**Session 03 – Class Activities & Exercises (Branching and sync with GitHub)**

* Look at some Source Forge projects activities and some open source projects official sites. There are many open source project sites e.g. Google code etc. We just use Source Forge site to see what statistics, collaboration and development tools are offered.
* Continue working with the TortoiseGit exercises from last session
* Setup branches in Git server to carry team development.
* Synchronize with the remote GitHub repository.
* Use the revision graph

## Activity 01: (Open Source)

Visit the <http://sourceforge.net> web site. Find out the following information. i.e. Click on More Details. Click Browse,

|  |  |
| --- | --- |
| 3 Most Popular Projects |  |
| 3 Most Downloads (Last Week) |  |
| 3 Most Downloads (Over All-Time) |  |
| 3 Top Best Rating Projects (See Software Directory, Rating) |  |

For TikiWiki CMS: In SourceForge, search for Tikiwiki CMS project. Between the SourceForge site and the official site <http://info.tikiwiki.org>, find out the following:

Look at the Tiki Admin Group: What contribute the admin team? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Latest 3 releases and their dates?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How many downloads has been supported last week? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What is the Rating/Ranking of the project? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name 3 related projects branched from TikiWiki product. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

From the official web site, find out what supports are available to the users and developers communities.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Activity 02: (TortoiseGit) – The exercise shows how to add program components come from another project.

\*\* This session the following activity relies on the activities that you have done in last session.

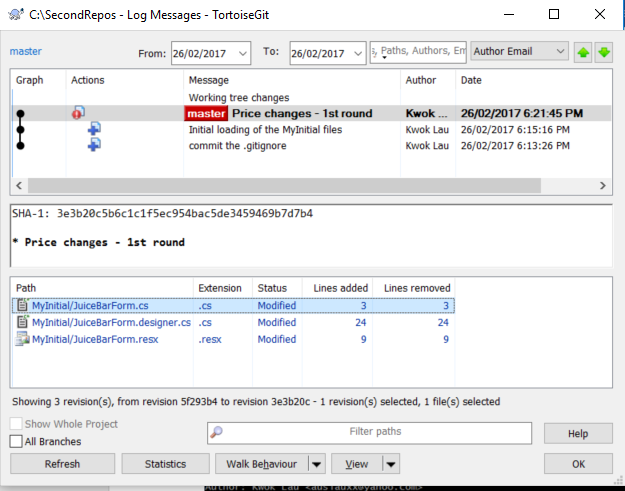
Last session you have set up the TortoiseGit with the **.gitignore** file & set up the local git repository to load the C# project in the repository.

\*\*Note:

Last session, in the **C:\SecondRepos,** we had copied a project, add (staged the files) and commit the file for version control in the local git repository. i.e. You had initially committed the application (MyInitial project in JuiceBar context) into the repository. Then we further developed on the project in the working directory and committed the modification into the repository.

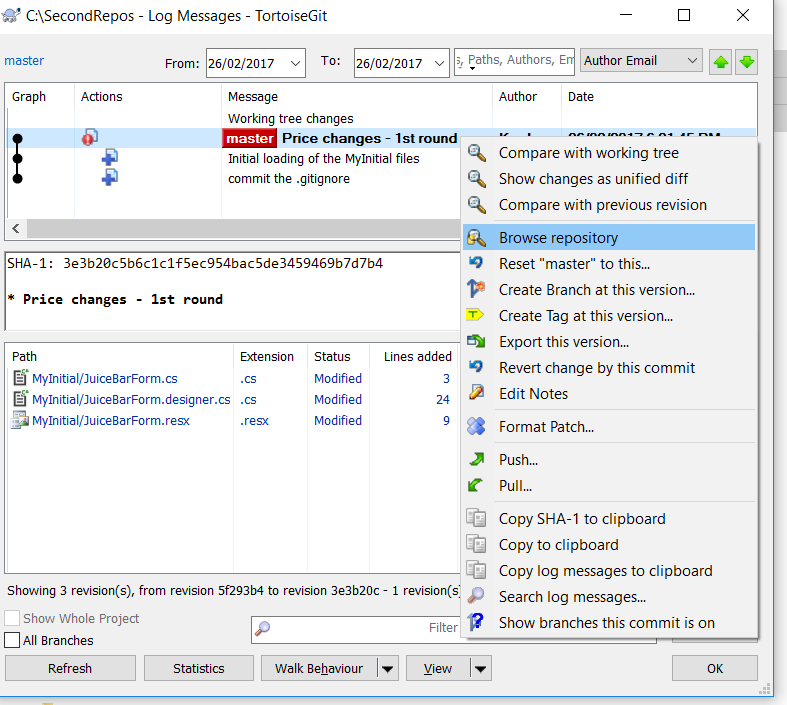
You should able to see the following, if

Right click on **C:\SecondRepos** 🡪 **TortoiseGit** 🡪 **Show Log**

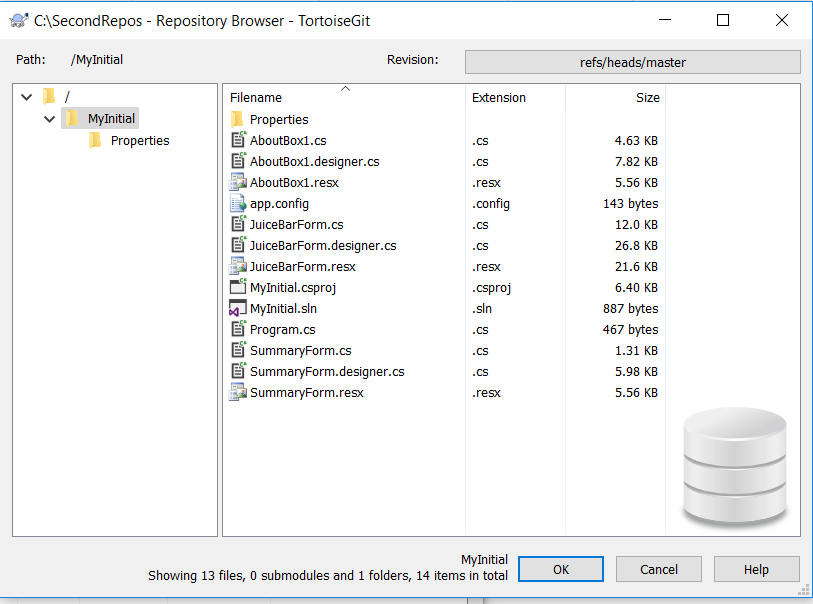


Right click here to browse into the repository.

i.e. Right click on the “**Price Changes – 1st round**” 🡪 Choose **Browse Repository**.



Then you can see your files & directories in the repository as shown. e.g.

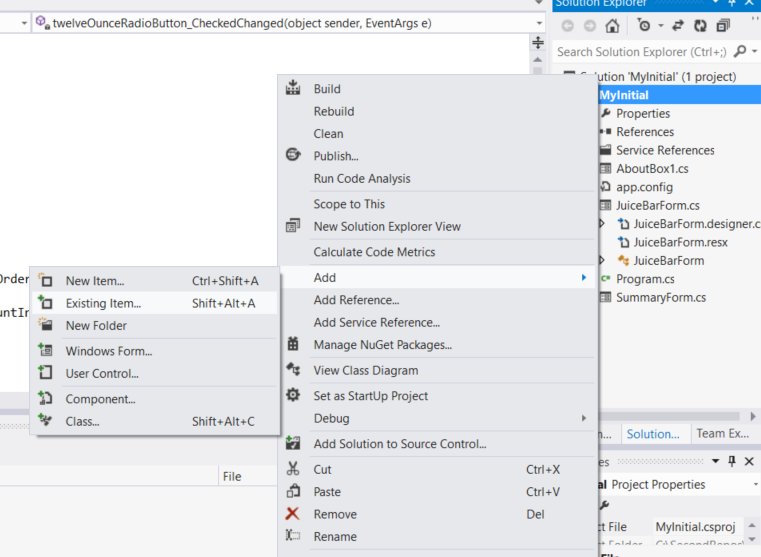


These are the source files for the C# project. You do not see the bin or release folder

Note: These are the source files needed to re-generate the executable codes.

Next is to further develop the project by adding some existing files from another project. Copy the Session03\StudentFiles\Ticketing folder into **c:\temp\TicketEx**.

In **C:\SecondRepos\MyInitial** working folder, open the **MyInitial** project by Visual Studio. In the **Solution Explorer**, right click on the **MyInitial** Application. Choose **Add** 🡪 **Existing Item**. Add the **TicketsForm.cs, TicketsForm.Designer.cs** and the **TicketPrice.cs** files, (i*.e note: add one file at a time*) which can be found from the **Ticketing** project, into the **MyInitial** project.



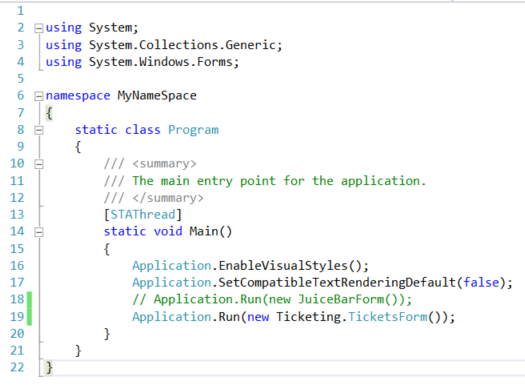
Add existing item i.e. you can use this option to add some components from another project.

(\*\*Note: you do not need to add TicketsForm.resx since it is a generated file)

Change the **Solution name** and the **Application name** from **MyInitial** to **Sales**.

In the **Program.cs** file, change the **Startup** form to be **TicketsForm**. Run the application and **save** all files.

Change the startup form to **Ticketing.TicketsForm.**



Change the startup form in the **Program.cs** file.

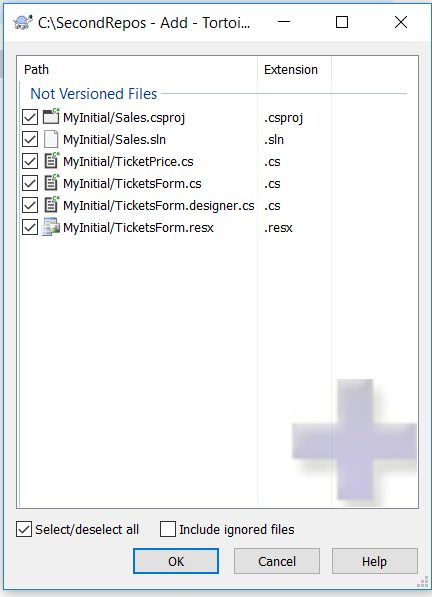
Run the solution. Fix any error if you have. Look at the repository folder, the directory should have an overlay icon **** . It indicates some file(s) has been modified and needs committing.

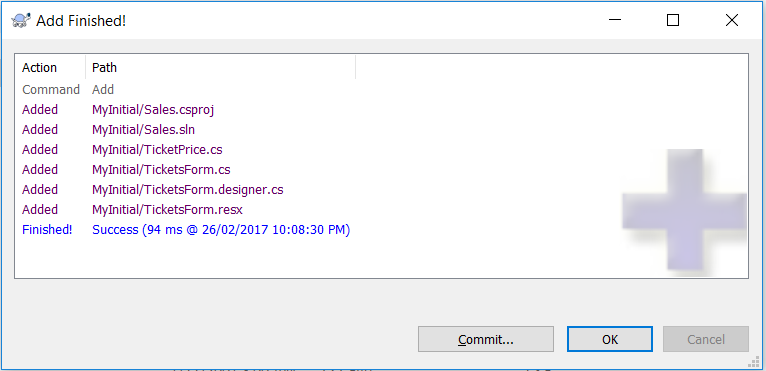
Why the **TicketsForm.cs, TicketsForm.Designer.cs** and the **TicketPrice.cs** files do not have overlay **icons?**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Browse into the sub-directories. What are the files have the overlay icon. **Save** and close the **Sales** Project from Visual Studio.Now you need to add (stage) the files for modification. **Right click on the repository 🡪 TortoiseGit 🡪 Add**.

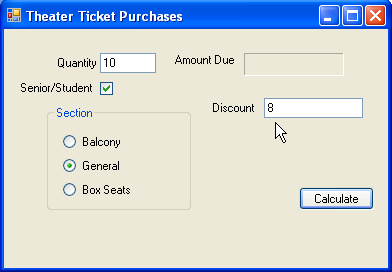




Click **Commit** and type the comments as “ **Added the TicketsForm.cs & designer file and TicketPrice.cs**”.

You should see the repository has overlay icon like this .

Now, you carry out further development with the following requirements.

****

Modify the **Ticket** form by adding a text box called **txtDiscount**. When the user clicks the Section radio button, the $5 discount is displayed in the txtDiscount text box. It allows the user to change the txtDiscount amount e.g. $8, $7, $6 before the user clicks on the **Calculate** button to get the Amount Due.

During the development, you should execute a Commit after you have setup the text box on the form interface. Execute the program. You also should execute another Commit after you have changed the program codes. Execute the program. Doing these, you should be generating different hash index as vision on the files being modified and those changes are committed to the repository.

Are you able to see the files in the repository with the time-stamps? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Are there any files in the ignore list exist in the repository?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Activity 03: (TortoiseGit – Synchronize with Remote Repository)**

*Concept behind in synchronizing local Git repository with the remote GitHub repository:*





*Local Repository “ThirdRepos”. TortoiseGit client uses the private key.*

*Need to use PuttyGen to create a private & public key pair.*



***Git Sync***

*Remote Repository “ThirdRepos”. GitHub uses the Public key.*

*Typical Git workflow:*  


*There are different types of branches we may use are:*

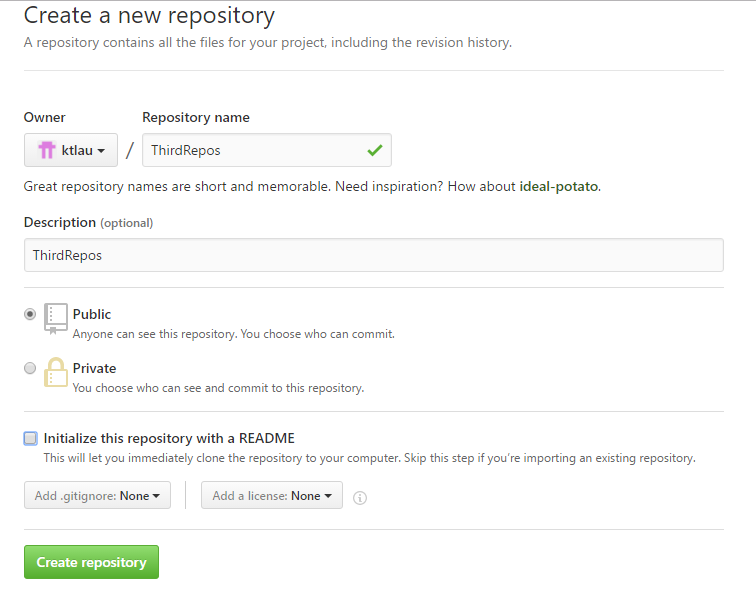
* *Master branches (original)*
* *Feature branches*
* *Release branches*
* *Hotfix branches*

*Each of these branches have a specific purpose.*

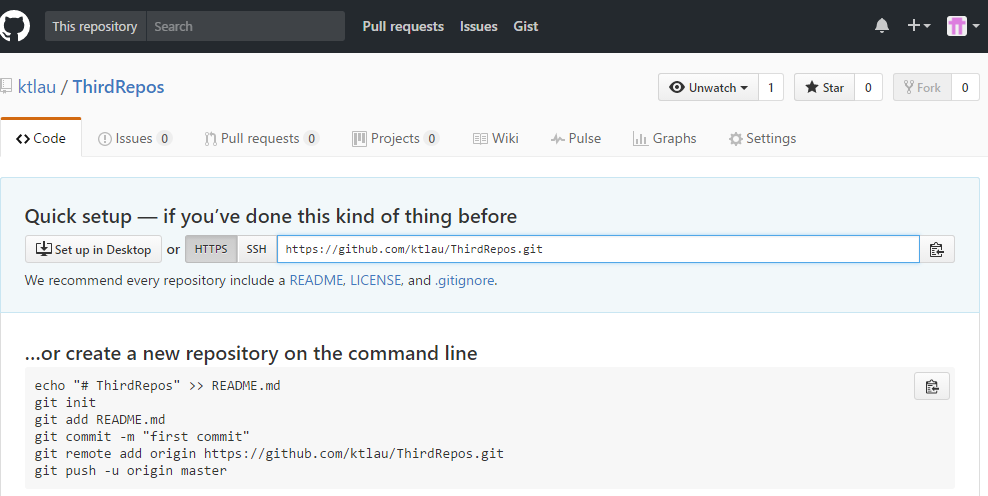
* ***master****: Original branch. Often it branches off to* ***develop*** *🡪* ***feature branches*** *🡪* ***develop*** *🡪* ***release*** *🡪* ***master.*** *Once the quality assurance is satisfied with the quality, the release branch is merged into* ***master*** *(and also back to develop). This is then what is shipped/used by the customers.*
* ***feature:*** *All features / new functions / major refactorings are done in* ***feature*** *branches, which branch of and are merged back from/into the* ***develop*** *branch (usually after some kind of peer review).*
* ***release:*** *When enough features have accumulated or the next release time frame comes near, a new release branch is branched of from develop, which is solely dedicated to testing/bugfixing and any cleanup necessary (e.g. changing some path names, different default values for instrumentation etc.).*
* ***hotfix:*** *If a major problem is found after release, it's fix is developed in a* ***hotfix*** *branch, that is branched off from master. Those are the only branches that will ever branch of master.*

*Note: Any commit in* ***master*** *is a merge commit (either from a release or a hotfix branch) and represents a new release that is shipped to the customer.*

1. Register an account in the GitHub (<https://github.com/>) if you have not done so.
2. Create a new repository called “**ThirdRepos**” in your GitHub account. Later we are going to sync the local repository to this remote repository in GitHub.

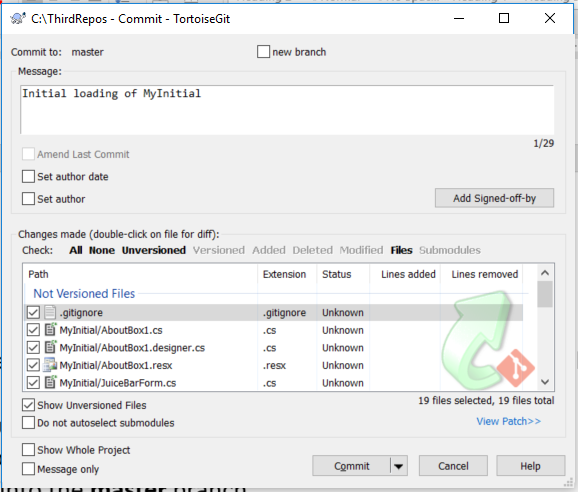


Choose Settings 🡪 Repositories 🡪 Choose your repository 🡪

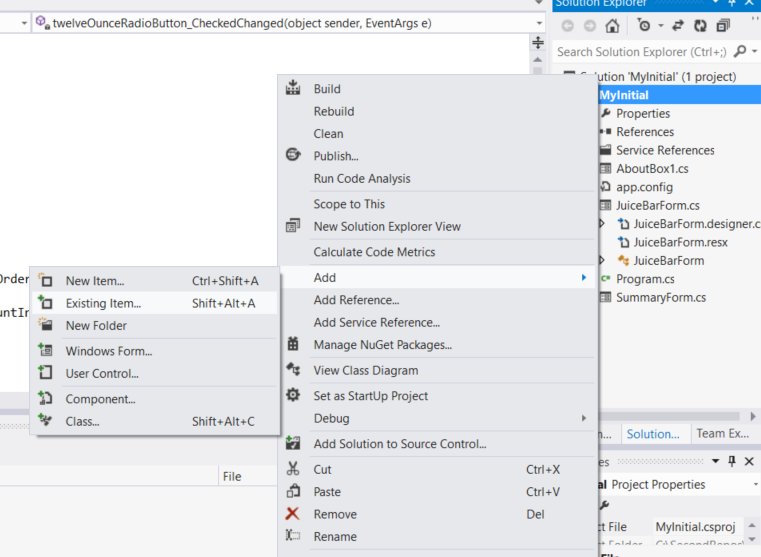




1. Create a local repository. In Windows Explorer, right click on c:\, create a local repository called **c:\ThirdRepos**.
2. Copy the **.gitignore** file into the repository directory. (Note: .gitignore available from Session 2)
3. Copy the **MyInitial** solution files into the repository.
4. Add (stage) the files for modification.
5. Commit the **MyInitial** into the **master** branch. (Note: remember to add message.)



1. In **C:\ThirdRepos\MyInitial** working folder, open the **MyInitial** project by Visual Studio. In the **Solution Explorer**, right click on the **MyInitial** Application. Choose **Add** 🡪 **Existing Item**. Add the **TicketsForm.cs, TicketsForm.Designer.cs** and the **TicketPrice.cs** files (i.e note: one file at a time) which can be found from the **Ticketing** project into the **MyInitial** project.



We deliberately change the solution name is to create some changes to the codes for merging later on and it also reflects what the program do.

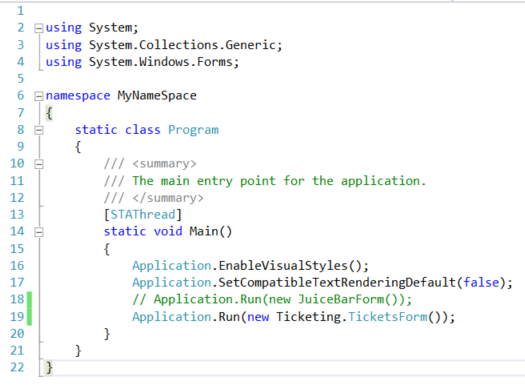
Add existing item i.e. you can use this option to add some components from another project.

(\*\*Note: you do not need to add TicketsForm.resx since it is a generated file)

Change the **Solution name** and the **Application name** from **MyInitial** to **Sales**.

In the **Program.cs** file, change the **Startup** form to be **TicketsForm**. Run the application and **save** all files.

Change the startup form to **Ticketing.TicketsForm**.



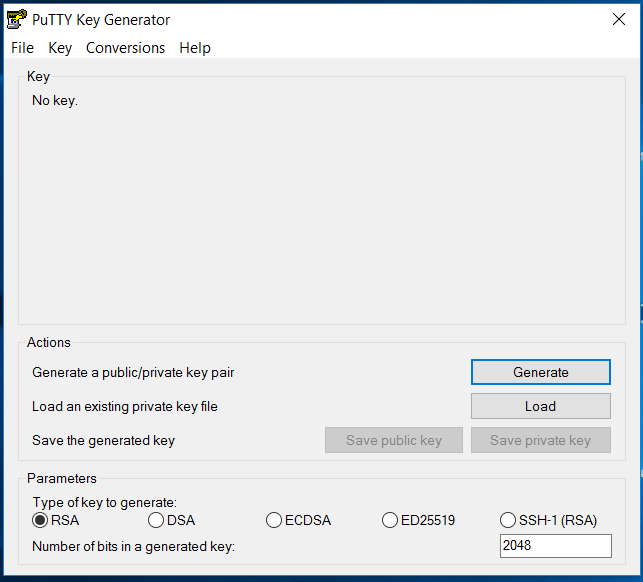
Change the startup form in the **Program.cs** file.

1. Run the solution. If no error, in TortoiseGit, **add** (stage) the changes and **commit** the changes to local repository. Remember to add message such as “**Added the Ticketing.TicketsForm & TicketPrice.cs**”
2. PuTTYgen allows you to generate a **secure shell (SSH) key pair (i.e. GNU Privacy Guard – or GPG).** You can save the **private key** on your local machine and provide the remote server you want to access with a copy of the **public key**. It would be good that you copy the public key and save it as a text file for later use.

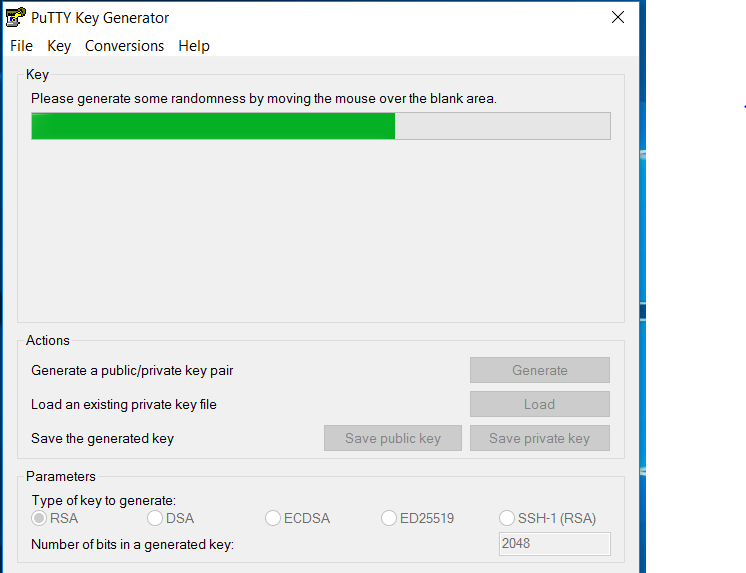
Download and install Putty and WinSSC from the official sites **(Note: only if there are not installed in your computer)**. They are also available in the L: drive under L:\its\files\applications folder as well.

1. In order to generate your key pair, do the following:

On the Windows application menu, choose application **puTTY** 🡪 run **puTTYgen** application.

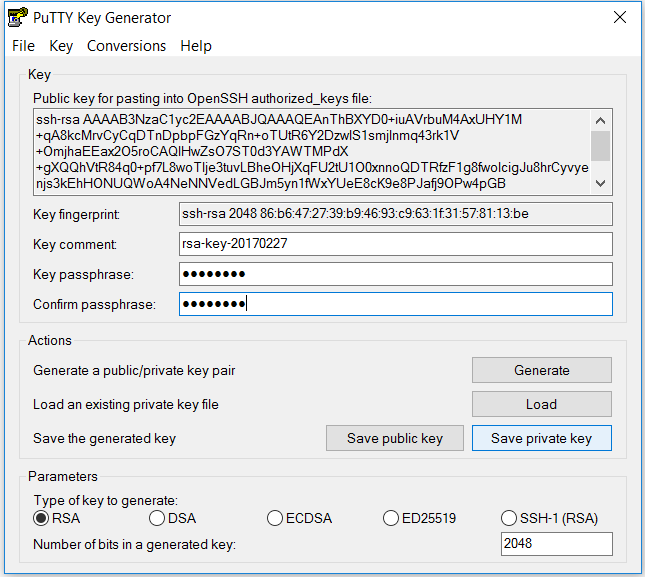


1. Click on the **Generate** button. Wiggle the cursor under the green bar to generate some randomness used to encrypt your private key. Once the green bar has filled up PuTTYgen will have created your key pair.



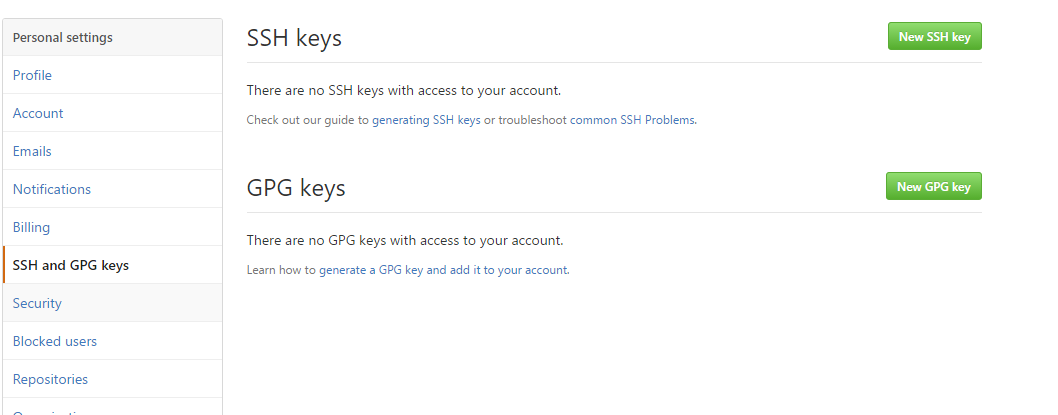
*Move the mouse to generate some random input*.

1. Save your key pair. Firstly, make sure that you protect your private key with a suitable pass phrase (e.g. mypassword etc.) by typing it into the Key passphrase and Confirm passphrase inputs. You should then click **Save private key** button and save the key with a **.ppk** extension. Make a note of where you have saved the private key file.



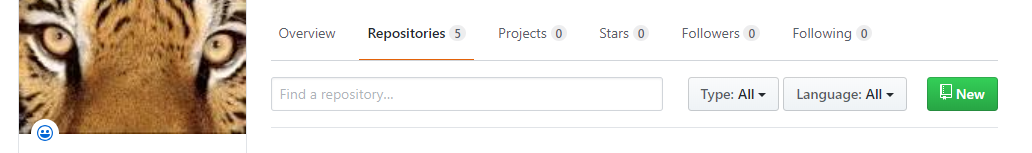


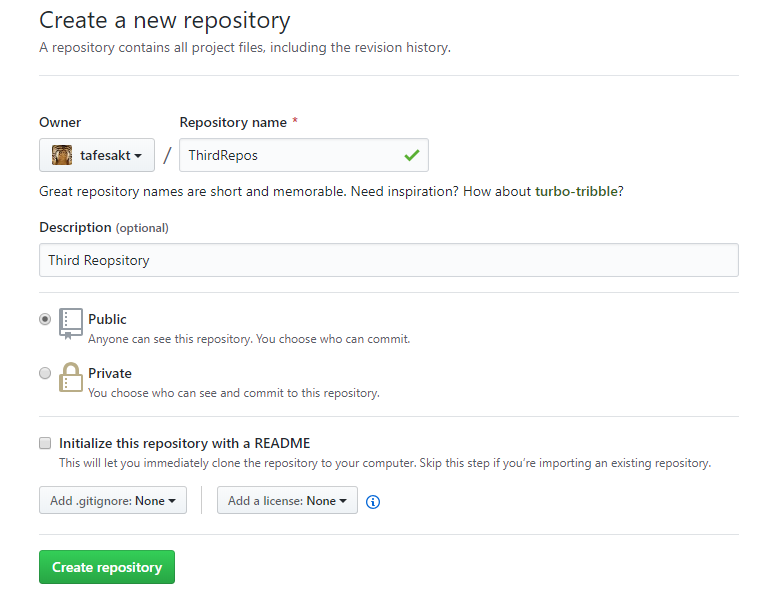
1. You need to provide server with a copy of your new SSH public key. Select and copy the text in the "Public key” for pasting into the **Setting 🡪 SSH and GPG key** section on your account in your GitHub. You may need to provide your GitHub account password to complete this.





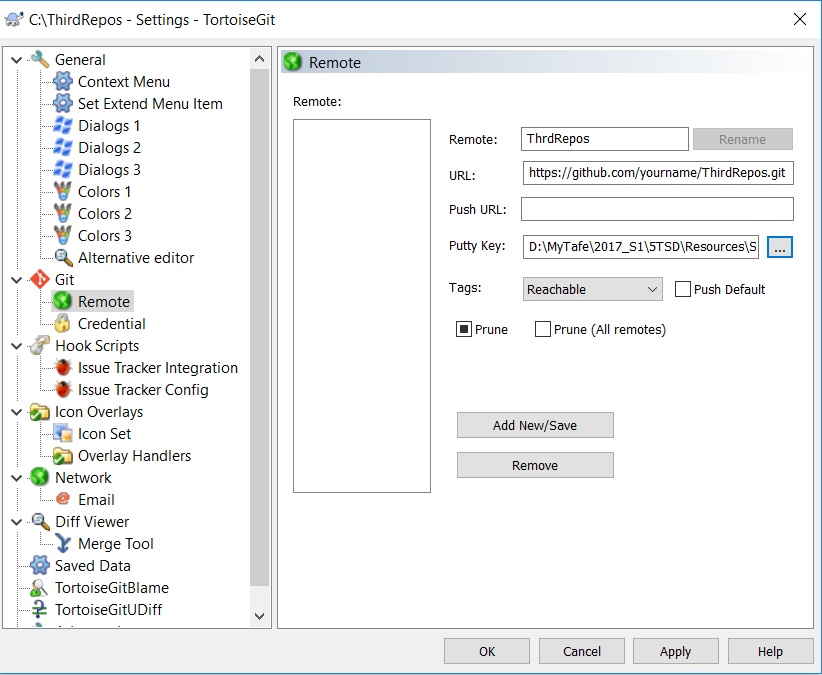
In the GitHub, create a repository called ThirdRepos 🡪 Your repositories, click the **New** button.





1. Let’s push the local repository to the GitHub after commit to the master branch. On the TortoiseGit settings 🡪 Git 🡪 Remote. Enter the following information:

* **Remote:** ThirdRepos
* **URL:** [https://github.com/*yourname*/ThirdRepos.git](https://github.com/yourname/ThirdRepos.git)
* **Putty key:** the file path of your private key *xxxxxxx.ppk*.



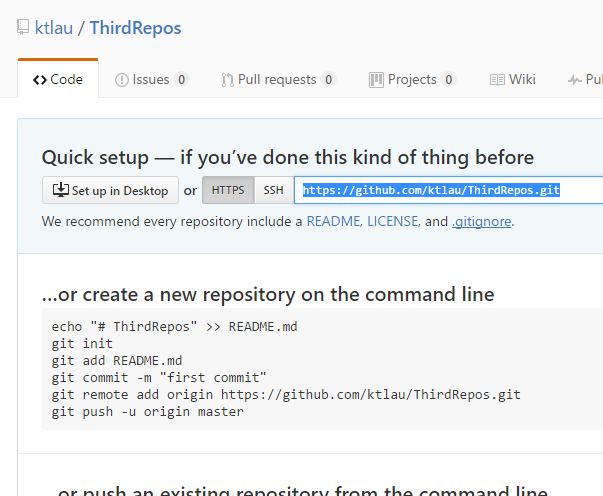
*Substitute with your name.*

Click **OK**.

1. **Using SSH keys with TortoiseGit client.**

You can specify the Key Location for TortoiseGit the following way:

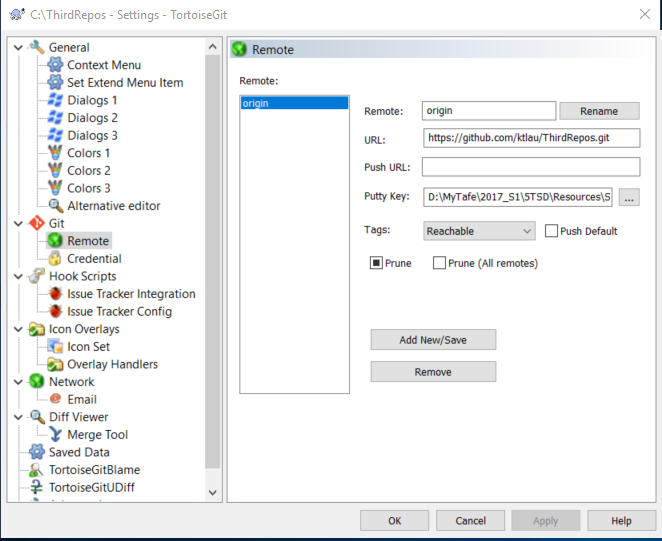
* Open an Explorer Window.
* Open the Contextmenu and Navigate TortoiseGit > Settings
* In the now opened window Navigate to Git > Remote
* Set the Path to your Putty Key in the corresponding Input Box. (See the following picture for TortoiseGit setting). Copy the SSH link from the GitHub repository. i.e. <https://github.com/yourname/ThirdRepos.git>.



You should use the SSH url address for user authentication.

In the Setting 🡪 Git 🡪 Remote 🡪

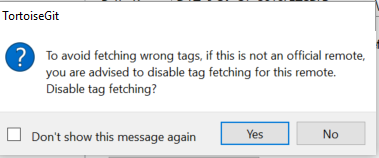
Fill the following:



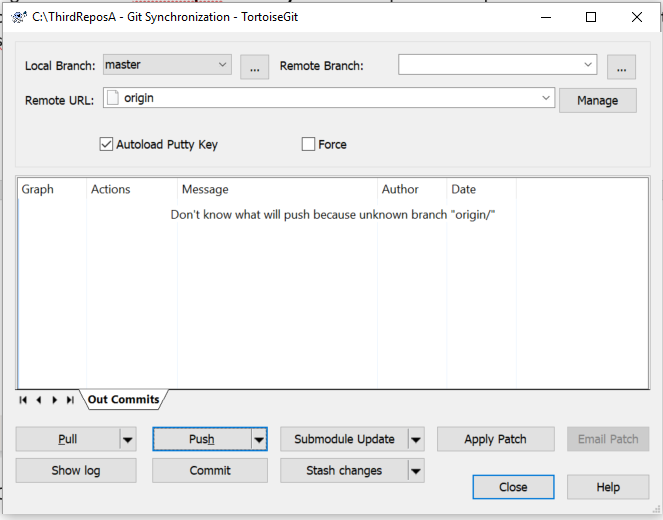
Use the SSH url instead.

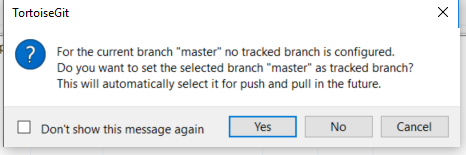
Just paste the URL of the GitHub repository. Leave the remote name blank and it will fill it as “origin” by default.

click the **Add New/Save button**. Choose **Yes** for the following option.

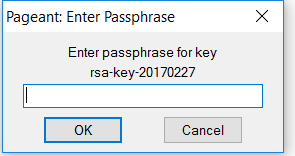


1. Right click on the **ThirdRepos** 🡪 **Git Syn**. Since the previous step had set the Remote URL as origin (you can click the **Manager** button to view the origin settings), click the **Push** button to syn the local repository with the remote repository.

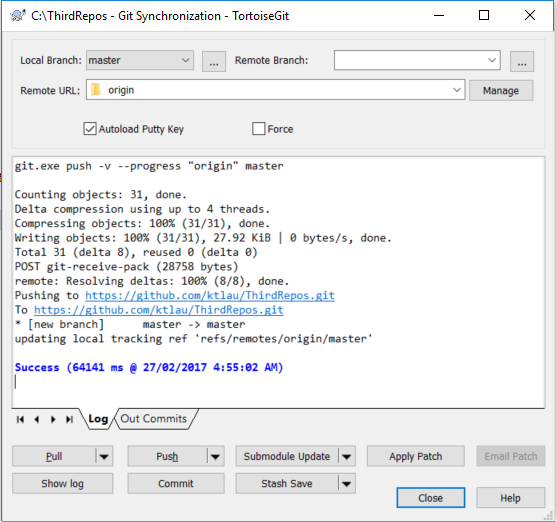




Click **Yes**.



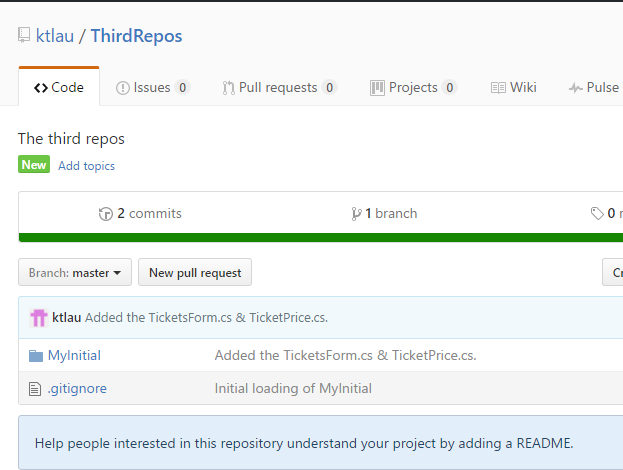
Enter your passphrase i.e. mypassword etc. & the GitHub login.



If you got error “git did not exit cleanly (exit code 128)”. It’s probably because your SSH key has been compromised. Make a new one and add it to your GitHUB account. Then try the push again.

*\*\* You should always keep your key pairs & the passphrase together.*

Click **Close.** Visit the GitHub to see whether the project has been pushed to the remote repository.



**Activity 03: (TortoiseGit – Branching)**

1. This activity follows on from the Activity 02.

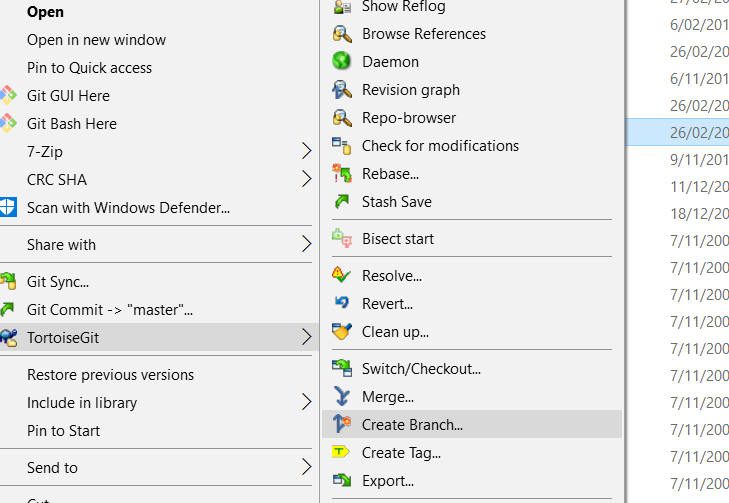
master

feat1.1\_ticket\_type feat1.2\_discount\_type

To create a branch, make sure all changes have committed back to the repository.

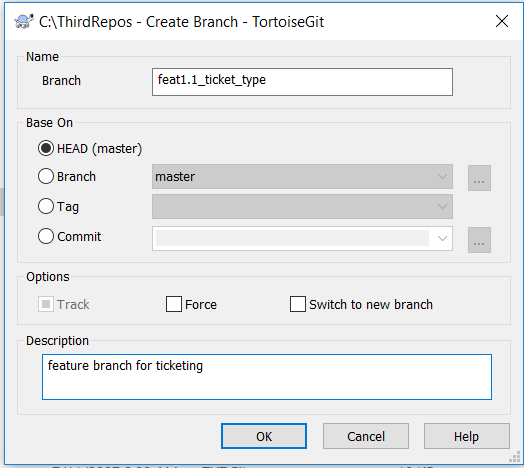
Right click on the repository 🡪 **TortoisGit 🡪 Create Branch**

Assuming that the working directory is checkout from the revision that you want to branch out. In our case is the head revision of the **master** branch at the moment.

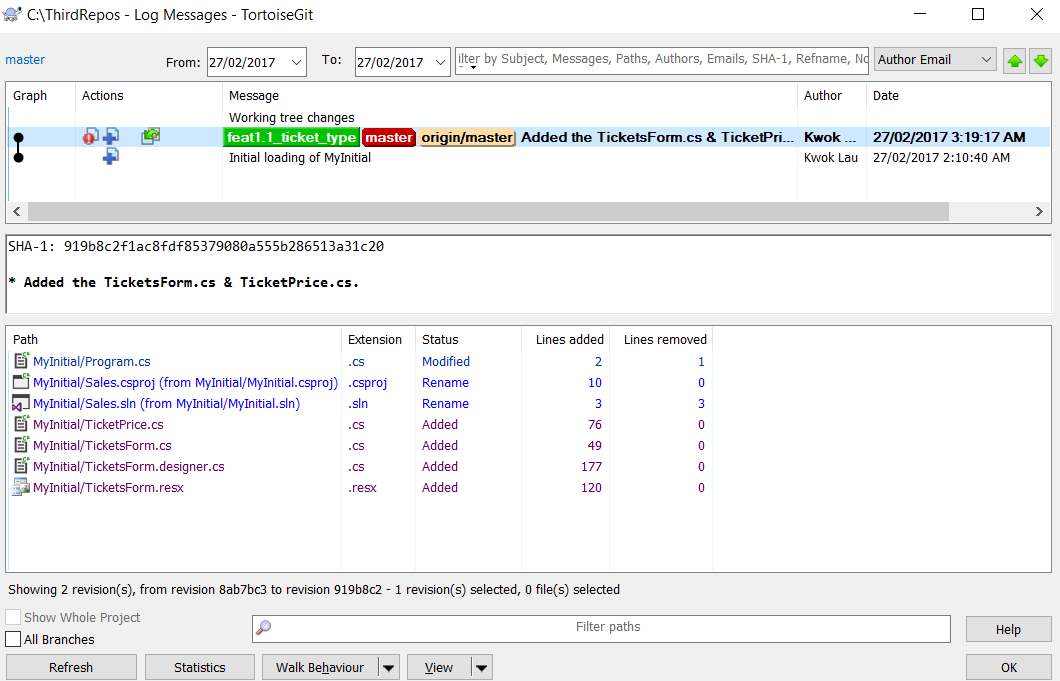


Enter the name of the branch as: **feat1.1\_ticket\_type**

Enter Description: **feature branch for ticketing**



See what has happened by using the **Show log**.

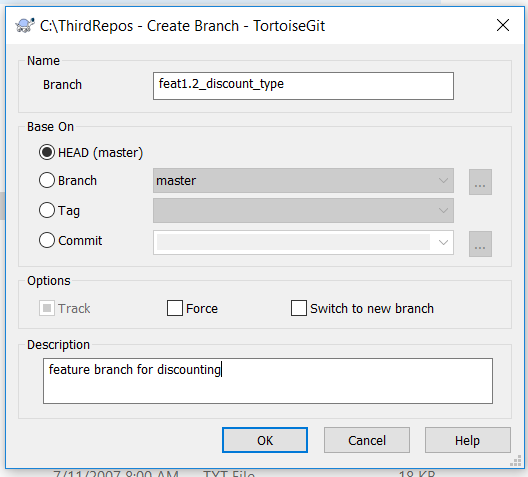


We are also creating another branch for **feat1.2\_discount\_type** based on the head revision of the **master**.

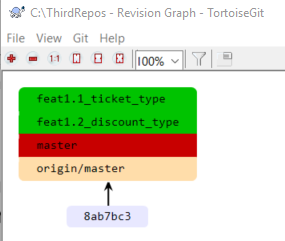
Right click on the repository 🡪 **TortoisGit 🡪 Create Branch**

Enter the name of the branch as: **feat1.2\_discount\_type**

Enter Description: **feature branch for discounting**



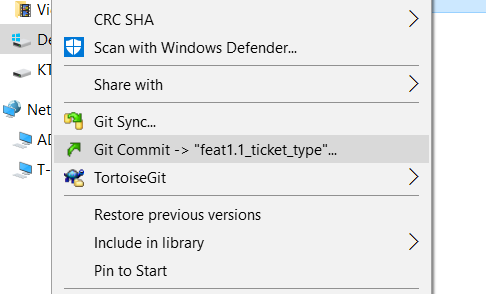
TortoiseGit 🡪 Revision Graph, you should be able to see:



You can switch between the branches. i.e. **TortoiseGit 🡪 Switch/Checkout**.

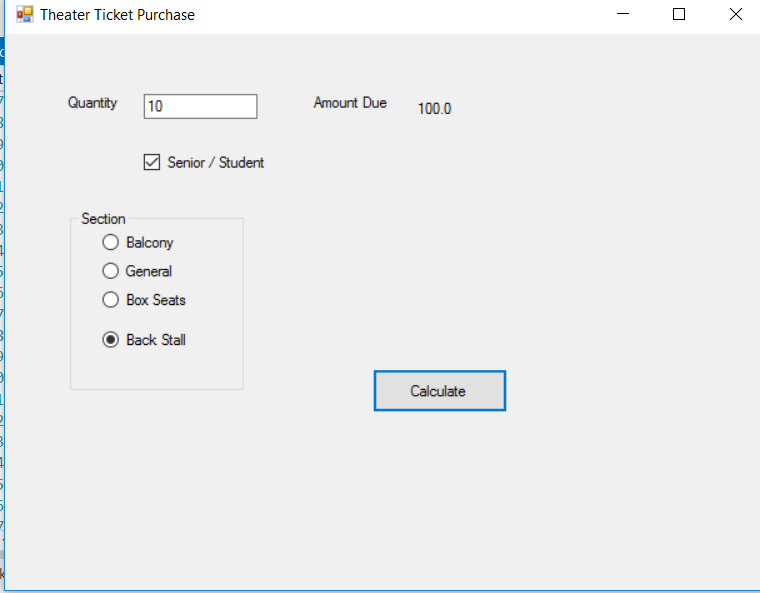
Let’s develop the two branches independently.

To develop **Feat1.1\_ticket\_type**: Right click on the repository folder, Switch/Checkout to the **feat1.1\_ticket\_type** branch.

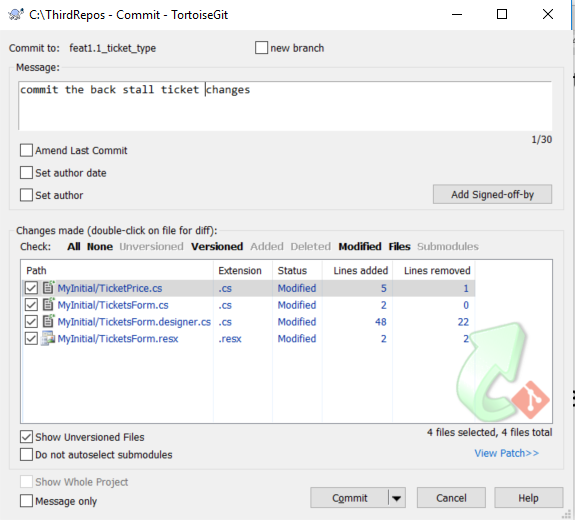


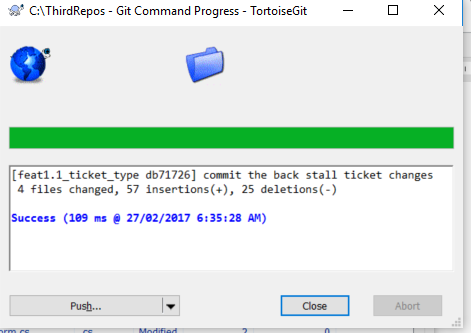
The working folder is pointing at the head revision of the feat1.1 branch now.

Your development task is to modify the ticket form to include another ticket type called **Back Stall**. The price is **15.0m**. e.g.

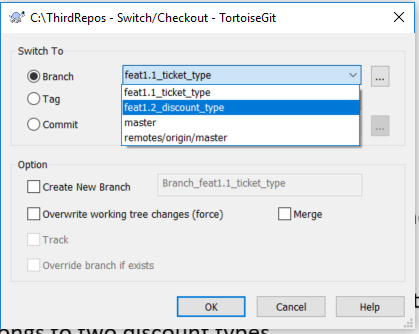


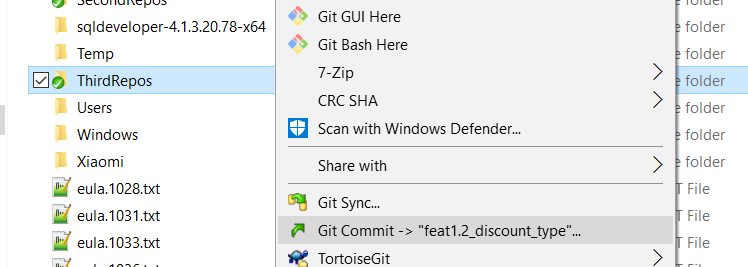
Choose **TortoiseGit 🡪 Commit**. Add the commit message as “**commit the back stall ticket changes**”.





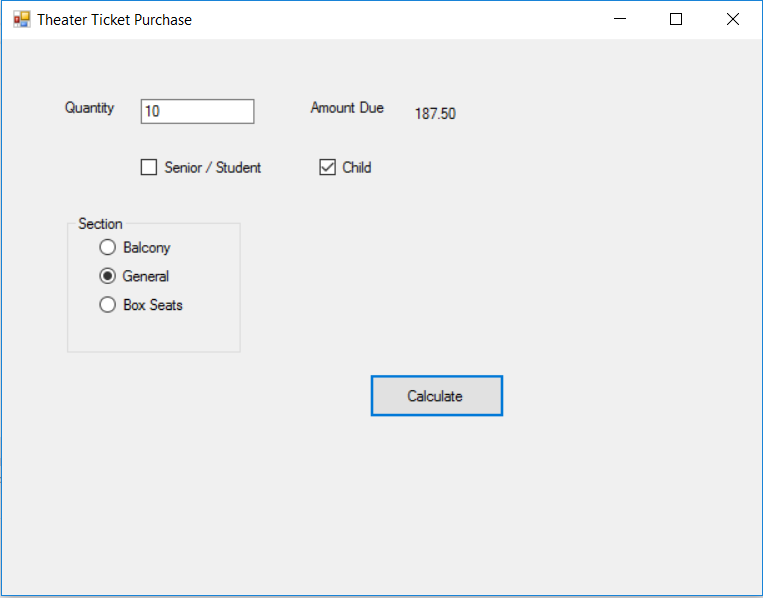
To develop **Feat1.1\_ticket\_type**: Right click on the repository folder, Switch/Checkout to the **feat1.2\_discount\_type** branch.



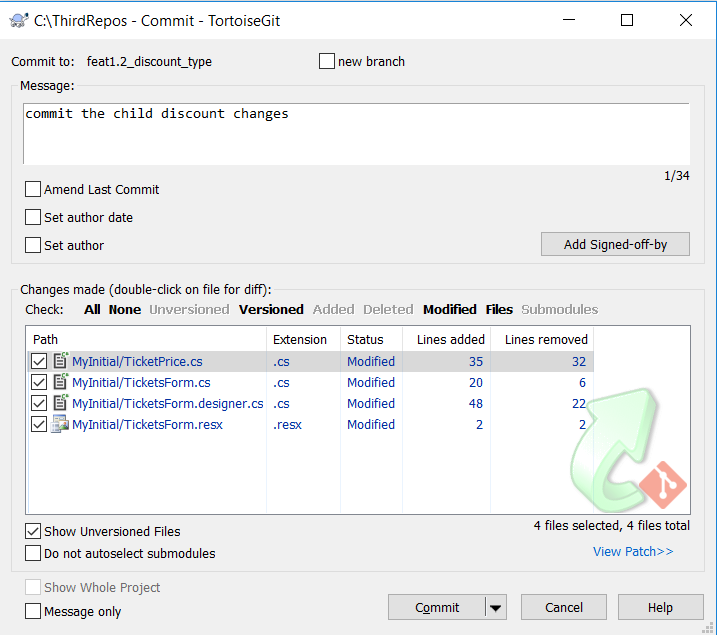


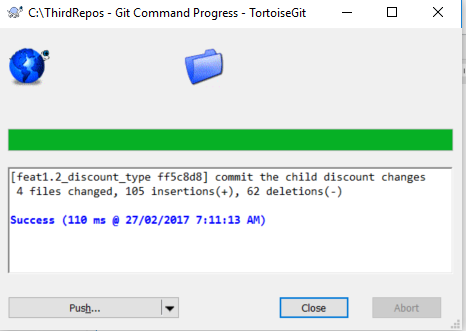
The working folder is pointing at the head revision of the feat1.2 branch now.

Your task is to modify the ticket form to include another discount type i.e. **Child below 12** has a **discount 10m**. No person belongs to two discount types. e.g.

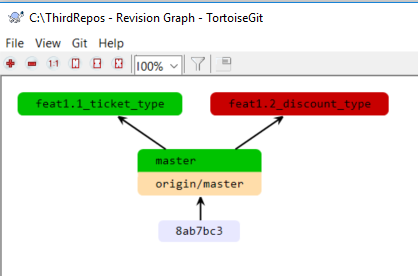


Choose **TortoiseGit 🡪 Commit**. Add the commit message as “**commit the child discount changes**”.





Look at the Revision Graph. It shows both branches came off from the master branch.

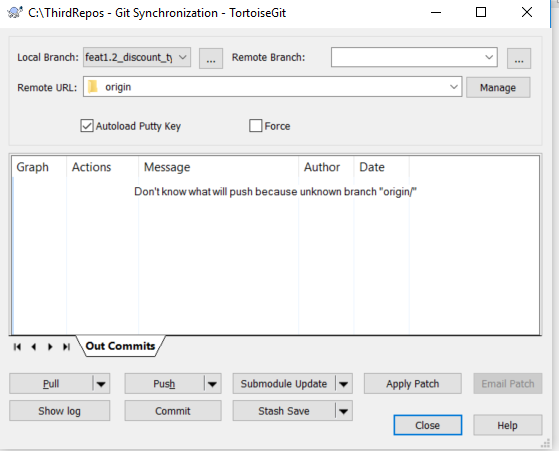


TortoiseGit Color means:

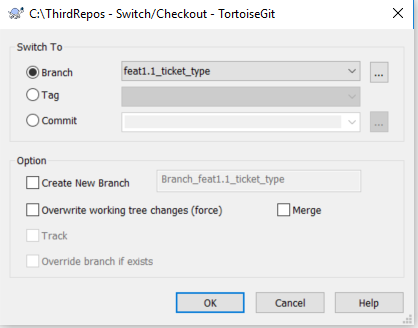
* Current Branch: Red
* Local Branch: Green
* Remote Branch: Peach
* Tag: Yellow
* Stash: Gray

Now let’s push all committed changes to the GitHub.

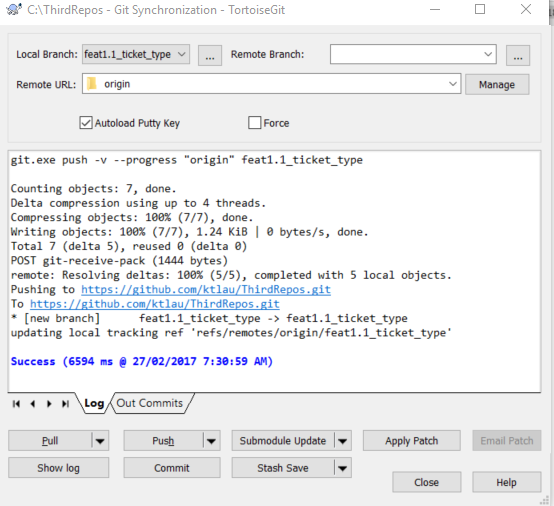
* We do the feat1.2 first since the working folder is pointing at it. Click **Git Syn** 🡪 **make sure that it is feat1.2\_discount\_type** 🡪 Click **Push.**



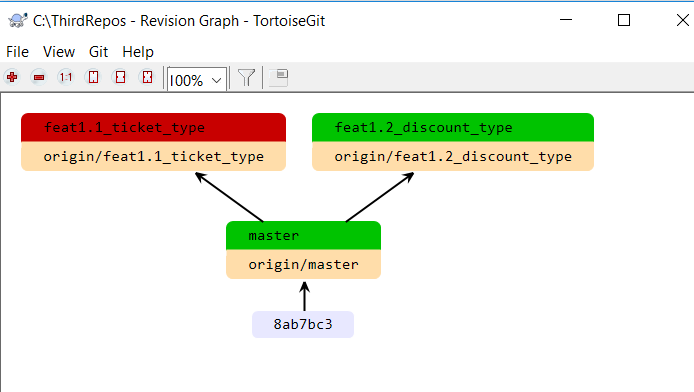
* Now we do the feat1.1. Choose **TortoiseGit** 🡪 **Switch/Checkout** 🡪 **Choose feat1.1\_ticket\_type** branch.



Click **Git Syn** 🡪 make sure that it is **feat1.1\_ticket\_type** 🡪 Click **Push.**



Finally see the Revision Graph.



Research: What is a node on the Revision graph? Hints: See the help menu on the TortoiseGit.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_