Name: Proxy - Schutzproxy
Art: Strukturelles Entwurfsmuster (Virtual-, Smart-, Protection- oder Smart Proxy
Ziel: Kontrolle des Zugriffs auf ein Objekt durch Stellvertreter (Proxy)

Einsatzgebiete: Zugriffskontrolle, Lazy initializion, Remote-Zugriff, Caching + Logging

Vorteile: -Trennug von Zugriffslogik und Fachlogik
-Nachträgliche Erweiterung
-Erhöht Kontrolle + Sicherheit
-Vielseitig einsetzbar

Nachteile: -Zusätzlicher Komplexitäts-Overhead
-Aufwand bei vielen Rollen und Regeln

Warum exestiert das Pattern?:Um kontrollierten, oder verzögerten Zugriff auf Objekte zu ermöglichen, ohne die Implementierung zu ändern. Wie im Beispiel mit dem FCBayern Mitglied: Ein normaler User ohne Mitgliedschaft kann sich nicht einloggen.

Codebeispiel: FC Bayern Mitglied möchte sich auf der Mitgliedswebsite einloggen. Die wichtigsten Ausschnitte

```java
public interface Anmeldung {
    boolean login(String name, String passwort);
}
```

1. Gemeinsames Interface

```java
public class RealAnmeldung implements Anmeldung {
    @Override
    public boolean login(String name, String passwort) {
        System.out.println("Anmeldung erfolgreich, Willkommen " + name + "!");
        return true;
    }
}
```

2. Das verifizierte ZielObjekt

```java
    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        if ("FcBayernMitglied".equalsIgnoreCase(role)) {
            System.out.println("Zugriff erlaubt fuer: " + method.getName());
            return method.invoke(target, args);
        } else {
            System.out.println("Zugriff verweigert fuer: " + method.getName());
            return false;
        }
    }
}
```

3. ProxyHandler mit Rollenprüfung

```java
import java.lang.reflect.Proxy;
```

4. Dieser Import ermöglicht die Erstellung von Proxy-Objekten + Klassen

```java
import java.lang.reflect.InvocationHandler;
```

5. Behandelt Proxy-Objekte