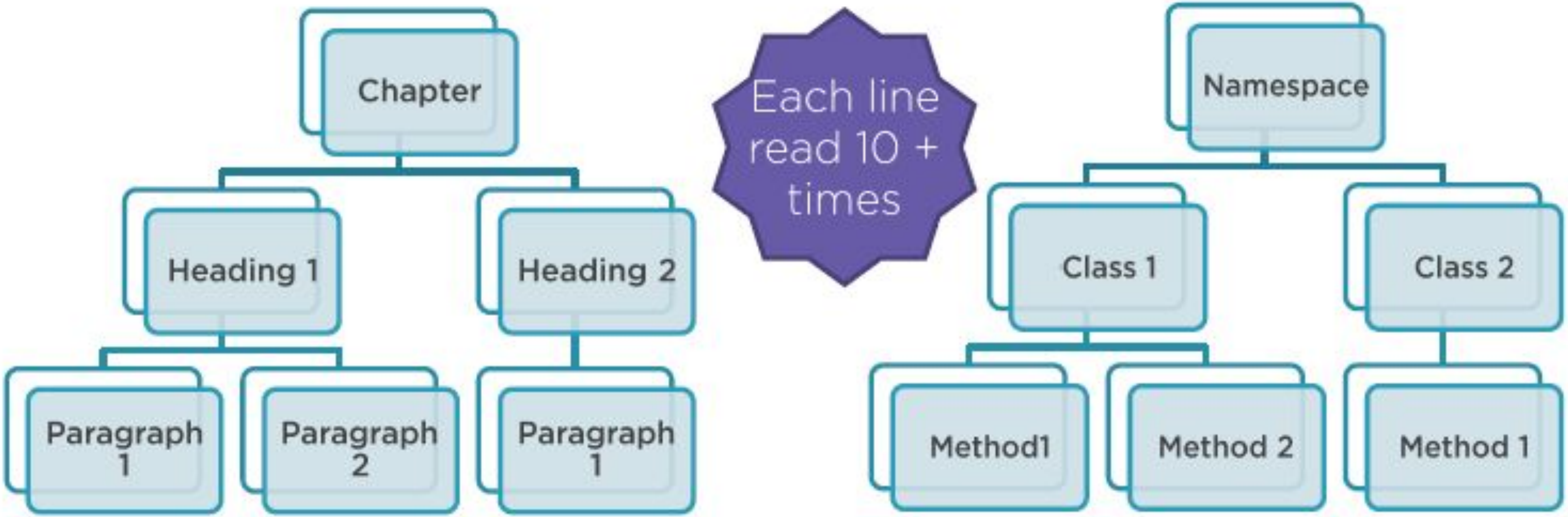
Code, git, github



Chapter

Heading 1

Heading 2

Paragraph 1

Paragraph 2

Paragraph 1

Each line read 10 + times

Namespace

Class 1

Class 2

Method1

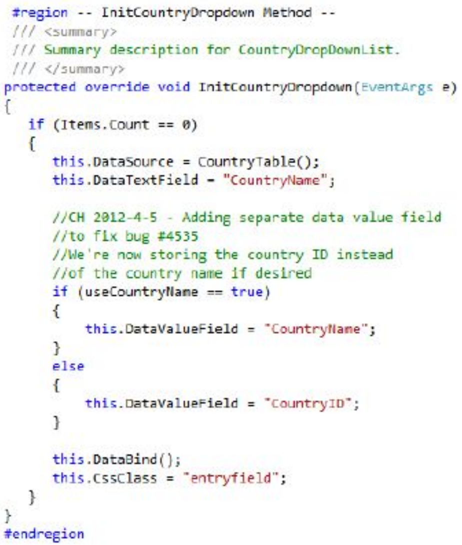
Method 2

Method 1

Avoid

- Unnecessary comments
- Poorly named structures
- Huge classes
- Long methods
- Repetition

🚫

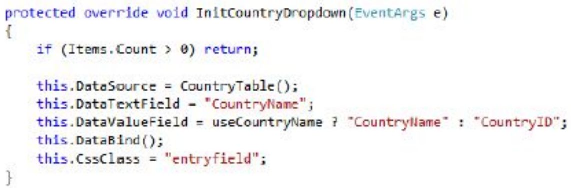


```
#region -- InitCountryDropdown Method --
 /// <summary>
 /// Summary description for CountryDropDownList.
 /// </summary>
protected override void InitCountryDropdown(EventArgs e)
{
    if (Items.Count == 0)
    {
        this.DataSource = CountryTable();
        this.DataTextField = "CountryName";

        //CH 2012-4-5 - Adding separate data value field
        //to fix bug #4535
        //We're now storing the country ID instead
        //of the country name if desired
        if (useCountryName == true)
        {
            this.DataValueField = "CountryName";
        }
        else
        {
            this.DataValueField = "CountryID";
        }

        this.DataBind();
        this.CssClass = "entryfield";
    }
}
#endregion
```

✓



```
protected override void InitCountryDropdown(EventArgs e)
{
    if (Items.Count > 0) return;

    this.DataSource = CountryTable();
    this.DataTextField = "CountryName";
    this.DataValueField = useCountryName ? "CountryName" : "CountryID";
    this.DataBind();
    this.CssClass = "entryfield";
}
```

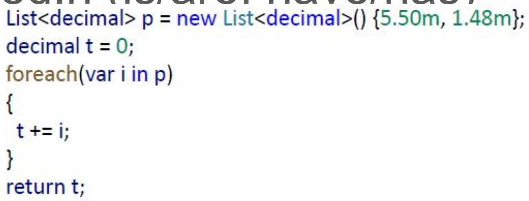
Goal: Density. More in each "eye full"

Naming

- Classes - nouns, single responsibility
- Methods - GetRegisterdUser, IsValidSubscription, ImportDocuments, SendEmail
- Avoid side effects for methods (it only checks, gets, delete, but not everything together)
- Booleans - isOpen, done, isactive, hasLoggedIn (is/are, have/has)
- Avoid using and, if, of in method name
- Don’t use abbreviations
- Be positive

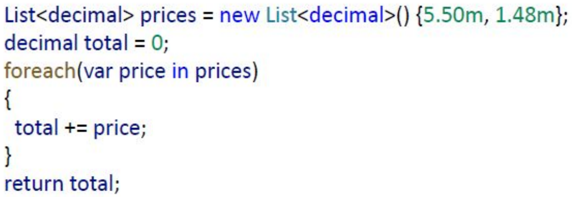
(isLoggedIn vs isNotLoggedIn)

🚫



```
List<decimal> p = new List<decimal>() {5.50m, 1.48m};
decimal t = 0;
foreach(var i in p)
{
  t += i;
}
return t;
```

✓



```
List<decimal> prices = new List<decimal>() {5.50m, 1.48m};
decimal total = 0;
foreach(var price in prices)
{
  total += price;
}
return total;
```

Classes, methods

🚫

WebsiteBO

Utility

Common

MyFunctions

✓

User

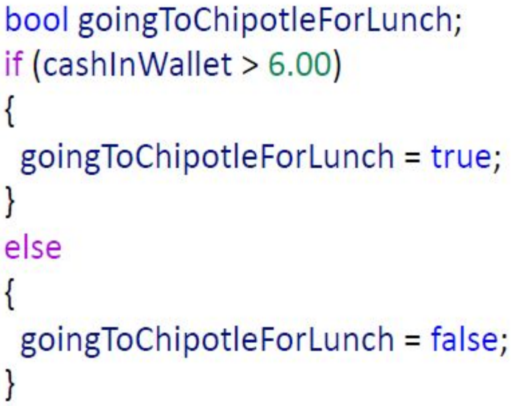
Account

QueryBuilder

ProductRepository

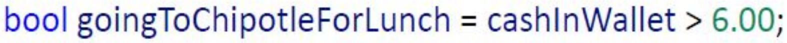
Assign booleans implicitly, ternary operations

🚫



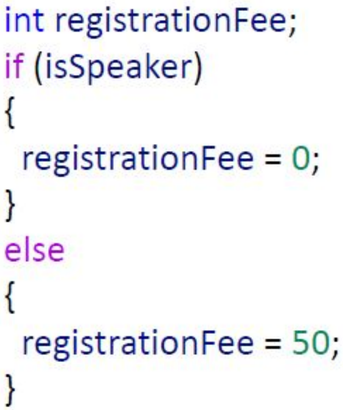
```
bool goingToChipotleForLunch;
if (cashInWallet > 6.00)
{
 goingToChipotleForLunch = true;
}
else
{
 goingToChipotleForLunch = false;
}
```

✓



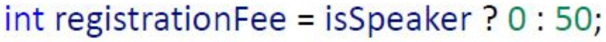
```
bool goingToChipotleForLunch = cashInWallet > 6.00;
```

🚫



```
int registrationFee;
if (isSpeaker)
{
 registrationFee = 0;
}
else
{
 registrationFee = 50;
}
```

✓



```
int registrationFee = isSpeaker ? 0 : 50;
```

Magic Numbers

🚫

```
if (age > 21)
{
 // body here
}
```

✓

```
const int legalDrinkingAge = 21;

if (age > legalDrinkingAge)
{
  // body here
}
```

🚫

```
if (status == 2)
{
 // body here
}
```

✓

```
if (status == Status.active)
{
  // body here
}
```

🚫

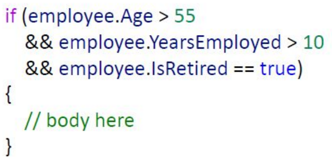
```
if (employeeType == "manager")
```

✓

```
if (employee.type == EmployeeType.Manager)
```

Intermediate Variables

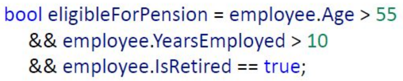
🚫



```
if (employee.Age > 55
  && employee.YearsEmployed > 10
  && employee.IsRetired == true)
{
  // body here
}
```

← What question is this asking?

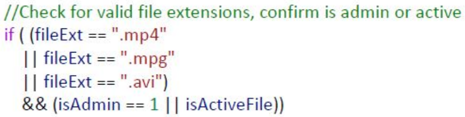
✓



```
bool eligibleForPension = employee.Age > 55
    && employee.YearsEmployed > 10
    && employee.IsRetired == true;
```

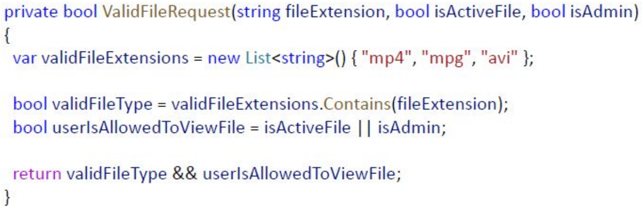
Encapsulate Complex Conditionals

🚫



```
//Check for valid file extensions, confirm is admin or active
if ( (fileExt == ".mp4"
   || fileExt == ".mpg"
   || fileExt == ".avi")
  && (isAdmin == 1 || isActiveFile))
```

✓



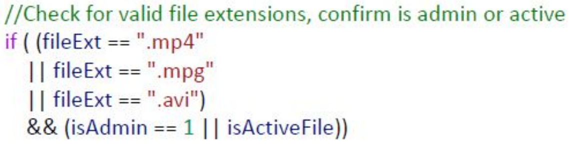
```
private bool ValidFileRequest(string fileExtension, bool isActiveFile, bool isAdmin)
{
 var validFileExtensions = new List<string>() { "mp4", "mpg", "avi" };

 bool validFileType = validFileExtensions.Contains(fileExtension);
 bool userIsAllowedToViewFile = isActiveFile || isAdmin;

 return validFileType && userIsAllowedToViewFile;
}
```

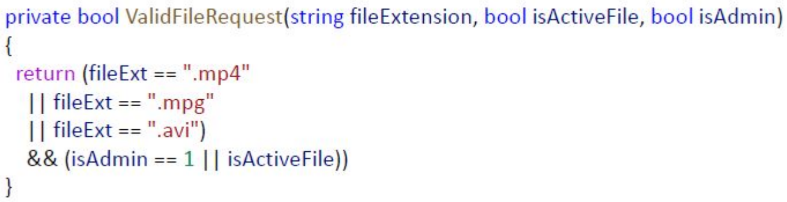
Encapsulate Complex Conditionals

🚫



```
//Check for valid file extensions, confirm is admin or active
if ( (fileExt == ".mp4"
   || fileExt == ".mpg"
   || fileExt == ".avi")
  && (isAdmin == 1 || isActiveFile))
```

✓



```
private bool ValidFileRequest(string fileExtension, bool isActiveFile, bool isAdmin)
{
 return (fileExt == ".mp4"
   || fileExt == ".mpg"
   || fileExt == ".avi")
  && (isAdmin == 1 || isActiveFile))
}
```

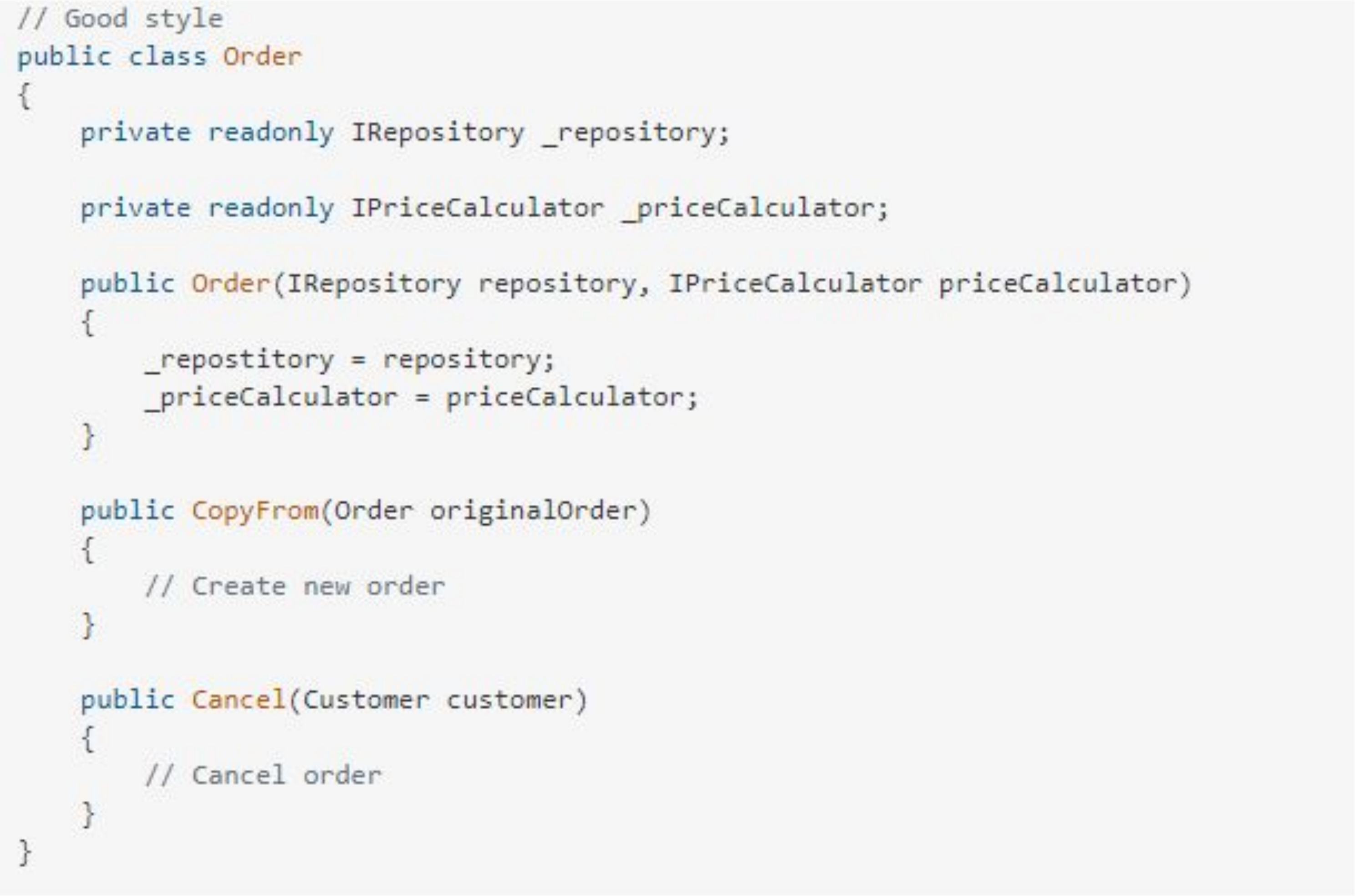
Before

```
if
  if
   while
     do
     some
     complicated
     thing
   end while
  end if
end if
```

After

```
if
  if
     doComplicatedThing()
  end if
end if
```

```
doComplicatedThing()
{
  while
    do some complicated thing
   end while
}
```



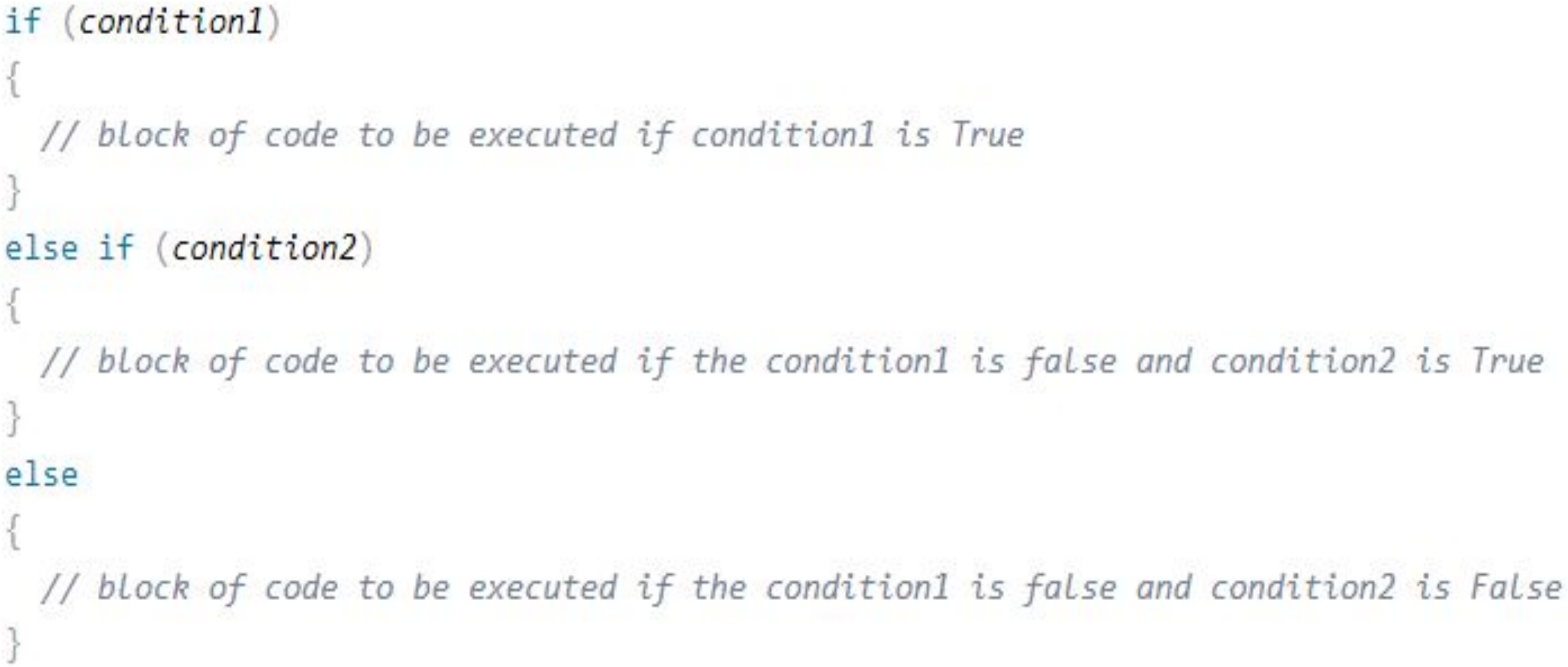
```
// Good style
public class Order
{
    private readonly IRepository _repository;

    private readonly IPriceCalculator _priceCalculator;

    public Order(IRepository repository, IPriceCalculator priceCalculator)
    {
        _repostitory = repository;
        _priceCalculator = priceCalculator;
    }

    public CopyFrom(Order originalOrder)
    {
        // Create new order
    }

    public Cancel(Customer customer)
    {
        // Cancel order
    }
}
```



```
if (condition1)
{
  // block of code to be executed if condition1 is True
}
else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is True
}
else
{
  // block of code to be executed if the condition1 is false and condition2 is False
}
```

.gitignore

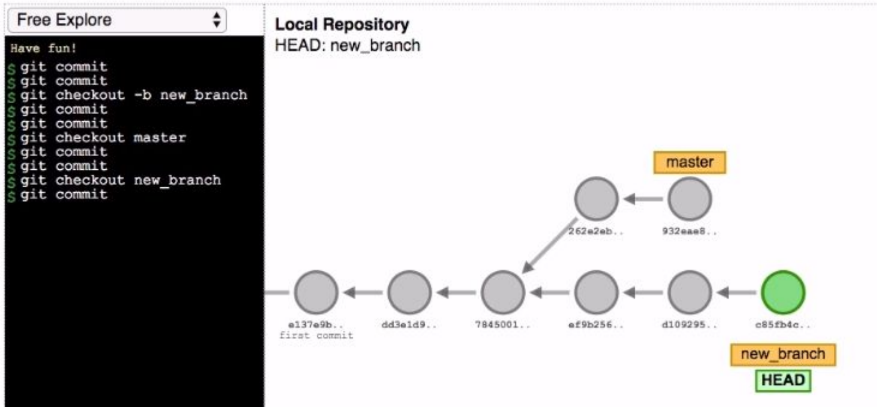
gitignore.io

Do it yourself

- Create a new repository
- Add .gitignore (gitignore.io)
- Create a new branch from main
- Add some files to the branch (commit, push)
- Create a pull request to main
- Merge the new branch with main
- Pull (fetch) newest main
- Create a new branch from main

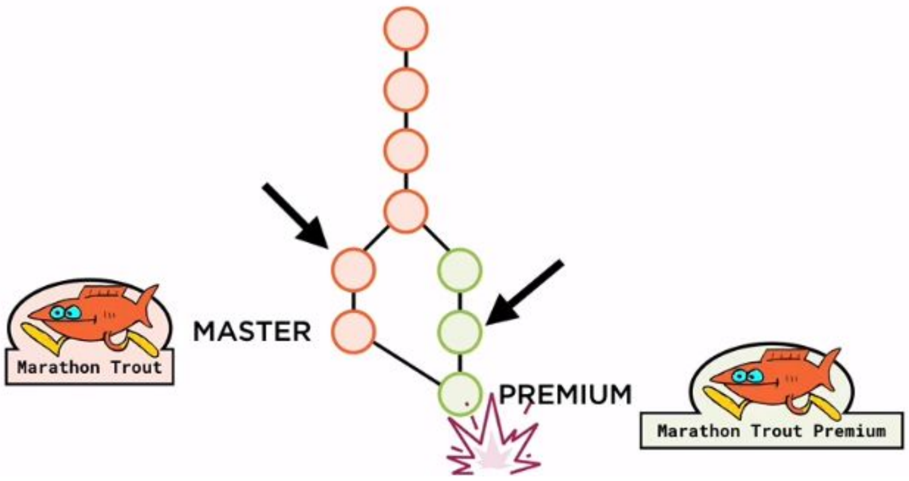
Git - version control system

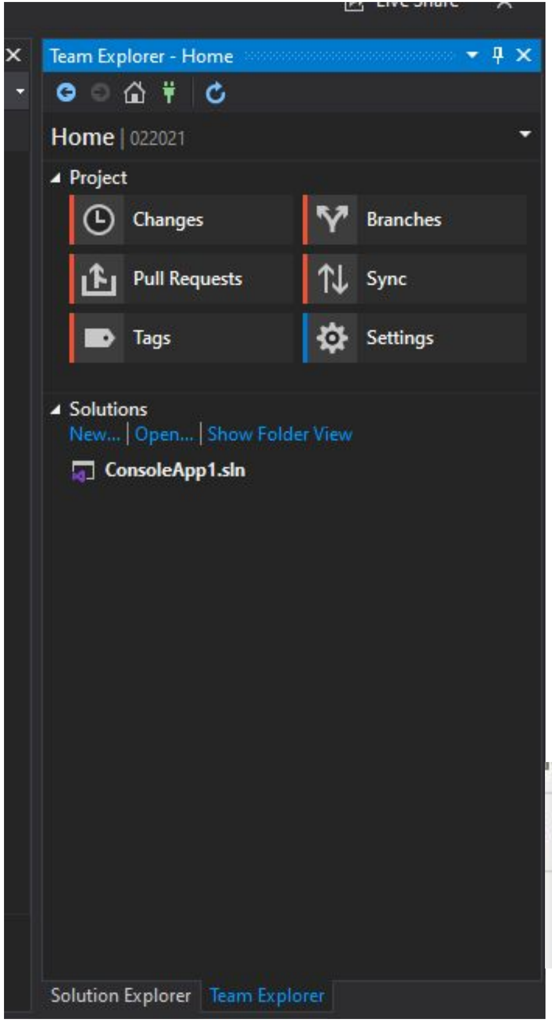
- Sharing code with others
- History tracking
- https://git-school.github.io/visualizing-git/

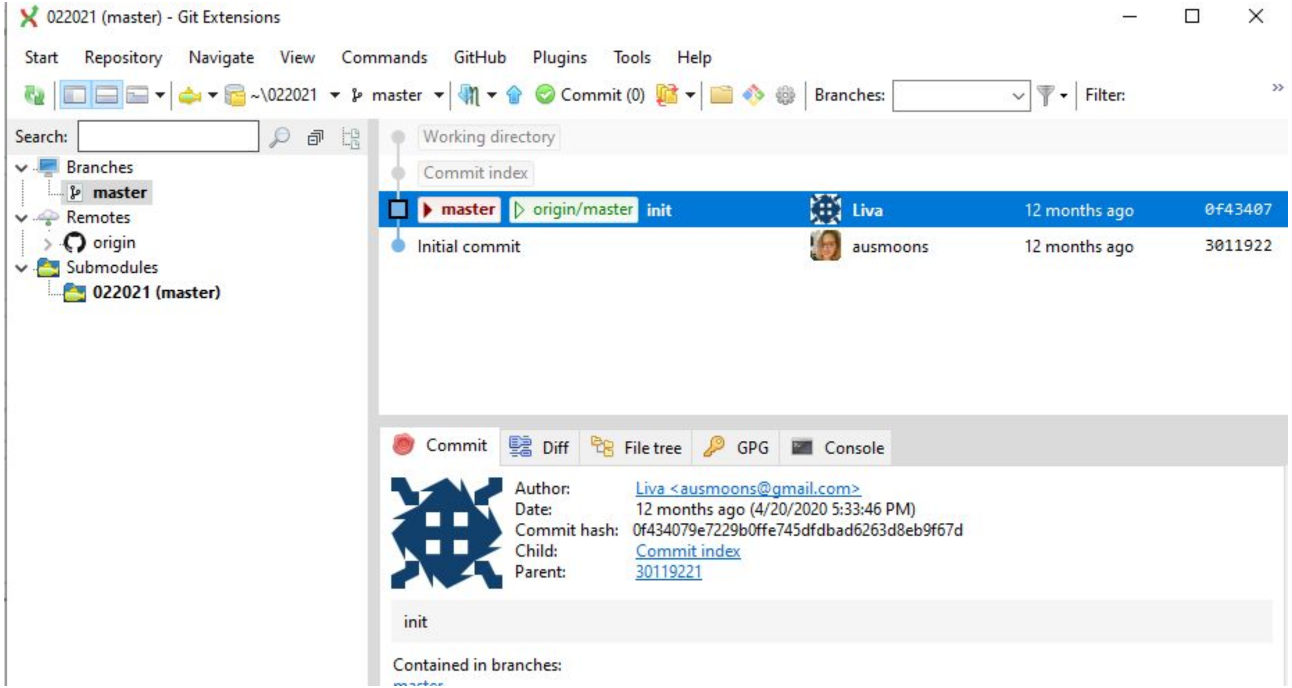


```
Have fun!
$ git commit
$ git commit
$ git checkout -b new_branch
$ git commit
$ git commit
$ git checkout master
$ git commit
$ git commit
$ git checkout new_branch
$ git commit
```

Local Repository
HEAD: new_branch





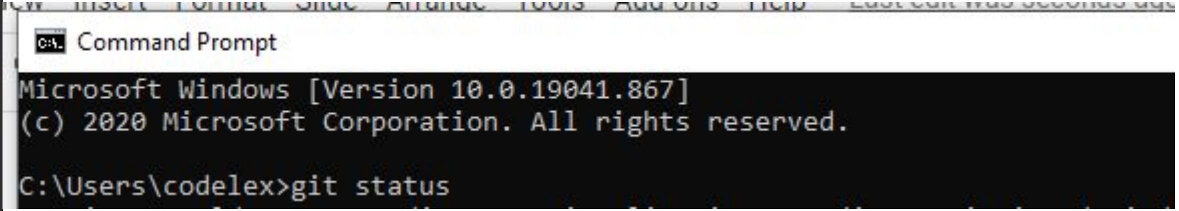


master origin/master init		Liva	12 months ago	0f43407
Initial commit		ausmoons	12 months ago	3011922

Author: Liva <ausmoons@gmail.com>
Date: 12 months ago (4/20/2020 5:33:46 PM)
Commit hash: 0f434079e7229b0ffe745dfdbad6263d8eb9f67d
Child: Commit index
Parent: 30119221

init

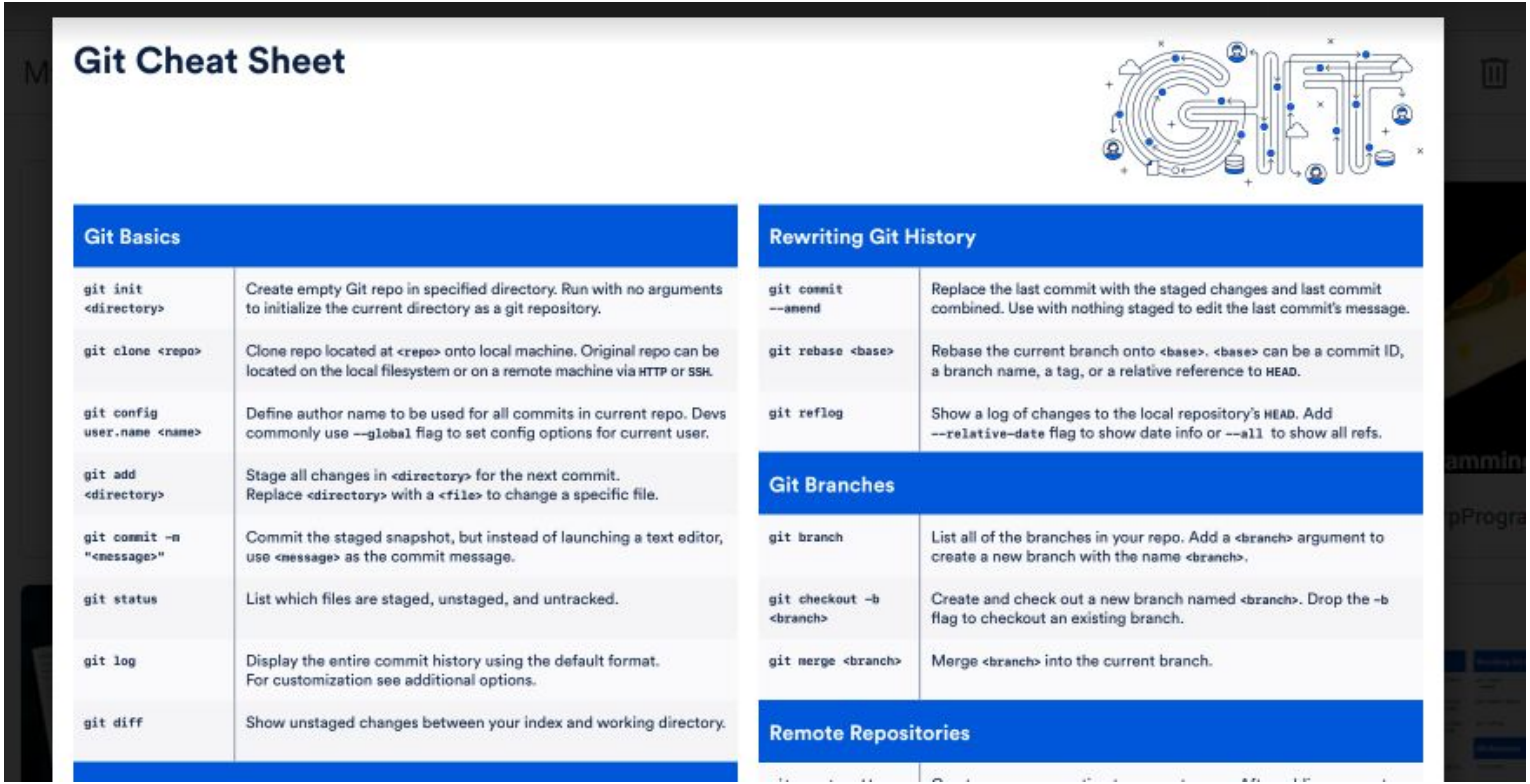
Contained in branches:



```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\codelex>git status
```

https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet



Git Cheat Sheet

Git Basics

`git init <directory>`	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
`git clone <repo>`	Clone repo located at `<repo>` onto local machine. Original repo can be located on the local filesystem or on a remote machine via `HTTP` or `SSH`.
`git config user.name <name>`	Define author name to be used for all commits in current repo. Devs commonly use `--global` flag to set config options for current user.
`git add <directory>`	Stage all changes in `<directory>` for the next commit. Replace `<directory>` with a `<file>` to change a specific file.
`git commit -m "<message>"`	Commit the staged snapshot, but instead of launching a text editor, use `<message>` as the commit message.
`git status`	List which files are staged, unstaged, and untracked.
`git log`	Display the entire commit history using the default format. For customization see additional options.
`git diff`	Show unstaged changes between your index and working directory.

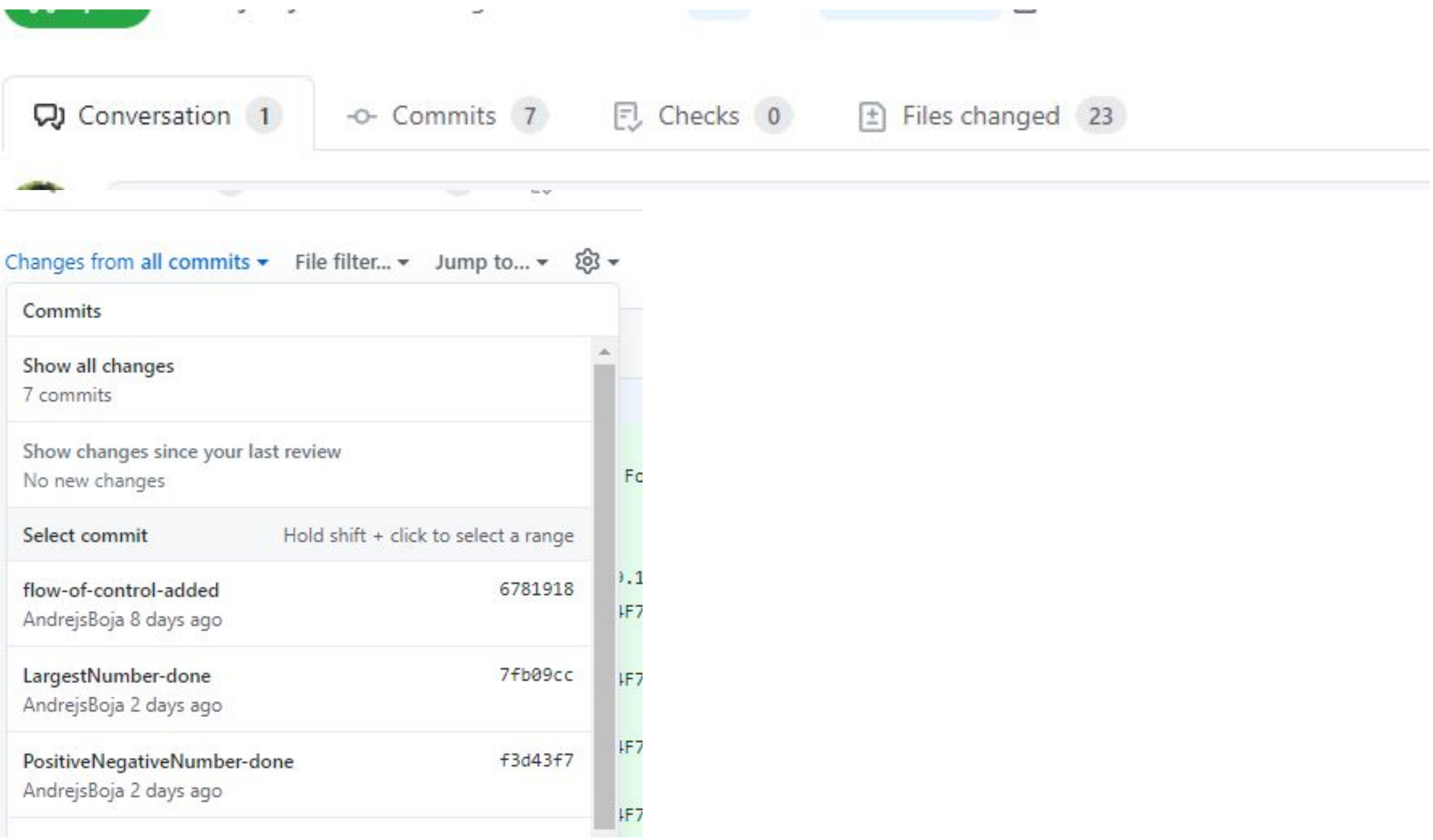
Rewriting Git History

`git commit --amend`	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
`git rebase <base>`	Rebase the current branch onto `<base>`. `<base>` can be a commit ID, a branch name, a tag, or a relative reference to `HEAD`.
`git reflog`	Show a log of changes to the local repository's `HEAD`. Add `--relative-date` flag to show date info or `--all` to show all refs.

Git Branches

`git branch`	List all of the branches in your repo. Add a `<branch>` argument to create a new branch with the name `<branch>`.
`git checkout -b <branch>`	Create and check out a new branch named `<branch>`. Drop the `-b` flag to checkout an existing branch.
`git merge <branch>`	Merge `<branch>` into the current branch.

Remote Repositories



Show all changes
7 commits

Show changes since your last review
No new changes

Select commit — Hold shift + click to select a range

flow-of-control-added — 6781918
AndrejsBoja 8 days ago

LargestNumber-done — 7fb09cc
AndrejsBoja 2 days ago

PositiveNegativeNumber-done — f3d43f7
AndrejsBoja 2 days ago

References

Clean Coding Principles in C# by Cory House - Pluralsight