## Modifica di un progetto Java

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

1. Creare una *fork* del progetto https://github.com/carmelo-cina-sal/Mesi sul tuo account 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. Spostarsi sotto la directory del nuovo progetto appena clonato.
4. 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 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 sul registro si trova da 1 a 12, dovrà farlo sul file **File1.java**.
   Chi si trova dalla posizione da 13 a 23 lavora sul file **File2.java** sottraendo 12 alla propria posizione, ossia 13 (13-12=1) dovrà far restituire la scritta "Gennaio", 14 lavorerà per far restituire "Febbraio" (14-12 =2) ecc.
5. Per poter aggiungere tali modifiche al repository, 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 selezionando il branch *NomeCognome* invece del branch main. ***Nel titolo della Pull request metti il tuo Nome e Cognome, nel commento incolla i comandi inseriti sulla bash di git***. Sei non sei arrivato a fare la pull request, incolla i comandi in un file di testo e invialo a carmelo.cina@iistommasosalvini.edu.it giuseppe.schiavone@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 | **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]` |

`$ git add [file]`
Crea uno snapshot del file in preparazione al versioning

`$ git diff --staged`
Mostra le differenze tra staging e ultima modifica

`$ git reset [file]`
Rimuovi un file dall'area di staging, ma mantieni le modifiche

`$ git commit -m "[descriptive message]"`
Salva gli snapshot dei file in maniera permanente nello storico

Elenca lo storico di versione per il file specificato, incluse rinominazioni

`$ git diff [first-branch]...[second-branch]`
Mostra la differenza tra due branch

`$ git show [commit]`
Mostra i metadati e i cambiamenti della commit specificata

**Annullare commit**
*Elimina errori e altera lo storico dei cambiamenti*

`$ git reset [commit]`
Annulla tutte le commit effettuate dopo [commit], preservando i cambiamenti locali

`$ git reset --hard [commit]`
Elimina tutto lo storico e i cambiamenti fino alla commit specificata

`$ git restore [file]`
Allinea il file locale al repository

**Sincronizzare i cambiamenti**
Collegati a un URL remoto e ottieni lo storico dei cambiamenti

`$ git fetch [remote]`
Scarica lo storico dei cambiamenti dal repository remoto

`$ git merge [remote]/[branch]`
Unisci il branch remoto con quello locale

`$ git push --set-upstream [remote] [branch]`
Carica tutti i cambiamenti dal branch locale su GitHub

`$ git pull`
Scarica lo storico e unisci i cambiamenti

`$ git branch [branch-name]`
Crea un nuovo branch