La pierre et le sabre

Suite et fin

Environnement GitHub (Rappel)

Mettre à jour votre dépôt local avec le contenu du dépôt distant.

- En ligne de commande :
  - Avec un explorateur de fichier se placer dans le répertoire du projet `'PierreSabre'` qui se trouve sous votre workspace (vous devez y trouver le fichier `.gitignore`), puis cliquer droit (sans rien sélectionner) et cliquer sur `'Git Bash Here'`.
  - Dans Git Bash (ouvert directement dans le répertoire du projet `'PierreSabre'`), tapez la commande `'git pull'`.

Historisation périodique de votre projet.

- Avec un explorateur de fichier retrouver sous votre projet `'PierreSabre'` (vous devez y trouver le fichier `.gitignore`) puis cliquer droit (sans rien sélectionner) et sélectionner `'Git Bash Here'`.

- Sous Git Bash : utiliser les commandes git :
  - `git add .`
  - `git commit -m <Intitulé des modifications>`
    exemple : `git commit -m "TP5 Fin Humain"`
  - `git push`

- Une fenêtre peut s'ouvrir (figure ci-contre) en vous demandant de vous identifier,
  Sélectionner `"Sign in with your browser"`,

- Une autre fenêtre s'ouvre (extrait ci-dessous), sélectionner "Authorize GitCredentialManager"

Java

On rappelle qu'on désire réaliser un programme Java permettant d'écrire facilement des histoires de Samouraïs dans lesquelles apparaissent des commerçants, des ronins et des yakuzas.

=> créer une classe **`HistoireTP5`**, dans le package **`histoire`**, qui contient une méthode **`main`**, et qui vous servira de support pour tester tous les objets et les méthodes que vous allez écrire. Toutes les autres classes seront placées sous le package **`personnage`**.

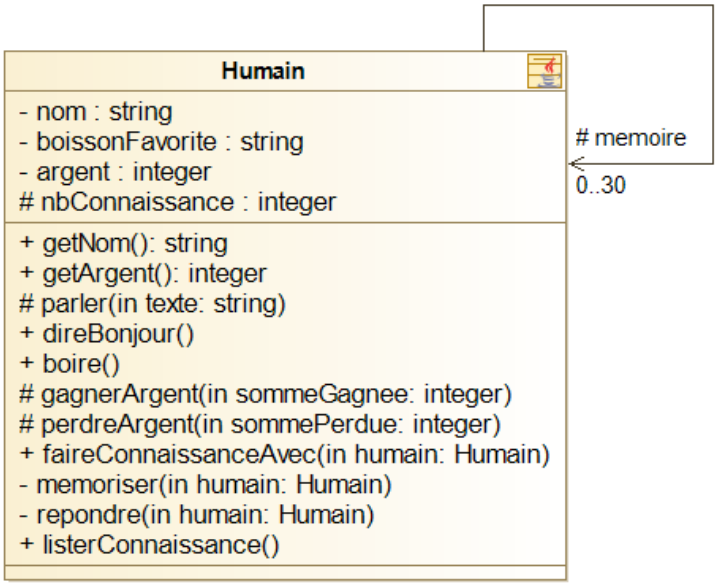
1- Les humains ont une mémoire

Les humains se rappellent des personnes qu'ils croisent. Ils ont une mémoire de 30 humains, qu'ils peuvent enrichir en faisant connaissance avec des humains de nos histoires. Au-delà de 30 humains, ils oublient la personne dont ils ont fait connaissance il y a le plus longtemps afin de se souvenir de la nouvelle personne.

A chaque fois qu'un humain fait connaissance avec un autre humain, il lui dit bonjour et l'ajoute à sa mémoire. Il en est de même pour l'autre humain.

Ci-dessous la classe **`Humain`** et le diagramme de séquence de la méthode **faireConnaissanceAvec(Humain humain)**.

Diagramme de classe



L'outil utilisé pour les diagrammes UML est Modelio.

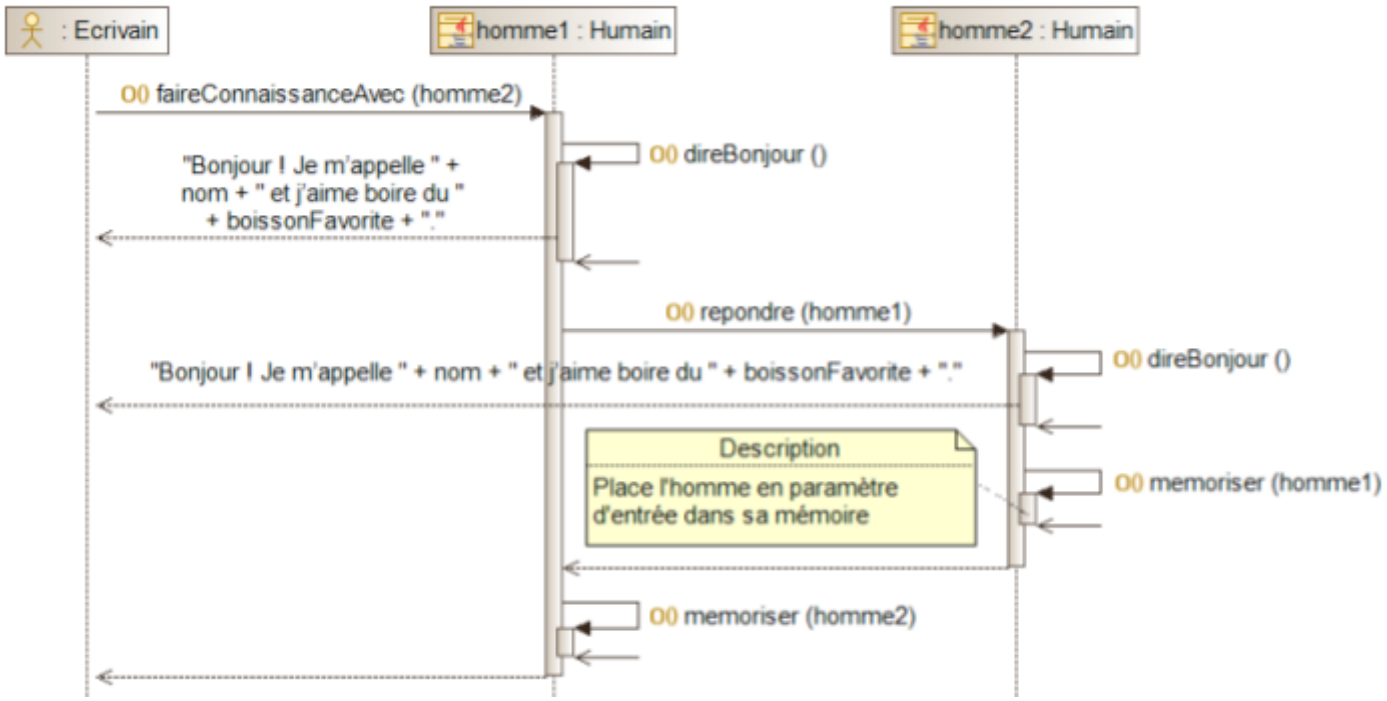
Cet outil est basé sur Eclipse.

Ainsi Modelio et Eclipse peuvent se partager le même workspace.

Cela permet de générer du code à partir de diagramme de classes mais aussi de faire l'opération inverse : récupérer les informations nécessaires aux diagrammes en utilisant le code écrit dans Eclipse.

Malheureusement il affiche des "in" au niveau des paramètres d'entrée des opérations. Il ne faut pas que vous en teniez compte.

Diagramme de séquence



a. En suivant ce qui est indiqué dans le diagramme de classes et le diagramme de séquence, écrire la méthode **void faireConnaissanceAvec(Humain autreHumain)** dans la classe Humain ainsi que toutes les méthodes nécessaires à son fonctionnement.

b. Écrire la méthode **listerConnaissance()** qui affiche l'ensemble des noms des connaissances présentes dans la mémoire d'un humain comme cela est décrit dans le scénario à la question ci-dessous.

c. Tester dans la classe **`HistoireTP5`** pour obtenir le scénario ci-dessous. Les objets créés sont :
   - Commercant marco = new Commercant("Marco", 20);
   - Commercant chonin = new Commercant("Chonin", 40);
   - Commercant kumi = new Commercant("Kumi", 10);
   - Yakuza yaku = new Yakuza("Yaku Le Noir", "whisky", 30, "Warsong");
   - Ronin roro = new Ronin("Roro", "shochu", 60);

(Marco) - Bonjour ! Je m'appelle Marco et j'aime boire du thé.
(Roro) - Bonjour ! Je m'appelle Roro et j'aime boire du shochu.
(Marco) - Bonjour ! Je m'appelle Marco et j'aime boire du thé.
(Yaku Le Noir) - Bonjour ! Je m'appelle Yaku Le Noir et j'aime boire du whisky.
(Marco) - Bonjour ! Je m'appelle Marco et j'aime boire du thé.

(Chonin) - Bonjour ! Je m'appelle Chonin et j'aime boire du thé.
(Marco) - Bonjour ! Je m'appelle Marco et j'aime boire du thé.
(Kumi) - Bonjour ! Je m'appelle Kumi et j'aime boire du thé.
(Marco) - Je connais beaucoup de monde dont : Roro, Yaku Le Noir, Chonin, Kumi
(Roro) - Je connais beaucoup de monde dont : Marco
(Yaku Le Noir) - Je connais beaucoup de monde dont : Marco

d. Modifier la taille de la mémoire d'un humain afin qu'elle ne retienne que 3 personnes et relancer le test. La mémoire de Marco est maintenant :
(Marco) - Je connais beaucoup de monde dont : Yaku Le Noir, Chonin, Kumi

Corriger si besoin vos méthodes, puis remettre la mémoire avec une taille de 30.

e. Sauvegarder votre travail sous GitHub (cf p1 - Historisation périodique de votre projet - exemple de message pour le commit "les humains ont des connaissances")

2- Le yakuza est fier de son clan

Lorsque le Yakuza dit bonjour, on veut qu'il annonce son clan en plus de sa présentation normale.

a. Dans la classe **`Yakuza`**, redéfinir la méthode **`direBonjour`** de la classe **`Humain`** pour que l'affichage soit conforme au scénario ci-dessous.
Ne pas oublier d'utiliser le tag **`@Override`**.
ATTENTION si vous faites un copier / coller c'est que vous n'utilisez pas l'héritage correctement ! Donc analysez ce qui est semblable de ce qui est différent dans les deux méthodes **`direBonjour`** afin de produire un code maintenable.

b. Relancer le scénario précédent, maintenant quand Yaku dit bonjour il donne son appartenance à son clan.

(Yaku Le Noir) - Bonjour ! Je m'appelle Yaku Le Noir et j'aime boire du whisky.
(Yaku Le Noir) - Mon clan est celui de Warsong.

c. Sauvegarder votre travail sous GitHub (cf p1 - Historisation périodique de votre projet - exemple de message pour le commit "le clan des Yakuzas")

3- Les samouraïs

Les samouraïs sont des ronins liés à un seigneur (représenté par une chaîne de caractères, contenant le nom). Le nom de son seigneur est donné au constructeur. Lorsqu'un samouraï se présente, il annonce quel seigneur il sert.

a. Créer la classe **`Samourai`**.

b. Dans la classe **`Samourai`**, redéfinir la méthode **`direBonjour`** de la classe **`Humain`** pour que l'affichage soit conforme au scénario ci-dessous. Ne pas oublier d'utiliser le tag **`@Override`**.

c. Tester dans la classe **`HistoireTP5`** pour obtenir le scénario ci-dessous. L'objet créés est : Samourai akimoto = **new** Samourai("Miyamoto", "Akimoto", "saké", 80);

(Akimoto) - Bonjour ! Je m'appelle Akimoto et j'aime boire du saké.
(Akimoto) - Je suis fier de servir le seigneur Miyamoto.
(Marco) - Bonjour ! Je m'appelle Marco et j'aime boire du thé.
(Akimoto) - Je connais beaucoup de monde dont : Marco

Alors que tous les autres humains ne peuvent boire que leur boisson favorite, le samouraï a une super capacité spéciale : il peut boire n'importe quelle boisson.
C'est le seul à pouvoir, par exemple, boire du thé en journée, prendre des bières à l'apéro et accompagner son repas de saké.

d. Coder la méthode **boire(String boisson)** qui surcharge celle de **`Humain`** pour que l'affichage soit conforme au scénario ci-dessous.

e. Tester dans la classe HistoireTP5 :

(Akimoto) - Qu'est-ce que je vais choisir comme boisson ? Tiens je vais prendre du thé.

f. Sauvegarder votre travail sous GitHub (cf p1 - Historisation périodique de votre projet - exemple de message pour le commit "les samourais")

4- Les traîtres

Un traître est un samouraï qui peut rançonner **ranconner(Commercant commercant)** des commerçants en leur prenant deux dixièmes de leur argent.

Il possède un attribut représentant son niveau de traîtrise, celui-ci étant nul lors de la création d'un traître.

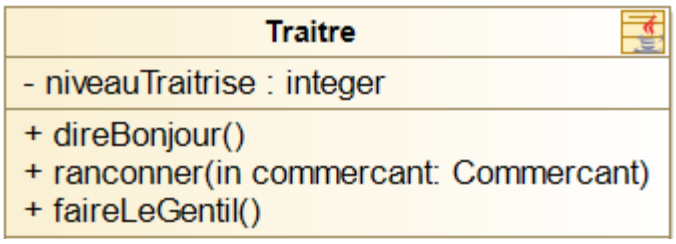
Lorsqu'il dit bonjour, il affiche son niveau actuel de traîtrise.

Chaque fois qu'il rançonne un commerçant son niveau de traîtrise augmente de 1.
Il ne peut plus rançonner si son niveau atteint 3.

Enfin, il peut faire croire qu'il est gentil **faireLeGentil()** en faisant des dons à n'importe quel humain de sa connaissance de 1 vingtième de son argent (il dira qu'il fait ami-ami). Son niveau de traîtrise diminue de 1 (sans jamais descendre en dessous de 0).

Ci-dessous, la classe **`Traitre`** et les diagrammes de séquence des méthodes **`ranconner`** et **`faireLeGentil`**.

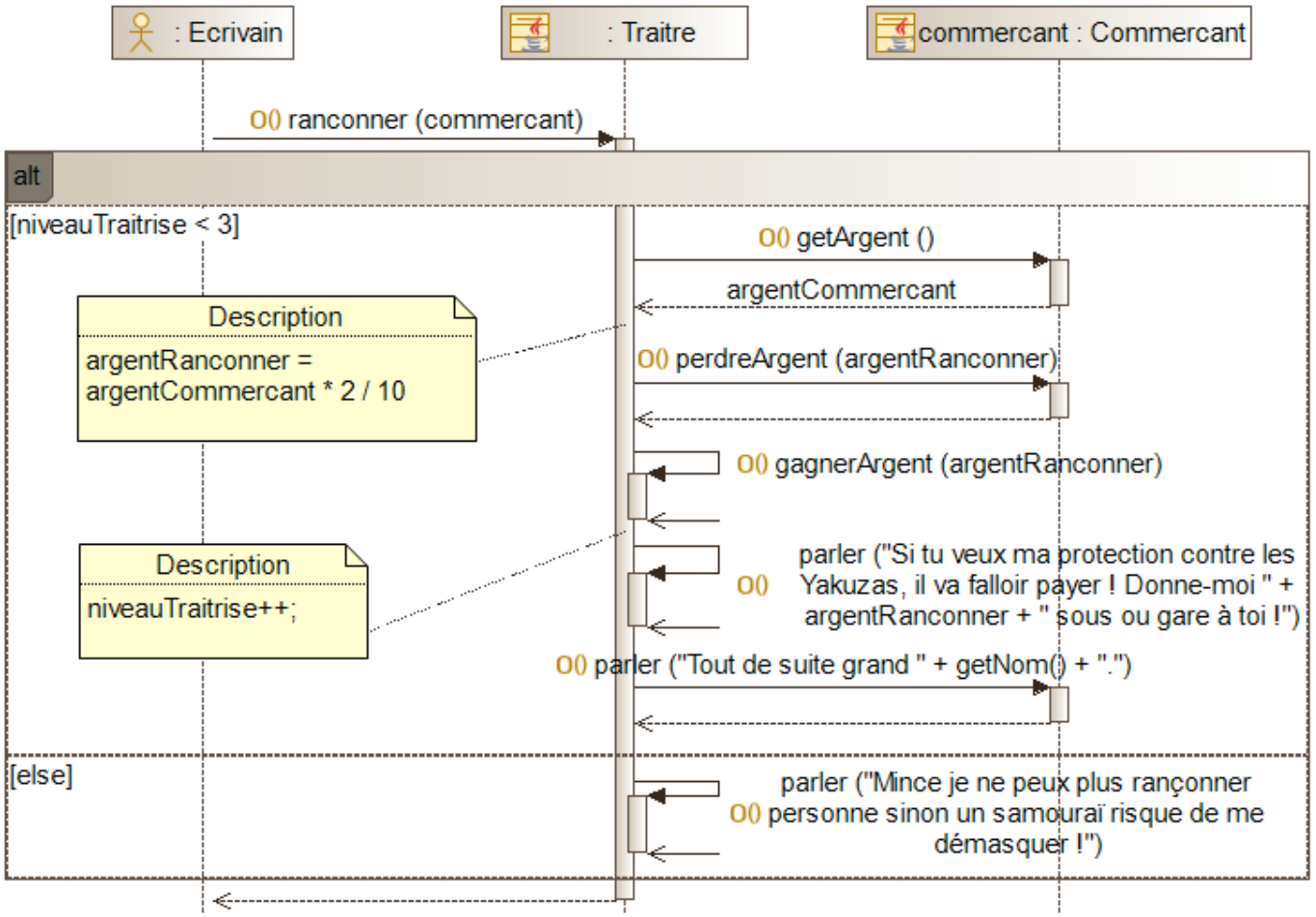
La classe Traitre



a. Créer la classe **`Traitre`** avec son attribut et son constructeur.

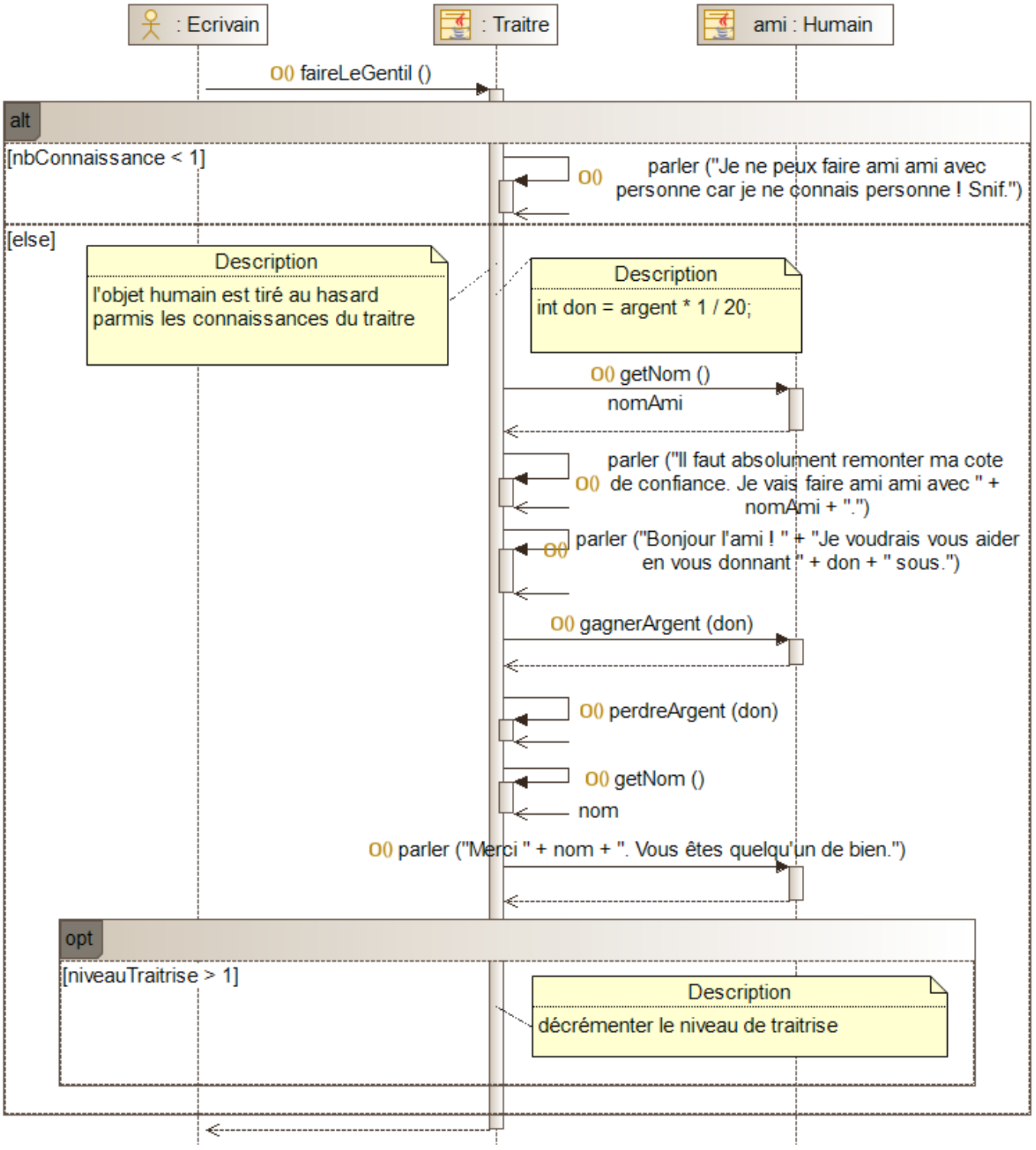
b. Dans la classe **`Traitre`**, redéfinir la méthode **`direBonjour`** de la classe **`Samourai`** pour que l'affichage soit conforme au scénario ci-dessous. Ne pas oublier d'utiliser le tag **`@Override`**.

Diagramme de séquence de la méthode **`ranconner`**



c. En suivant ce qui est indiqué dans le diagramme de classes et le diagramme de séquence, écrire la méthode **`ranconner`** dans la classe **`Traitre`**.

Diagramme de séquence de la méthode **`faireLeGentil`**



d. En suivant ce qui est indiqué dans le diagramme de classes et le diagramme de séquence, écrire la méthode **`faireLeGentil`** dans la classe **`Traitre`**.

e. Tester dans la classe **`HistoireTP5`** pour obtenir le scénario ci-dessous. L'objet créés est : Traitre masako = **new** Traitre("Miyamoto", "Masako", "whisky", 100);

Ci-dessous une trace possible (masako peut donner soit à yaku soit à akimoto quand il veut faire le gentil).

(Masako) - Je ne peux faire ami ami avec personne car je ne connais personne ! Snif.
(Masako) - Si tu veux ma protection contre les Yakuzas, il va falloir payer ! Donne-moi 2 sous ou gare à toi !
(Kumi) - Tout de suite grand Masako.
(Masako) - Si tu veux ma protection contre les Yakuzas, il va falloir payer ! Donne-moi 8 sous ou gare à toi !
(Chonin) - Tout de suite grand Masako.
(Masako) - Si tu veux ma protection contre les Yakuzas, il va falloir payer ! Donne-moi 4 sous ou gare à toi !
(Marco) - Tout de suite grand Masako.
(Akimoto) - Bonjour ! Je m'appelle Akimoto et j'aime boire du saké.
(Akimoto) - Je suis fier de servir le seigneur Miyamoto.
(Masako) - Bonjour ! Je m'appelle Masako et j'aime boire du whisky.
(Masako) - Je suis fier de servir le seigneur Miyamoto.
(Masako) - Mais je suis un traître et mon niveau de traîtrise est : 3. Chut !
(Masako) - Mince je ne peux plus rançonner personne sinon un samouraï risque de me démasquer !
(Masako) - Bonjour ! Je m'appelle Masako et j'aime boire du whisky.
(Masako) - Je suis fier de servir le seigneur Miyamoto.
(Masako) - Mais je suis un traître et mon niveau de traîtrise est : 3. Chut !
(Yaku Le Noir) - Bonjour ! Je m'appelle Yaku Le Noir et j'aime boire du whisky.
(Yaku Le Noir) - Mon clan est celui de Warsong.
(Masako) - Il faut absolument remonter ma cote de confiance. Je vais faire ami ami avec Yaku Le Noir.
(Masako) - Bonjour l'ami ! Je voudrais vous aider en vous donnant 5 sous.
(Yaku Le Noir) - Merci Masako. Vous êtes quelqu'un de bien.
(Masako) - Bonjour ! Je m'appelle Masako et j'aime boire du whisky.
(Masako) - Je suis fier de servir le seigneur Miyamoto.
(Masako) - Mais je suis un traître et mon niveau de traîtrise est : 2. Chut !
(Roro) - Bonjour ! Je m'appelle Roro et j'aime boire du shochu.

f. Sauvegarder votre travail sous GitHub (cf p1 - Historisation périodique de votre projet - exemple de message pour le commit "les traitres").

5- Humains, Ronin, Samouraï et Traître : les mêmes origines mais pas les mêmes combats !

Faire un diagramme de classes contenant les classes **Humain**, **Ronin**, **Samourai** et **Traitre**.

Ajouter les méthodes sans omettre les paramètres d'entrée et de sortie ainsi que la visibilité :

- **getNom**, **direBonjour**, **boire** et **memoriser** dans la classe **Humain**
- **provoquer** dans la classe **Ronin**
- **direBonjour** et **boire** dans la classe **Samourai**
- **direBonjour** et **ranconner** dans la classe **Traitre**

Nous avons déjà créé l'objet **masako** :

```
Traitre masako = new Traitre("Miyamoto", "Masako", "whisky", 100);
```

Créer les objets suivants :

```
Samourai akira = new Traitre("Miyamoto", "Akira", "whisky", 50);
Ronin hiro = new Traitre("Miyamoto", "Hiro", "saké", 20);
Humain nori = new Traitre("Miyamoto", "Nori", "whisky", 70);
```

Quelles méthodes (et de quelles classes) peuvent être utilisées par les objets : **masako** ? **akira** ? **hiro** ? **nori** ?

Sous Moodle suivez le lien : "wooflash - Question 4", et donner les méthodes qui peuvent être utilisées (2 niveaux de difficulté - questions 1-4 difficulté niveau 1, question 5 difficulté niveau 2).

6- Les grand-mères : les derniers personnages mais pas les moins importants !

Une grand-mère est un humain assez particulier : sa mémoire est seulement de 5 humains. Au-delà de 5 humains, elle ne peut plus retenir de personnes supplémentaires et elle s'exclame : "Oh ma tête ! Je ne peux plus retenir le nom d'une personne supplémentaire !".

Ensuite elle passe ses journées à ragoter, en se demandant si chacune de ses connaissances est un commerçant, un Ronin, un Samouraï...

Par contre elle détecte tout de suite avec son flair incroyable les traîtres.

Enfin, une grand-mère ne boit que de la tisane.

Comme elle ne sait pas vraiment dire qui est qui, nous allons lui fournir une méthode privée **String humainHasard()** qui renvoie aléatoirement une chaîne de caractère contenant "Commerçant" ou "Ronin" ou une des autres noms de type de personnages. Pour cela vous utiliserez un objet de la classe **Random** et un énuméré **TypeHumain** (privée à la classe) contenant les différents types de personnages de l'histoire ainsi que "habitant" pour un humain.
Indice : pour récupérer toutes les valeurs d'un énuméré vous pouvez utiliser un tableau. `TypeHumain[] types = TypeHumain.values();`

Pour détecter si un objet est d'un type particulier, il existe un mot-clé en java instanceof qui permet de tester l'appartenance à une classe en particulier :

```
if (monObjet instanceof maClasse){
    faireQuelquechose();
}
```

Note : attention à l'usage abusif de instanceof (discuté en CTD).

Utilisez-le pour détecter les traîtres !

a. Ajoutez la classe GrandMere.
   - Redéfinir dans la classe **GrandMere** la méthode **memoriser** (vous devez donc relâcher un peu la protection de cette méthode au niveau de la classe Humain).
   - créer les méthodes **String humainHasard()** et void **ragoter()**.

b. Tester dans la classe **`HistoireTP5`** pour obtenir le scénario ci-dessous. Ci-dessous le code à écrire.

```
GrandMere grandMere = new GrandMere("Grand-Mère", 10);
grandMere.faireConnaissanceAvec(akimoto);
grandMere.faireConnaissanceAvec(yaku);
grandMere.faireConnaissanceAvec(masako);
grandMere.faireConnaissanceAvec(kumi);
grandMere.faireConnaissanceAvec(marco);
grandMere.faireConnaissanceAvec(chonin);
grandMere.ragoter();
```

Ci-dessous une trace possible (ragoter utilisant la méthode **`humainHasard`**).

(Grand-Mère) - Bonjour ! Je m'appelle Grand-Mère et j'aime boire du tisane.
(Akimoto) - Bonjour ! Je m'appelle Akimoto et j'aime boire du saké.
(Akimoto) - Je suis fier de servir le seigneur Miyamoto.
(Grand-Mère) - Bonjour ! Je m'appelle Grand-Mère et j'aime boire du tisane.
(Yaku Le Noir) - Bonjour ! Je m'appelle Yaku Le Noir et j'aime boire du whisky.
(Yaku Le Noir) - Mon clan est celui de Warsong.
(Grand-Mère) - Bonjour ! Je m'appelle Grand-Mère et j'aime boire du tisane.
(Masako) - Bonjour ! Je m'appelle Masako et j'aime boire du whisky.
(Masako) - Je suis fier de servir le seigneur Miyamoto.
(Masako) - Mais je suis un traître et mon niveau de traîtrise est : 2. Chut !
(Grand-Mère) - Bonjour ! Je m'appelle Grand-Mère et j'aime boire du tisane.
(Kumi) - Bonjour ! Je m'appelle Kumi et j'aime boire du thé.
(Grand-Mère) - Bonjour ! Je m'appelle Grand-Mère et j'aime boire du tisane.
(Marco) - Bonjour ! Je m'appelle Marco et j'aime boire du thé.
(Grand-Mère) - Bonjour ! Je m'appelle Grand-Mère et j'aime boire du tisane.
(Chonin) - Bonjour ! Je m'appelle Chonin et j'aime boire du thé.
(Grand-Mère) - Oh ma tête ! Je ne peux plus retenir le nom d'une personne supplémentaire !
(Grand-Mère) - Je crois que Akimoto est un ronin
(Grand-Mère) - Je crois que Yaku Le Noir est une grand-mère
(Grand-Mère) - Je sais que Masako est un traître. Petit chenapan !
(Grand-Mère) - Je crois que Kumi est un habitant
(Grand-Mère) - Je crois que Marco est un samouraï

c. Sauvegarder votre travail sous GitHub (cf p1 - Historisation périodique de votre projet - exemple de message pour le commit "les grands meres").