## Modifica di un progetto Java

Anche in questa esercitazione sei uno sviluppatore che deve effettuare una nuova modifica al progetto aziendale sviluppato in Java aggiungendo una nuova funzionalità a una classe.
Più in dettaglio, i passi da effettuare sono:

1. Creare una *fork* del progetto https://github.com/carmelo-cina-sal/Mesi su GitHub. Se l'operazione è stata fatta correttamente, dovrai trovarti il progetto sul tuo repository.
2. Effettuare la *git clone* della nuova repository sul tuo PC locale, sotto *c:\Utenti\studente\PortableGit\repository*.
3. Il progetto contiene due file Java, File1.java e File2.java. Ognuno di questi file allo stato attuale vi chiede di inserire un numero e in output non restituiscono nulla. Il progetto finale invece prevede che questi due file restituiscano, dato un numero in ingresso, il mese corrispondente, es. se inserisco 1 mi viene restituito "Gennaio", 2 "Febbraio" ecc. Ti viene richiesto di implementare il metodo di ***File1.java*** che restituisce il nome del mese corrispondente alla tua posizione nel registro. Se sei nella posizione 1, dovrai solo implementare il codice che restituisce la scritta "Gennaio", così via fino a 12.
   Chi si trova dalla posizione da 13 a 23 ricomincia daccapo, ossia 13 dovrà far restituire la scritta "Gennaio" ma questa volta lavorando su ***File2.java***.
4. Correggere eventuali bug.
5. Per poter aggiungere tali modifiche, sarà necessario utilizzare il comando *git add*.
6. Al fine di allineare il tuo repository remoto, effettua la *commit* sul tuo repository locale e infine effettua la *push* in remoto.
7. Su GitHub, dalla copia del tuo repository, effettua la Pull request al progetto principale. ***Nel commento della Pull request, incolla i comandi inseriti sulla bash di git***. Sei non rei arrivato a fare la pull request, incolla in un file di testo e invialo a carmelo.cina@iistommasosalvini.edu.it.

Di seguito uno schema di riepilogo dei comandi principali di Git.

| | |
|---|---|
| **Configurazione globale**<br>Configurazione dell'utente valida per tutti i repository<br><br>`$ git config --global user.name "[name]"`<br>Imposta il nome che vuoi mostrare sulle tue commit<br><br>`$ git config --global user.email "[email address]"`<br>Imposta l'email che vuoi mostrare sulle tue commit | **Creare repository**<br>Crea un nuovo repository o clonane uno esistente da un URL<br><br>`$ git init [project-name]`<br>Crea un nuovo repository locale con il nome specificato<br><br>`$ git clone [url]`<br>Scarica un progetto esistente e il suo storico di cambiamenti |
| **Effettuare modifiche**<br>Rivedi i cambiamenti al codice e prepara una commit<br><br>`$ git status`<br>Elenca tutti i file nuovi o modificati<br><br>`$ git diff`<br>Mostra le differenze non ancora nell'area di staging<br><br>`$ git add [file]`<br>Crea uno snapshot del file in preparazione al versioning<br><br>`$ git diff --staged`<br>Mostra le differenze tra staging e ultima modifica<br><br>`$ git reset [file]` | **Rivedere lo storico**<br>Esplora l'evoluzione dei file del progetto<br><br>`$ git log`<br>Elenca lo storico di versione per il branch corrente<br><br>`$ git log --follow [file]`<br>Elenca lo storico di versione per il file specificato, incluse rinominazioni<br><br>`$ git diff [first-branch]...[second-branch]`<br>Mostra la differenza tra due branch<br><br>`$ git show [commit]`<br>Mostra i metadati e i cambiamenti della commit specificata |

| | |
|---|---|
| Rimuovi un file dall'area di staging, ma mantieni le modifiche<br><br>`$ git commit -m "[descriptive message]"`<br>Salva gli snapshot dei file in maniera permanente nello storico | |
| **Annullare commit**<br>Elimina errori e altera lo storico dei cambiamenti<br><br>`$ git reset [commit]`<br>Annulla tutte le commit effettuate dopo [commit], preservando i cambiamenti locali<br><br>`$ git reset --hard [commit]`<br>Elimina tutto lo storico e i cambiamenti fino alla commit specificata | **Sincronizzare i cambiamenti**<br>Collegati a un URL remoto e ottieni lo storico dei cambiamenti<br><br>`$ git fetch [remote]`<br>Scarica lo storico dei cambiamenti dal repository remoto<br><br>`$ git merge [remote]/[branch]`<br>Unisci il branch remoto con quello locale<br><br>`$ git push [remote] [branch]`<br>Carica tutti i cambiamenti dal branch locale su GitHub<br><br>`$ git pull`<br>Scarica lo storico e unisci i cambiamenti |