ADAP CW#11

Projektdaten

Projekt name:	3D-Printing
Projekt repository:	https://github.com/**ModischFabrications**/wahlzeit
This week's tag:	adap-hw10 on master
Homework diff:	https://github.com/ModischFabrications/wahlzeit/compare/adap-hw09...adap-hw10
CI:	https://travis-ci.org/**ModischFabrications**/wahlzeit
Docker Hub Repo:	https://hub.docker.com/r/**modischfabrications**/wahlzeit

Hausaufgaben

Implementation

Neue Klassen (falls vorhanden)

PatternInstance

Veränderte Klassen (falls vorhanden)

Nur Annotations, siehe Fragen

Erklärung

Ich habe zuerst die Implementierung des Instanz-Sharings der Coordinates überarbeitet. Dabei habe ich die Factory mit synchronized annotiert, um die Klasse Threadsafe zu machen. Außerdem habe ich die Map mit dem Hash statt dem eigentlichen Objekt geschlüsselt, um eine saubere Trennung zwischen Key und Value einzuführen. Dies führt außerdem zu einer einfacheren Implementierung.

Als zweites habe ich die Annotation "PatternInstance" selbst implementiert. Wichtig war dabei die Annotation "@Repeatable" und die damit verbundenen values, die eine Mehrfachnennung erlaubt.

Dann habe ich das Projekt systematisch nach Patterns durchsucht, die gefundenen Patterns werden in den Fragen erläutert.

Zuletzt habe ich alle Änderungen gepusht und den letzten Commit mit dem oben genannten Tag für die Abgabe markiert.

Fragen

What is the purpose of the design patterns you documented?

Does the use of these design pattern make sense to you? Why?

What are the drawbacks of using these design patterns here?

Decorator (LoggingEmailService)

Purpose: E-Mail Service durch Logging ergänzen

Useful: Wirkt etwas gestellt, logging kann auch akzeptabel gut in der Klasse selbst implementiert werden

Drawbacks: höhere Komplexität durch Vergrößerung der Klassenmenge und Verstecken der konkreten Implementierung.

Abstract Factory (PhotoFactory)

Purpose: Photos ohne konkrete Implementierung erstellen

Useful: Akzeptabel. Gut ersetzbare Photoklasse durch die Entkopplung von der Implementierung, ansonsten aber nur größerer Overhead.

Drawbacks: Vergrößert Aufwand für Konstruktor

Singleton (PhotoFactory)

Purpose: Global einzigartige Instanz

Useful: Sinnvoll, aber gefährlich. Singletons erschweren Dependency Injection, sind im Fall von Factories aber häufig verwendet und meist nach Konvention.

Drawbacks: Erschwert Testen

Adapter (DatastoreAdapter)

Purpose: Google Datastore von Implementierung entkoppeln

Useful: Sinnvoll, um aufwändige Dependency (Google) aus der restlichen Implementation zu verstecken.

Drawbacks: Größere Distanz verringert Performance.

Flyweight (*Coordinate)

Purpose: Value Objects mit Shared Instances nachbilden

Useful: In diesem Kontext zu aufwendig und wenig anwendbar durch floating point errors

Drawbacks: vergrößert Komplexität der Klasse, in unserem Fall ohne großen Zugewinn.