Thane Richard (GitHub: @thaneofcawddor)
Matteo Bjornsson (GitHub: @matteobjornsson)

ESOF 322 Homework # 4

Question 1a:

1. The system we downloaded was Sweet Home 3D. We downloaded this Open Source software from SourceForge. The software is written in Java.

2. This software allows a user to design an interior space using a GUI. Users can draw a floorplan, decorate, and rearrange the furniture, then experience the space in 3D.

3. The program has 64,782 lines of code. This was calculated using another program available on SourceForge that counted the number of right curly braces and terminating semicolons. We limited the count to just the Java files in the source folder. There were many other files, such as HTML, .dll files, and other resources and properties we did not include in the count. Another way we estimated these lines of code was to simply count the number of lines in each file using the following command in Windows command line:

   ```
   (for /r %f in (*.java) do @type "%f") | find /c /v ""
   ```

   This command puts all the java files into a string and counts the lines. The number of lines in the java documents was 143,798.

Question 1b:

1. Found 8 files that possibly contain design patterns.

   C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\j3d\Object3DBranchFactory.java
   Possible patterns: Factory

   C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\model\ObserverCamera.java
   Possible patterns: Observer

   C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\swing\ObserverCameraPanel.java
   Possible patterns: Observer

   C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\swing\SwingViewFactory.java
   Possible patterns: Factory

C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\viewcontroller\Object3DFactory.java
Possible patterns: Factory

C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\viewcontroller\ObserverCameraController.java
Possible patterns: Observer

C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\viewcontroller\ViewFactory.java
Possible patterns: Factory

C:\Users\Matteo\Desktop\SweetHome3DSourceCode\SweetHome3D-6.2-src\src\com\eteks\sweethome3d\viewcontroller\ViewFactoryAdapter.java
Possible patterns: Adapter, Factory

2. The program looks at the names of classes and files within the software and essentially does a string search of the naming convention of the files against the names of different patterns.

3. I think the method is moderately successful, but relies on the developer adhering to a Strategy Pattern naming scheme. A better way would be to look at the actual performance of the code, but I question the overall ability of any program to read source code and discern the overarching design pattern used in the software design. It would be akin to looking through the specific commands given to individuals in an Army and trying to discern the strategy being deployed for a battle or war. It is not impossible, but each subset may be performing a task that itself is not the overarching strategy pattern but is, through its actions, performing a critical function as a cog in the larger machine. For example, one could look at total uses of calls or classes of different types that pass through an interface as a way of inferring the use of the Adapter Pattern. But employing a high volume of Adapter Pattern calls may be a false positive when in fact the overarching design strategy is in fact the Strategy Pattern and performing an adapter role is part of implementing one of the Strategy Pattern's sub-strategies per the user. In other words, the absolute count of a Strategy Pattern's usage is not a sound method for determining its role in a piece of software's design.

Question 2:

a. I created a new repository on GitHub and then cloned that repository (which just had an empty README.md) to my computer. I then copied the homework assignments into that directory, staged them as changes, committed them, then pushed them.

b.

Once the file (dummyfile.txt) was created in the project folder, I ran:
`git add .`
This adds the file to the staging area. Then I ran
`git commit -m "adding dummy file for ESOF assignment"`
This commits the file to my local repository. Then I push those local changes to Github
`git push origin master`
I was on the master branch the entire time (I could have checked this with git branch) and now the master branch on GitHub is updated.