Assign-02

The DisneyCharacter Class

Believe it or not, you have all had the lessons necessary to begin writing your own classes! One of the scariest things is to do it for the first time … this exercise will ease you through it. In order to minimize the scary factor – you will be developing a class to model Disney Characters (Mickey, Minnie, etc.) – because hey – there's nothing scary about Mickey Mouse! (Right? ☺)

In this exercise, you will be

- Using source-code repositories again (as you did in SEF last semester) to store your code for this assignment
- Developing 3 source files to store in your repository
    - DisneyCharacter.h – the file that contains the class definition for this exercise
    - DisneyCharacter.cpp – the source file that contains the method bodies (the code) for the class
    - testDisneyCharacter.cpp – your *testharness* (main function) which you will use to test your class

The requirements description below give you details on the DisneyCharacter class in terms of the data-members (attributes) of the class as well as the methods.

STEP 1 – Connecting to your Source Repository and Checking in the Initial Solution

1. Follow the instructions in the Creating-A-RiouxSVN-Repository document as well as the Installing-Tortoise-SVN document in order to create and prepare your computer for connecting to your personal repository code space.
2. As instructed in the Installing-Tortoise-SVN document in eConestoga, create a local directory on your computer (or even one of the lab computers) and call it **C:\SETRepo**
3. Connect to your repository (remembering your repository's URL and your login credentials using the Tortoise SVN client and perform an *SVN Checkout*)
4. Now start Visual Studio (VS) and create an **empty** project called DisneyCharacter in the **C:\SETRepo** directory (as show in Diagram 1 at the end of this document)
5. Let's setup the skeleton of the source files for this project …
    - Add a file called DisneyCharacter.h to the *Header Files* section in the VS Solution.
    - Add a file called DisneyCharacter.cpp to the *Source Files* section in the VS Solution and
    - Also add a file called testDisneyCharacter.cpp to the Source Files
    - This is shown in Diagram 2 at the end of this document – just so that the files weren't empty, I placed a single line comment at the start of each file. You do the same.
6. The reason I wanted to ensure that the source files weren't empty is because I want to **add** and **commit** the files to my repository now
7. When adding VS solutions to a repository – you ***do not want to add all files*** that are generated by VS. This is because some of the files change each and every time you open the solution, or are different on different people's computers …
    a. Imagine you were working on a project with a number of other people – all from the same source files. If all VS files were committed – then each and every member of the team would have commits to do every time they opened the project – even if they made no source code changes!!
8. So now – save the VS solution and project and exit VS. Go to the C:\SETRepo\DisneyCharacter folder and erase the DEBUG folder. Then go into the C:\SETRepo\DisneyCharacter\DisneyCharacter folder and erase that DEBUG folder. You do not need or want these 2 folders in your golden copy of the repository,
    a. As well, if you venture into the C:\SETRepo\DisneyCharacter\.vs folder (which is hidden) and drill further down into the "…DisneyCharacter\v15\" folder, you may very well see a folder titled **ipch**. Erase this folder – this is where VS hides the incremental pre-compiled header files … these files take up unnecessary space in your repository.

b. Once you've erased these files and folders – go back to the C:\SETRepo folder and right-click on the top-level **DisneyCharacter** folder (as shown in Diagram 3)
   i. Choose the *Tortoise-SVN* option from the pop-up and the *Add* option from the sub-menu – follow the prompts
   ii. Right-click on the same folder again and select *SVN Commit* – remember when prompted for a commit comment – we want to follow best practices – so give it a comment … perhaps "initial commit of DisneyCharacter solution and project directory structure and files"

9. Now your VS Solution and Project are ready to go, you have skeleton source files and they have been added to your repository …
   a. You are ready to proceed to Step 2 below – the class definition requirements and coding …
   b. Please refer to Step 3 below for instructions on saving (updating) your code in your repository (if you complete this exercise over multiple programming sessions). This section also discusses the *due-date* for this exercise.

STEP 2 – The Requirements and Programming

This should be a simple one … I want you to write a class definition for a class to be known as **DisneyCharacter**. Here are the particulars of the class:

The Class Particulars

- The class has the following data members
  - name
    - a *string*[1] of maximum 50 characters
    - if upon construction of the DisneyCharacter object or modification of the name data-member, the name is being set to a *string* longer than 50 character – you need to truncate the string at 46 characters and append the character sequence " …" to the end (that is a leading space before the "…"
  - creationDate
    - a *string* of the format yyyy-mm-dd (example value "2012-01-30")
  - numMovies
    - an integer representing the number of movies that the character has been in
  - whichPark
    - a single character value indicating which of the DisneyWorld / DisneyLand parks the character can be found in – allowable values are:
      - 'M' for Magic Kingdom
      - 'S' for Disney Studios
      - 'A' for Animal Kingdom
      - 'E' for Epcot
      - 'C' for California Adventure
      - 'N' to indicate the character is not placed

- The class should have the following methods
  - a constructor which takes values for all 4 attributes and sets them

[1] You can choose to use a char[] for this data member or a string object … either way, you still need to be able to limit the maximum length to 50 characters …

Assign-02

The DisneyCharacter Class

- need to ensure that the name *string* length is not exceeded – otherwise do as required
- need to ensure that incoming value for whichPark data member is valid. If not, mark the character as *Not Placed*.
- need to ensure that incoming value for numMovies is greater than or equal to zero. If not, set to zero.
- need to ensure that incoming value for creationDate is in the required format. If not, leave blank

o a constructor which takes only the **name** and **creationDate** attributes and defaults the **numMovies** attribute to a value of 0, and the **whichPark** attribute to 'N'

- need to ensure that the name *string* length is not exceeded – otherwise do as required
- need to ensure that incoming value for creationDate is in the required format. If not, leave blank
- no other input validation is necessary in this constructor

o a destructor

- which prints out message to say "<name attribute> destroyed." When invoked
  - e.g if the DisneyCharacter is "Tigger" then it states "Tigger destroyed"

o accessors for *each* of the attributes

o a mutator for each of the **numMovies** and **whichPark** attributes

- your mutator's – like good mutators should perform validation on the incoming parameter value
- and only set the data-member's value if the incoming value is valid
- if the value is not valid – then the current value for the data-member is left alone
- your mutator should also return a ***bool*** data-type to indicate whether the incoming value was valid or not

o also declare the following public methods

- ShowInfo(void) – this method is called in order to print out the current value of all of the data members for the object
  - In presenting the information to the user … ***think like the user*** … what makes sense – what would be understandable to them?
- PlaceCharacter(char whichPark) – this method is called in order to place or position the character at a particular park
- SameMovies(DisneyCharacter& anotherCharacter) – this method is called upon to set the number of movies that this Disney Character has been in to the same number of movies as the specified character (from the parameter list)

- Please follow and use *best practices* in terms of **encapsulation**
  - If you are not sure what I mean by this requirement – ask me …
- Place only the class definition in the .H file and **<u>place all of the method body code in the .CPP file</u>**
- Commenting the source code is ***not required in this exercise***

<u>The Test-Harness Particulars</u>

- Your *testharness* source code module (i.e. your main() function) is called upon to use the DisneyCharacter class interactively. In this code:
  - Instantiate a DisneyCharacter object called mickey (i.e. name="Mickey", creationDate="1929-01-01", numMovies=100, whichPark='M')
  - Instantiate a DisneyCharacter object called minnie (i.e. name="Minnie", creationDate="1930-01-01")
  - try setting the number of Minnie's movies equal to Mickey's by calling the SameMovies() method
    - <u>Question</u>: Which object (Minnie or Mickey) is this method called on – and which object (Minnie or Mickey) is passed into it as a parameter?
  - try dumping out the object's member values by calling ShowInfo() – for both Mickey and Minnie

- try placing Minnie in the Epcot park.

STEP 3 – Putting your Code back into the Repository

Once you're done programming the exercise – or just want to take a break … you need to save it to your repository. So save all open files in VS and exit it.

- Using Windows Explorer – go back to **C:\SETRepo**
- You should notice that the Tortoise SVN icons on the folders and files have changed
  - Please note that depending on the College network – this may or may not work. There is a background process that runs on the computers called "TSVNCache" which is always connecting to your repository and seeing if your local copy of the files has changed or is different than what is in the repository
  - You may see something like Diagram 4 (at the end of this document)
- When committing your source code changes to any repository, *best practices* say that you **need to commit each individual source code file** (i.e. each .H and .CPP (or whatever language you are programming in)) **by itself**
  - We do this so that our commit message to the repository (which becomes part of the file's change log) is specific to the changes that you made in that file and that file alone!
  - So go into your C:\SETRepo\DisneyCharacter\DisneyCharacter directory and **right-click *SVN-Commit***
    - On DisneyCharacter.h – enter a commit comment specific to the changes you made in that file
    - On DisneyCharacter.cpp – enter a commit comment specific to those changes and finally
    - On testDisneyCharacter.cpp – and enter a commit comment specific to that file
  - You would do this individual file level commit for all files (usually source file) that you care to have individual commit messages on. In larger system solutions, this may also include configuration files, text files, etc.
- Just as a catch-all commit after you've commited each source file – go back to the **C:\SETRepo** directory
  - If TSVNCache is operating correctly and keeping the SVN icons up-to-date on your directory and files – you may still see a red exclamation mark on the DisneyCharacter folder
  - It might be a good idea – until you get the hang of a repository to go back to the top-level directory of your solution (after you've committed the individual files that you know you've changed separately) and do a **right-click *SVN Commit*** here
    - So – you should be in **C:\SETRepo** and you right-click and commit on the top-level DisneyCharacter folder
      - If there really are no more modified files (i.e. files which are different in your local copy than in the repository) – then the commit "files" window will be blank – and you're done !
      - If there are some files that are different in your local copy – they should be VS administration-type files only – and these are safe to commit en-masse with one commit comment

What to Submit

Regardless of whether you choose to store your class definition in a repository, please ZIP up the 3 source files [DisneyCharacter.h, DisneyCharacter.cpp and testDisneyCharacter.cpp] and submit them to the assignment drop-box by the deadline.

Assign-02

The DisneyCharacter Class

Diagrams and Screen Captures

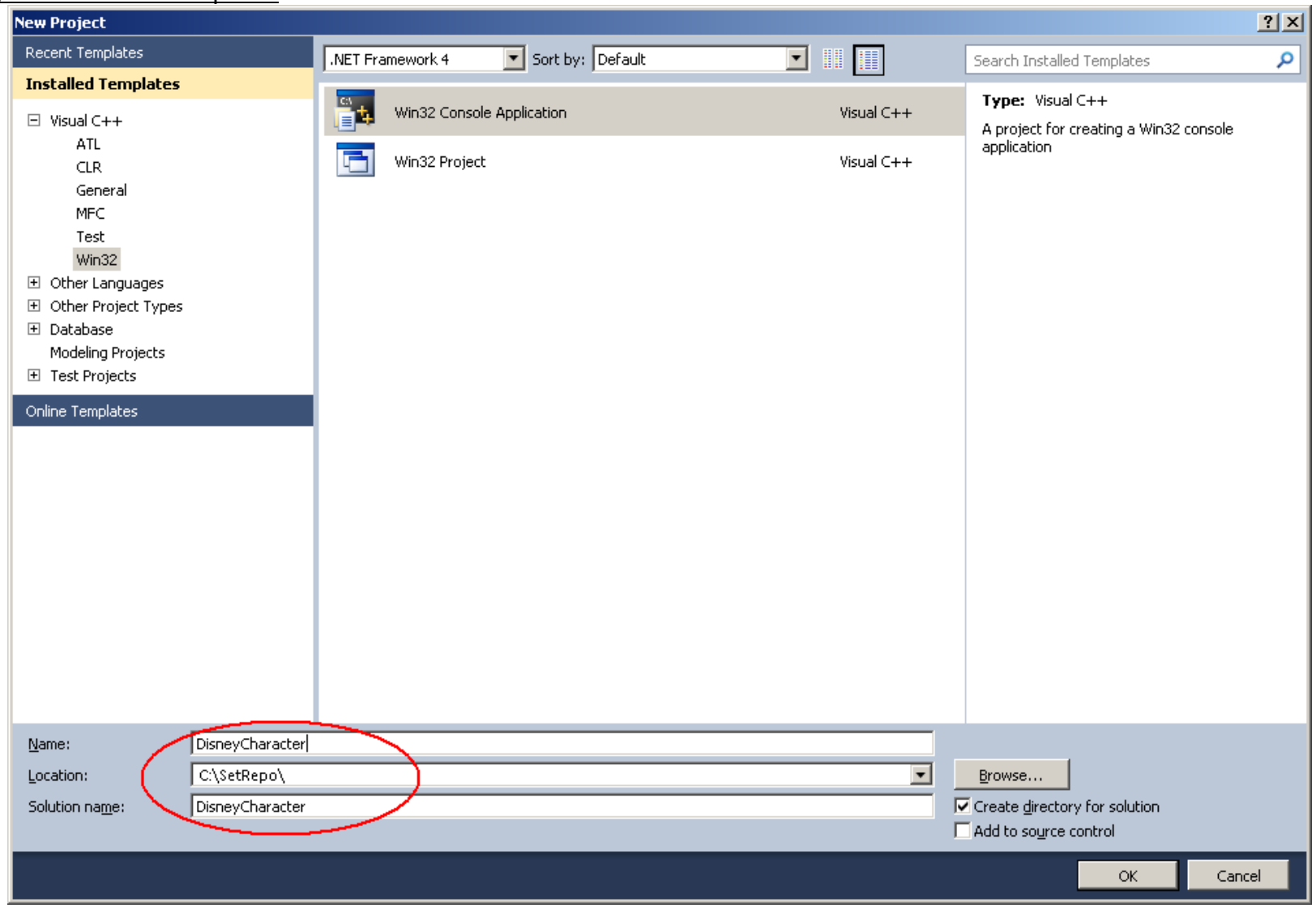

Diagram 1 : Making sure to create the VS project within the signed out repository

Assign-02

The DisneyCharacter Class

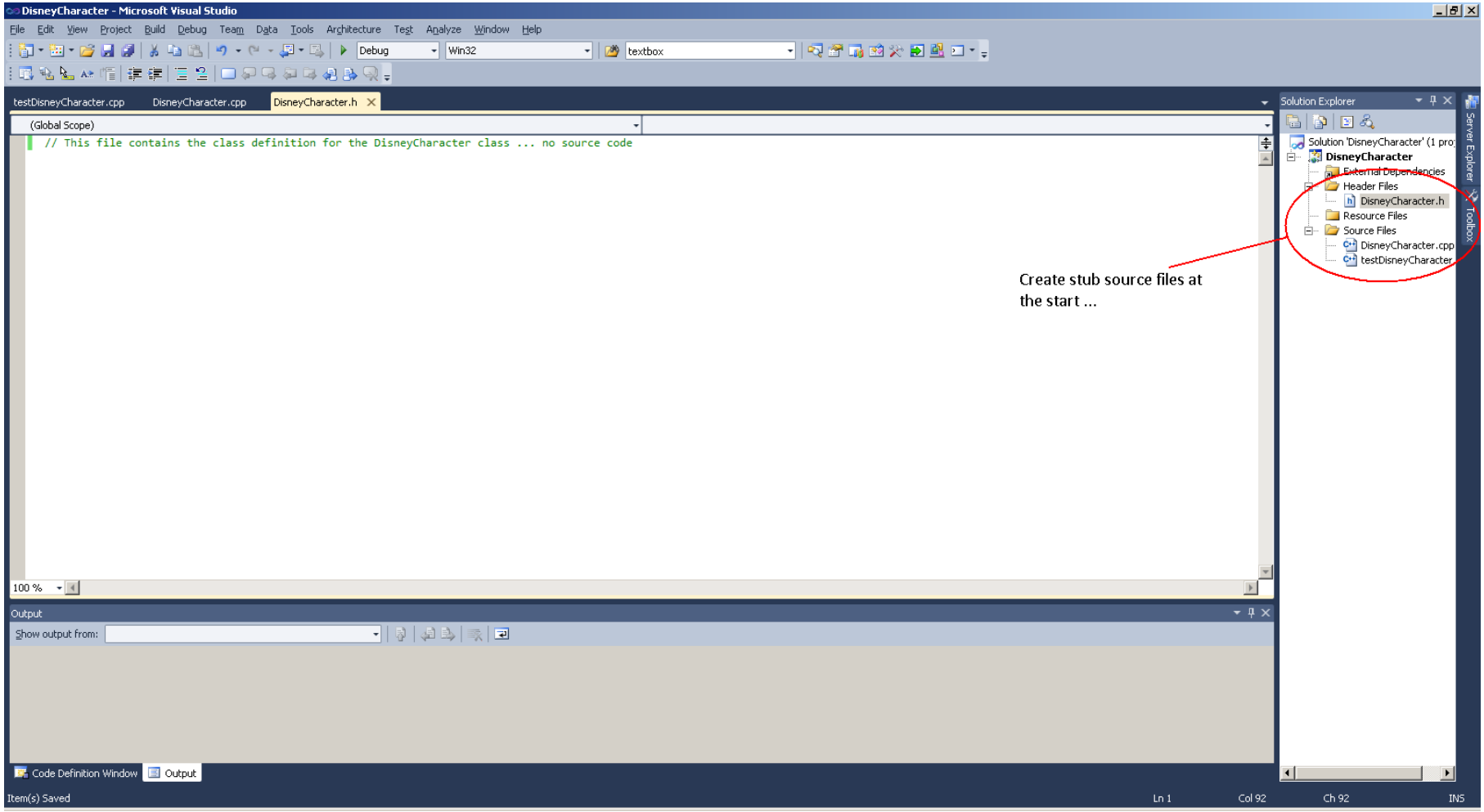

Diagram 2 : Creating stub source files so we can add the files to the repository

Assign-02

The DisneyCharacter Class

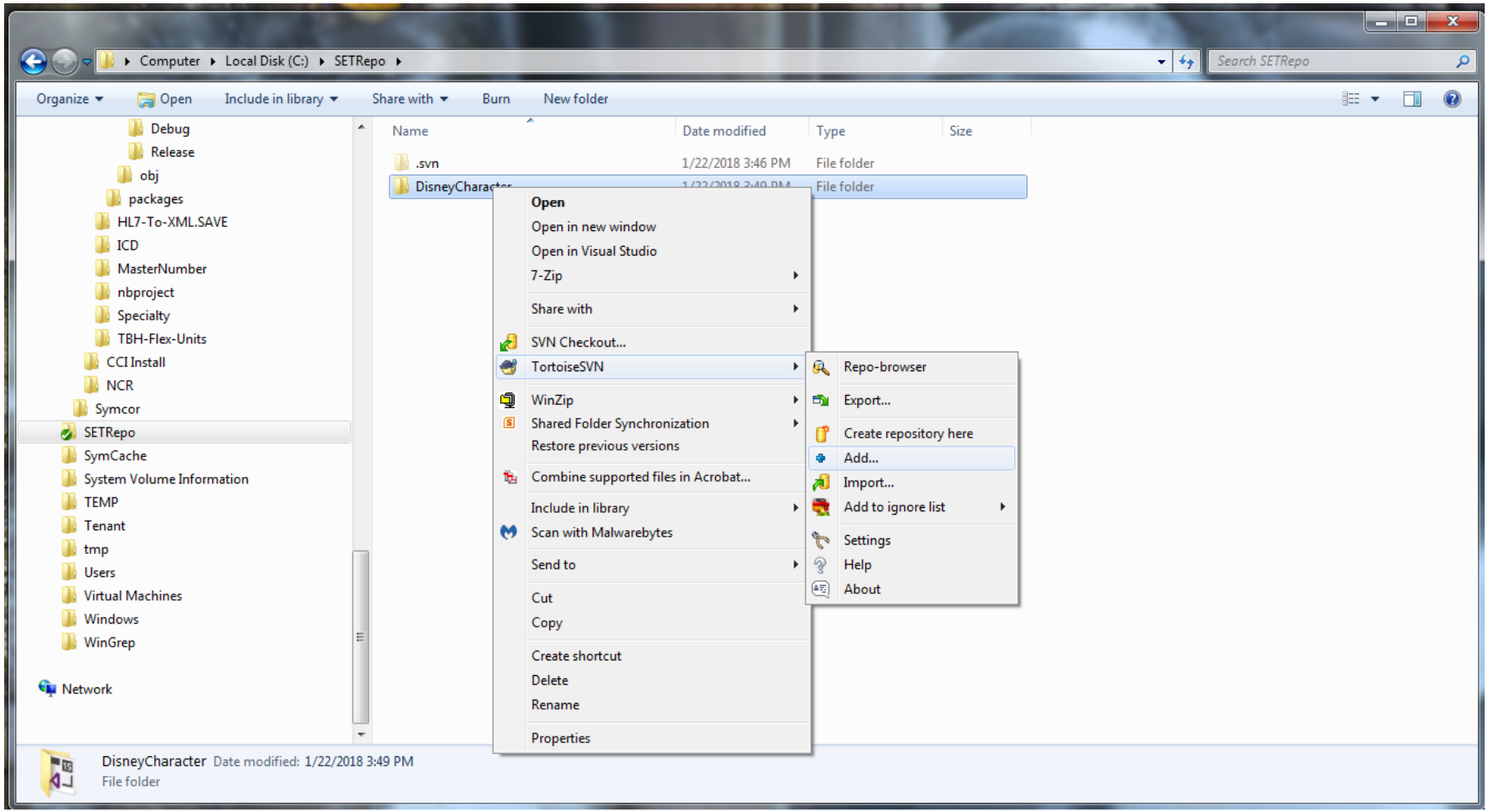

Diagram 3 : Adding the initial DisneyCharacter directory structure and files

Due Date : Feb 16, 2018 by 11:00pm
(in eConestoga Dropbox)

OOP - Winter 2018

Assign-02

The DisneyCharacter Class

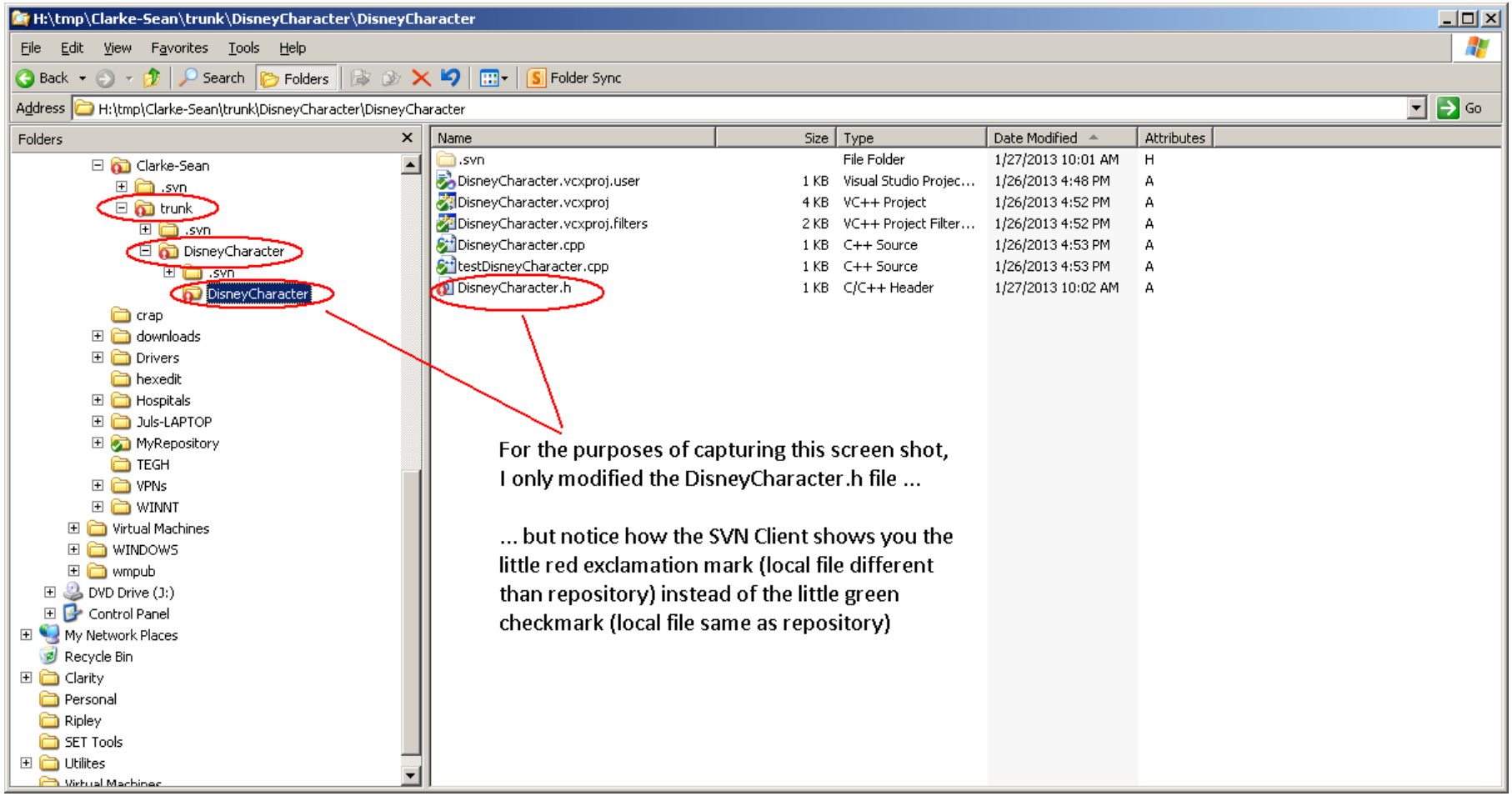

Diagram 4 : After the coding ... Tortoise SVN indicates what has changed ...