Vorname	Nachname	Beruf	Dauer
Marten	Meißner	Fachinformatiker AE	07.22-07.24

Wochenbericht KW 26. (2023.06.19. - 25.)

Letztens bekamen wir neues Lernmaterial, diese nutzen wir in dieser Woche, um uns mit dem Thema Creation Pattern einzuarbeiten. Um uns in das Thema Creation Pattern näher einzuarbeiten, lasen wir uns das gleichnamige Kapitel in dem sehr bekannten Buch „Design Patterns. Elements of Reusable Object-Oriented Software". Als Creation Pattern wird bezeichnet, die Architektur nach einem gewissen Muster aufzubauen. Wie Datenklassen erstellt werden und wie sie im Entwicklungsverlauf umgesetzt werden. Von den 23 Designmuster, die in diesem Buch vorgestellt werden, beziehen sich auch fünf Muster auf die Erstellung von Objekten und welche Merkmale, die Klassen für eine ausreichend abstrahierte Architektur benötigt wird. Da die Einzel beschriebenen Muster unabhängig voneinander funktionieren konnten wir uns unabhängig von dem Buch jedes Muster für sich allein stehend lesen und versuchen zu verstehen, da der Aufbau und nutzen dieser Muster doch sehr abstrakt ist. Deswegen beschäftigten wir uns zuerst mit der Architektur der Singelton Klassen. Eine Singelton Klasse ist so aufgebaut, dass sie sich genau nur einmal aufrufen kann, weil sie nur indirekt aufgerufen werden kann über eine Instanz. Diese Instanz fragt sich bei jeden Aufruf, ob es schon ein Objekt dieses Typen gibt und verweist auf dieses, alternativ erstellt es sich ein einziges Mal. Dies wird gerne genutzt, um im gesamten Aufbau nur ein einziges Objekt dieser Klasse zu erstellen. Z. B. bei einem State oder wenn es eine mehrsprachige App werden soll, dann kann jeder ausgegebene Text über dasselbe Objekt übersetzt werden. Dies hat den Vorteil, dass das Objekt in der gesamten Software benutz werden kann und bei jeden Aufruf dasselbe Objekt aufzurufen und nicht unbeabsichtigt ein ähnliches Objekt aufgerufen wird. Da dieses Objekt wie eine globale Variable funktioniert, hat sie auch ähnliche Schwachstellen, so kann bei einem Aufruf auch Attribute verändert werden, die an einer anderen Stelle im Programm fatal sein können. Zudem kann es auch theoretisch passieren, dass ein Singelton doch zweimal erstellt werden kann, da diese Designvorlage nicht Threadsave ist, das heißt sie kann bei der Erstellung in zwei Prozessoren Warteschlangen vorhanden sein und aus Versehen zweimal gleichzeitig erstellt werden. Dennoch kann es in bestimmten Situationen hilfreich sein, eine Singelton zu verwenden, weil sie bestimmte Abläufe sehr vereinfacht, aber auch hier ist es wichtig nicht zu viele Singelton zu erstellen, da dieses auch wieder zu unsauberen Programmierstil führen kann. Ein weiteres Design pattern ist das Prototyping. Bei Prototyping wird sich immer auf ein Prototyp Objekt bezogen und bei jedem erstellen eines Objekt dieser Klasse wird im Programm nur eine Kopie des Prototyps erstellt. Dieser Klon des eigentlichen Objekts hat dann den Vorteil, dass es während der Laufzeit überall erstellt werden kann und auch gelöscht werden kann. Dies kann gut angewendet werden, wenn häufig ähnliche Objekte gebraucht werden, die sich nur wenig unterscheiden. Ein weiterer Grund das Prototyping zu wählen ist, wenn Objekte in paralleler Hierarchie verwendet werden soll. Nachteilig ist, dass in jeder Unterklasse die jeweilige Klon-Funktion implementiert werden muss und zusätzlich noch in manchen Sprachen auch registriert werden. Beim Erstellen eines Klons gibt es zudem noch zu beachten, wie der Klon erstellt wird, in manchen Sprachen wird ein Hallo Copy oder Deep Copy erzeugt. Einen genauen Klon des Prototyps, bei einer Memberwise Copy wird ein Objekt erzeugt, welches mit einem Pointer auf den Speicher den Prototypen verweist, wenn hier der Klon geändert wird, wird auch der Prototyp angepasst.

Kontrolliert am: _____ Unterschrift :_____