AUFGABE: Refactoring

1. Wo befindet sich der Refactoring Katalog?
2. Wieviel Refactorings sind dort enthalten?
3. Welche Refactorings beherrscht Ihre IDE?
4. Was ist Ihr Lieblingsrefactoring?
5. Bitte mindestens ein eigenes (!) Vorher- / Nachher Code-Beispiel.

zu 1.)

Der Refactoring Katalog ist zu erreichen unter https://refactoring.com/catalog/.

zu 2.)

Der Refactoring Katalog beinhaltet 66 Refactorings.

zu 3.)

Meine IDE *Ecplise* beherrscht für ein Java-Projekt folgende Refactorings:

- Change Function Declaration (via *Change Method Signature...*)
- Combine Functions into Class
- Encapsulate Field / Variable
- Extract Variable / Function / Class / Superclass
- Move Field / Function / Type
- Parameterize Function (via *Change Method Signature...*)
- Pull up constructor body / Field / Method
- Push down Field / Method
- Remove dead code (via Warnings, Quickfix, *Organize Imports*)
- Rename Field / Variable
- Replace Constructor with Factory Function (via *Introduce Factory...*)

zu 4.)

https://refactoring.com/catalog/replaceSubclassWithDelegate.html

Getreu dem Motto *Composition Over Inheritance*. Kommunikation zwischen mehreren unabhängigen Objekten, die einen Verbund bilden, ist besser als Code, in dem enthaltene Fehler mehrfach vererbt werden.

zu 5.)

Beispiele:

RefactoringInheritance: *Replace Subclass with Delegate*, Auftrennen von Mehrfachvererbung in über Interface gekoppelte Klassen.

Mehrere Datenquellen für Netzwerk, Datenbank und Dateisystem erben von der selben abstrakten Klasse `DataSource`, die die Routinen `open()`, `close()` und `fetch(...)` definiert.

Nach dem Refactoring gibt es nur noch die konkrete Klasse `DataSource`, die über *dependency injection* eine Connector-Klasse bekommt. Connectoren sind: `DBConnector`, `FileConnector`, `NetworkConnector`.

Vereinheitlicht sind diese über das Interface `IConnectable`. Die Klasse `DataSource` weiss also gar nicht mehr, welche Art Connector verwendet wird, die Connectoren wiederum können frei implementiert werden. Wird Code zwischen den Connectoren wiederverwendet, kann dies sowohl über Vererbung als auch über weitere Komposition gelöst werden. Die `DataSource-Klasse` bekommt davon allerdings nichts mehr mit.

RefactoringPullUp: Verschieben der Methode `verifyPassword(...)` in die Basisklasse `DataSource`, nachdem sich herausgestellt hat, dass diese Methode von allen Subklassen verwendet wird.