دستور کار ۳

objectives

- Branches in git
- Vim editor
- Encapsulation in java

branches

- In programming we usually divide problems into subproblems.
- It is easier to solve each subproblem in a different branch
- A branch represents an independent line of development.
- each person usually has his own branch
- after development branches get merged

common git branch commands

- `git branch`
  - to list the (local) branches
- `git branch <branch>`
  - create a branch but do not checkout to it
- `git branch -D <branch>`
  - Force delete the specified branch, even if it has unmerged changes. This is the command to use if you want to permanently throw away all of the commits associated with a particular line of development.
- `git branch -d <branch`
  - Delete the specified branch. This is a “safe” operation in that Git prevents you from deleting the branch if it has unmerged changes.

common git branch commands

when deleting a branch, the local branch will be deleted but the branch on the remote repository will remain. to interact with the remote branches use the commands below:

- `git push origin  --delete <branch>`

or

- `git push origin :<branch>`

common git commands

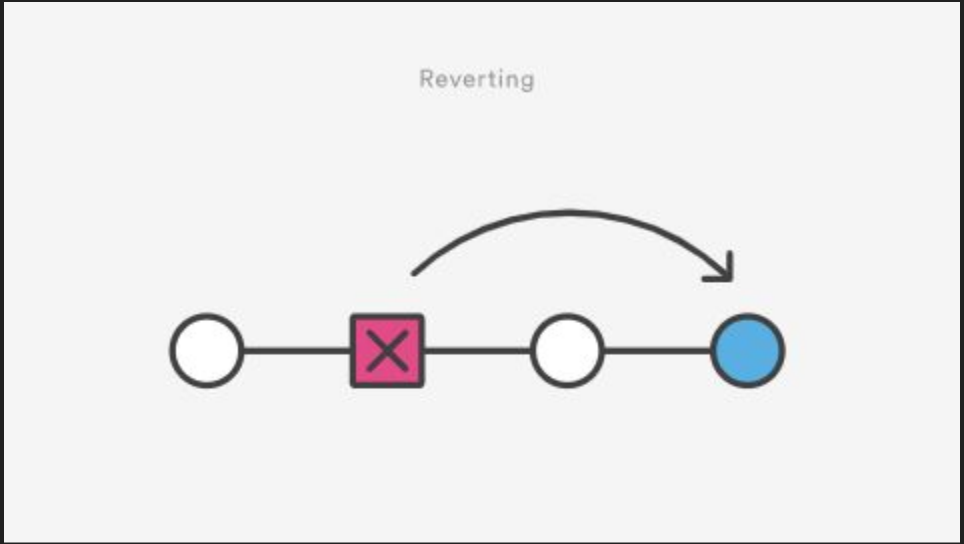
- `git branch -m <branch>`
  - Rename the current branch to <branch>.
- `git branch -a`
  - List all local and remote branches.
-

vim editor

vim has 3 modes

- **normal mode**
  - This is the default mode when you open Vim.
  - In normal mode, you can navigate through the text, perform editing commands, and execute various operations.
  - It is primarily used for moving the cursor, searching, copying, pasting, deleting, and executing commands.
  - To enter normal mode from other modes, press the Esc key.
- **insert mode**
  - In insert mode, you can enter and edit text.
  - It is similar to traditional text editors, where you can directly type and modify the content.
  - To enter insert mode from normal mode, press the 'i' key.
  - To exit insert mode and return to normal mode, press the Esc key.
- **visual mode (we will not use this)**
  - Visual mode allows you to select and manipulate blocks of text.
  - To enter visual mode from normal mode, press the 'v' key.
  - To exit visual mode and return to normal mode, press the Esc key.

git revert



git revert

- Sometimes you do something wrong with a commit, for example you fix a bug but that turns out to produce some other bugs in the code. so you decide to remove any changes done in that specific commit.
- you use the `git revert` command.
- here are the steps:
    - use `git log --oneline` to view all the commits on the branch
    - next select the hash of the commit you want to revert changes. for example in `ef7ef22 add a.txt` , `ef7ef22` is the hash of the commit.
    - if no conflicts occur, you should revert the changes
    - sometimes you need to handle conflicts, so open the files that have conflicts, solve them and then commit.

git amend

- useful when editing the last commit
- both commit message and actions
- steps:
  - edit the files you want
  - add them using `git add .`
  - commit the changes using the following command:
    - `git commit --amend -m ‘message’`
  - this command modifies the history, replacing the latest commit with the new commit

Encapsulation

- assuming you know about classes, objects, methods and constructors
- every class has a default constructor that is an empty constructor
- Encapsulation ensures that the internal state of an object is accessible and modifiable only through controlled interfaces, typically in the form of public methods or properties. These methods, often referred to as accessors and mutators or getters and setters, allow the external code to interact with the object in a controlled manner.

Encapsulation

```
public class Person{
      private String name;
      private String id;
      private int[] grades;
      public Person(String id, String name) {
            this.id = id;
            this.name = name;
            this.grades = new int[10];
      }
      public void sayHello() {
            System.out.println("Hello World!");
      }
      public int getName() {
            return name;
      }

}
```

Encapsulation

```
Person p1 = new Person("18271", "Ali");

Person p2 = new Person("84178", "Mahdi");

String p1Name = p1.getName();

System.out.println(p1Name);

String p2Name = p2.getName();

System.out.println(p2.getName());
```

Let’s Code

what to do?

1. see manual 3 for instructions
2. after you're finished, commit the changes with the message ‘finished manual 3’
3. then edit the last commit with git amend and commit the changes with the message ‘modified last commit’
4. the history of the commands will be evaluated

good luck