

# Introduction au développement Java en ILU1

## 1 ➡ Première approche de l'utilisation de Git et GitHub

### Outillage

Pour utiliser un gestionnaire de version, vous devez utiliser deux outils : un permettant de gérer votre dépôt à distance (nous utiliserons GitHub) et un autre permettant de gérer votre projet sur votre ordinateur (la version dépend de votre système d'exploitation).

Pour utiliser GitHub, vous devez avoir un compte. Si ce n'est pas déjà le cas, inscrivez-vous gratuitement sur [github.com](https://github.com). Une fois connecté, vous aurez accès à une interface pour gérer vos projets.

Sur les ordinateurs de la fac, sous l'environnement windows, l'outil "git for windows" est déjà installé. Pour vos ordinateurs personnels, veuillez suivre les instructions du document "Installation des outils - Git - Eclipse"

### Sujet

#### 1.1 Créer un projet sur GitHub et le cloner sur votre ordinateur

Afin de prendre en main l'outil de gestion de version, vous devez :

- créer un projet sur GitHub : "ProjetTest" qui aura pour description "essai de l'outil". Ce projet devra être public, sans ajout de readme, de .gitignore ni de licence.
- cloner ce projet sur votre ordinateur à un endroit que vous choisirez.  
Pour cela, aller sur votre disque personnel (Z:). Vous reconnaîtrez votre identifiant. Ensuite vous créez les répertoires nécessaires pour obtenir ce chemin : Z:/ILU1/TPs/BacASable.  
Sous le répertoire BacASable, cloner votre projet :  
`git clone URL de votre projet distant`
- sous votre projet ProjetTest, créer un fichier texte HelloWorld.txt contenant "Hello"

Ces 3 étapes ci-dessus sont reprises en image dans le document "1.1 créer & cloner un projet". Attention vous devrez avoir acquis ces étapes sans aide à la fin de l'ILU1.

#### 1.2 Pousser vos modifications depuis votre ordinateur sur github

Sous le terminal, procéder aux étapes suivantes :

- Déplacez-vous sous votre projet.  
exemple : `cd ProjetTest`
- Pour dire à Git de suivre les modifications de ce fichier en particulier, utilisez la commande :  
`git add nom du fichier`  
exemple : `git add HelloWorld.txt`

Si vous souhaitez suivre l'ensemble des fichiers sous le répertoire, utilisez la commande :

*git add .*

- Un commit est une "photo" des modifications que vous avez apportées au projet. Pour enregistrer les modifications localement, utiliser la commande :  
*git commit -m "message décrivant les changements"*

exemple : *git commit -m "ajout du fichier HelloWorld"*

- Envoyer les commits de votre machine locale vers le dépôt GitHub.  
*git push*
- Retourner sur votre dépôt GitHub dans votre navigateur et actualiser la page. Vous verrez le fichier HelloWorld.txt avec son contenu.

Ces étapes ci-dessus sont reprises en image dans le document "1.2 pousser les modifications". Comme précédemment vous devrez avoir acquis ces étapes sans aide à la fin de l'ILU1."

## Résumé

1. **Cloner** : cela permet de travailler localement sur un projet GitHub.
2. **Ajouter et committer** : ces actions enregistrent les changements dans un historique.
3. **Pousser sur GitHub** : vous partagez vos modifications en ligne, pour vous ou pour d'autres utilisateurs.

Entraînez-vous à modifier le fichier HelloWorld en ajoutant et/ou supprimant du texte. En ajoutant un nouveau fichier comme un fichier pdf. Il faut que la manipulation devienne facile.



## Prise en main d'un IDE

### Préparation de l'environnement Eclipse

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

- Ouvrir Eclipse, sur les ordinateurs de la fac choisir la version **Eclipse 2022-03**
- Placez vous sous un environnement que vous choisissiez. Par exemple sous le répertoire **Z:/ILU1/TPs/BacASable/workspace**  
DANS TOUS LES CAS, que vous soyez sur votre ordinateur personnel ou sur un ordinateur de l'université, terminer votre chemin par le dossier **workspace**

### Installation des plugins Eclipse

Nous allons utiliser le plugin sonarQube dans les TPs.

#### *sonarQube*

Pour rappeler ce qu'on a vu en cours, un "*Lint*er" est un outil qui assiste le développeur en analysant statiquement le code pour en contrôler et en améliorer la qualité.

#### *Installation*

Pour ajouter des plugins à Eclipse, 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".

Redémarrer Eclipse

Retrouvez les étapes de la préparation de l'environnement et l'installation de sonarQube en image dans le document "2.1 préparation environnement Eclipse". Comme précédemment vous devrez avoir acquis ces étapes sans aide à la fin de l'ILU1."

## Utilisation de l'IDE Eclipse

L'avantage d'utiliser un IDE (Integrated Development Environment) c'est qu'il intègre plusieurs fonctionnalités et outils qui améliorent la qualité du développement. Une fois certaines habitudes prises, il permet de soutenir la productivité du développeur en écrivant certaines parties de code pour lui ou en automatisant certaines tâches.

Parmi les fonctionnalités que nous utiliserons beaucoup, il y a :

- la création d'un projet Java,
- la création de squelettes de code,
- la coloration syntaxique,
- la complétion des symboles (de variables ou méthodes accessibles).

Nous allons reprendre l'écriture des classes vues en TD en expliquant comment le faire rapidement. Donc si vous avez déjà réécrit le TD, on vous propose de le recommencer !

### Créer les différents éléments

Retrouver cette partie en vidéo sur Moodle : 3-1-creationElements.mp4

#### *Créer un projet Java*

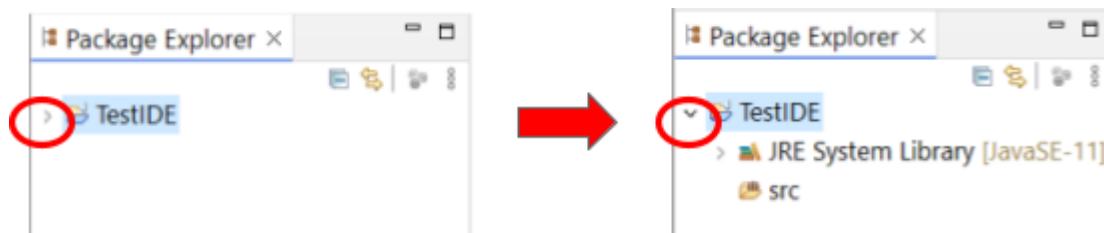
File > New > Java Project

Donner un nom représentant le sujet de votre projet (pas le numéro de séance de TP) par exemple "TestIDE".

Dans la partie "Module" décocher l'option : "Create module-info.java file".

Cliquer sur le bouton "Finish"

Développer l'arbre :



#### *Créer un paquetage (package)*

Cliquer droit sur le dossier "src" puis New > Package

Donner un nom représentant ce que contiendra le package, ici "navire".

**Convention d'écriture :** le nom d'un package doit être entièrement écrit en minuscule.

### Créer une classe Java

Cliquer droit sur le package "navire" puis New > Class

Donner le nom de la classe que vous voulez créer, ici "Element".

**Convention d'écriture** : le nom d'une classe doit :

- commencer par une majuscule,
- être au singulier,
- ne pas comporter d'accents.

S'il est composé de plusieurs mots ils seront accolés et chaque début de mot commencera par une majuscule (CamelCase, ex : SousMarin).

### Créer les attributs d'une classe

Créer l'attribut *touche* de type boolean initialisé à false.

```
private boolean touche = false;
```

**Convention d'écriture** : le nom d'un attribut doit :

- commencer par une minuscule,
- ne pas comporter d'accents.

S'il est composé de plusieurs mots ils seront accolés et chaque début de mot commencera par une majuscule (ex : sousMarin).


Reprendre les deux derniers points pour créer la classe Bateau possédant deux attributs *nom* de type *String* et *taille* de type *int*.

Toujours dans la classe Bateau ajouter l'attribut élément comme un tableau d'éléments. Utiliser l'IDE pour le faire rapidement :

private El -> CTRL + espace -> ENTRER

```
public class Bateau {
    private String nom;
    private int taille;
    private El

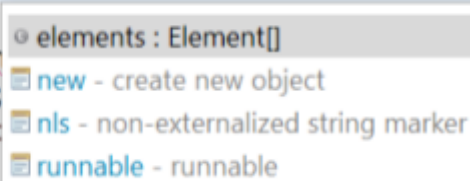
    public Bateau(String nom, int taille) {
        this.nom = nom;
        this.taille = taille;
    }
}
```



ajouter les crochets [] + espace puis CTRL + espace -> ENTRER

```
public class Bateau {
    private String nom;
    private int taille;
    private Element[]

    public Bateau(String nom, int taille) {
        this.nom = nom;
        this.taille = taille;
    }
}
```



## Générer le code standard

Un IDE permet de générer certaines parties du code. Selon la configuration de votre IDE Eclipse, celui-ci ajoute un commentaire :

```
// TODO Auto-generated method stub
```

Cela indique qu'il s'agit d'un code généré et qu'il faut le personnaliser.

Une fois que vous avez vérifié qu'il ne faut rien modifier ou au contraire que vous avez modifié ou complété le code alors vous devez supprimer ce commentaire.

Vous pouvez vous même ajouter des commentaires `// TODO` afin de noter les tâches à réaliser. Par exemple pour vous rappeler qu'une méthode n'est pas terminée et indiquer les différents points que vous devez encore réaliser.

Pour voir l'ensemble des tâches que vous devez terminer sélectionner :

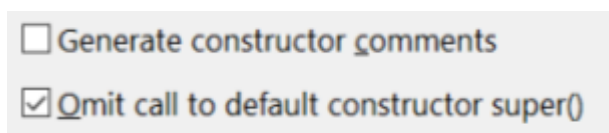
Window > Show View > Task  Tasks

### Générer le constructeur

Toujours dans la classe Bateau, générer le constructeur :

Source > Generate Constructor using Fields...

Sélectionner les attributs qui seront affectés par les paramètres d'entrée du constructeur et cocher la case "Omit call to default constructor super()" comme dans les deux illustrations ci-dessous.



Cliquer sur "generate"

```
public Bateau(String nom, int taille) {  
    this.nom = nom;  
    this.taille = taille;  
}
```

Compléter le constructeur en créant le tableau :

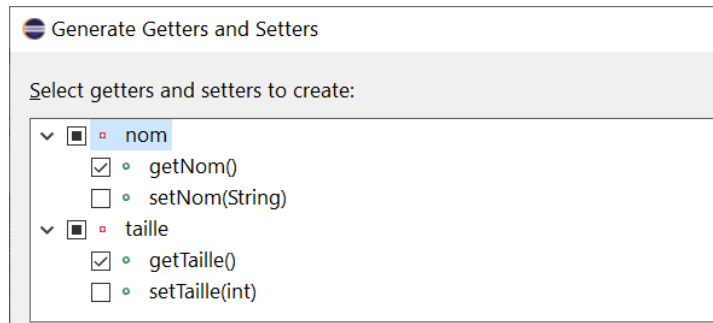
```
elements = new Element[taille];
```

## Générer les getters / setters

1. Pour générer plusieurs getter et setter :

Source > Generate Getters and Setters...

Développer l'arbre afin de choisir exactement les getters et setters dont vous avez besoin.



Cliquer sur "generate"

Remarque : un getter sur un attribut de type boolean s'écrit is + le nom de l'attribut. Par exemple, générer dans la classe Element le getter sur touche :

taper is puis Ctrl + espace

```
private boolean touche;
```

```
public boolean isTouche(){
    return touche;
}
```

Supprimer tous les getter et setter qui ont été générés dans la classe Bateau (dans ce programme nous n'avons besoin que du getter sur le nom).

2. Pour générer un getter ou un setter en particulier

Ecrire directement à l'endroit où vous souhaitez générer :

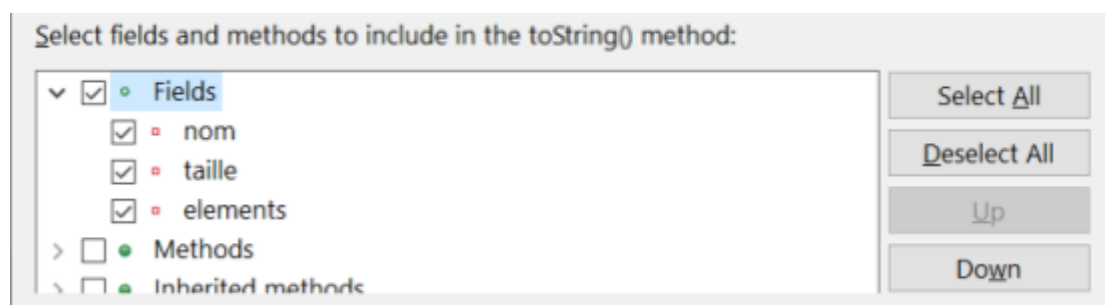
- un getter : get puis appuyer sur les touches Ctrl + la barre d'espace ou is puis appuyer sur les touches Ctrl + la barre d'espace
- un setter : set puis appuyer sur les touches Ctrl + la barre d'espace

Générer le getter sur le nom.

## Générer la méthode toString

La méthode toString() est une méthode qui est automatiquement appelée lorsque l'on souhaite obtenir une représentation textuelle (String) de l'objet, comme c'est le cas pour un affichage par exemple. Vous pouvez la personnaliser ou utiliser celle générée par Eclipse en choisissant les champs que vous souhaitez voir apparaître.

Source > Generate toString()...



Cliquer sur "generate"

```
@Override
public String toString() {
    return "Bateau [nom=" + nom + ", taille=" + taille
        + ", elements=" + Arrays.toString(elements) + "];"
}
```

*Mettre en forme le code selon les bonnes pratiques*

A utiliser régulièrement et au minimum à chaque fois que vous avez terminé d'écrire une classe. Formater permet une meilleure lecture, pour vous et pour vos collaborateurs qui doivent relire votre code.

Source > Format (Ctrl + Shift + F)

Pour que le formatage fonctionne il ne faut pas qu'il reste des erreurs de compilation.

*Générer la méthode main*

Il s'agit de la méthode qui sera exécutée lorsque vous ferez un "run" sur cette classe.

Ecrire directement à l'endroit où vous souhaitez générer la méthode main :  
main puis appuyer sur les touches Ctrl + la barre d'espace.

```
public static void main(String[] args) {
}
```



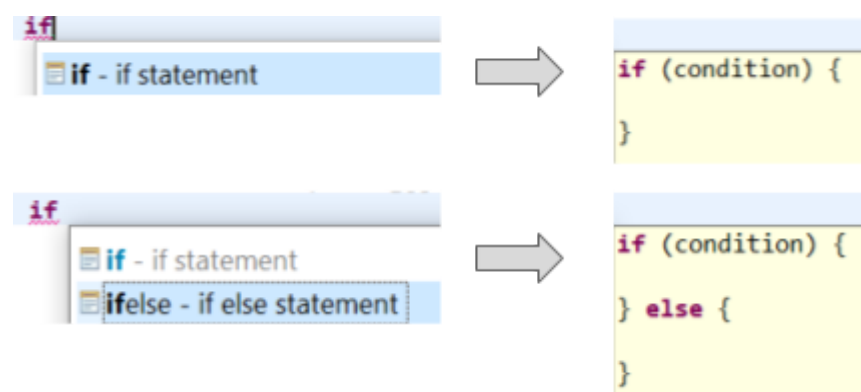
## Coder rapidement

### *Les structures algorithmiques*

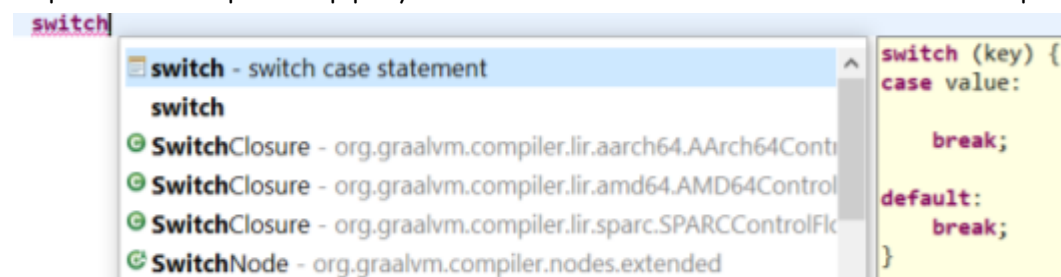
Se placer dans la méthode main.

En tapant quelques lettres Eclipse génère un squelette à compléter. Pour passer d'un champ au suivant taper sur la touche tabulation.

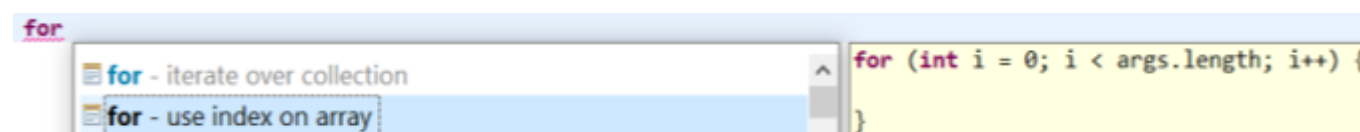
Taper if puis appuyer sur les touches Ctrl + la barre d'espace vous permet de choisir entre le if ou le ifelse



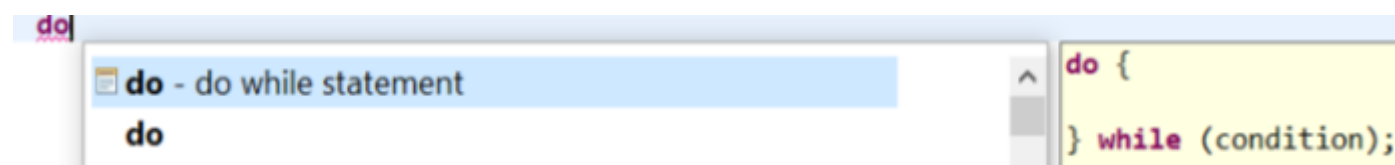
Taper switch puis appuyer sur les touches Ctrl + la barre d'espace



Taper for puis appuyer sur les touches Ctrl + la barre d'espace



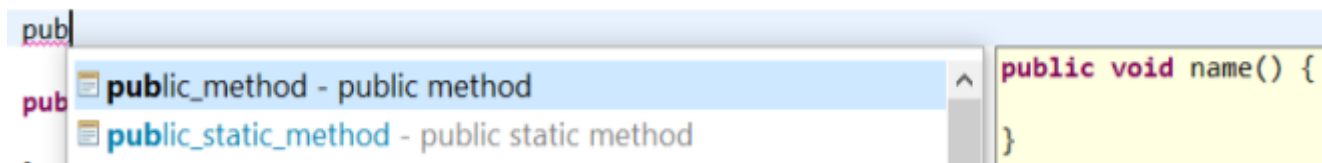
Taper do puis appuyer sur les touches Ctrl + la barre d'espace



Supprimer les essais des structures algorithmiques ci-dessus.

## Les méthodes

Dans la classe taper `pub` puis appuyer sur les touches `Ctrl + la barre d'espace` pour créer la méthode



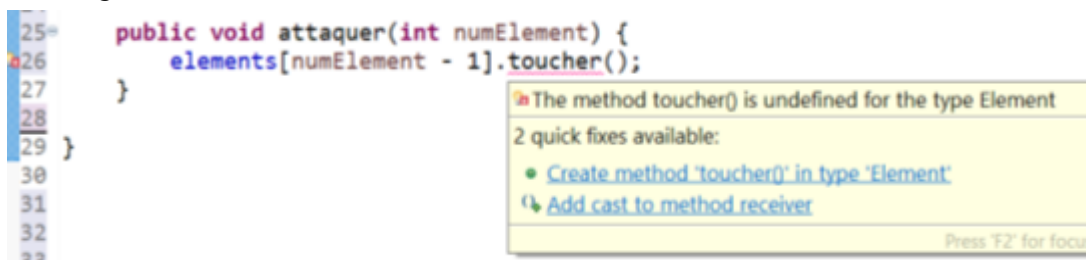
Vous pouvez vous déplacer avec la touche de tabulation pour modifier le paramètre de sortie, le nom de la méthode et les paramètres d'entrées.

```
public void attaquer(int numElement) {  
}
```

Une méthode peut être créée lorsque l'on en a besoin dans une autre méthode. Par exemple, j'ai besoin d'une méthode `toucher()` dans la classe `Element` que je n'ai pas encore écrite. Appuyer sur l'ampoule à gauche, puis appuyer sur les touches `Ctrl + la barre d'espace`, la méthode est ainsi générée.

Eclipse détermine si la méthode à générer possède un paramètre de retour ou non.

L'IDE génère la méthode dans la classe concernée :



```
public class Element {  
    private boolean touche = false;  
  
    public boolean isTouche() {  
        return touche;  
    }  
  
    public void toucher() {  
        // TODO Auto-generated method stub  
    }  
}
```

Il vous reste à compléter la méthode `toucher()` :

- supprimer la ligne permettant de repérer les méthodes à compléter  
*//TODO Auto-generated method stub*
- compléter la méthode  
`touche = true;`

## Les instructions

`Ctrl + espace` -> liste des paramètres, des variables locales, des attributs puis des méthodes que vous pouvez utiliser.

Afin de limiter le choix, taper la première lettre de ce que vous recherchez.

`syso` `Ctrl + espace` -> `System.out.println();`

Autres...

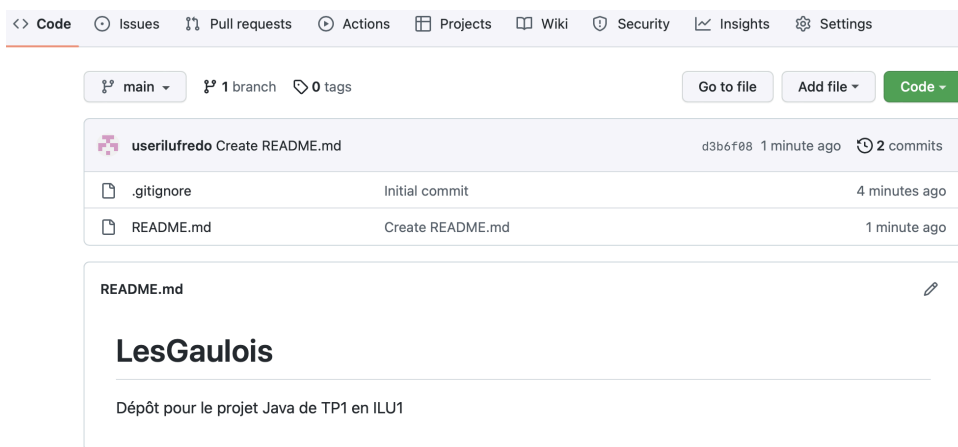
Sauvegarder l'ensemble des fichiers : `Ctrl + Shift + S`

### 3 Préparation de l'environnement Git pour suivre votre projet Java sous Eclipse

#### Création du projet sur GitHub

Reprenons les étapes de la partie 1.1 à quelques changements près :

- créer un projet sur GitHub intitulé "LesGaulois" qui aura pour description "suivi des TPs de POO". Ce projet devra être public, sans ajout de readme. Ajouter un fichier .gitignore avec le template Java. Ne pas ajouter de licence.
- Pour obtenir l'URL à copier, afin de le cloner sur votre ordinateur, cliquez sur le bouton vert : Code.



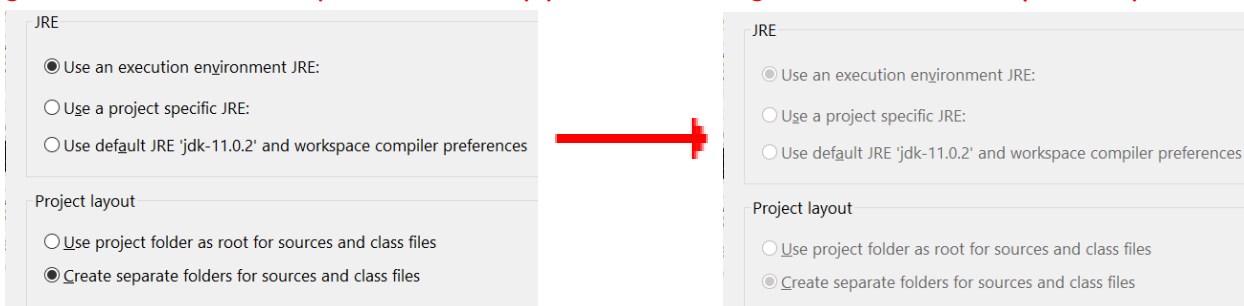
- Placez-vous sous le répertoire : Z:/ILU1/TPs/workspace, ouvrez votre terminal et cloner votre projet :  
*git clone LesGaulois*

#### Ouverture du projet sur l'IDE Eclipse

Ouvrir Eclipse sur le même workspace où vous avez cloné le projet LesGaulois.

Créer le projet sous Eclipse File > New > Java Project

Donner un nom représentant le sujet de votre projet, ici "LesGaulois". **Veiller à utiliser le même nom de projet que le nom du dépôt. Toutes les options deviennent grisées. Si ce n'est pas le cas, appelez l'enseignant de TPs : il y a un problème.**



Dans la partie "Module" décocher l'option : "Create module-info.java file".  
Cliquer sur le bouton "Finish"

## Écrire le code des classes vu en TD

En utilisant les différentes techniques vues dans la partie 2, écrire les différentes classes vues en TD. Une démonstration vous est proposée dans la vidéo 3-3-coderRapidement.mp4 disponible sous Moodle.

Les classes à implémenter sont données ci-dessous sous le package "personnages". Si vous avez bien suivi la partie précédente, seul ce qui est en rouge doit être écrit, tout le reste doit être généré en allant dans l'onglet source ou en utilisant le CTRL + espace.

```
public class Gaulois {
    private String nom;
    private int force;

    public Gaulois(String nom, int force) {
        this.nom = nom;
        this.force = force;
    }

    public String getNom() {
        return nom;
    }

    public void parler(String texte) {
        System.out.println(prendreParole() + "\"" + texte + "\"");
    }

    private String prendreParole() {
        return "Le gaulois " + nom + " : ";
    }
}

public class Romain {
    private String nom;
    private int force;

    public Romain(String nom, int force) {
        this.nom = nom;
        this.force = force;
    }

    public String getNom() {
        return nom;
    }

    public void parler(String texte) {
        System.out.println(prendreParole() + "\"" + texte + "\"");
    }

    private String prendreParole() {
        return "Le romain " + nom + " : ";
    }
}
```

*Finir ce sujet avant le prochain TP et penser à pousser vos modifications sur github !*