Consignes pour rendre un travail.

Vous rendrez vos TP via votre dépôt GitLab pour POO. Le **respect des échéances est obligatoire** et sera un critère d'évaluation.
Dans votre dépôt, vous créerez un dossier par TP à rendre, en respectant le nom de dossier indiqué dans le sujet de TP. Dans ce dossier on trouvera **nécessairement** :

- le dossier `src/` contenant les sources,
- le dossier `test/` contenant les fichiers de test
- le fichier `test4poo.jar`

  NB : il est probable qu'il soit nécessaire de forcer l'ajout de ce fichier archive sur votre dépôt car votre `.ignore` est probablement paramétré pour les rejeter, et c'est normal, cela se fait en ajoutant l'option `-f` (pour « **f**orcer ») lors du `add` : « `git add -f test4poo.jar` »
- le(s) fichier(s) « manifeste » nécessaire(s) à la production du (ou des) jar exécutable(s),
- un fichier texte simple, nommé `readme.md`.

  Ce fichier doit utiliser la mise en forme *markdown* (`.md`). Vous trouverez des éléments sur cette syntaxe, simple mais beaucoup utilisée, dans la zone *Documents* du portail de POO.

  Consultez toujours la mise en forme de votre fichier `readme.md` dans l'interface de votre dépôt git pour la corriger si nécessaire.

  Dans ce fichier `readme.md` vous indiquerez

  - les noms des membres du binômes,
  - un paragraphe présentant le TP et ses objectifs
  - une description le cas échéant de ce qui n'a pas été fait ou qui ne fonctionne pas correctement (en précisant dans quel(s) cas) ou ce qui a été fait en plus dans votre programme par rapport au cahier des charges du TP et toute information complémentaire que vous jugerez utile.
  - les instructions **précises** (vous donnerez des exemples de lignes de commande exactes à chaque fois) indiquant comment :
    * générer et consulter la documentation,
    * compiler les sources du projet,
    * compiler et exécuter les tests,
    * générer le(s) fichier(s) `.jar` **exécutable**(s),
    * exécuter le programme (avec et sans le jar exécutable) en décrivant les éventuels arguments à préciser dans la ligne de commande, avec des exemples précis.

    Pour cette étape, vous pouvez ajouter à votre dépôt un `makefile`. Dans ce cas vous créerez des cibles pour la compilation, la génération de la documentation, la création du jar et l'exécution. Et vous en préciserez l'usage dans le `readme.md`.
- ne déposez pas le dossier `docs/`, ni les fichiers `.class`, ni les fichiers `.jar` qui peuvent être générés par les commandes indiquées (configurez votre fichier `.gitignore`).

Vous nettoierez votre dépôt en en supprimant les fichiers inutiles (`.java~` ou autres) (vous devez configurer correctement le `.gitignore`).

Cela devrait paraître évident et il devrait être inutile de le préciser mais : **vous ne devrez jamais rendre un tp sans avoir vérifier que les différentes étapes que vous présentez dans votre `readme.md` s'exécutent sans problème : compilation, génération des documentations et des tests, exécution du progamme** (vous ne devez par exemple pas avoir de *warning* lors de la génération de la documentation).

«how to» POO

Voici une rapide synthèse des commandes utiles pour les TP de POO.
Toutes ces commandes s'exécutent depuis le dossier du tp `tpX/`.
Vous pouvez créer un fichier `Makefile` qui permet l'exécution de ces commandes.

Structure du dossier pour un TP
Si le dossier `poo/` correspond à votre depôt local git pour l'UE POO.

```
|- poo/
    |- readme.md
    |- .gitignore
    |- tpX/
        |- readme.md
        |- manifest-tpX
        |- test4poo.jar
        |- src/
            |- pack1/
                 |- MyMainClass.java
                 |- SomeClass.java
                 |- ...
                 |- subpack1/
                      |- ...
            |- pack2/
                 |- OtherClass.java
        |- test/
            |- SomeClassTest.java
```

génération de la documentation

```
···/poo/tpX> javadoc -sourcepath src -d docs -subpackages pack1 pack2
```

compilation

```
···/poo/tpX> javac -sourcepath src -d classes src/pack1/MyMainClass.java
```

ou

```
···/poo/tpX> javac -sourcepath src -d classes src/pack1/*.java
```

compilation des tests

```
···/poo/tpX> javac -classpath test4poo.jar test/SomeClassTest.java
```

ou

```
···/poo/tpX> javac -classpath test4poo.jar test/*.java
```

exécution des tests

```
···/poo/tpX> java -jar test4poo.jar SomeClassTest
```

exécution du programme

```
···/poo/tpX> java -classpath classes pack1.MyMainClass
```

création d'un jar exécutable

```
···/poo/tpX> jar cvfe appli.jar pack1.MyMainClass -C classes .
```

ou

```
···/poo/tpX> jar cvfm appli.jar manifest-tpX -C classes .
```

exécution du fichier jar

```
···/poo/tpX> java -jar appli.jar
```