



Introduction au développement

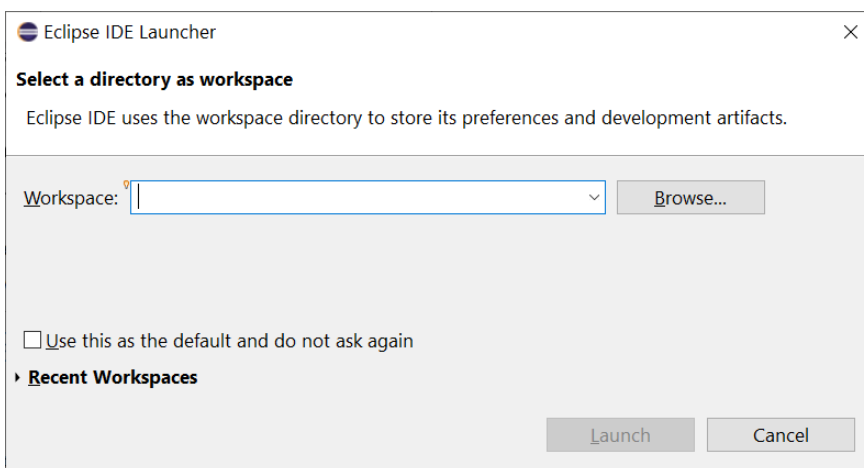
Une histoire de Gaulois

① Préparation de l'environnement Eclipse

Ouverture De l'IDE Eclipse

Nous allons utiliser l'IDE Eclipse, vous devrez apprendre à l'utiliser : cela fait partie des compétences que vous devez acquérir.

Aller sous l'environnement Windows et choisir la version **Eclipse 2022-03**



Si la fenêtre ci-dessus ne s'ouvre pas, faire File>Switch Workspace>Other...

A l'aide du bouton **browse** placez-vous sous votre **espace personnel** :

- Sélectionner "Ce PC" puis choisir le lecteur "Z:"
- Créer des répertoires afin d'organiser votre espace de travail (clic droit puis Nouveau\Dossier
Exemple d'organisation : **Z:\SRI\1A\COO\workspace**)

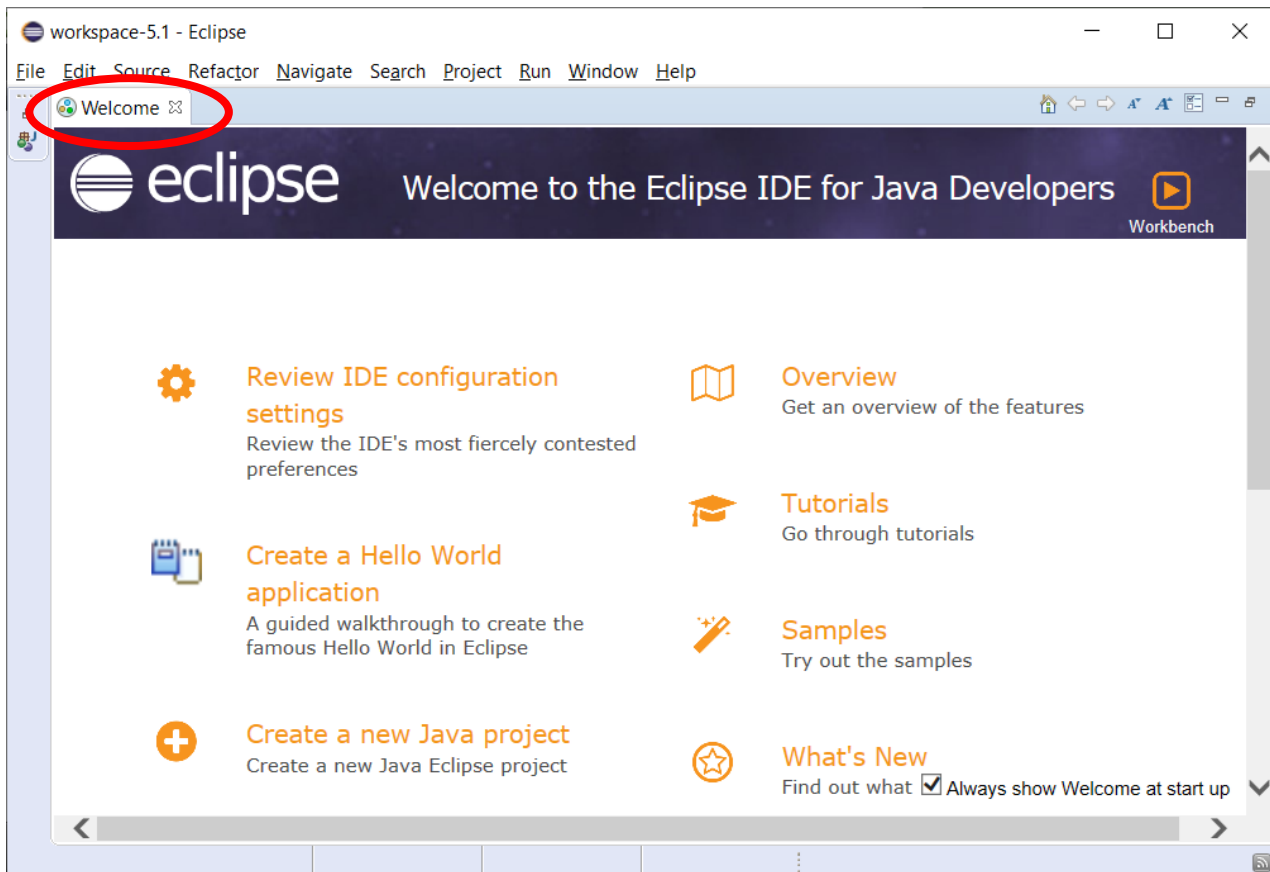
DANS TOUS LES CAS que vous soyez sur votre ordinateur personnel ou sur un ordinateur de l'université **terminer obligatoirement votre chemin par le dossier workspace**

Exemple de chemin complet :

- à l'université : Z:\SRI\1A\COO\workspace
- sur votre ordinateur personnel :
C:\Users\chaudet\Documents\SRI\1A\COO\workspace



Vous arrivez sur une fenêtre Welcome : il faut la fermer avec la croix au niveau de l'onglet



Installation des plugins Eclipse

Nous allons utiliser le plugin sonarQube. Il s'agit d'un "Linter" : un outil qui assiste le développeur en analysant statiquement le code pour en contrôler et en améliorer la qualité.

Installation

Pour ajouter des plugins, aller sur l'onglet "Help" puis sélectionner "Eclipse Marketplace..."

Dans la barre de recherche "Search" taper le nom du plugin à ajouter : "sonarQube" puis appuyer sur le bouton "Go".

Sélectionner le plugin de "sonarQube" et cliquer sur le bouton "Install".

Si besoin, accepter le contrat puis cliquer sur le bouton "Finish".

Lorsqu'Eclipse le demande, redémarrer Eclipse.



② Préparation de l'environnement Git

Pour ceux qui travaillent sur leur machine ou pour travailler chez vous, vous devez installer git :

- Pour Windows : <https://gitforwindows.org/>
- Pour macOS : installer homebrew (<https://brew.sh/>) si vous ne l'avez pas déjà, puis faites :
\$ brew install git tig
- Pour Linux Ubuntu : installer le paquet git-all en faisant :
\$ sudo apt-get update && sudo apt-get install git-all

Pour les autres versions de linux, le lien d'installation est : <https://git-scm.com/download/linux>

Nous allons utiliser l'outil de gestion de version git et le site d'hébergement GitHub. Ceci se décompose en 4 étapes :

- la création de votre compte sur GitHub
- la création de votre dépôt distant pour ce sujet de TP
- le clonage local de votre projet
- l'historisation périodique de votre projet

Création de votre compte GitHub



Sign in to GitHub

Username or email address

Password

[Forgot password?](#)

Sign in

[Sign in with a passkey](#)

New to GitHub? [Create an account](#)

Connectez-vous sur le site <https://github.com> cliquez sur le bouton Sign Up, créez un nouveau compte avec l'adresse mail de l'université : il s'agit de vos premiers pas or github est public, se sera votre compte bac à sable. Trouver un pseudo d'utilisateur et un mot de passe solide. Après réception d'un code par mail et son enregistrement sur la plateforme, vous êtes prêt pour les étapes suivantes.



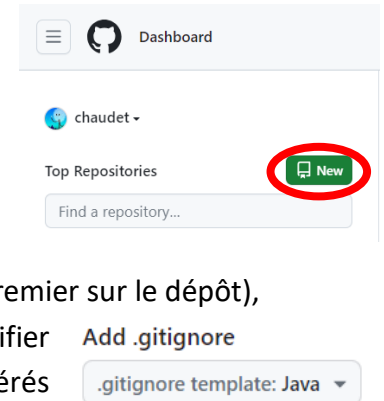
3 Création du projet « Les Gaulois »

Création de votre dépôt distant pour ce sujet de TP

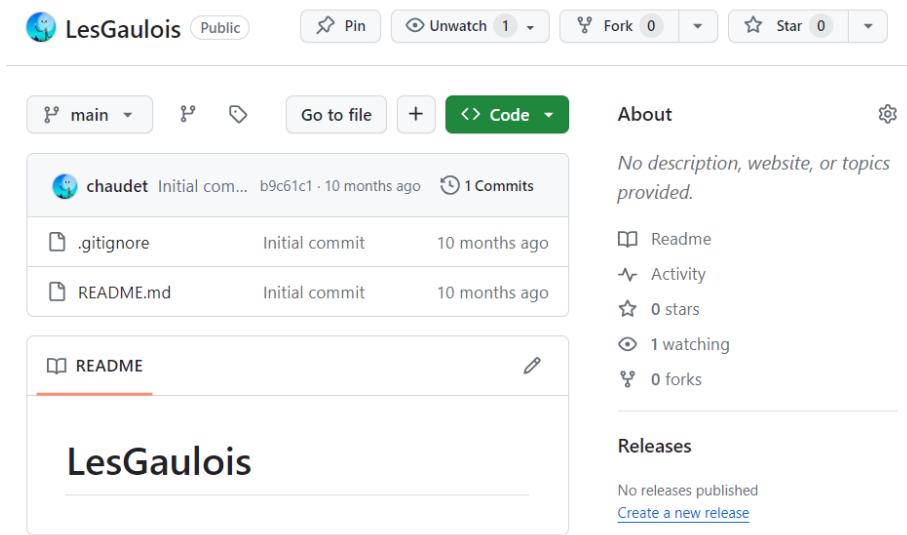
Connectez-vous et placez-vous sur la page de vos dépôts (*repositories*).

Cliquez sur le bouton New pour démarrer la création du dépôt. Renseignez :

- le nom du dépôt (pour ce TP : "LesGaulois"),
- la description (exemple : "Dépôt pour les premiers TPs de Java"),
- la visibilité 'Public' (au moins pour celui-ci),
- l'ajout d'un fichier 'README' initial (qui définit ce qu'il faut lire en premier sur le dépôt),
- l'ajout d'un fichier .gitignore en utilisant le template java (pour spécifier les fichiers qui ne seront pas automatiquement rajoutés car considérés comme inutiles sur le dépôt)
- pas de précision de licence pour le moment.

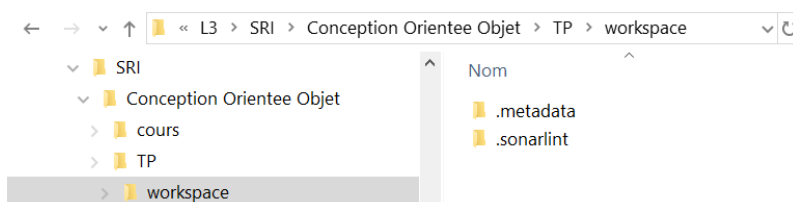


Une fois la création confirmée, vous devriez obtenir une première version de votre dépôt distant comparable à l'image ci-dessous.



Clonage local initial de votre projet

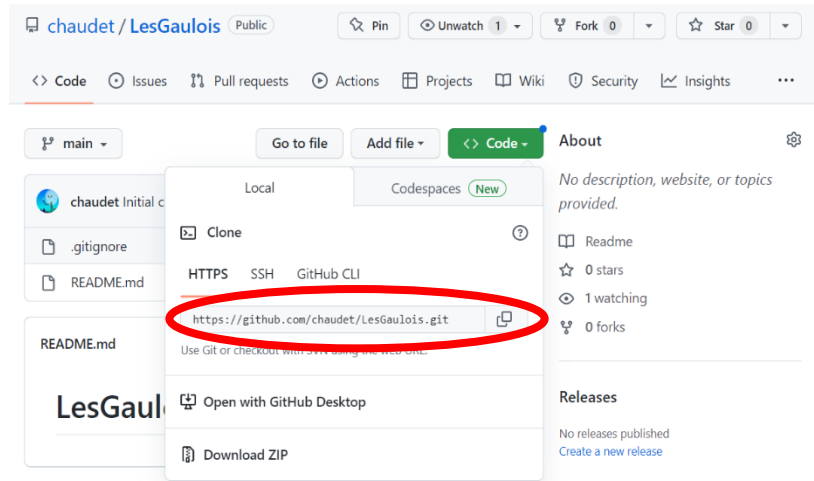
Ouvrez votre explorateur de fichier et positionnez-vous sous le dossier workspace que vous avez créé quand vous avez ouvert Eclipse.



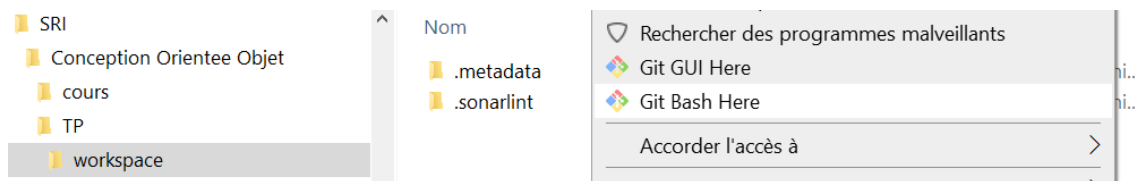


Sur **github** pour cloner le projet :

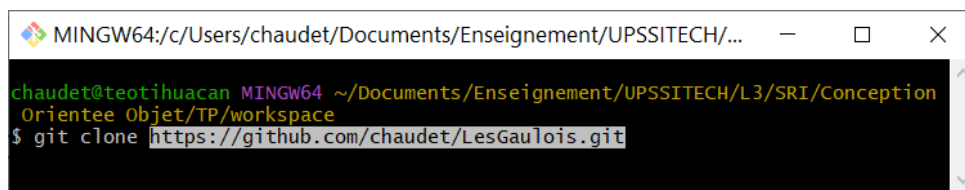
En restant sur le mode HTTPS, on peut copier l'URL du dépôt présentée dans la pop-up au moyen du bouton à droite du champ de texte.



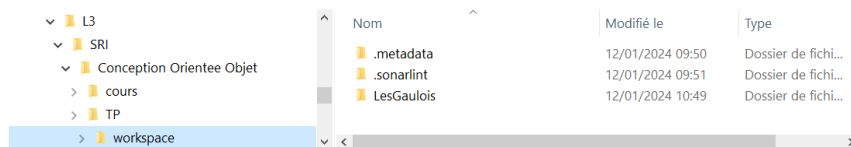
Dans votre explorateur de fichier cliquer droit sous votre dossier 'workspace' puis ouvrir 'Git Bash Here'.



Vous pouvez alors utiliser la commande git 'git clone'+ l'URL copiée dans le répertoire 'workspace' (le ctrl+v ne fonctionne pas il faut cliquer droit et sélectionner coller).



Vous pouvez voir le répertoire 'LesGaulois' sous votre explorateur de fichier.



Toujours sous Git Bash configurer votre identité Git en utilisant les commandes suivantes :

git config --global user.name "**mon_user_name**"

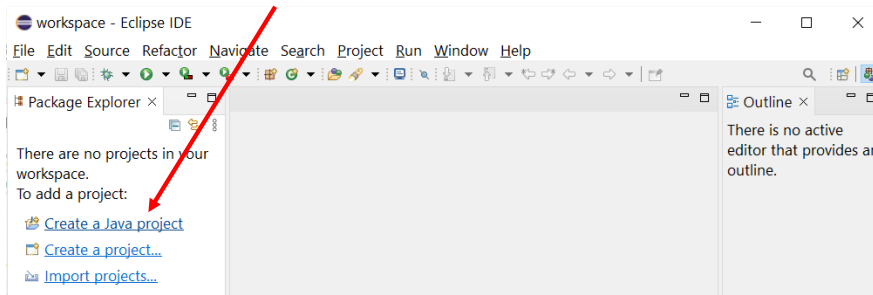
git config --global user.email "**mon_adresse**"

où **mon_user_name** est à remplacer par votre nom d'utilisateur GitHub et **mon_adresse** est à remplacer par votre adresse mail utilisée pour la création de votre compte GitHub. **Attention : bien mettre les guillemets.**

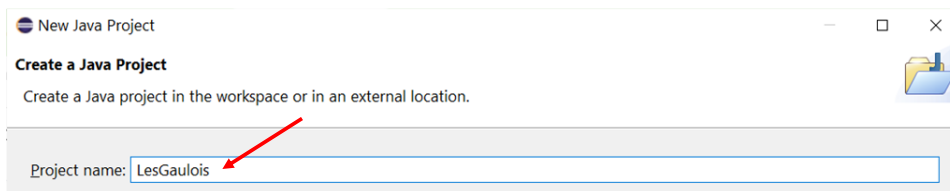


Ensuite, pour vérifier, vous pouvez utiliser la commande : `git config --global -l`

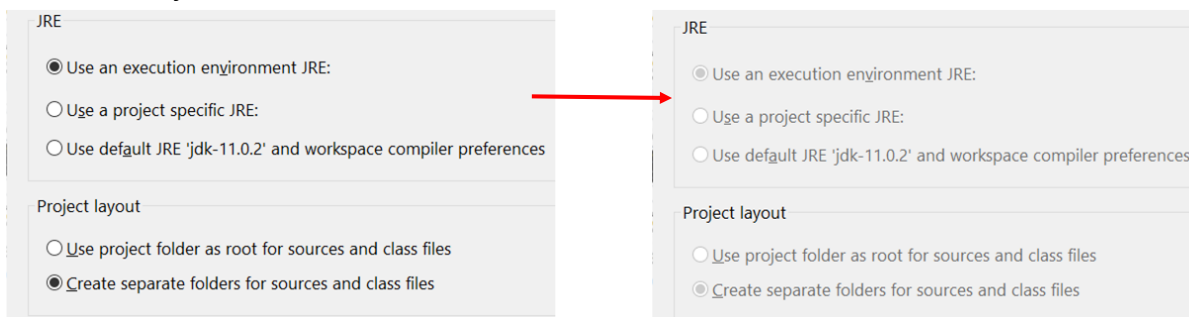
Il vous reste à récupérer le projet sous Eclipse. Créer un nouveau projet Java



Donner exactement le même nom que sur github



Quand vous collez le nom de votre projet Java toutes les options sur la JRE et sur Project Layout ne doivent plus être sélectionnables, si ce n'est pas le cas arrêtez tout et appelez le chargé de TP. Décocher « create module-info.java file ».



Historisation périodique de votre projet

Lorsque vous avez fini une question du sujet de TP (ce qui vous a amené à créer ou modifier plusieurs classes), vous devez procéder à l'historisation de votre dépôt local et la mise à jour du dépôt distant.

Dans votre explorateur de fichier cliquer droit sous votre dossier 'LesGaulois' puis ouvrir 'Git Bash Here'. Il faut réaliser trois commandes successives :

- Enregistrement des modifications à historiser dans l'index au moyen de la commande :
`'git add .'` (n'oubliez pas le point)
- Enregistrement dans le dépôt local des fichiers mis à l'index avec la commande :
`'git commit -m <Intitulé des modifications>'`
Exemple : `git commit -m "TP1 fin de l'exercice 1"`
De manière plus professionnelle il faudrait donner la nature des modifications effectuées.
- Mise à jour du dépôt distant avec la commande : `'git push'`¹

¹ A la première utilisation de la commande, une authentification par le navigateur peut être demandée. Complétez cette identification dans votre navigateur et revenez sur votre terminal continuer votre processus Git.



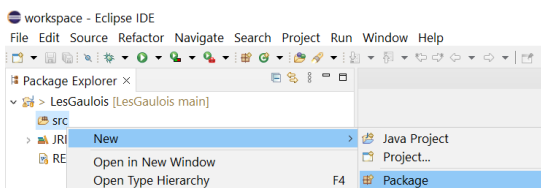
Récupération du dépôt distant

A l'issue d'un TP sur une machine de l'université, vous aurez donc modifié le dépôt local de la machine de TP et le dépôt distant. Si vous voulez travailler sur une machine personnelle en dehors de la séance, il vous faut mettre à jour votre dépôt local avec le contenu du dépôt distant :

- Pour la première fois sur une nouvelle machine : il faudra reprendre la procédure décrite ci-dessus depuis 'Clonage local initial de votre projet').
- Pour les fois suivantes, à chaque fois que vous commencez à travailler chez vous ou à l'université, dans votre explorateur de fichier cliquer droit sous votre dossier 'LesGaulois' puis ouvrir 'Git Bash Here'. Tapez la commande 'git pull'.

4 Implémentation Java « Les Gaulois »

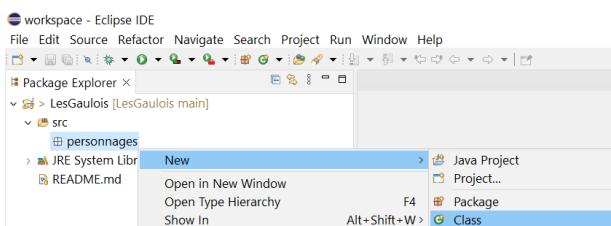
1. Créer le paquetage « personnages ». Attention à la casse : un nom de paquetage doit être écrit en minuscule. Décocher « create module-info.java file ».



Déplier le projet « LesGaulois » puis cliquer droit sur « src ».

Sélectionner « New » puis « Package ».

2. Créer une classe nommée « Gaulois ». Attention à la casse : un nom de classe doit commencer par une majuscule.



Cliquer droit sur le paquetage « personnages ».

Sélectionner « New » puis « Class ».

- a. Chaque gaulois possède un nom et une force (entier) correspondant à son état actuel. Ecrire les attributs correspondants.
- b. Le nom du personnage et sa force sont donnés à la création de l'objet. Ecrire le constructeur de la classe.
- c. La classe contient un getteur sur son nom.
- d. Historiser votre projet en suivant :

5 Création du projet « Les Gaulois » - Historisation périodique de votre projet - en page 6.

S'il cela ne fonctionne pas demander à votre enseignant mais ne passer pas à l'étape suivante sans avoir poussé sous github.



3. Créer le paquetage « test_fonctionnel » et dans ce paquetage créer la classe « TestGaulois ».

Générer la méthode main : écrire le nom de la méthode « main » à l'intérieur des accolades de la classe TestGaulois.

```
Gaulois.java  *TestGaulois.java ×
1 package test_fonctionnel;
2
3 public class TestGaulois {
4     main
5 }
6
```

main - main method

Puis appuyer sur les touches Ctrl + barre d'espace puis appuyer sur la touche Entrée.

La méthode main sera générée :

```
1 package test_fonctionnel;
2
3 public class TestGaulois {
4     public static void main(String[] args) {
5     }
6 }
7 }
```

La méthode main() est imposée par la machine virtuelle pour reconnaître le point d'entrée d'une application.

Dans cette méthode créer un objet correspondant à un gaulois :

- asterix qui a pour nom = "Astérix" et pour force 8

Eclipse vous propose d'importer la classe Asterix depuis le paquetage personnages (ampoule dans la marge), accepter.

4. Ajouter la ligne de code suivante : `System.out.println(asterix);` et exécuter le programme.

Appuyer sur le bouton run :



Le résultat se trouve dans la console :

```
Problems  Javadoc  Declaration  Console ×
<terminated> TestGaulois [Java Application] C:\Prograr
personnages.Gaulois@368239c8
```

J'obtiens le résultat qui a pour forme : `personnages.Gaulois@368239c8`

- `personnages` est le nom du paquetage.
- `Gaulois` le nom de la classe.
- `@368239c8` est la représentation hexadécimale de l'objet `asterix`².

Donc quand vous demandez l'affichage d'un objet c'est le résultat que vous obtenez : il n'y a pas d'erreur. Par contre ce que l'on voudrait afficher c'est la valeur de l'attribut « nom » de l'objet « asterix ».

Corriger la commande « `System.out.println(asterix);` » pour obtenir l'affichage suivant : `Astérix`.

² Il s'agit d'une représentation hexadécimale du résultat de la méthode `hashCode()` de l'objet.



5. Quelques questions sur la visibilité. La classe « Gaulois » se situe dans le package « personnages ».

Pour cela nous allons créer trois espaces de test différents :

- Celui de la classe **TestGaulois** du package « test_fonctionnel ». Nous appellerons ce main dans ce sujet « main A » afin de le différencier des autres,
- Créer dans le package « personnages » la classe **TestPersonnages**. Cette classe ne contient qu'une méthode main que nous appellerons « main B » dans ce sujet afin de le différencier des autres,
- Créer un main à la fin de la classe **Gaulois** que nous appellerons « main C » dans ce sujet afin de le différencier des autres.

Dans chacun des « main » copier les 2 instructions écrites dans les questions 3 et 4.

Modifier la visibilité de la méthode **getNom** et remplir chaque ligne ci-dessous.

COCHEZ dans le tableau vos observations : ☒ on peut afficher Astérix, ☐ on ne peut pas.

Critère de visibilité devant le getteur getGaulois()	main A dans une classe en dehors du package	main B dans une classe dans le même package	main C A l'intérieur de la classe
public	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Aucun critère	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
protected	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
private	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Dans les bonnes pratiques nous utiliserons :

- pour les attributs les critères de visibilité private (et protected dans un contexte d'héritage),
- pour les méthodes les critères de visibilité public, private (et protected dans un contexte d'héritage).

6. Dans la classe **Gaulois** créer la méthode **parler** qui prend en paramètre un texte à afficher. Cette méthode affiche à l'écran « Le gaulois », son nom, deux points, guillemets ouvrants, le texte reçu en paramètre d'entrée puis guillemet fermant et un point. Par exemple :

Le gaulois Astérix : « Bonjour à tous ».

Tester la méthode dans le main de la classe **TestGaulois**.

Historiser votre projet en suivant :

③ Création du projet « Les Gaulois » - Historisation périodique de votre projet - en page 6.

7. **Créer une classe** nommé « Romain » dans le package « personnage ». Chaque romain possède un nom et une force (entier) correspondant à son état actuel. Ecrire les attributs correspondants.

Le nom du personnage et sa force sont donnés à la création de l'objet.

La classe contient un getteur sur son nom.



8. Dans la classe **Romain** reprendre la méthode `parler` de Gaulois en remplaçant « Le gaulois » par « Le romain ». Dans la méthode `main` de la classe « **TestGaulois** » créer un objet correspondant à un romain :
- minus (nom = "Minus", force = 6)

Faire parler le Romain, exemple :

Le Romain Minus : « UN GAU... UN GAUGAU... »

9. Dans la classe **Romain** créer la méthode `recevoirCoup` prenant en paramètre d'entrée un entier représentant la force du coup. L'attribut `force` sera diminué de la force du coup. Si la force du romain diminue en dessous de 0 alors elle sera ramenée à 0 (une force ne pouvant pas être négative !).

Si la force est > 0 le romain dira « Aïe ! » sinon « J'abandonne... ».

Dans le `main` de la classe **TestGaulois** faire recevoir à minus deux coups successifs de force 3. Vous devez OBLIGATOIREMENT utiliser une boucle `for`.

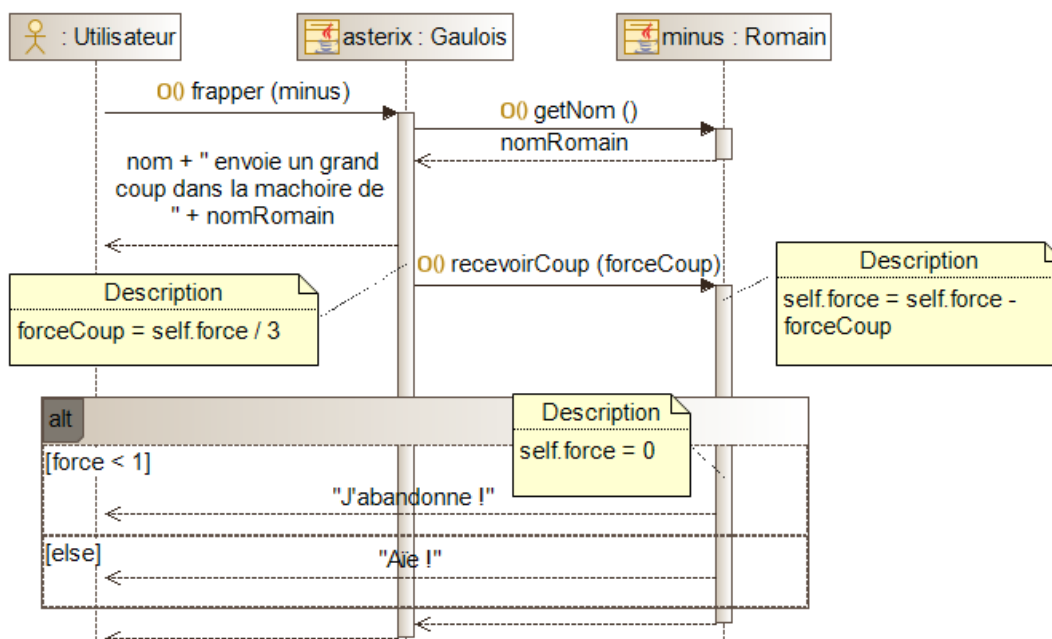
Affichage attendu :

```
Le gaulois Asterix : « Bonjour à tous ».  
Le romain Minus : « UN GAU... UN GAUGAU... ».  
Le romain Minus : « Aïe ! ».  
Le romain Minus : « J'abandonne... ».
```

10. Dans la classe **Gaulois** créer la méthode `frapper`, prenant en paramètre d'entrée un objet de type **Romain**.

Faire afficher à l'écran : le nom du gaulois, " envoie un grand coup dans la mâchoire de " et le nom du romain frappé.

Le romain reçoit le coup avec pour force celle du gaulois divisé par 3.



Dans le `main` de la classe **TestGaulois** mettre en commentaire la boucle de la question précédente.

Ecrire les instructions pour qu'Astérix frappe trois fois Minus (Minus abandonne au troisième coup).