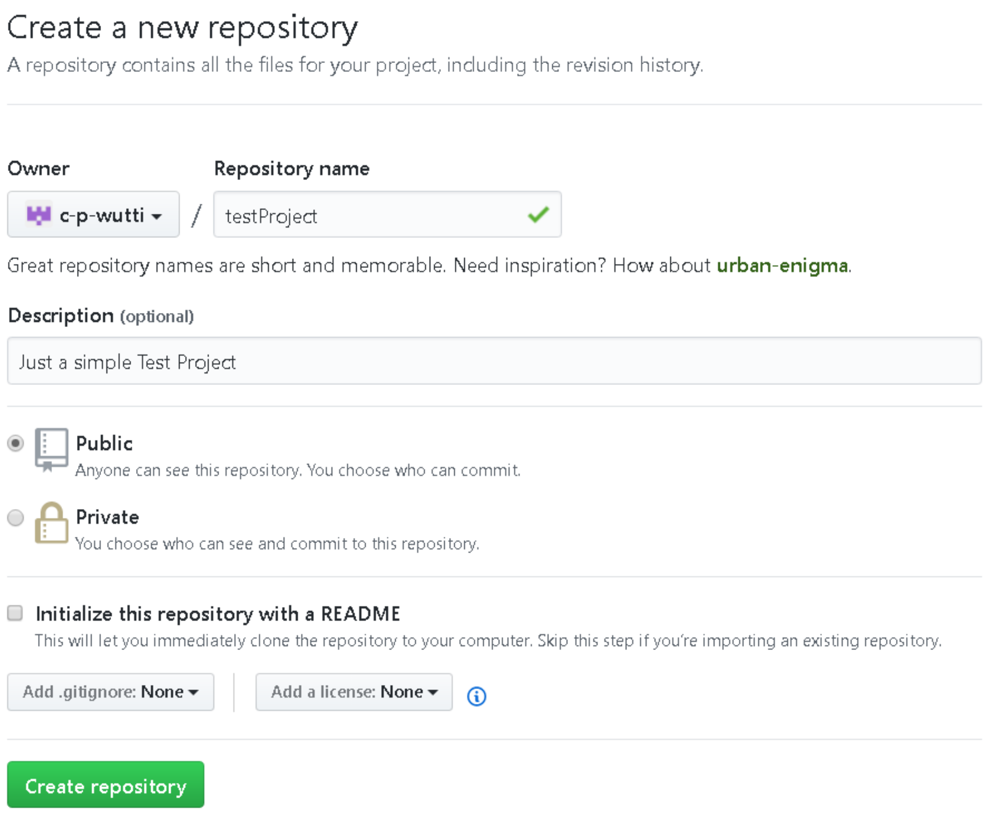
Elaboration 403_GitHub

13th March 2017

Create new global repository on GitHub



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: c-p-wutti / Repository name: testProject

Great repository names are short and memorable. Need inspiration? How about urban-enigma.

Description (optional): Just a simple Test Project

[✓ Public] Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

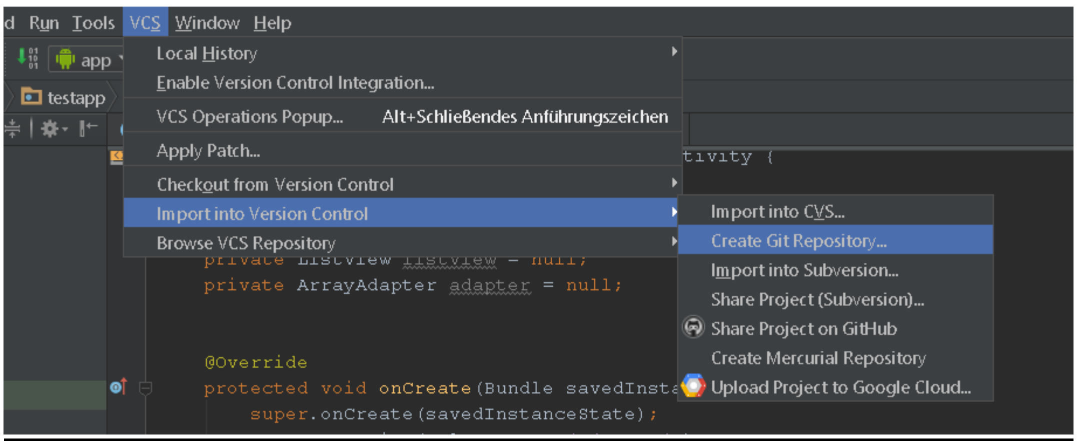
Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None

Create repository

Initialise a local repository for your app with Android Studio


d Run Tools VCS Window Help
Local History
Enable Version Control Integration...
VCS Operations Popup... Alt+Schließendes Anführungszeichen
Apply Patch...
Checkout from Version Control
Import into Version Control
Browse VCS Repository
Import into CVS...
Create Git Repository...
Import into Subversion...
Share Project (Subversion)...
Share Project on GitHub
Create Mercurial Repository
Upload Project to Google Cloud...
app
testapp

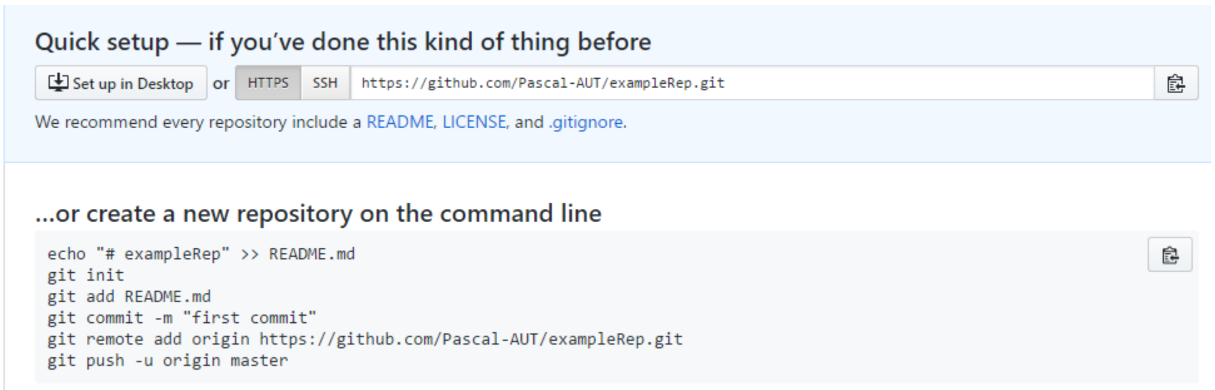
```
tivity {
    private ListView listview = null;
    private ArrayAdapter adapter = null;

    @Override
    protected void onCreate(Bundle savedInsta
        super.onCreate(savedInstanceState);
```

Select your current project folder in the upcoming file dialogue. It might be necessary to log in to your Git account.

Elaboration 403_GitHub

13th March 2017

Go to the path of your project and choose the option *'Git Bash Here'* from the context menu, execute the commands you see on the page of your created repository.



Quick setup — if you've done this kind of thing before

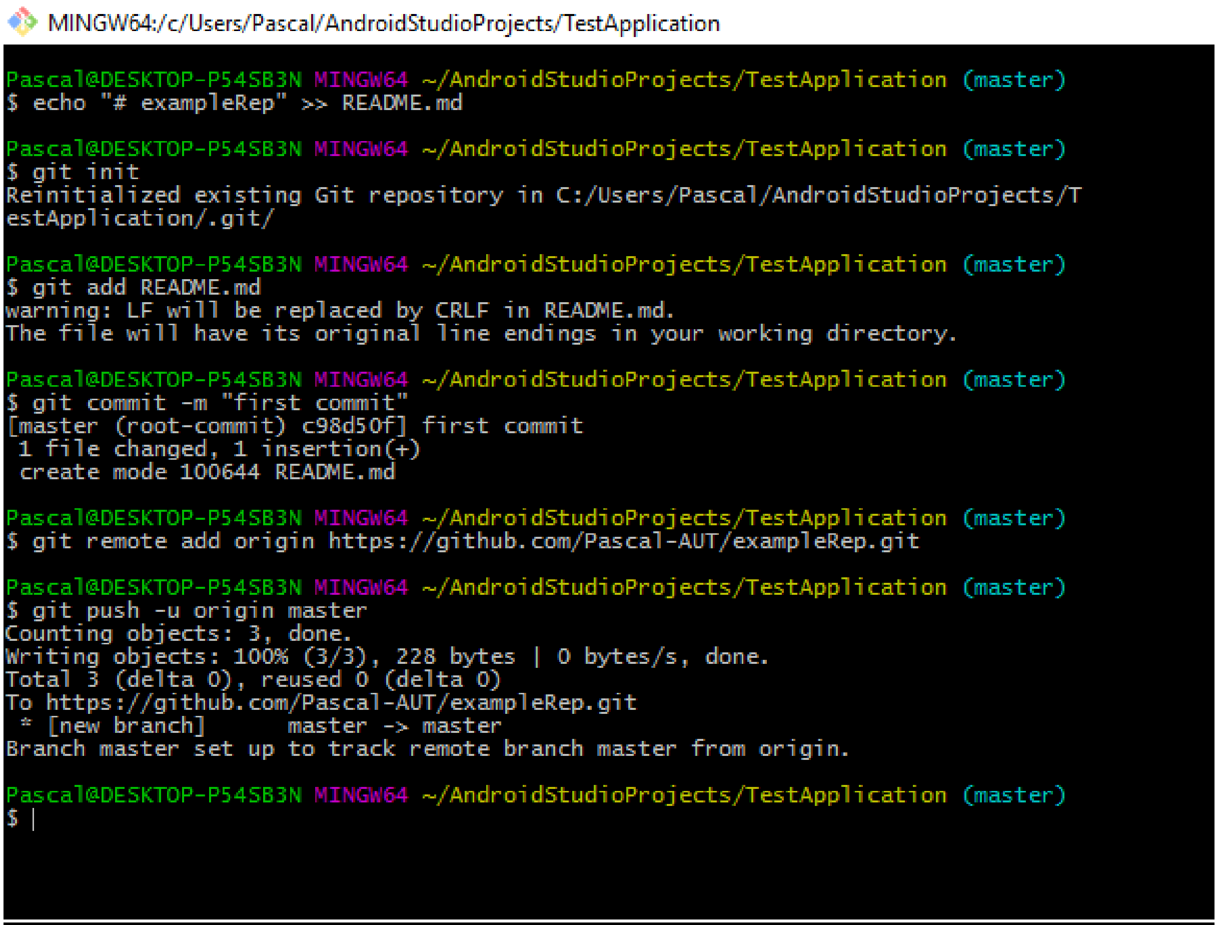
```
https://github.com/Pascal-AUT/exampleRep.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# exampleRep" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Pascal-AUT/exampleRep.git
git push -u origin master
```

Git is initialised for this folder (already happend in Android Studio, just to be sure). A readme-file is created and added to your local repository. You commit your repository and add an appropriate message. Finally, you push the changes to the global repository.



MINGW64:/c/Users/Pascal/AndroidStudioProjects/TestApplication

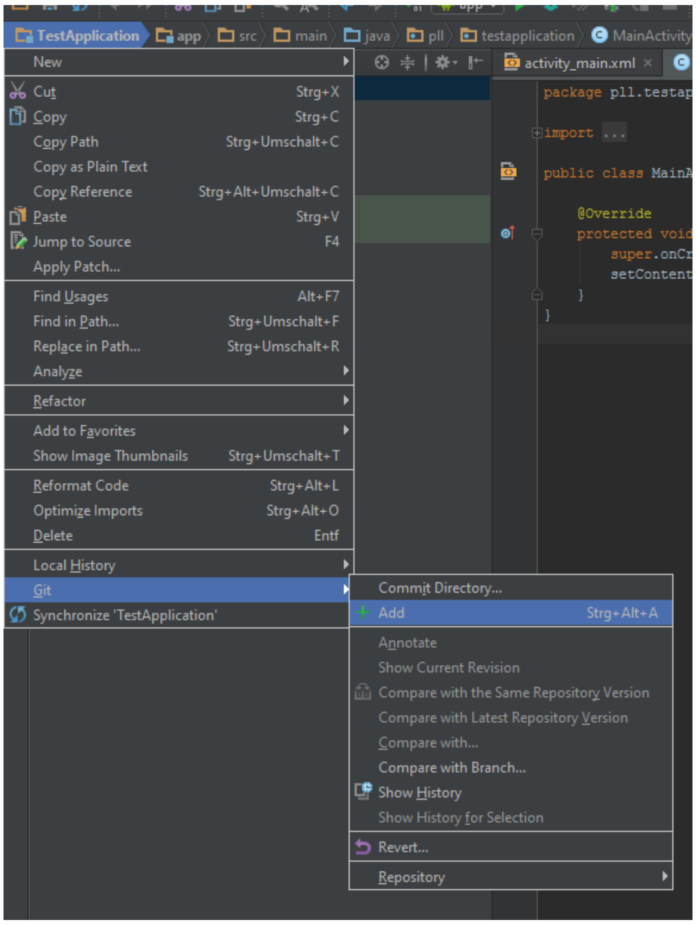
```
Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ echo "# exampleRep" >> README.md

Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ git init
Reinitialized existing Git repository in C:/Users/Pascal/AndroidStudioProjects/T
estApplication/.git/

Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ git add README.md
warning: LF will be replaced by CRLF in README.md.
The file will have its original line endings in your working directory.

Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ git commit -m "first commit"
[master (root-commit) c98d50f] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ git remote add origin https://github.com/Pascal-AUT/exampleRep.git

Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ git push -u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 228 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Pascal-AUT/exampleRep.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

Pascal@DESKTOP-P54SB3N MINGW64 ~/AndroidStudioProjects/TestApplication (master)
$ |
```

Elaboration 403_GitHub

13th March 2017

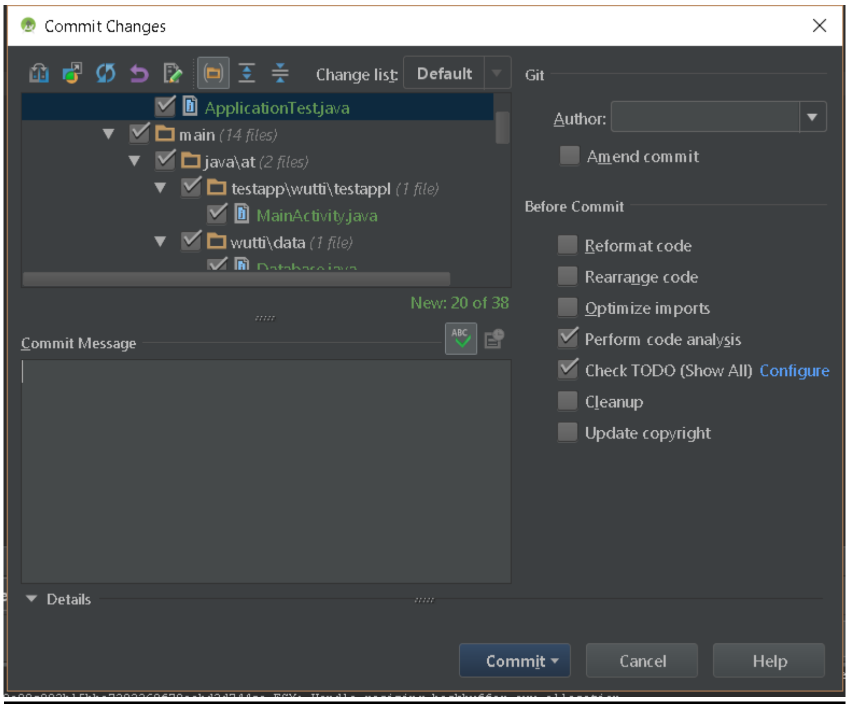
Commit the project in the local repository

First, you need to add all the classes and other files you want to push to your local repository.



```
package pll.testap
import ...
public class MainA
    @Override
    protected void
        super.onCr
        setContent
    }
}
```

In the same context menu and select *'Commit Directory ...'* (do not forget a commit message)



ApplicationTest.java
main (14 files)
java\at (2 files)
testapp\wutti\testappl (1 file)
MainActivity.java
wutti\data (1 file)
New: 20 of 38

Author:
☐ Amend commit

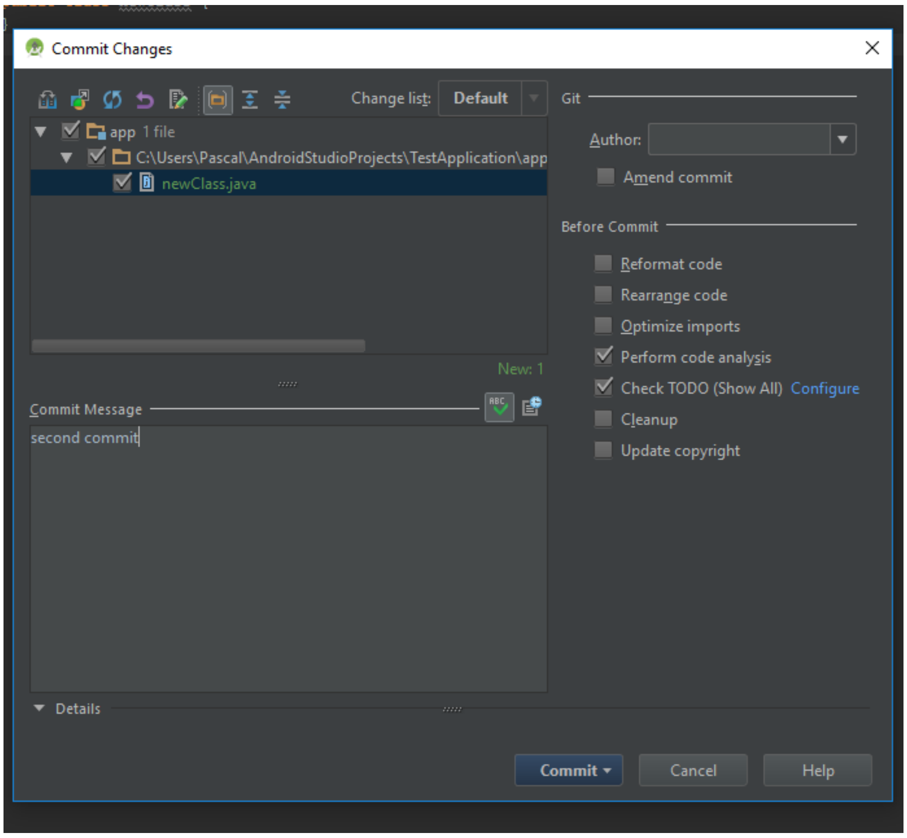
☐ Reformat code
☐ Rearrange code
☐ Optimize imports
☑ Perform code analysis
☑ Check TODO (Show All)
☐ Cleanup
☐ Update copyright

Elaboration 403_GitHub

13th March 2017

Push the local repository to the global repository

You can either add a new created class immediately to Git or add it later via the Add- button as shown above. If a class is displayed red, it is not added to the commit. To finally add your changed files to the global repository, you need to execute a push.



Commit Changes

Change list: Default

- [✓] app 1 file
  - [✓] C:\Users\Pascal\AndroidStudioProjects\TestApplication\app
    - [✓] newClass.java

New: 1

Commit Message

second commit

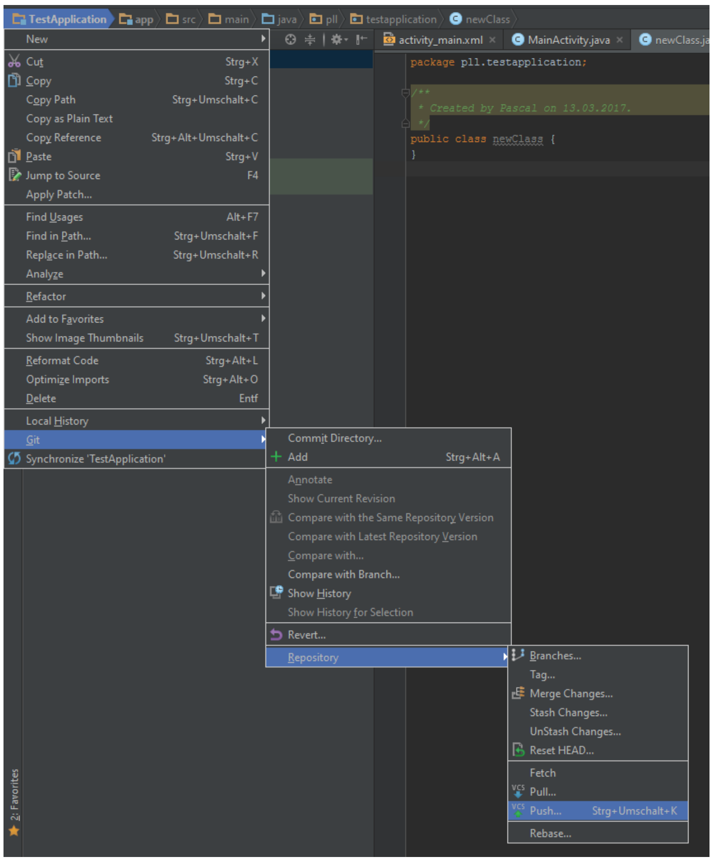
Git

Author:

- [ ] Amend commit

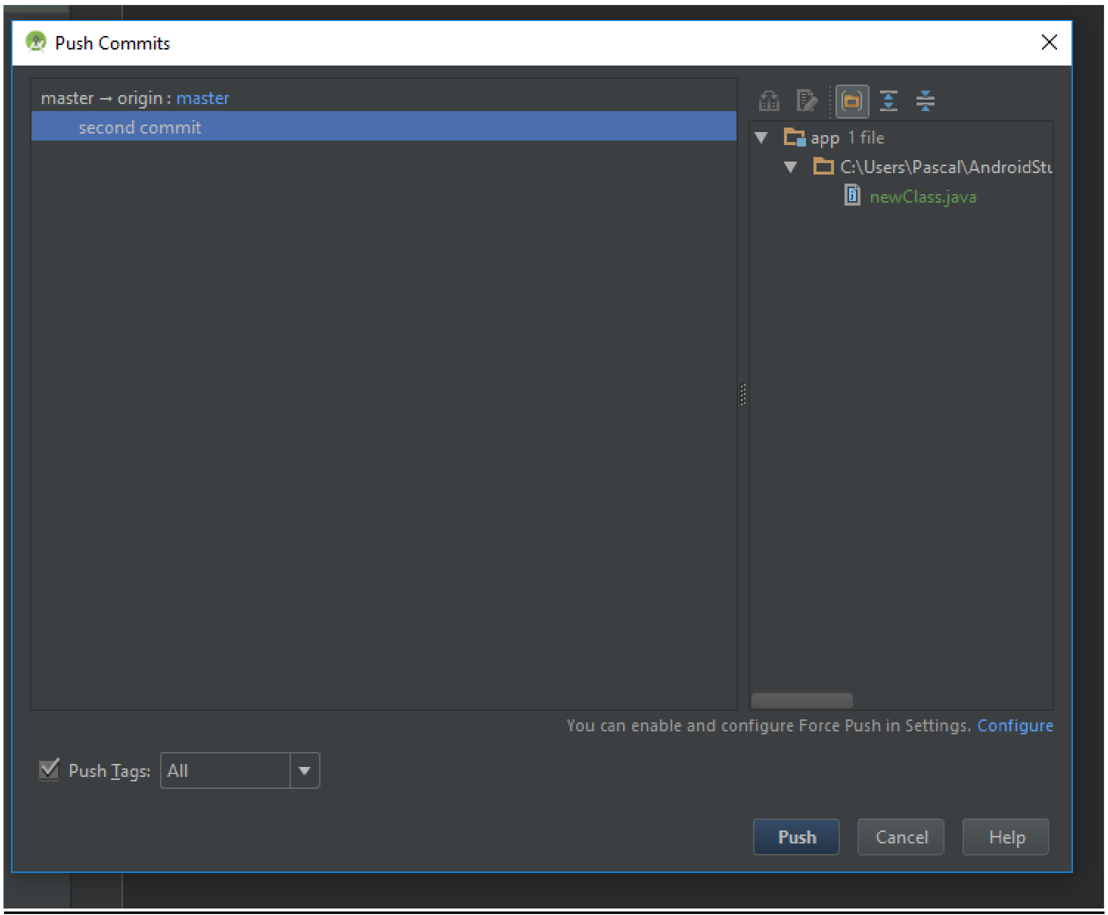
Before Commit

- [ ] Reformat code
- [ ] Rearrange code
- [ ] Optimize imports
- [✓] Perform code analysis
- [✓] Check TODO (Show All) Configure
- [ ] Cleanup
- [ ] Update copyright



```
package pll.testapplication;

/**
 * Created by Pascal on 13.03.2017.
 */
public class newClass {
}
```

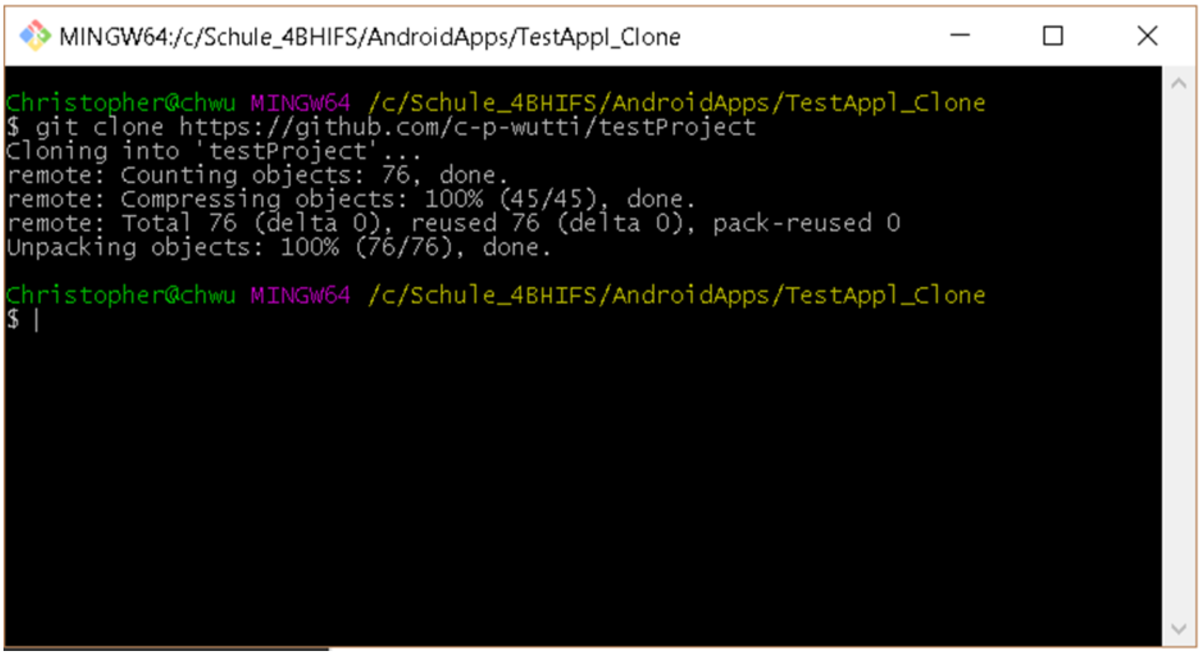
Group I – Lagger, Šarković, Weiler, Wutti



Here you can select which commit you want to push to the global repository. On the left you can see all concerned files.

Clone the global repository to another local repository (e.g. different PC)

Navigate to new directory, open the Git Bash and execute the following command.



```
Christopher@chwu MINGW64 /c/Schule_4BHIFS/AndroidApps/TestAppl_Clone
$ git clone https://github.com/c-p-wutti/testProject
Cloning into 'testProject'...
remote: Counting objects: 76, done.
remote: Compressing objects: 100% (45/45), done.
remote: Total 76 (delta 0), reused 76 (delta 0), pack-reused 0
Unpacking objects: 100% (76/76), done.

Christopher@chwu MINGW64 /c/Schule_4BHIFS/AndroidApps/TestAppl_Clone
$ |
```

Elaboration 403_GitHub

13th March 2017

Manage simple conflicts

In general, it is highly advisable to execute a pull in the first place.

Situation:

Developer-1 makes changes in class X and pushs to the global repository
Developer-2 makes changes in class Y and also wants to push it

The best way to avoid merge conflicts and other problems is to make use of Branching.

Branching is the best practice to work on different versions of a repository at the same time
By default, your repository has one branch named master. Branches are used to make changes before finally committing them to master.

When you create a branch, you make a copy of the master as it was at this time. If someone else made changes to the master branch while you were working on your branch, you could pull those updates.

Link to the repository used in the tutorial

https://github.com/Pascal-AUT/exampleRep