CST8116 Assignment 03 (22S)

Using Git within Eclipse.

# Instructions

* This assignment focuses on using the Git version control system, within an Eclipse Environment.
* **Expectation**: you have already reviewed and completed instructions for Git installation and configuration
  + Installing and initial configuration: See Hybrid 05 content area, within week 05 lecture notes content area
  + Using Git in Eclipse: See Hybrid 08 content area, within week 10 lecture notes content area
    - The handout here will walk you through this lab assignment as well.
  + **If you have not installed Git, or worked through the handouts for Hybrid 05 and 08 do so now.**
* This assignment handout has brief step-by-step instructions
* You will need to screen shot your running program
* You will need to screen shot your Git History for the project in Eclipse
* Submit your source code file, and your Git Repository as a zip archive.

(and follow Lab Professors Directions)

# Steps

## Create a Git Repository for the project (Use Windows File Explorer for this step)

* Create a Git Repository using Eclipse for your Assignment 3 Project, see hybrid handouts.
* Create the following location on your hard drive for the repository:

**C:\CST8116\_Assignment\_03\_Your\_Name**

Your\_Name should be your full name, as see in ACSIS separating each name with underscores.

* You will need to zip this folder and submit it after completing all of the steps in this lab.

## Create an Eclipse Project

* Name the project: Assignment 3

## Add the Project files to the Git Staging area

* Add your project to the Git staging area within Eclipse (See hybrid handout(s) for information on Team > Share Project, as well as how to direct Eclipse to use the repository folder you created above.
* Then Right-Click Project and use Team > Add to Index

## Verify .ignore file

* Make sure that the .ignore file has an entry for /bin/
* If you cannot see the .ignore file reference the hybrid handout on how to modify the package explorer filters.

## First Commit

* Commit the initial project files
  + Enter a commit message   
    “New Project Assignment03 by *YourFullName*”
  + Omit the double-quotes.
  + replace YourFullName with your actual name
  + Add a signed-off by entry with your name and email using the interface (see hybrid handout)
  + Commit.

## Create a source code file and add to version control

* Add a class to the project naming it Assignment03*YourFullName* where *YourFullName* is your full name as it appears in ACSIS. For Example Stanley Pieda would name the class File Assignment03StanleyPieda
* Check off the box to create a main method.
* Right-click on the file and add it to the Git Staging Area (Team > Add To Index)

## Second Commit

* Commit the new source code file.
  + Enter a commit message  
    “Added source code file Assignment03*YourFullName* with method main”
    - Omit the double-quotes shown on the line above.
  + replace YourFullName with your actual name
  + Add a signed-off by entry with your name and email using the interface (see hybrid handout)
  + Commit.

## Copy starter code into source code file

* Copy the starter code located here (see end of handout for fuller code listing) into method main.
* Modify the code so that it prints out your actual name, instead of “Your name”
* Add programmer comments to the file in the usual manner for our course.
  + You will be asked to fix a mistake in the program in a later step.
  + Currently if you run the program as presented here, it will not produce the correct output.
  + See the Appendix at the end of this handout for what the program is expected to do, versus how it crashes.

int roll = 0;

int[] diceRolls = new int[10];

int totalRolls = 0;

// sample the rolls

for(int count = 0; count < 1000; count++) {

roll = (int)(Math.random() \* 10) + 1;

diceRolls[roll - 1] = diceRolls[roll - 1] + 1; // adjust 1-10 value to 0-9 for index

}

// run a report

**for(int index = 0; index <= diceRolls.length; index++) { // crashes**

System.out.printf("Count of %d is: %d%n", (index + 1), diceRolls[index]);

totalRolls = totalRolls + diceRolls[index];

}

## System.out.println("Total rolls were: " + totalRolls);

## System.out.println("Program by Your Name");

## Save the changes to the file. (Just normal Save button in Eclipse).

## Third Commit

* Commit the changed source code file.
  + Enter a commit message  
    “Added programmer comments, starter code in main, added my name.”
    - Omit the double-quotes as seen on the line above
  + Add a signed-off by entry with your name and email using the interface (see hybrid handout)
  + Commit.

## Run the program / fix the bug / run the program & Screen Shot.

* Run the program and observe the mistake (it likely crashes)
* Fix the problem with the loop that generates the report (see bolded line above marked as //crashes)
* Run the program and take a screen shot of the corrected program output for your MS Word document.

## Fourth Commit

* Commit the changed source code file.
  + Enter commit messages, use Separate lines, blank line between each.   
    “Corrected the problem with the loop.”

“The loop counter is being used as the subscript for the array, the stopping condition was incorrectly set as index <= diceRolls.length when it needed to be index < diceRolls.length.”

* + Add a signed-off by entry with your name and email using the interface (see hybrid handout)
  + Commit.

## Modify the program / run the program & Screen Shot.

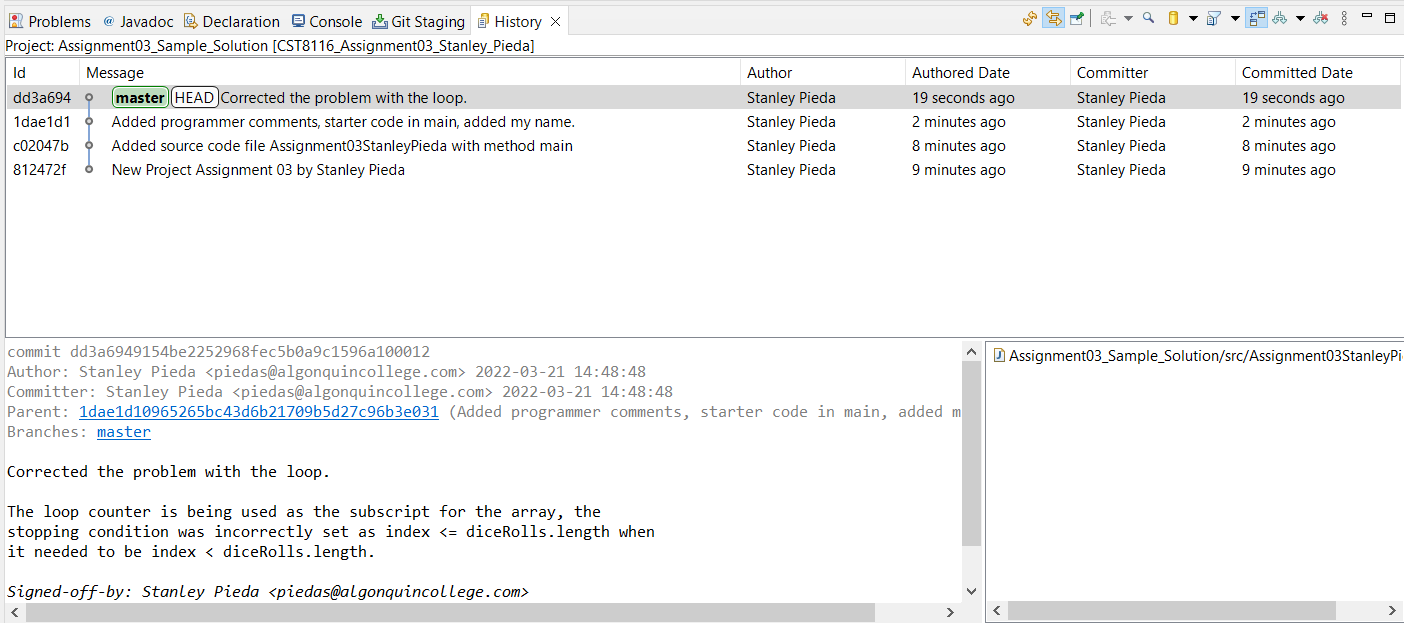
* Modify the program so that it uses a 16-sided dice
* Report the number of even rolls and the number of odd rolls

## Fifth Commit

* Commit the changed source code file.
  + Enter commit messages, use Separate lines, blank line between each.   
    “Modified to uses 16-sided dice and report total even and odd rolls”
  + Add a signed-off by entry with your name and email using the interface (see hybrid handout)
  + Commit.

## Take screen shot of Git history in Eclipse

* View the Git History in Eclipse.
* Ensure that your Project (not just the source code file) is selected in the Package Explorer so that the full history is shown. (Right-click, Team > Show in history)
* Take a screen shot that shows your commit: id, message, author, authored date, committer, and committed date.
* This is an example created by Stanley Pieda



Submit MS Word document with screen shots, your Java file, and a zip archive of the Git Repository you created.

Follow any additional instructions provided by your lab professor.

Your lab professor may verify your repository using command line tools, e.g. >git log to determine that it closely matches your screen shot taken from Eclipse.

# Microsoft Word Document Format

Screen Shot Running Program

Screen Shot Git History in Eclipse

# Grading (10 points)

|  |  |  |  |
| --- | --- | --- | --- |
| Criteria | Missing / Poor (0) | Below Expectations (1) | Meets Expectations (2) |
| Screen Shot, Program Execution | Missing or poorly done (missing student name in screen shot) | Shows program executing successfully, may only have part of student’s name in output. | Shows program executing successfully, student’s full name as part of output. |
| Screen Shot, Git History | Missing or poorly done (missing student name in screen shot) | Shows some but not all of the requested commits, or commit text not correct. May only have part of the student’s name in the screen shot. | Shows all four requested commits and closely matches sample screen shot. Student’s full name is visible in the screen shot. |
| Source Code Comments | Missing or poorly done | Missing some of: Programmer comments at top of file, descriptive comment for class and method name. | All completed: Programmer comments at top of file, descriptive comment for class and method name. |
| Source Code Fixed | Missing or poorly done or is un-modified starter code. | Problem with program not fixed correctly. | Problem with program is fixed. |
| Git Repository | Missing from submission or only a project folder and not a full Git Repository. | Does not have requested folder name for repository, git log command may show inconsistencies when compared to screen shot. | Has requested folder name for repository, git log command closely matches the history seen in the screen shot. |

# Submission Requirements

* Upload these items to Brightspace by the due date:
* MS Word document containing screen shots.
* Java file with programmer comments and source code.
* Assignment 03 Git Repository within a zip archive (CST8116\_Assignment\_03\_*Your\_Name*.zip)
  + *Your\_Name* should be your full name as seen in ACSIS.
* Follow your lab professor’s instructions regarding lab submissions for their lab section.

# Appendix: Sample **git log**

* Students are not required to submit a screen shot like this one, it is here for demonstration only.
* Your lab professor will open a command prompt-window to the root folder of the repository, in this case the extracted zip folder that you (student) submitted along side your MS Word document.
* Use the command:

git log

* Use the space-bar if there are multiple pages of output if attempting this (optional).
  + It seems the first time the git log is run there is a great deal of output, use q to quit, then cls to clear screen, then git log again.
* Important: Enter a ‘q’ for quit, to exit the log tool.
* The output from the git-log should closely match the screen shot in Eclipse of your Git History.

# 

# Appendix: Sample Program Runs

* This is an example of what is happening with the starter code. The program crashes somewhere.

(Line number is 29 as seen here at the end of the error message, your exact line number might be different)

Count of 1 is: 110

Count of 2 is: 107

Count of 3 is: 94

Count of 4 is: 91

Count of 5 is: 97

Count of 6 is: 95

Count of 7 is: 109

Count of 8 is: 100

Count of 9 is: 98

Count of 10 is: 99

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10

at Assignment03StanleyPieda.main(Assignment03StanleyPieda.java:29)

* This is an example of what is desired. Specific counts of dice-face values will differ due to the random number generator.

Count of 1 is: 81

Count of 2 is: 107

Count of 3 is: 110

Count of 4 is: 103

Count of 5 is: 110

Count of 6 is: 83

Count of 7 is: 92

Count of 8 is: 95

Count of 9 is: 105

Count of 10 is: 114

Total rolls were: 1000

Program by Stanley Pieda

Appendix: Fuller Starter Code (Has Additional Comments)

public static void main(String[] args) {

// program rolls a 10 sided dice 1000 times and records the number

// of occurrences of each result.

// See:https://en.wikipedia.org/wiki/Dice#Common\_variations

// Math.random() returns 0.0 to 1.0 excluding 1.0

// i.e. we could get 0.99999999999 etc. but not 1.0

// so Math.random() times upper range of 10 is 0 to 9

// add 1 to get correct range, then cast to int to remove any decimal fraction.

// Hint: it is easy to make a mistake with array indexes as people

// (and dice) count from 1 upwards, while programmers using arrays

// count from 0 upwards.

int roll = 0;

int[] diceRolls = new int[10];

int totalRolls = 0;

// sample the rolls

for(int count = 0; count < 1000; count++) {

roll = (int)(Math.random() \* 10) + 1;

diceRolls[roll - 1] = diceRolls[roll - 1] + 1;

}

// run a report

for(int index = 0; index **<=** diceRolls.length; index++) { // crashes

System.out.printf("Count of %d is: %d%n", (index + 1), diceRolls[index]);

totalRolls = totalRolls + diceRolls[index];

}

System.out.println("Total rolls were: " + totalRolls);

}