Resolving Merge Conflicts

A merge conflict refers to the event of attempting to merge branches which each have commits which are conflicting because they are trying to change the same code in different ways. Simple conflicts can be handled by Git on its own but oftentimes human intervention is required to resolve conflicts. If these conflicts are not fixed, Git will not allow the merging of the pull request. There are multiple ways of resolving conflicts. For example, developers can coordinate on their own and decide which branch's code should be used and they can rewrite the code in the other branch to coordinate. This can be handled by manually rewriting and modifying a branch or it can be handled right in Github online where the interface will prompt you to select which branch's codc you wish to use. Git also offers merge tools which will essentially accomplish the same thing as previously mentioned but on a command line interface as opposed to on Github's website. Using the command "git mergetool --tool=emerge" will show three windows, the branches being merged with conflicts and the branch (typically in this case this will be the master or main branch) which the other branches will be merged into. The windows will show you where the conflicts are and you can select which branch you wish to use in each of the conflicting areas by typing A or B. Merge conflicts are practically bound to happen in development teams with multiple developers working on multiple branches trying to merge into one main branch. Being able to easily detect and fix merge conflicts is essential and why Git's merge tools are so useful.

Below is a diagram which shows the process of creating a pull request with two branches into a main branch, solving conflicts, then merging.