4. **Git Rebase & Git Merge**  
The different within 'git rebase' and 'git merge' is

- Git Rebase -> Git rebase actually reapplies our commits on top of another base branch, it doesnt merge histories commits etc.

- Git Merge -> Git merge actually joins two or more current development commits OR histories commits on the branch together into one.

Example Merge & Rebase:

>Niki completed her feature/voicecall

>Jack completed his feature/videocall

>There would be a master branch.

[MERGE]

1. Niki merge his branch into master

2. Jack want to keep all history on master branches.

3. Jack use merge method by doing recursive strategy.

3. Jack making sure his local and remote master branches are in sync

4. Jack 'git checkout master' Then 'git pull' Master branch.

5. Jack 'git checkout feature/videocall' branch then Merge into master keeping all commits history.

[REBASE]

1. Niki merge his branch into master

2. Jack want to re-write history on master branches

3. Jack making sure his local and remote master branches are in sync

4. Jack 'git checkout master' Then 'git pull' Master branch.

5. Jack 'git rebase master feature/videocall' branch then Git will rewind and reapply the commits

6. **Git Graph**

Git Graph

Available Branch

> Master

> Hotfix

> Release

> Develop

> Feature

> Feature

Storyline

Vito, Niki and Jack are in the same Team, which is Team A.

>Team A asked to continue an application development

>Team A start at Master on version v0.1

>Team A Initiate a Develop Branch

--> Niki initiate a hotfix branch to fix bug in the mean time

-> Vito initiate a feature branch to start development

-> Jack initiate a feature branch to start development

--> Niki finish the hotfix, her hotfix branch up to develop branch and master branch and she created new version of master after she ask for merge a pull request

-> Vito and Jack merged their branch into new branch after taking newest commits from develop branch

-> Develop branch up to release branch

--> release branch has new commits and updated

-> Master branch updated with release update with new version v1.0

-> Development Continue using release branch.

Class, Interface and Abstract  
  
Class:

A class is a blue-print prototype model.

A class would have Modifiers, class name, superclass / interface (if there is any), constructors and Method

Interface:

an interface is an abstract type that tells the compiler which property names a given object can have

Abstract Class

Abstract class is a class that doesnt have any method and body.  
  
Text

Description automatically generated

The rest of the code is on ZIP as Code.