TP1- Garder un historique complet des commits avec rebase et obtenir un commit de merge (no-ff)

On veut dans ce Tp profiter du meilleur des mondes rebase et merge.

Fonctionnellement, nous voulons qu'un commit de merge soit créé pour que l'évènement de merge soit représenté par un commit dédié. C'est un choix fonctionnel qui peut être valable notamment dans le cas de l'implémentation d'un feature.

Etape 1 : préparation de l'environnement

Créer une local repository dans un répertoire de travail

```
git init
```

********** Sur la branch master

Créer un fichier A.txt

```
touch A.txt
git add .
git commit -m "A"
```

Créer un fichier B.txt

```
touch B.txt
git add .
git commit -m "B"
```

A partir de master, créer la branch password_reset

```
git branch password_reset
```

********** La branch password_reset évolue de son côté

Passer sur la branch password_reset

```
git checkout password_reset
```

Créer un fichier C.txt

```
touch C.txt
git add .
git commit -m "C"
```

Créer un fichier D.txt

```
touch D.txt
git add .
git commit -m "D"
```

********** La branch master continue d'évoluer

Passer sur la branche master

```
git checkout master
```

Créer un fichier E.txt

```
touch E.txt
git add .
git commit -m "E"
```

Créer une branche newsletter à partir de la branche master

```
git branch newsletter
```

********** La branch newsletter évolue de son côté

Passer sur la branch newsletter

```
git checkout newsletter
```

Créer un fichier F.txt

```
touch F.txt
git add .
git commit -m "F"
```

Créer un fichier G.txt

```
touch G.txt
git add .
git commit -m "G"
```

********** La branch master continue d'évoluer

Passer sur la branche master

```
git checkout master
```

Créer un fichier H.txt

```
touch H.txt
git add .
git commit -m "H"
```

Créer un fichier I.txt

```
touch I.txt
git add .
git commit -m "I"
```

********** Fin du scenario de préparation de l'environnement

Nous avons terminé la préparation de notre environnement

Vérifier l'historique de notre repository

```
git log --all --graph --oneline --decorate
```

Etape 2 : procéder aux merges

Objectif : conserver l'historique de tous les commits, tout un créant un commit dédié au moment du merge. On utilisera --no-ff (no fast forward) pour volontairement créer le **commit de merge J** (pour la fusion de password_reset dans master) et le **commit de merge K** (pour la fusion de newsletter dans la branch master)

```
A---B---E---H---I-------J-------K ← master
                  \     / \     /
 password_reset →   C---D   F---G   ← newsletter
```

- **Scenario password_reset** : au niveau du commit I, la branche password_reset (qui contient les commit C et D) est fusionnée avec la branche master : cela donnera le commit J dans la branche master
- **Scenario newsletter** : au niveau du commit J, la branche newsletter (qui contient les commit F et G) est fusionnée avec la branche master : cela donnera le commit K dans la branche master

Déroulement du scenario password_reset

```
On se place dans la branch password_reset
```

```
git checkout password_reset
```

```
On applique tous les commits de la branch master à la branch
password_reset
```

```
git rebase master password_reset
```

ou

git rebase master , puisque nous sommes dans la branche password_reset

```
On passe de

A---B---E---H---I ← master
     \
```

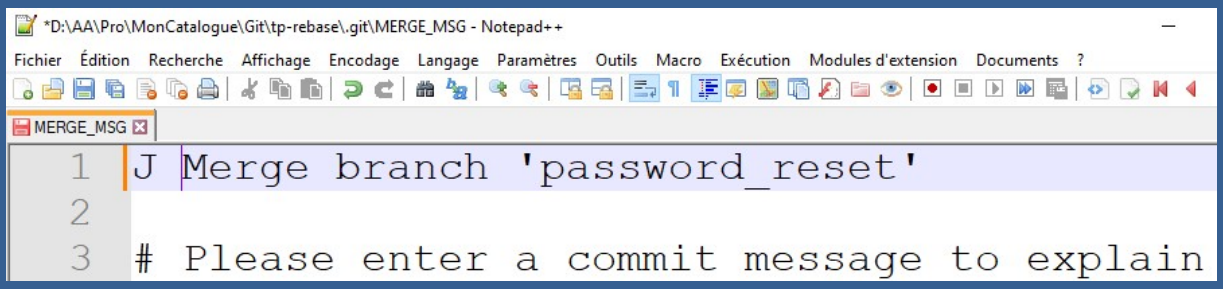
```
      C---D ← password_reset
À

A---B---E---H---I ← master
                \
                 C---D ← password_reset
```

On se place sur la branch master

```
git checkout master
```

On procède au merge

```
git merge password_reset --no-ff
```



Sauvegarder et Fermer

Le merge étant fini , on pourrait supprimer la branche password _reset avec la commande ci-dessous. Cependant, on va la conserver pour plus tard.

```
Si on veut la supprimer :
git branch -d password_reset
```

La branch password_reset a été fusionnée dans master .Vérifier l'historique de notre repository avec

```
git log --all --graph --oneline --decorate
```

Déroulement du scenario newsletter

La branch newsletter (qui contient les commit F et G) est fusionnée avec la branch master au niveau du commit J: cela donnera le commit K dans la branche master

```
On se place dans la branche newsletter
```

```
git checkout newsletter
```

```
 On applique tous les commits de la branch master à la branch
newsletter
```

```
git rebase master newsletter
```

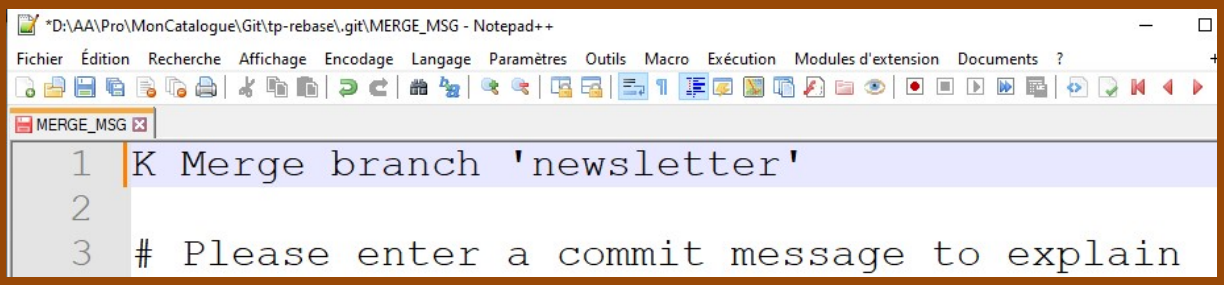
```
On passe de

          F---G ← newsletter
         /
A---B---E---H---I---J ← master

À

                     F---G ← newsletter
                    /
A---B---E---H---I---J ← master
```

```
On se place sur la branch master
```

```
git checkout master
```

```
On procède au merge
```

```
git merge newsletter --no-ff
```



Sauvegarder et fermer

Le merge étant fini, on pourrait supprimer la branche newsletter avec la commande ci-dessous. Cependant, on va la conserver pour plus tard.

```
git branch -d newsletter
```

Dans l'historique, nous retrouverons toujours les réferences des commits effectués dans les branches password_reset et newsletter même si les références à ces dernières n'existent plus.

Vérifier l'historique de notre repository avec

```
git log --all --graph --oneline --decorate
```

Si les branches fusionnées n'ont pas été supprimées, il est possible de voir leur référence dans notre branche master :

```
git checkout master
git branch --merged
```