Aufgabe 04a: Decorator

1) Study the chapter about the Decorator pattern in [1].
2) Get an overview of the C# Stream classes (System.IO).
3) Take a look at the Java Stream decorators:
   http://www.docjar.com/html/api/java/io/InputStream.java.html,
   http://www.docjar.com/html/api/java/io/FilterWriter.java.html,…
4) Implement an En-/Decryption mechanism (e.g. rot13) using the decorator pattern. To be truly usable, your class shall extend a Stream class (e.g. System.IO.Stream). It shall work just as the other .NET stream classes. As you already know that for sure there will come some more decorator classes later on, introduce an extra class from which the rot13-decorator all further decorators inherit (see lecture slides).

Aufgabe 04b: Factory + Observer/Delegate + Singleton

Extend the ImageViewer-Application (our Proxy example 3c). In case you didn't implement 3c, ask colleagues (mention them in the comments).

1) Implement a factory for the image creation (decide on the url-String; if file-based: RealImage, if http-based:ProxyImage)
2) Implement a notification mechanism that allows the picturebox to update.
3) Implement the Factory as a singleton
4) Read the chapter about the Flyweight pattern in [1]. Expect questions.

Work in teams of 2 (name both authors in all the source files)!

[1] Gamma et al. „Design Patterns. - Elements of Reusable Object-Oriented Software."

Deadline: Sonntag, 23.10. 16:00