Object-Oriented Programming

Nguyen Thi Thu Trang, trangntt@soict.hust.edu.vn

Hands-on lab guidelines

1. You have to follow the instructions in the hands-on lab and complete all exercises. If you can not finish them at class, please do it at home.
2. Commit and push all your results as soon as possible (when you finish, even at class) to github (through git) **BEFORE 10PM** of the next day of the lab class (i.e. Wednesday).
3. Git and github tutorial:
    a. Git Tutorial:
        i. Learn Git commands in https://git-scm.com/book/en/v2
        ii. https://git-scm.com/docs/gittutorial
        iii. https://www.atlassian.com/git/tutorials
        iv. https://www.youtube.com/watch?v=HVsySz-h9r4&list=PL-osiE80TeTuRUfjRe54Eea17-YfnOOAx
    b. Github Tutorial:
        i. Git tutorial 1: creating github repo and sharing project in Eclipse: https://www.youtube.com/watch?v=cdsMIX9gB94
        ii. Git tutorial 2: Committing, pushing, pulling and resolving conflicts with git and Eclipse: https://www.youtube.com/watch?v=M88sKbRDR8Y
4. Guidelines to submit your result:
    a. Install Git (with optional tools: Source Tree, eGit for Eclipse…): https://www.youtube.com/watch?v=HVsySz-h9r4
    b. Create an account on in https://github.com
    c. Create a **private** repository naming "OOP.DSAI.20202.StudentID.StudentName" (individually for your in-class tasks). If you don't follow this naming convention, your repository will be ignored.
    d. Add "trangntt.for.student" (trangntt.for.student@gmail.com) as a member of your repository
    e. Clone a repository, create a branch (the default and main one is master):

```
cd <local_working_folder>
git clone <repository_url>
cd <repository_name> /* e.g. OOLT.ICT.20192.StudentID.StudentName */
git branch /* let you know which branch you're working on */
git checkout -b <your_branch> /* create and switch to your new branch */
```

Then you can make any changes such as create a new file, modify an existing file, delete a file…

    a. Commit and push after changing some resources:

```
cd <local_working_folder/repository_name>
git add -A /* -A: for all operations or . for without deletions in the current folder */
git status /* check status of the stage */
git commit -m "Your comment for the update" /* commit with comment*/
git push origin <your_branch> /* push from local repo to remote repo */
```

After checking carefully the new branch code, the leader can go to the Bitbucket to *Create merge request*, then *Approve merge request* to merge the code to the master branch.

b. Pull without conflicts

```
git stash /* run this command if you want to ignore your current change */
git pull origin <your_branch>
```

c. Pull and resolve conflicts

After you commit and push to the remote repository, if there is any conflict when creating merge request, you need to resolve conflicts.

```
git checkout master
git pull origin master
git checkout <your_branch>
git rebase master
git add -A
git status
git rebase –continue
```

You need to resolve conflicts by checking the resources that have conflicts. There are both versions in the resources so that you can observe and make decisions, e.g.

Having conflig:

```
<<<<<<< HEAD /* master branch in the remote repository */
      suggestor.setClientId(clientID);
      suggestion = suggestor.translate(input).replaceAll("-", " ");
======= /* your branch in the local repository */
      suggestion = suggestor.translate(input);
>>>>>>> Your comment for the update
```

Merging (merge your local repository version with the replaceAll() method, remove the first statement of the remote repository):

```
      suggestion = suggestor.translate(input).replaceAll("-", " ");
```

Then continue, force to push the merged version to the remote repository

```
git push origin <your_branch> -f
```