|  |
| --- |
| WELTEC Bachelor of Information Technology |
| English League Table |
| Creating a program and adding Git and GitHub version control systems capabilities |

|  |
| --- |
| Francois Mindiel  ID: 2804582  Submission Date: 04/08/16 |
| Lecturer: Chalinor Baliuag |
|  |
|  |

Contents

[Project description 2](#_Toc458087975)

[English Premier League Table 2](#_Toc458087976)

[Git 2](#_Toc458087977)

[GitHub 2](#_Toc458087978)

[Screenshots of project 3](#_Toc458087979)

[Local Repository (Git) 3](#_Toc458087980)

[Exporting to GitHub 4](#_Toc458087981)

[Importing from GitHub 4](#_Toc458087982)

[Resolving conflicts 5](#_Toc458087983)

[UML Diagram 6](#_Toc458087984)

[References 7](#_Toc458087985)

[Appendix 8](#_Toc458087986)

[Main class 8](#_Toc458087987)

[Team class 10](#_Toc458087988)

[Figure 1: creating a local repository 3](#_Toc458087262)

[Figure 2: committing changes to local repository 3](#_Toc458087263)

[Figure 3: history view of local repository 3](#_Toc458087264)

[Figure 4: creating repository on GitHub 4](#_Toc458087265)

[Figure 5: exporting local repository to GitHub 4](#_Toc458087266)

[Figure 6: importing remote repository to Eclipse 4](#_Toc458087267)

[Figure 7: error pushing changes to GitHub 5](#_Toc458087268)

[Figure 8: branching and merging changes 5](#_Toc458087269)

[Figure 9: UML class diagram of project 6](#_Toc458087270)

# Project description

This project is meant for us to show our skills to develop a simple program and use tools to create snapshots of the program, where we can revert to a previous version of the latter if need be.

## English Premier League Table

The developed program will be written in Java using Eclipse IDE, and will create and store information for the English Premier League teams, such as the number of matches won, lost, goals scored, and points earned. It will also display the teams and rank them from first to last accordingly. For the sake of simplicity, a few rules of the competition have not been applied, such as teams with the same number of points are not further sorted by goal difference.

For the Version control systems used, Git and GitHub will be used. Both tools are complementary of each other.

## Git

Git is a local based repository tool with complete history and full version-tracking capabilities, independent of network access or a central server (Git (software), 2016). It also has characteristics such as branching, merging, and tools to visualise these.

## GitHub

GitHub is also a version control system, but compared to Git, it is a remote repository. This means that the repositories are stored on in the cloud. It provides a safe way to store the current and previous iterations of the program, but also gives the option of sharing the code to other collaborators, who can either only see or modify the code. Again the code can be branched and merged later on.

# Screenshots of project

## Local Repository (Git)

After creating a project, along with a class, we can initialise the Git tool to create a local repository of the project on the computer.

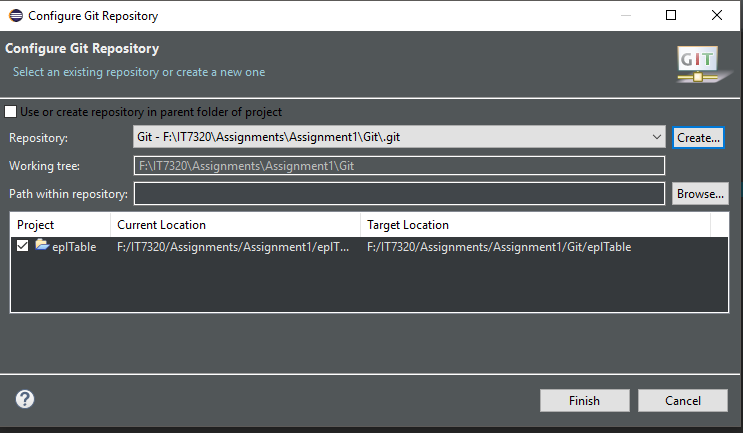


Figure 1: creating a local repository

After making changes to the program, we can commit the changes by pressing the "commit" button, and add a description of the changes made to it.

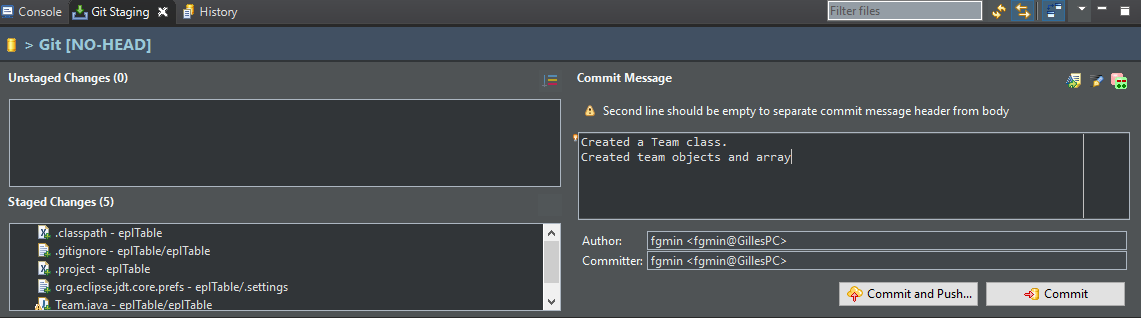


Figure 2: committing changes to local repository

The diagram below shows the master branch with the commit changes.

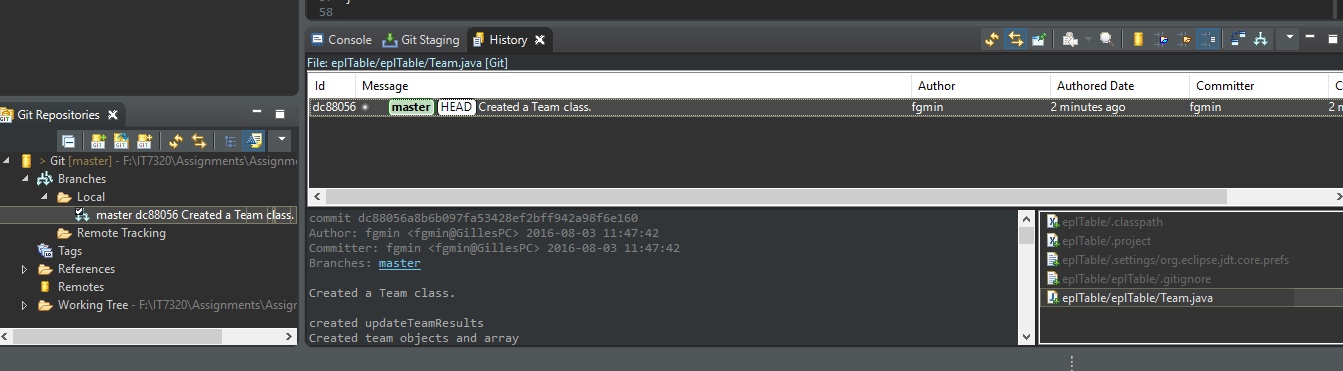


Figure 3: history view of local repository

## Exporting to GitHub

After creating a repository on GitHub, the local repository is linked to GitHub using the remote repository URI.

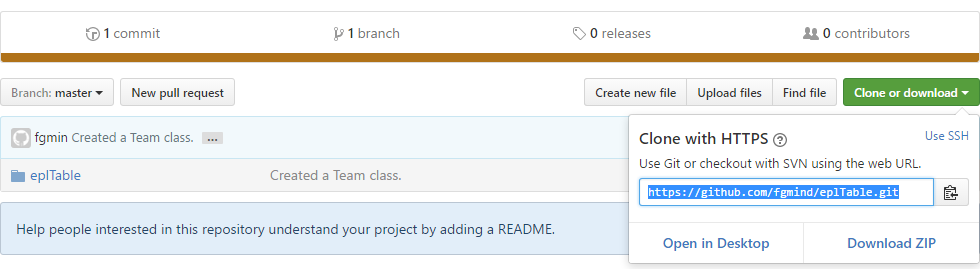


Figure 4: creating repository on GitHub

When they are linked, the local repository can be pushed to the remote one to store the committed changes on GitHub. The figure below shows that the local and remote repositories have the same ID, which means they are the same.

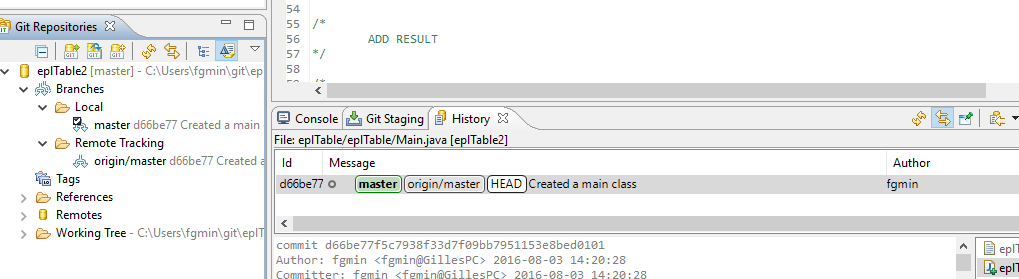


Figure 5: exporting local repository to GitHub

## Importing from GitHub

If we want, we can import a remote repository, in this case from GitHub, to the local IDE, by importing it using the remote repository URI. We then fetch it from GitHub to have the same copy than the cloud one, as shown below.

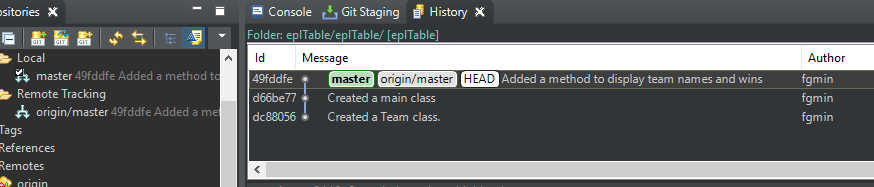


Figure 6: importing remote repository to Eclipse

## Resolving conflicts

In a case where 2 collaborators have fetched a project from GitHub, worked on it and the first collaborator saves and pushes changes back to GitHub, then the second collaborator commits his changes and pushes it to the remote repository, a conflict occurs. This is due to the previous local repository ID of the second collaborator not matching the remote one. In order to resolve such issue he needs to fetch the new remote repository and then push his changes back to the cloud.

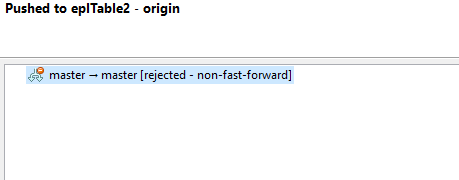


Figure 7: error pushing changes to GitHub

It can also be good to branch the project to create changes, and when satisfied, merge them back to the main branch.

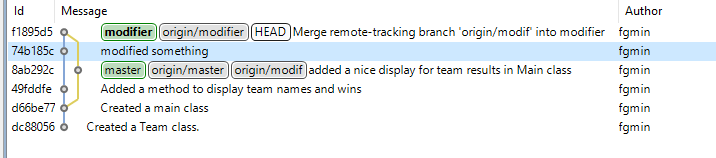


Figure 8: branching and merging changes

# UML Diagram

Unified Modelling language (UML) is a modelling language used to visualise, construct and document classes of a software system. UML is an important aspect involved in object-oriented software development and uses graphic notation to create visual models of software systems (Unified Modeling Language (UML), 2016).

A tool can be imported into Eclipse IDE to display the classes and their relationship. After importing and installing the tool, we can then create a new class diagram for the project.

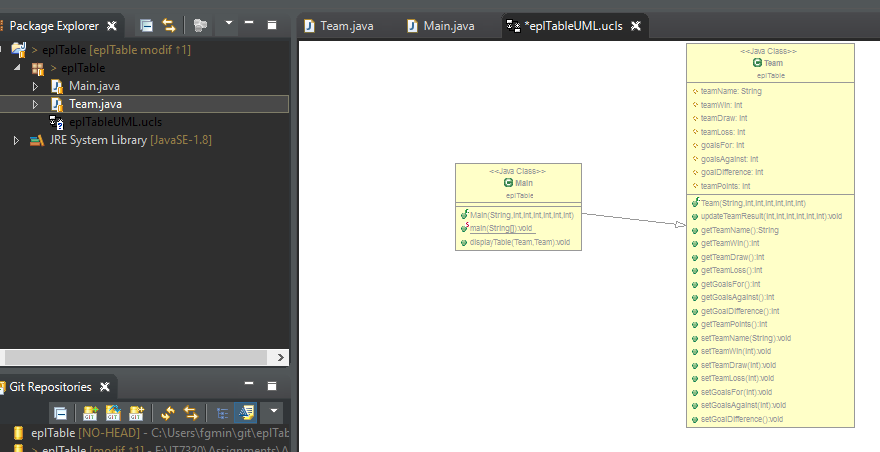


Figure 9: UML class diagram of project

# References

*Git (software)*. (2016, July 31). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Git\_(software)

*Unified Modeling Language (UML)*. (2016). Retrieved from Techopedia: https://www.techopedia.com/definition/3243/unified-modeling-language-uml

# Appendix

## Main class

**package** eplTable;

**public** **class** **Main** **extends** **Team**{

**public** **Main**(**String** name, **int** win, **int** draw, **int** loss, **int** gFor, **int** gAgainst, **int** points) {

**super**(name, win, draw, loss, gFor, gAgainst, points);

**public** **static** **void** **main**(**String**[] args) {

**Team**[] **eplTeams** = **new** **Team**[20];

eplTeams[0] = **new** Team("manUtd", 9, 0, 0, 9, 0, 27);

eplTeams[1] = **new** Team("chelsea", 0, 3, 0, 2, 2, 3);

eplTeams[2] = **new** Team("liverpool", 0, 0, 1, 0, 2, 0);

eplTeams[3] = **new** Team("arsenal", 3, 1, 0, 4, 0, 10);

eplTeams[4] = **new** Team("stoke", 0, 0, 0, 0, 0, 0);

eplTeams[5] = **new** Team("everton", 0, 0, 0, 0, 0, 0);

eplTeams[6] = **new** Team("middlesbrough", 0, 0, 0, 0, 0, 0);

eplTeams[7] = **new** Team("bornemouth", 0, 0, 0, 0, 0, 0);

eplTeams[8] = **new** Team("sunderland", 0, 0, 0, 0, 0, 0);

eplTeams[9] = **new** Team("westbrom", 0, 0, 0, 0, 0, 0);

eplTeams[10] = **new** Team("leicester", 0, 0, 0, 0, 0, 0);

eplTeams[11] = **new** Team("hull city", 0, 0, 0, 0, 0, 0);

eplTeams[12] = **new** Team("swansea", 0, 0, 0, 0, 0, 0);

eplTeams[13] = **new** Team("crystalPalace", 0, 0, 0, 0, 0, 0);

eplTeams[14] = **new** Team("burnley", 0, 0, 0, 0, 0, 0);

eplTeams[15] = **new** Team("watford", 0, 0, 0, 0, 0, 0);

eplTeams[16] = **new** Team("manCity", 0, 0, 0, 0, 0, 0);

eplTeams[17] = **new** Team("southampton", 0, 0, 0, 0, 0, 0);

eplTeams[18] = **new** Team("westHam", 0, 0, 0, 0, 0, 0);

eplTeams[19] = **new** Team("tottenham", 0, 0, 0, 0, 0, 0);

// DISPLAY TABLE

**System**.***out***.println("Rank \tTeam Name \tWins \tDraws \tLosses \tPoints");

**int** **rank** = 1;

**for** (**int** **i** = 114; i >= 0; i --){

**for** (**Team** **eplTeam** : eplTeams) {

**if** (eplTeam.teamPoints == i) {

**if** (eplTeam.teamName.length()< 8 ) {

**System**.***out***.println(rank + "\t" + eplTeam.teamName + "\t\t" + eplTeam.teamWin + "\t" + eplTeam.teamDraw + "\t" + eplTeam.teamLoss + "\t" + eplTeam.teamPoints);

}

**else**{

**System**.***out***.println(rank + "\t" + eplTeam.teamName + "\t" + eplTeam.teamWin + "\t" + eplTeam.teamDraw + "\t" + eplTeam.teamLoss + "\t" + eplTeam.teamPoints);

}

rank++;

}

}

}

/\*

ADD RESULT

\*/

/\*

GO BACK TO DISPLAY TABLE

\*/

}

**public** **void** **displayTable**(**Team** a, **Team** b){

**System**.***out***.println(a.getTeamName());

**System**.***out***.println(b.getTeamWin() +" "+ b.getTeamDraw() );

## Team class

**package** eplTable;

**public** **class** **Team** {

**protected** **String** teamName;

**protected** **int** teamWin;

**protected** **int** teamDraw;

**protected** **int** teamLoss;

**protected** **int** goalsFor;

**protected** **int** goalsAgainst;

**protected** **int** goalDifference;

**protected** **int** teamPoints;

**public** **Team**(**String** name,**int** win, **int** draw, **int** loss, **int** gFor, **int** gAgainst, **int** points){

**this**.setTeamName(name);

**this**.updateTeamResult(win, draw, loss, gFor, gAgainst, points);

**this**.teamWin = win;;

**this**.teamDraw = draw;

**this**.teamLoss = loss;

**this**.goalsFor = gFor;

**this**.goalsAgainst = gAgainst;

**this**.goalDifference = gFor - gAgainst;

// this.teamPoints = points;

}

**public** **void** **updateTeamResult**(**int** win, **int** draw, **int** loss, **int** gFor, **int** gAgainst, **int** points){

teamWin += win;

teamDraw += draw;

teamLoss += loss;

goalsFor += gFor;

goalsAgainst += gAgainst;

goalDifference += (gFor - gAgainst);

}

**public** **String** **getTeamName**() {

**return** teamName;

}

**public** **int** **getTeamWin**() {

**return** teamWin;

}

**public** **int** **getTeamDraw**() {

**return** teamDraw;

}

**public** **int** **getTeamLoss**() {

**return** teamLoss;

}

**public** **int** **getGoalsFor**() {

**return** goalsFor;

}

**public** **int** **getGoalsAgainst**() {

**return** goalsAgainst;

}

**public** **int** **getGoalDifference**() {

**return** goalDifference;

}

**public** **int** **getTeamPoints**() {

**return** teamPoints;

}

**public** **void** **setTeamName**(**String** teamName) { // NOT NECESSARY

**this**.teamName = teamName;

}

**public** **void** **setTeamWin**(**int** teamWin) {

**this**.teamWin = teamWin;

}

**public** **void** **setTeamDraw**(**int** teamDraw) {

**this**.teamDraw += teamDraw;

}

**public** **void** **setTeamLoss**(**int** teamLoss) {

**this**.teamLoss += teamLoss;

}

**public** **void** **setGoalsFor**(**int** goalsFor) {

**this**.goalsFor += goalsFor;

}

**public** **void** **setGoalsAgainst**(**int** goalsAgainst) {

**this**.goalsAgainst += goalsAgainst;

}

**public** **void** **setGoalDifference**() {

goalDifference += (goalsFor - goalsAgainst);

}

}