گیت

۱ - از git stash command زمانی استفاده می‌کنیم که بخواهیم وضعیت فعلی تغییرات را ذخیره نماییم. در واقع در روند کار هنوز به جایی نرسیده‌ایم که از git commit –m استفاده نماییم، اما می‌خواهیم تغییرات اعمال شده تا اینجا را ذخیره کرده تا در فرصت بعدی آن را تکمیل نماییم. با استفاده از این دستور، اصلاحات مدنظر ما در جایی دیگر ذخیره می‌گردد و هنوز HEAD به commit قبلی اشاره می‌کند. با استفاده از git stash list می‌توانیم به لیست stash های قبلی دست یابیم. با استفاده از دستور git stash show می‌توانیم وضعیت فایل در state مدنظرمان را ببینیم و برای تغییرات موقتی می‌توانیم از git stash apply استفاده نماییم.

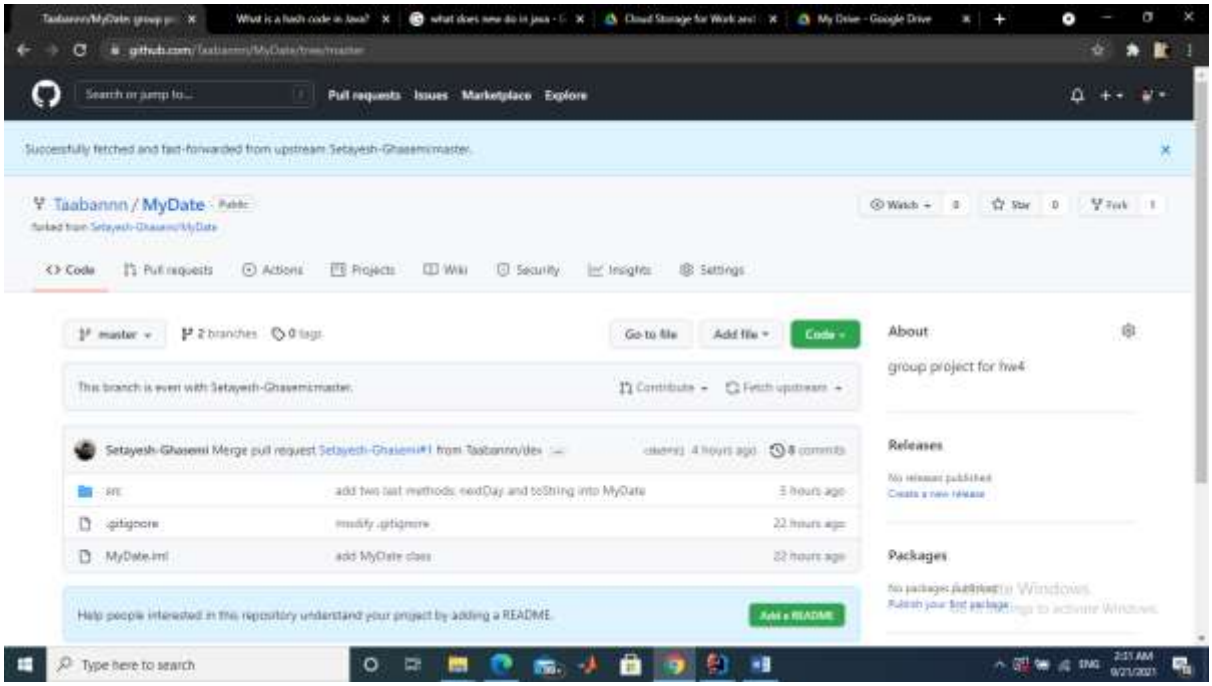
۲ – با استفاده از git checkout و با داشتن شناسه‌ی کامیت، می‌توانیم تغییرات کامیت مدنظر را مشاهده نماییم. (محدودیتی نیز در انتخاب کامیت‌ها وجود ندارد.) با استفاده از دستور git revert مقدار HEAD به commit یکی مانده به آخر در تاریخچه ی کامیت ها اشاره می‌کند. با استفاده از دستور git reset –hard HEAD~۱ تمام تغییرات ایجاد شده تو سط آخرین کامیت به طور کلی از بین می رود ودر این حالت HEAD به کامیت یکی مانده به اخری اشاره می‌کند، در حالی که با استفاده از دستور git reset –soft HEAD~۱ تغییرات ایجاد شده تو سط آخرین کامیت حفظ می شود ودر این حالت نیز HEAD به کامیت یکی مانده به اخری اشاره می‌کند.

۳ – با استفاده از دستور git rebase در صورتیکه نام شاخه‌ای در جلوی آن ذکر شود، به طور اتوماتیک قبل از انجام کار دیگری به آن شاخه switch می‌کند و در غیر اینصورت در همان شاخه‌ی فعلی باقی می‌ماند. در صورتی که عملیات merge شاخه‌ها موفقیت آمیز نباشد با استفاده از دستور git rebase --continue می توان مشکل را حل نمود یا با استفاده از دستور git rebase --skip می توان کامیتی را که با شکست مواجه شده bypass نمود و یا با استفاده از git rebase --abort می توان فایل‌های git/rebase. را پاک نمود و با استفاده از git rebase --onto می‌توانیم تغییرات دلخواه را در درخت نمایش دهنده‌ی گیت ایجاد نماییم.

۴– در واقع git workflow یک دستورالعمل یا توصیه نامه برای چگونگی استفاده از گیت به منظور افزایش بازدهی (و یکدست بودن تغییرات ایجاد شده توسط کسانی که به گیت برای ویرایش فایلها دسترسی دارند.) است. به عنوان مثال یه بخش مرکزی برای merge نمودن شاخه‌های فرعی با شاخه‌ی اصلی ایجاد می‌نماییم. اینگونه تمامی کاربران به یک ورژن از فایلها دسترسی دارند و بعد از اعمال تغییرات توسط بخش مرکزی هربار آن را آپدیت می‌کنند. این گونه conflict ها را راحت‌تر می توان مدیریت نمود زیرا هر کاربر برای merge کردن شاخه‌ی کاری خود با شاخه‌ی اصلی مجبور است تا هربار برای merge نمودن به

بخش مرکزی یک pull request بفرستد و پس از بررسی‌های انجام شده توسط بخش مرکزی آن شاخه‌ی فرعی با شاخه‌ی اصلی merge می‌شود.

۵ – در شکل ۱ نتیجه‌ی کار و repository مشترک را می‌بینید. (در این کلاس پیاده‌سازی توابع toString و nextDay با من بوده است.)



شکل ۱