si vous n'en disposez pas encore d'un 😉

❑ Implémentez le diagramme de classes suivant

(vous vérifierez l'architecture avec une petite **rétro-conception** 😉)



Remarques :

→ Pour les méthode `faireDesBetises`, `faireDes Affaires` et `faireUnBraquage` , un simple `//TODO` suffira comme implémentaion.

→ Pour la méthode `décrire` le résultat attendu à la console donné ci-dessous devrait vous aider 😉

❑ Récupérerez la classe **`FarWestMain`** disponible sur le **gist** : https://unil.im/farwest (https://gist.github.com/iblasquez/5597413cb37a30a2e7e8129f2d6aedf1)

Faites-en sorte que ce code compile, puis exécutez-le pour vérifier et valider le comportement de votre application. Si votre implémentation est correcte, vous devriez obtenir sur la console un affichage similaire à :

```
--------------------------------------------------
Les personnages de la caravanne sont :
Lucky Luke! Je suis cow-boy et mon cheval est Jolly Jumper de race appaloosa
Joe Dalton! Je mesure 150 cm et je suis Libre
Averell Dalton! Je mesure 190 cm et je suis Libre
Zacharie Martins! Je suis inventeur
Phineas ! J'ai 10ans
--------------------------------------------------
Les personnages capable de tirer au pistolet sont :
Lucky Luke ! Je tire plus vite au pistolet : ..Pan..Pan..Pan
Joe Dalton ! Je tire au pistolet : .......Pan......Pan......Pan
Averell Dalton ! Je tire au pistolet : .......Pan......Pan......Pan
Zacharie Martins ! Je tire très mal au pistolet : ............Pan......
```