How to navigate the OWP tutorials

A basic knowledge of ***git*** is required for navigating these tutorials. But, we understand that many readers might not be familiar with git and this guide is meant to target such readers. This guide, in no way, is a guide to understanding git. It is here just to be a complete summary of commands required to properly go through these tutorials. It will be a complete guide, in the sense that users need not have any other git experience and can just read this guide to have enough information for traversing the tutorial at each step properly.

Why a guide to traversing the tutorial?

Well, because the main aim of these tutorials, are to let the user see the code (along with comments) at different stages of the project and understand how a project evolves. The way to expand the code is equally important for creating a good application.
In git, whenever a certain step of a project is complete, that is, when we have a set of changes which are linked to each other and are, in some way, self contained, we group them together. For example, adding 4 new functions to a library which allow creating and deleting a file on disk can be considered to be 1 step of development. Hence the developer can "mark" this change to be in one group. This doesn't mean that the changes will be stored separately, this just means that an invisible marking will be created which will remember that these 4 functions were added together because they do something.
Each such grouping of changes is stored in a ***commit*** in git terminologies. So in these tutorials, changes made at each step have been grouped in one commit each. Hence changes made in going from step 1 to step 2 are in another commit and hence can be viewed separately than the changes made in going from step 2 to step 3. We put proper description of this grouping so a git user can know which "commit" contains which changes.
Enough literature, let's dig into actual working.

Installing git

If you are on windows, you can download ***git*** from its official site at *https://git-scm.com/*
If you are on linux, you can get git from your respective app stores. For example, on Ubuntu, you can use the following command to install git:

```
sudo apt-get install git
```

On other linux versions, you can use similar commands.

Working with the tutorial:

Downloading the code-

You can either download the zip from github and extract it in some directory. Then you can open command prompt or terminal and cd to the directory of the tutorial.
Or you can download the code via git. First open a command prompt or terminal and cd to the directory where you will keep the tutorial. Now issue the following command to download the tutorial from github:

```
git clone https://github.com/DaemonLab/OWP_Java_Inventory_Manager.git
```

This will create a folder named “OWP_Java_Inventory_Manager” which will contain all the code and other files. Now cd into the directory using:

```
cd OWP_Java_Inventory_Manager
```

Viewing all the groups of changes (commits) present in the tutorial-

Assuming you are in the tutorial’s directory, issue the following command to see all the commits present:

```
git log --oneline
```

You should see something like this:

```
f4f8c54 Step 6 -- Add printing code and some other cosmetics
cbbaa7a Step 5 -- Allow proper withdrawing of items
40b41af Step 4 -- Allow viewing and editing database in another window
cefa75b Step 3 -- HomeWindow and AddItemsScreen
9dbd4d0 Step 2 -- Create Login Prompt and DatabaseHelper
c84978b Step 1 -- Installing everything and testing SQL connection!
```

The hexadecimal numbers on the first column of each line are the commit IDs, that is the string which identifies the group, like our names. The string after the first word on each line, is the string which tells about the changes that were done during that commit. As you can see each step has its own commit.

Viewing the state of code at a certain step-

If you wish to see, how the code looked at, say, step 3, then look at the commit ID of the commit which added step 3. Here, for example, the commit ID cefa75b added Step 3. So issue the following command:

```
git checkout cefa75b
```

You should replace the commit ID with the commit ID which you want to see. After issuing this command, the code in all the files will change and if you now open the source folder, you cannot see ‘ViewEditScreen.java’ or ‘ReceiptPrinter.java’. This is because they were introduced in later steps of the tutorial. You might get a warning after issuing this command that you are now in detached HEAD mode, that's ok.
If you issue *git log --oneline* now, you will get the list of commits only upto the commit that has been currently checked out. You can still use *git checkout* to go to the state of code at any time.

Restoring the code state to final state (as it was when downloaded)-

You can use *git checkout* to checkout the last commit and that will restore your files to the state that they were when you downloaded them. But, generally we don't remember each commit's ID and hence it won't be possible to use *git checkout* to get to latest commit from a commit that we are checking out (as we can't see them in *git log* anymore).
Hence, we use the following command to get back to latest state of the code:

```
git checkout master
```

It will restore the files to as they were when you cloned/downloaded the repository.

Error while restoring state: git asks to "commit your changes or stash them"-

Well, it means that you checkout out the code at certain stage and made some changes in files, which is highly recommended for understanding how things work. But git cannot restore the state of code now because it senses changes that were not there earlier in the commit. So, you either need to commit the changes or remove them. For storing/committing your changes learn more about git from other git tutorials.

We will first remove all the local changes made which were not in the commit by:

```
git checkout .
```

be sure to put the dot at the end of the command, this command removes all the local changes which were not in the commit.

Now you can safely switch to master with *git checkout master* which is original state of code when downloaded.

How to view just the changes that were added in a certain step-

The real power of git commits are in the fact that they do not store the state of files but the changes that were made. So, for example, you want to know what changes were made in the files in going from step 2 to step 3 (commit IDs 9dbd4d0 and cefa75b respectively) then use the following command:

```
git diff 9dbd4d0 cefa75b
```

remember that changes are shown as the code went from state of the first ID in the command to state of the second commit ID. Hence keep the commit ID of step 2 before that of step 3.

You will see some lines with white text which will showcase which file the following changes were made in. There will be some lines with green text which shows the lines that were added in that commit. Lines with red text show the lines that were deleted in that commit. Lines with normal greyed out text are there just to establish the locality of changes and show lines which were not affected by the commit. Git treats edits in a line as 2 changes, deleting the original line and adding a new one.

This way you can see the changes that were done in each step. Remember that git also keeps track of readme and other files so you will also see changes done in those files in every step.

Feel free to contact us by opening a new issue on the repository for further clarifications on how to navigate the tutorial.