Vorname	Nachname	Beruf	Dauer
Marten	Meißner	Fachinformatiker AE	07.22-07.24

Wochenbericht KW 28. (2023.07.10. - 16.)

Nachdem wir uns mit den Creation Pattern vertraut gemacht haben, haben wir uns nun mit den Structure Pattern beschäftigt.
Structure Pattern werden genutzt, um Klassenstrukturen so aufzubauen, dass sie vielseitig sein können, aber dennoch so abstrakt sind, dass sie nicht voneinander abhängig sind oder vielleicht Klassen, die nicht zusammen passen zusammen, trotzdem miteinander funktionieren können, wie bei dem Adapter Pattern.
Das Adapter Pattern wird genutzt, um verschiedene inkompatiblen Klassen eine Verbindung zu erstellen.
In dem Buch Design Pattern von den „Gang of Four"(Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides) wird das Adapter Pattern mit dem Beispiel eines Zeichenprogramms erklärt.
In diesem Zeichenprogramm gibt es die Klasse Shape in einem shape Objekt können verschieden Zeichenformen und Zeichnung verarbeitet und ausgegeben werden, aber dieses Objekt ist nicht mit der Klasse Text kompatible. Hier eignet sich das Adapter Pattern, da hier nur eine Schnittstelle benötigt wird, um vorhanden Code wiederzuverwenden und nicht unnötig viel neuen Programmcode zu schreiben. In diesem Beispiel wird nur eine neue Klasse TextShape benötigt, die von Shape und von Text erben kann, jetzt kann Textshape genauso verwendet werden wie ein normale Shape Objekt. Nur das hier auch Text hinzufügt werden kann. Eine auf den ersten Blick ähnliches Pattern ist das Bridge Pattern.
Im Bridge Pattern werden nicht zwei inkompatiblen Klassen näher gebracht, vielmehr wird hier eine Klasse abstrahiert und eine Klasse erweitert.
Hier habe ich ein Beispiel gefunden, in diesem Beispiel geht es um refactoring an bestehendem Code.
Wenn z.B. eine Funktion sehr komplex ist oder nicht so ausgereift veröffentlicht wurde, könnte hier eine Bridge erstellt werden, die auf Ebene der Superklasse eine Verbindung zur ersten Klasse und dessen Hierarchiestruktur spiegelt.
Mit dieser neuen Klasse kann nun ein refactoring durchführen und später leicht mit der ersten Klasse ersetzt werden.
Durch das Bridge Pattern können wir die Implementation sehr schön trennen von abstrakten Datenklassen.
Ein weiteres Konzept ist das Decorator Pattern.
Mit dem Decorator Pattern versucht man eine komplexe Objekt-Struktur so zu vereinheitlichen, dass es einfacher ist ein Basisobjekt zu erweitern und dadurch die Interaktion mit andern Objekten zu verbessern.
Durch das Decorator Pattern wird eine hohe Flexibilität erreicht, da hier Basisobjekte von decorator getrennt werden, dadurch können Basis- und Decorator-objekte besser weiter verwendet werden, da sie weniger Komplex sind.
Zum Beispiel kann es eine Basisklasse Gericht geben, die mit viele kleine einzelnen Gerichte erweitert werden kann.
Z.B. ist ein Komplexes-Gericht „Schnitzel mit Pommes" schlecht umbauen, wenn anstatt Pommes lieber Salat dazu möchte, bei dem Decorator Pattern wäre eine Aufteilung viel effizienter. Wenn Schnitzel ein eigenes Gericht ist, Pommes und Salat auch, dann könnten einfach weitere Beilagen erzeugt und kombiniert werden.
In diesem Beispiel könnte z. B. ein Gericht aktiviert werden, mit dem Objekt Schnitzel und leicht um einen Salat oder Pommes oder beides erweitert werden.

Kontrolliert am: _____ Unterschrift :_____