Links:

- **Git book**
- Git History
- Version Control
- Git
- Slides
- Workflow

Initialization:

- Download and install GIT (http://git-scm.com/download/win))
- Sign up to GITHUB
- Create a new repository to GITHUB server (Name it as you want)

Lab 4.1 Follow commands and do the following issues. Teacher can check in the server that everything has been done as told. Save all the git commands in a log file and return the file (Add comments to lines to explain what was done). You can save the bash history:

$history >mylog.log.

Case 1: Merge:

- Start Git BASH
  - Generate SSH key

```
Start SSH agent `eval "$(ssh-agent -s)"
```

Note: GITHUB add key: Settings/Deploy keys (Give some name 'mykey' and paste the key)

  - Create project folder (You can name it as repository name or anything else)
  - Change default directory to created one (Git Bash)
  - Add existing project to GITHUB (Follow link after 4 and guidance below)
    - After 'git init' command add a new file to folder:
      - Create a new text file named 'ShortCat'
      - Copy the cat below as content (Picture 1)
    - Add the files in your new local repository. This stages them for the first commit.
      - git add .
    - Commit the files that you've staged in your local repository.
      - git commit -m "First commit"
    - Copy remote repository URL field: the top of your GitHub repository's Code tab page, click to copy the remote repository URL (right 'Copy to Clipboard').
    - In the Command prompt, add the URL for the remote repository where your local repository will be pushed.
      - git remote add origin ***[remote repository URL]***
    - Check the new remote
      - git remote -v
  - Push the changes in your local repository to GitHub.
    - git push origin master

Now your data is in version control and saved also in remote repository.

- Create local branch ***b1***
- Stretch the cat longer (picture 2) in master branch. Then Add & Commit
    - This happens in master branch
- Switch to branch b1 (Checkout branch b1, you can see active branch 'git branch')
- Put ears to the cat (picture 3) in branch b1. Then Add & Commit
- Switch back to master branch
- Merge branch b1 to master (git merge b1 -m "merge")
    - Now cat is long and has ears (Both changes are in master copy)
- Delete branch b1 (Repository browser)
- Push changes to remote repo and check that file exists in correct form in server.

```
  /       \
 |  #    # |
 \     @   |
  \   _|_ /
  /       \______
 / _______ ___   \
 |_____   \   \__/
  |    \__/
  |       |
  /        \
 /   ____   \
 |  /    \  |
 | |      | |
/  |      |  \
\__/      \__/
```

Picture 1: Shortcat.

```
  /       \
 |  #    # |
 \     @   |
  \   _|_ /
  /       \______
 / _______ ___   \
 |_____   \   \__/
  |    \__/
  |       |
  |       |
  |       |
  /        \
 /   ____   \
 |  /    \  |
 | |      | |
/  |      |  \
\__/      \__/
```

Picture 2: Shortcat (longer)

```
 /\”””””””/\
 |  #    # |
 \     @   |
  \   _|_ /
  /       \______
 / _______ ____  \
 |_____   \   \__/
  |    \__/
  |       |
  /        \
 /   ____   \
 |  /    \  |
 | |      | |
/  |      |  \
\__/      \__/
```

Picture 3: Ears (b1)

```
 /\”””””””/\
 |  #    # |
 \     @   |
  \   _|_ /
  /       \______
 / _______ ____  \
 |_____   \   \__/
  |    \__/
  |       |
  |       |
  |       |
  /        \
 /   ____   \
 |  /    \  |
 | |      | |
/  |      |  \
\__/      \__/
```

Picture 4:After merge (longer AND ears)

Case 2: Merge and conflict

- Create branch ***b2***
- In master branch put ‘|’ as belt for the cat (Picture 5). Then Add & Commit
- Switch to branch ***b2***
- Put ‘X’ as belt for the cat (Picture 5). Then Add & Commit
- Switch back to master branch
- Merge branch b2 to master
    - Now there is a conflict → Resolve (Edit file in conflict part as you want it to be)
    - Select one of the belts then remember add & commit after resolve
- Delete branch ***b2*** (Repository browser)
- Push changes to remote repo and check that file exists in correct form in server.

```
  /\”””””””””/\
  |  X     X  |
  \     @     |
   \   _|_  /
   /        \_______
  / _______ ____    \
  |______   \    \__/
   |     \__/
   |        |
   ||||||||||
   |        |
   /         \
  /   _____   \
  |  /     \  |
  | |       | |
 /  |       |  \
 \__/       \__/
```

Picture 5: Belt for master

```
  /\”””””””””/\
  |  X     X  |
  \     @     |
   \   _|_  /
   /        \_______
  / _______ ____    \
  |______   \    \__/
   |     \__/
   |        |
   |XXXXXXX|
   |        |
   /         \
  /   _____   \
  |  /     \  |
  | |       | |
 /  |       |  \
 \__/       \__/
```

Picture 6: Belt for branch

Lab 4.2 Follow commands and do the following issues. Save all the git commands in a log file and return the file (Add comments to lines to explain what was done). You can save the bash history to a file: $history >mylog.log. Even here there are 2 members you can do this also by yourselves (2 GIT Bash sessions, SSH key should be generated only once). Just use 2 folders as local repos.

First Change:

- Create a new repository in GITHUB called 'Lab42'
- Generate SSH keys
- Create a folder for your local repository and initialize it as local git repo
- Create a new Java class file 'MyApp.java' and
  - Copy the source files from Appendix A.
    - Compile and run the file
    - Add & Commit and push whole project to remote repository Lab42

Using remote branch

1. First Change:
   - Create a local branch and switch to branch
     - git checkout b1
   - Add following method:
     - changeBalance(int newvalue)
     - Add also method call in main to test that it works
   - Compile and run
   - Add&Commit + push local branch to server(git push origin b1)
     - Now in GITHUB you can see remote branch b1

2. Second Change
   - Go to master branch
   - Create a local branch b2 and switch to branch
   - git checkout b2
   - Edits file:
     - Add method changePrice() (Also add method call in main)
   - Compile and run
   - Add&Commit + push branch b2 to server(git push origin b2)

Now go to GITHUB. There are now 3 branches: master, b1 and b2. You task is to merge both branches to master and remove the branches.

3. In GITHUB do following: 'Pull request' is made in GITHUB server and then branches are merged. Branch is deleted after successful merge. Note: if you have a conflict you can resolve it in GITHUB with web editor.

4. Finally, checkout master and delete local branches and pull master from GITHUB. Now your file have both changes!

Lab 4.3 In the project there is Paper-Rock-Scissor (PRS) game. Using for example Netbeans and debugger test it and find errors. Your answers include where the errors were and what was the correction. Return corrected game app with error lines commented.

Should work like that: (red are user input, orange system answers for result of the game)

```
run:
Rock: 1, Paper: 2, Scissor: 3
1
Computer chooses paper
Lose
Go to score menu? y/n
n
Want to play again? y/n
y
Rock: 1, Paper: 2, Scissor: 3
2
Computer chooses scissor
Lose
Go to score menu? y/n
n
Want to play again? y/n
y
Rock: 1, Paper: 2, Scissor: 3
3
Computer chooses paper
Win!
Go to score menu? y/n
n
Want to play again? y/n
y
Rock: 1, Paper: 2, Scissor: 3
1
Computer chooses rock
Tie!
Go to score menu? y/n
y
Display Score: display
Reset Score: reset
Exit score menu: exit
exit
Want to play again? y/n
n
BUILD SUCCESSFUL (total time: 23 seconds
```

APPENDIX A

```
public class main {
   public static void main(String[] args) {
                    Product product = new Product("Edam", 3.3, 120);
                    System.out.println("Product value is " + product.countValue());
                    product.printProduct();
    }
}

 class Product {
                    private String name;
                    private double price;
                    private int amount; //Amount in storage
                    public Product(String name, double price, int amount) {
                                        this.name = name;
                                        this.price = price;
                                        this.amount = amount;
                    }
                    public double countValue() {
                                        return(amount * price);
                    }

                    public void printProduct() {
                                        System.out.printf("Product %s, price %4.1f and balance %d pcs",
name,price,amount);
                    }
}
```