# Lab: Git and GitHub

Problems for exercises and homework for the ["Programming Fundamentals" course @ SoftUni](https://softuni.bg/trainings/3605/programming-fundamentals-january-2022).

# Create a GitHub Developer Profile

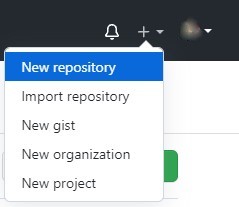
## Create a GitHub Profile

Register for a free **developer account at GitHub**: <http://github.com>. Submit your developer profile's URL as the output of this homework.

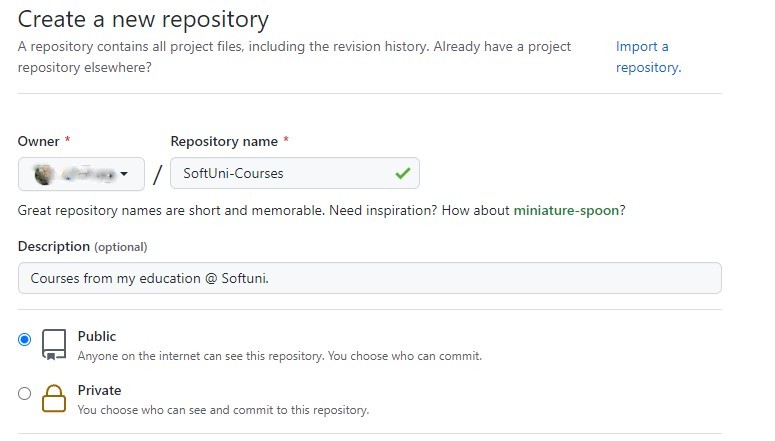


# Create a GitHub Repo and Upload your SoftUni Project

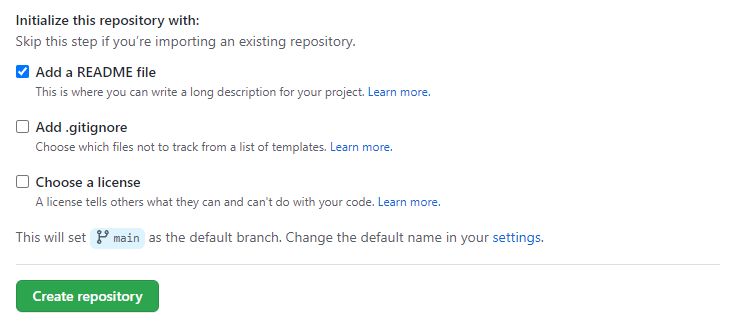
In the upper-right corner of any page, use the drop-down menu, and select **New repository**.



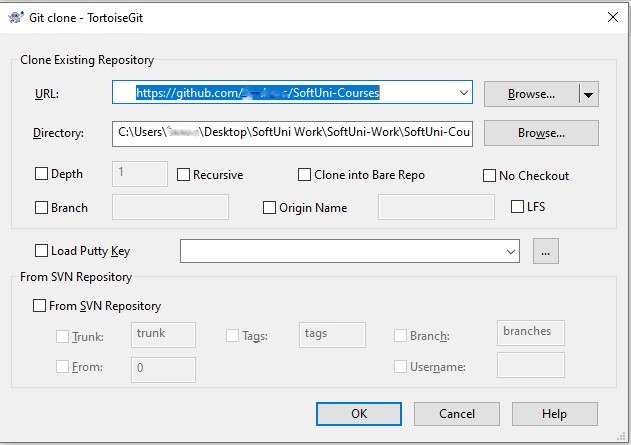
We choose a name according to the topic of our project. We can do it public or private.



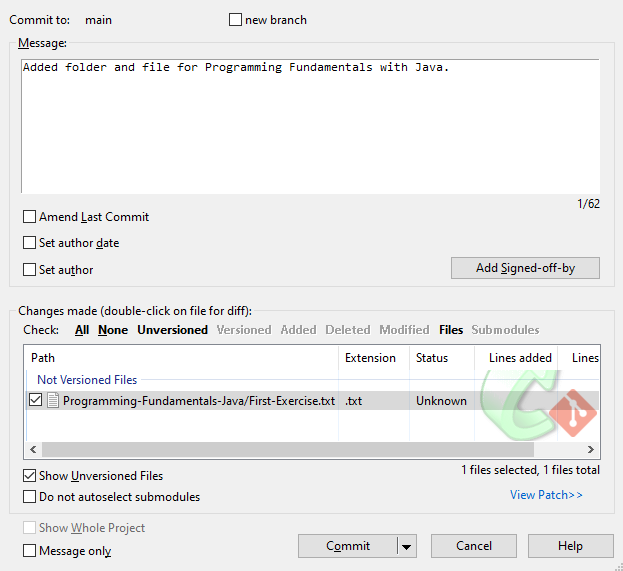
Select Initialize this repository with a **README** and **click** on **Create repository**.



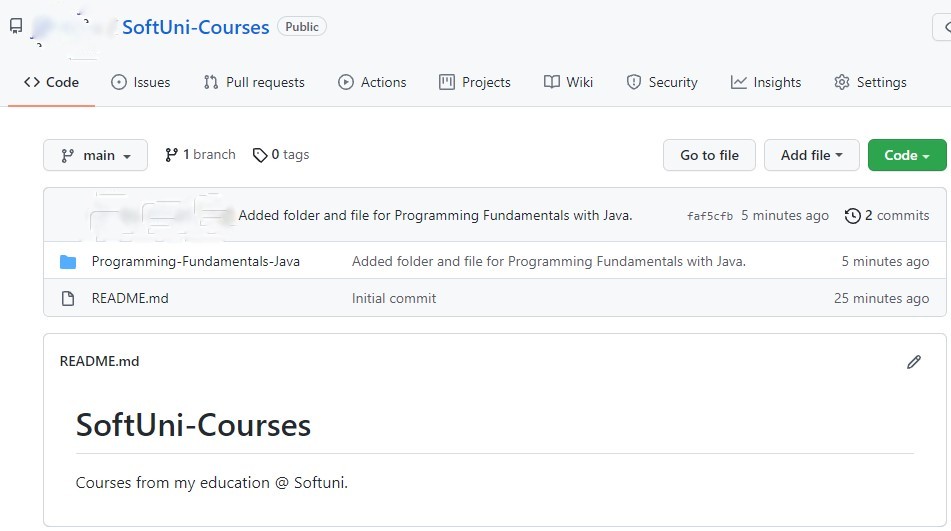
Then we select the folder on the computer in which we want to download the repo. **Right button** in folder **-> Git Clone**.



And click on **OK**.  
For example, in the folder we downloaded from GitHub (SoftUni Courses), we **created** a folder with the name **Programming-Fundamentals-Java** and in it, we **created** a file with the name **First-Exercise**. We return to folder SoftUni Courses and right button in folder -> Git commit -> "main".



And click on **Commit**. Then in the lower-left corner **click** on **Push.** We **return to GitHub** and see that the changes are **reflected**.

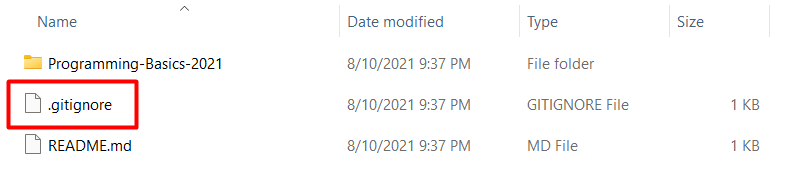


**We are ready!**

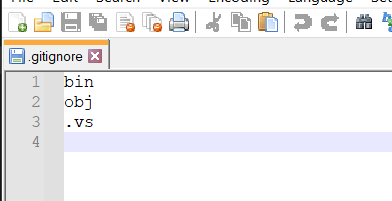
# Configure the .gitignore file

A .gitignore file specifies intentionally untracked files that Git should ignore.

Create a file called .gitignore in your project’s directory.



Each line in a **gitignore** file specifies a pattern.



If you are not sure what files you must ignore, use <https://www.toptal.com/developers/gitignore>

# Conflict + Resolve

## Make a Conflict

Update content in both directories differently:

* On your **TortoiseGit** clone create a "**test.txt**" file and add the line: "**Creating with Tortoise…**"
* On your **GitBash** clone create a "**test.txt**" file and add the line: "**Creating with Bash…**"

## Upload Changes

Upload Your Changes from **TortoiseGit** Clone. You can use TortoiseGit's "**Git Commit…**":





## Update Bash Clone

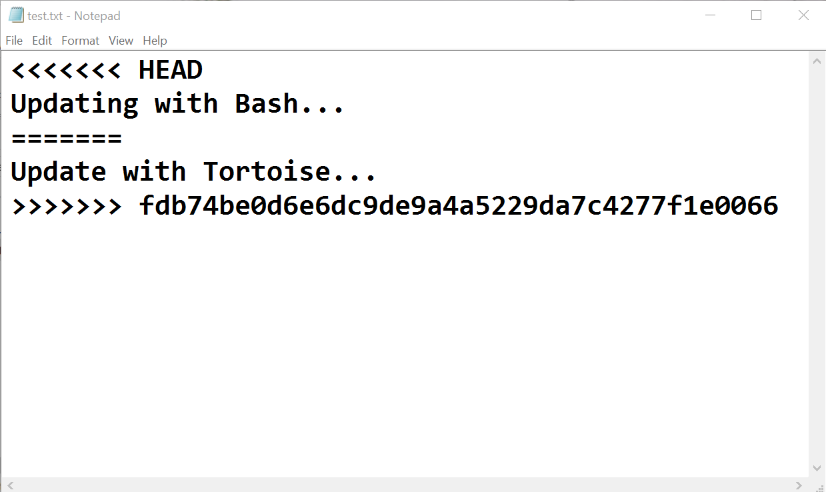
* Open your Git clone directory and open the **GitBash** console. Run the following commands:
  + Add all modified files to **staging** area
    - "**git add .**"
  + **Commit** your changes and a give commit message.
    - ''**git commit -m "Update test.txt.**""
  + **Update** your local repository
    - "**git pull**"



## Merge Conflict

Now you have merged conflict which you have to resolve

* + **Open** the "**test.txt**" file in your **GitBash** clone, it should look like this:



* + Remove the HEAD, ======, <<<<<<, >>>>>>> symbols and save the file.



* Now that you have resolved the **conflict - stage** the modified file, **commit** again and   
  **sync** with the remote repository.



## Merge Changes and Push to GitHub

You have updated the content of your remote repository, now try to update your TortoiseGit clone.

* Make additional changes to test.txt and **commit** them.



**\*Note** that if you make changes too simple TortoiseGit may **automatically** merge them.

* Now try to **push**. It turns out that we have our **remote** repository **updated** (the merge commit) and we do not have these changes on our **local** repository.



* So we have to **pull** new changes:



* Note that message: "Automatic merge failed; fix conflicts…". We have another conflict and we have to resolve it like we did earlier but small difference:
  + Go on the "**test.txt**" file. You should **open** the **file** and **remove** the same **symbols** that we have previously removed. Then right-click on the file - choose **TortoiseGit** -> **Resolve…** and click it.   
    A dialog window should open. Then you click "**Ok**" to try to **resolve** the conflict.









* Now our file is **clean** and we are ready for our final **commit**!

# Meet Your Colleagues

It’s time to meet a couple of **colleagues** at **SoftUni**. For this exercise, you must submit a **zip** file with all the solutions to the **problems below**.

## GitHub Profile Link

Create a new **text document**, called "1. GitHub Link.txt", and put a **link** to your **GitHub profile** inside it. The file should look something like this:



## GitHub Repository Screenshot

Take a **screenshot** of your **GitHub repository**, using something like a [snipping tool](https://support.microsoft.com/en-us/help/13776/windows-use-snipping-tool-to-capture-screenshots), then save the file as   
"2. GitHub Repo.jpg".

## Meet Some Colleagues

First and foremost, look around the hall and try to **make acquaintances** with your fellow students. After you meet someone, **note down** the following information about them in a **text document**:

* What is their **name**?
* Where are they **from**?
* What **hobbies/pastimes** do they enjoy?
* Why did they pick **SoftUni**?

Try to do this with **at least 3** students and also exchange **contact information** with them.

Hopefully, you made a couple of new friends from this exercise.

# Upload Homework to SoftUni

Put all of the text files and screenshots you created in a **zip file** and **send it as homework** on the **SoftUni** site.

# Teamwork

Work into **teams** of (about) 5 students in class:

* Online students work alone or form their teams.
* Each team selects a "**team leader**".

The team leader **creates a repository** in GitHub e.g. "**test-repo**".

## Add a File to GitHub

Team members add a few files:

1. Clone the "**test-repo**" into your computer (if not cloned yet).
2. Create a new file into your working directory:
   * Name the new file "**<your\_name>.txt**".
   * Put some text in it the file, e.g. "*My name is …*".
3. Commit the **new file** to your **local repository**.
4. Sync the changes to **upload your file to the remote repo**.
5. Browse the repo from <https://github.com/user/repo> to check whether your file is successfully uploaded in GitHub.

## Create a Git Conflict & Merge

* All team members create a common file **"config.txt****".**
* Each team member adds some settings in **"config.txt"**, e.g.:
  + **name = Peter**
  + **size = 100**
  + **email = peter@dir.bg**
* Each team member **commits** his local changes.
* Each team member **syncs** his changes.
  + The first member will succeed without **conflicts**.
  + The others will have a **conflict** to be merged.
  + **Resolve** the conflict:
    - **Edit** the merged changes + **commit** and **sync** again.