Git Basic Tutorial  
Classroom environment

2015-2016 ed.

Version: 1.07

Last Amendment: 21st February, 2016

Nuno Bettencourt

# Index

Index 2

Initial remarks 3

Course Outcomes 4

Environment 5

Tutorial 6

Step 1 – Setting up the environment 6

Instructor 6

Team 6

Step 2 – Creating JAVA classes in the project 7

Instructor 7

Team 7

Instructor 7

Step 3 – Adding Methods to an existing class 9

Instructor 9

Team 9

Instructor 9

Step 4 – Adding new methods and changing lines of existing code 10

Instructor 10

Team 10

Instructor 10

Project Issues 11

Issues For Step 2 11

Issues For Step 3 14

Issues For Step 4 17

References 20

Document History and Version Control 21

# Initial remarks

This tutorial is based on an existing Java project, therefore the usage of Eclipse (or Netbeans) as your IDE makes is simpler to follow.

Advise team to always use the same case when naming files and folders and check if a certain file with the same name but different case already exists. This mostly happens when dealing with multi-operating systems environments whereas some team members might use Linux and others Windows or even OS X.

In order to avoid Case Sensitive or Case Preserving issues, attendees may run the following line of code:

*git config --global core.ignorecase false*

Character typeset is also something that is worth investing some time into and make sure that the IDE is set to use UTF-8.

# Course Outcomes

* Each team will be able to:
  + Clone a GIT repository into their local development environment
  + Perform a fetch & pull operation in order to get the latest changes in the remote repository
  + Create commits in their local repository
  + Add meaningful messages to their commit
  + Amend local commit messages
  + Push local commits to a remote repository
  + Understand how and when a conflict occurs when pushing to a remote repository
  + Resolve conflicts when pushing commits to a remote repository that differs from their local one
    - By using a diff3 approach and looking at the base, local and remote changes
  + Perform their tasks according to predefined issues
    - By assigning each team a different issue
  + Close issues automatically based on commit messages with special keywords
* Each team will not learn:
  + How to switch between repository branches
  + How to merge branches
  + How to stash files
  + How to fork repositories
  + Amend local commit file changes
  + Amend remote commit messages
  + Amend remote commit file changes

# Environment

1. This tutorial is based on an Eclipse Java Project hosted on Bitbucket because:
   1. Bitbucket allows the concept of private repositories for educational members
      1. A special request to Github can be performed and instructor’s can have a two year coupon that allows them to have only up to five private repositories
   2. Bitbucket allows the export/import of all issues when a project is forked
      1. This is particularly interesting because all the work is based on issues and there should exist only one base project from which all the other instructors should fork and therefore tickets must be recreated on those forked projects (note: forking a project does not fork it’s issues)
2. The initial project was created using Eclipse
3. Each team local environment should be provided with some kind of remote viewing so that occasionally and in the best interest of the class, what is happening with a team can be displayed to all the others.
   1. Teamviewer, VNC, RDP are some of the suggested tools
4. Every team and instructor should have running:
   1. an internet browser:
      1. Bitbucket repository
      2. Bitbucket project issues
   2. a JAVA IDE that supports GIT (*e.g.* eclipse, netbeans, etc)
   3. any tool that provides git management (*e.g.* SourceTree, TortoiseGit, command line)
   4. a diff tool/application installed (filemerge, kdiff3)
5. Main links
   1. Bitbucket base project
      1. <https://bitbucket.org/nmb_isep/gittutorial/>
   2. KDiff3
      1. <http://kdiff3.sourceforge.net>

# Tutorial

This step consists in setting up the environment for the tutorial.

## Step 1 – Setting up the environment

### Instructor

1. For each lab class, each instructor should:
   1. Fork the existing base project into their own account
      1. <https://bitbucket.org/nmb_isep/gittutorial/>
      2. Suggested name for the repository: [COURSE]\_2015\_[CLASS]\_GitTutorial, where:
         1. [COURSE] can be replaced by some identifier, e.g. EAPLI
         2. [CLASS] can be replaced by some class identifier, e.g. 2DA
   2. Clone the project to their local computer
   3. Go to Bitbucket repository import/export settings[[1]](#footnote-1)
      1. Import all the issues that are located in a zip file inside the project folder named issues
      2. Go to Repository Settings, Hooks and add **Issues** hook
   4. Ask the class to divide into pairs
   5. Give each team permissions to join and edit the repository
      1. For each pair ask for one of the members’ e-mail, the one who will be pushing commits, in order to add that e-mail to the project users

### Team

1. Each team member enlisted in the project must have a Bitbucket account, otherwise registration is need
   1. Team members should have their name according to the following rule:
      1. First Name LastName [Member #], where “Member #” should be replaced by the member’s registration id, e.g. 1111111
2. Each team should clone the project to their local computer using Netbeans, Eclipse or any other tool
3. Each team should go to their Bitbucket repository and follow that repository by clicking in icon displayed in Fig. 1.



Fig. 1 - Watch icon

## Step 2 – Creating JAVA classes in the project

This step considers the addition of source files into the same repository.

### Instructor

1. Each class instructor should be assigned issue #1.
2. One team member should be assigned to an issue (from #2 to #12)
3. **Each instructor should wait for all teams to have cloned the repository**
4. Each instructor should perform the task described in issue #1
5. Each instructor should commit and push that issue to the repository
   1. This allows for all the teams to have a push error on their first commit

### Team

1. Each team should start programming their issue
2. When a team finishes the implementation**, each pair shall ONLY commit their changes** **LOCALLY** and **never push them to the remote server** until further notice.

### Instructor

1. After all teams have committed their changes locally, show that the project is still the same in Bitbucket apart from the instructor’s commit and push.
   1. Afterwards start asking each team to push their changes to the remote repository.
      1. Each team will be presented with the following conflict because local repositories are older than the remote.

Eclipse:

rejected non-fast-forward

SourceTree:

git -c diff.mnemonicprefix=false -c core.quotepath=false -c credential.helper=sourcetree push -v --tags origin master:master

Pushing to https://nmb\_isep@bitbucket.org/nmb\_isep/eapli\_gitconflicttutorial.git

To https://nmb\_isep@bitbucket.org/nmb\_isep/eapli\_gitconflicttutorial.git

**! [rejected] master -> master (non-fast-forward)**

error: failed to push some refs to 'https://nmb\_isep@bitbucket.org/nmb\_isep/eapli\_gitconflicttutorial.git'

hint: Updates were rejected because the tip of your current branch is behind

hint: its remote counterpart. Integrate the remote changes (e.g.

hint: 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Completed with errors, see above

1. The instructor should explain that this error happens because each local repository is different from the remote one. As a result, each local repository must be synched with the remote before each team can perform the push action.
   1. For this, each team should do a fetch and pull to their local repository (by using IDE’s sometimes it automatically pulls and merges but the idea is for student to understand that there are individual steps in the process fetch+pull+merge+commit).
   2. By pulling from the remote repository, the class Animal is pulled into the local repository.
      1. At this moment, the GIT application used for this will inform that there is an extra file in the project. A merge should be performed so that the Animal class becomes part of the project
         1. By doing this and committing, the following text will appear automatically in the commit message: "Merge branch 'master' of https://bitbucket.org/[user\_name]/[repository\_name]"
      2. At this point, each team can push their pending commits (exactly two)
         1. One exists for the addition of the source class mentioned in the attendees issue
         2. Another is the result of the merging with Animal class
      3. **NOTE**: if using eclipse pull (and probably Netbeans) the ide automatically merges the remote files (if no conflicts exist) and automatically performs a local commit
2. After explaining the conflict the instructor shall ask each team (individually) to push their changes at the same time that progress is being monitored in the Bitbucket repository.
3. In the end of this interaction, all teams apart form the last one to push will have their local repository out of sync from the remote repository. All teams (except the last one) should perform a pull.

## Step 3 – Adding Methods to an existing class

This step consists in changing existing source files by adding new behaviour to existing java classes.

### Instructor

1. Each team should be assigned a new issue (from #13 to #23)
   1. Each team should work on a different class than the one they worked previously.
      1. This forces all the teams to perform a pull before starting this iteration
   2. Each selected issue shall be assigned to a member of that team
      1. Otherwise no one would know what is currently being done and what can be done next
2. Each task consists in adding a void makeSound() method to an existing class

### Team

1. Each team should implement their task
2. Each team should perform a commit
3. **No team should push the commit until noticed to do so**

### Instructor

1. After each team have finished their commit the instructor should ask all teams to amend their commit messages in order to include the following message to each commit:
   1. This fixes issue #????
      1. More working examples:
         1. - This fixes issue #1
         2. - This fixes issue 1
         3. - This fixes issue #1, #2
         4. - This fixes #1
         5. - This fixes #1, #2
      2. NON WORKING EXAMPLES
         1. The following examples will not work because Bitbucket cannot recognize the keyword issue of the # character, therefore not being able to link to those issues.
            1. This fixes 1
            2. This fixes 1, 2
   2. Instructor should explain what this is for and that more keywords are also available
2. Ask each team (individually) to perform a push and notice how the issue will automatically be marked as solved in Bitbucket, until no more issued are in progress
   1. If tickets are not automatically marked as resolved, check if “Issues” hook was added to the repository
3. In the end, all teams should perform a pull so that their local repository is the same as the remote

## Step 4 – Adding new methods and changing lines of existing code

This step concerns adding new methods to existing classes and changing lines of existing code in a single class. All the teams will be working in the same file and multiple conflicts should emerge.

### Instructor

1. Each team should be assigned a new issue (from #24 to #34) – if there are not enough teams for all issues, try to assign #25, #29, #31 first for conflicts to happen.
   1. Each selected issue shall be assigned to a member of that team
   2. Some tasks will be related to adding new methods to the Duck class while others are related to changing existing lines of code.

### Team

1. Each team should implement their task
2. Each team should perform a commit
3. **No team should push the commit until noticed to do so**

### Instructor

1. On demand, each individual team should push their local commits to the remote repository, starting by the teams that changed some existing lines of code.
   1. **For teams with issues #25, 29, 31 the instructor should explain how to resolve a conflict when the same line of code has changed, by using a diff3 analysis[[2]](#footnote-2).**
   2. If referring to Fig. 2 this includes looking at the base file (A), remote file (B) and local file (C) in order to achieve file (D) and can be achieved using any of the diff tools mentioned earlier.

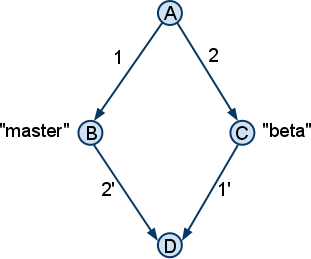


Fig. 2 - Diff3 Analysis

1. For teams that are just adding new methods, typically the only conflict they will encounter is the one of having an older version of the remote repository

## Project Issues

All the issues in this section already exist in the Gitbucket project repository.

### Issues For Step 2

#### Instructor

|  |  |
| --- | --- |
| Issue # | 1 |
| Name | Create Animal class |
| Description | Assign to Instructor only  ```  #!java  public class Animal {  public Animal() {  }  }  ```  Commit message:  Class Animal Added |

#### Team

|  |  |
| --- | --- |
| Issue # | 2 |
| Name | Create Bird class |
| Description | ```  #!java  public class Bird{  public Bird() {  }  }  ```  Commit message:  Class Bird Added |

|  |  |
| --- | --- |
| Issue # | 3 |
| Name | Create Cat class |
| Description | ```  #!java  public class Cat{  public Cat() {  }  }  ```  Commit message:  Class Cat Added |

|  |  |
| --- | --- |
| Issue # | 4 |
| Name | Create Cow class |
| Description | ```  #!java  public class Cow{  public Cow() {  }  }  ```  Commit message:  Class Cow Added |

|  |  |
| --- | --- |
| Issue # | 5 |
| Name | Create Cricket class |
| Description | ```  #!java  public class Cricket{  public Cricket() {  }  }  ```  Commit message:  Class Cricket Added |

|  |  |
| --- | --- |
| Issue # | 6 |
| Name | Create Dog class |
| Description | ```  #!java  public class Dog{  public Dog() {  }  }  ```  Commit message:  Class Dog Added |

|  |  |
| --- | --- |
| Issue # | 7 |
| Name | Create Donkey class |
| Description | ```  #!java  public class Donkey{  public Donkey() {  }  }  ```  Commit message:  Class Donkey Added |

|  |  |
| --- | --- |
| Issue # | 8 |
| Name | Create Duck class |
| Description | ```  #!java  public class Duck {  public Duck() {  }  }  ```  Commit message:  Class Duck Added |

|  |  |
| --- | --- |
| Issue # | 9 |
| Name | Create Pig class |
| Description | ```  #!java  public class Pig{  public Pig () {  }  }  ```  Commit message:  Class Pig Added |

|  |  |
| --- | --- |
| Issue # | 10 |
| Name | Create Rooster class |
| Description | ```  #!java  public class Rooster{  public Rooster() {  }  }  ```  Commit message:  Class Rooster Added |

|  |  |
| --- | --- |
| Issue # | 11 |
| Name | Create Sheep class |
| Description | ```  #!java  public class Sheep{  public Sheep() {  }  }  ```  Commit message:  Class Sheep Added |

|  |  |
| --- | --- |
| Issue # | 12 |
| Name | Create Turkey class |
| Description | ```  #!java  public class Turkey{  public () {  }  }  ```  Commit message:  Class Turkey Added |

### Issues For Step 3

#### Team

|  |  |
| --- | --- |
| Issue # | 13 |
| Name | Add sound to Bird |
| Description | Add the following method to Bird class.  ```  #!java  void makeSound() {  System.out.println("piu piu");  }  ```  Commit message:  Added sound behaviour to Bird. |

|  |  |
| --- | --- |
| Issue # | 14 |
| Name | Add sound to Cat |
| Description | Add the following method to class Cat.  ```  #!java  void makeSound() {  System.out.println("miau");  }  ```  Commit message:  Added sound behaviour to Cat. |

|  |  |
| --- | --- |
| Issue # | 15 |
| Name | Add sound to Cow |
| Description | Add the following method to class Cow.  ```  #!java  void makeSound() {  System.out.println("muuuuu");  }  ```  Commit message:  Added sound behaviour to Cow. |

|  |  |
| --- | --- |
| Issue # | 16 |
| Name | Add sound to Cricket |
| Description | Add the following method to class Cricket.  ```  #!java  void makeSound() {  System.out.println("cri cri");  }  ```  Commit message:  Added sound behaviour to Cricket. |

|  |  |
| --- | --- |
| Issue # | 17 |
| Name | Add sound to Dog |
| Description | Add the following method to class Dog.  ```  #!java  void makeSound() {  System.out.println("au au");  }  ```  Commit message:  Added sound behaviour to Dog. |

|  |  |
| --- | --- |
| Issue # | 18 |
| Name | Add sound to Donkey |
| Description | Add the following method to class Donkey.  ```  #!java  void makeSound() {  System.out.println("ió ió");  }  ```  Commit message:  Added sound behaviour to Donkey. |

|  |  |
| --- | --- |
| Issue # | 19 |
| Name | Add sound to Duck |
| Description | Add the following method to class Duck.  ```  #!java  void makeSound() {  System.out.println("qua qua");  }  ```  Commit message:  Added sound behaviour to Duck. |

|  |  |
| --- | --- |
| Issue # | 20 |
| Name | Add sound to Pig |
| Description | Add the following method to class Pig.  ```  #!java  void makeSound() {  System.out.println("oinc oinc");  }  ```  Commit message:  Added sound behaviour to Pig. |

|  |  |
| --- | --- |
| Issue # | 21 |
| Name | Add sound to Rooster |
| Description | Add the following method to class Rooster.  ```  #!java  void makeSound() {  System.out.println("cócórócócó");  }  ```  Commit message:  Added sound behaviour to Rooster. |

|  |  |
| --- | --- |
| Issue # | 22 |
| Name | Add sound to Sheep |
| Description | Add the following method to class Sheep.  ```  #!java  void makeSound() {  System.out.println("mêêê");  }  ```  Commit message:  Added sound behaviour to Sheep. |

|  |  |
| --- | --- |
| Issue # | 23 |
| Name | Add sound to Turkey |
| Description | Add the following method to class Turkey.  ```  #!java  void makeSound() {  System.out.println("glu glu");  }  ```  Commit message:  Added sound behaviour to Turkey. |

### Issues For Step 4

#### Team

|  |  |
| --- | --- |
| Issue # | 24 |
| Name | Add fly behaviour to Duck |
| Description | Add fly behaviour to Duck.  ```  #!java  void fly() {  }  ```  Commit message:  Added fly behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |
|  |  |

|  |  |
| --- | --- |
| Issue # | 25 |
| Name | Change Duck sound to “qua qua qua” |
| Description | Change the sound Ducks make to “qua qua qua”.  ```  #!java  System.out.println("qua qua qua");  ```  Commit message:  Changed Duck sound to “qua qua qua”.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 26 |
| Name | Add quack behaviour to Duck |
| Description | Add quack behaviour to Duck.  ```  #!java  void quack() {  }  ```  Commit message:  Added quack behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 27 |
| Name | Add swim behaviour to Duck |
| Description | Add swim behaviour to Duck.  ```  #!java  void swim() {  }  ```  Commit message:  Added swim behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 28 |
| Name | Add lay egg behaviour to Duck |
| Description | Add lay egg behaviour to Duck.  ```  #!java  void layEgg() {  }  ```  Commit message:  Added lay egg behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 29 |
| Name | Change Duck sound to “qua” |
| Description | Change the sound Ducks make to “qua”.  ```  #!java  System.out.println("qua");  ```  Commit message:  Changed Duck sound to “qua”.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 30 |
| Name | Add eat behaviour to Duck |
| Description | Add eat behaviour to Duck.  ```  #!java  void eat() {  }  ```  Commit message:  Added eat behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 31 |
| Name | Specify how loud each sound made by a Duck should be |
| Description | Change makeSound method to accept a loudness argument.  ```  #!java  void makeSound(int loudness)  ```  Commit message:  Changed Duck makeSound behaviour to set loudness level.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 32 |
| Name | Add sleep behaviour to Duck |
| Description | Add sleep behaviour to Duck.  ```  #!java  void sleep() {  }  ```  Commit message:  Added sleep behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 33 |
| Name | Add blink behaviour to Duck |
| Description | Add blink behaviour to Duck.  ```  #!java  void blink() {  }  ```  Commit message:  Added blink behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

|  |  |
| --- | --- |
| Issue # | 34 |
| Name | Add shake behaviour to Duck |
| Description | Add shake behaviour to Duck.  ```  #!java  void shake() {  }  ```  Commit message:  Added shake behaviour to Duck.  Fixes issue #? (replace ? by your issue number) |

# References

<http://blog.wuwon.id.au/2010/09/painless-merge-conflict-resolution-in.html>

# Document History and Version Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date Approved** | **Changed By** | **Approved By** | **Brief Description** |
| 1.00 | 2014/02/22 | NMB |  | Initial Draft |
| 1.05 | 2014/03/05 | NMB |  | Included Document History and Version Control  Fixes some sentence errors  Introduce some more setup options based on first lab experience feedback |
| 1.06 | 2015/09/17 | NMB |  | Fixes some sentence errors.  Remove reference to BitbucketCards  Adapt it to the MEI repository |
| 1.07 | 2016/02/21 | NMB |  | Remove MEI and LEI references. |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. [https://bitbucket.org/[user\_name]/[repository\_name]/admin/issues/import-export](https://bitbucket.org/%5buser_name%5d/%5brepository_name%5d/admin/issues/import-export) [↑](#footnote-ref-1)
2. Note that the Netbeans IDE does not support diff3 analysis. Please remind your students to install KDiff3[http://kdiff3.sourceforge.net] and properly configure the Git client tool (e.g., TortoiseGit, SourceTree) to use it as the default Merge tool. [↑](#footnote-ref-2)