Design Patterns - Lab 1

Setting up the working environment

Goal

We need to prepare our development environment to get started in our design patterns journey. Because we are preparing to be professional software engineers we need to get familiar with some of the standard tools used in the industry. Any software engineer should be able to use a **source control system (SCM)** and an **Integrated Development Environment (IDE)**.

Tasks

1. What is a source control system (SCM)?

   Source control system (SCM) refers to tools that help you keep track of your code with a complete history of changes.

2. Why should we use it?

   Because it helps us maintain a single source of truth and it helps facilitate collaboration and accelerates release velocity.

3. Create a github account (github.com) and send a message to your tutor in the classroom with your account id.

   Done

   .....

4. Install a command line git client on your machine (Git SCM):

   .....

5. Optional: install a graphical git client (Git SCM or any other)

   .....

6. Create a SP-Lab1 repository on github.com and clone the repository on your local machine using the command line git client.

   .....

   .....

7. Locally create a README file and add some content to it and after that add the file

to the SP-Lab1 repository

8. Follow a simple git tutorial and get familiar with the terminology and the concepts. Some resources to start with :
   a. https://www.datacamp.com/community/tutorials/git-push-pull
   b. https://www.youtube.com/watch?v=J_Clau1bYco

9. Git Cheat Sheet https://about.gitlab.com/images/press/git-cheat-sheet.pdf - most important and frequently used commands

10. What is an IDE? (Integrated development environment)
    An integrated development environment (IDE) is software for building applications that combines common developer tools into a single GUI.

11. Download and install IntelliJ IDEA (https://www.jetbrains.com/idea/download/). Select the Community edition (or Ultimate edition and user your e-uvt account to get a licence)

12. After installation create a new project called MyBook in the same folder where you have checked out the git project.

13. Now let's do some coding. Let's say we have a friend that wants to become a writer and he came to you and asked you to build a software that he can use to write his books. His books are all going to contain text, images and sometimes some tables. Write a class Book that would allow him to add this type of information and print the context. Your main function should look like this:

```
public static void main(String[] args) {
        Book discoTitanic = new Book("Disco Titanic");
        discoTitanic.createNewParagraph("Paragraph 1");
        discoTitanic.createNewParagraph("Paragraph 2");
        discoTitanic.createNewParagraph("Paragraph 3");
        discoTitanic.createNewImage("Image 1");
        discoTitanic.createNewParagraph("Paragraph 4");
        discoTitanic.createNewTable("Table 1");
```

```
        discoTitanic.createNewParagraph("Paragraph 5");
        discoTitanic.print();
    }
```

14. What is your print output:

```
Paragraph 1
Paragraph 2
Paragraph 3
Image 1
Paragraph 4
Table 1
Paragraph 5
```

15. Commit and push your changes to the Git repository; Git uses a 2-step commit process: *commit* command commits the changes locally, followed by the *push* command that pushes the locally committed changes to the remote repository. a. From command line
    i. Change directory to your repository folder
    ii. Run git status to see the status of your file
    iii. Run git add src/ to add all the fles from src folder; add other files as necessary
    iv. Run git commit -m “My initial commit”
        1. Configure the credentials when prompted
    v. Run git push to upload your local changes to github.com
    vi. Visit the repository’s homepage on github.com to confirm that everything’s ok
    b. From IntelliJ - see Commit and push changes to Git repository - IntelliJ IDEA for details
        i. If using the new UI, from the left toolbar select Commit window; if using old UI select Version Control window
        ii. Select the files you want to commit to your github repository (Main.java), right-click and then ‘Add to VCS’
        iii. Enter a message and then from the Commit button arrow select Commit and Push
        iv. Visit the repository’s homepage on github.com to confirm that everything’s ok
    c. **CAUTION**: The following folders and files should not be committed to your github repository: .idea folder and <project_name>.iml file as these contain IntelliJ IDEA’s local settings, and the out folder as this contains the binaries of your project; only include the relevant project files (Java source code, JSON/XML configuration files and other resources of your project); in order to configure git client to ignore certain files or folders, you can
        i. either create a file named .gitignore (actually the name is empty and the extension is gitignore) with the following content

```
# Project exclude paths
/out/
/.idea/
/*.iml
/.gitignore
```

ii. or exclude them from IntelliJ IDEA, which will create / update the .gitignore file for your

16. Pick-up the source code from a colleague and commit it to your repository. There are several possibilities to do this
    a. Fork an existing github repository (more details here)
        i. Navigate to the repository you want to fork
        ii. In the top-right corner of the page click Fork

        iii. WARNING: The owner will see your fork

    b. Copy sources from an existing repository
        i. Create your own repository and clone it locally, say in /my-repo folder
        ii. Clone your colleague's repository in another folder, say in /his-repo iii. Copy the files you need from /his-repo folder to /my-repo folder (keep the directory structure)
        iv. Commit and push changes from your repository

17. You may also want to learn more about working with Git, from An Intro to Git and GitHub for Beginners (Tutorial) or similar resources on the Web: a. *Pull* latest changes from remote repository to your local repository b. Manage *branches* of your repository
    c. *Merge* changes and resolving conflicts
    d. Working with *Pull Requests*