Vorname	Nachname	Beruf	Dauer
Marten	Meißner	Fachinformatiker AE	07.22-07.24

Wochenbericht KW 26. (2023.06.26. - 07.02.)

Zu dem Creation Pattern zählt auch die Architektur der „factory-method“ und der „abstract-factory“. Grob gesagt kann eine „factory“ ähnlich eingesetzt werden wie ein Konstruktor, nur mit dem Unterschied, dass ein Konstruktor immer das gleiche Objekt herstellen kann. Bei einer Factory hingegen kann die Superklasse eines Objektes auch unterschiedliche Objekte einer Unterklasse erstellen und hat einen Rückgabewert, den ein Konstruktor nicht hat. Dies kann z.B. genutzt werden, wenn ein Objekt verschiedene angewendet werden soll oder es verschiedene erbende Objekte erzeugt werden soll. Als kleines Beispiel hierfür würde ich eine Superklasse Hund erstellen, die eine Eigenschaft haben kann wie Wachhund. Wenn diese Eigenschaft vorhanden ist, dann kann die Factory in der Superklasse entscheiden, dass aus dem Hund ein Wachhund werden kann, wenn diese Attribute nicht vorhanden ist, dann wird es stattdessen ein Begleithund. In diesem Beispiel sind Wachhund und Begleithund, Unterklassen der Basisklasse Hund und besitzen damit auch alle Eigenschaften der Basisklasse. Mit dieser Architektur weise können voneinander erbende Objekte in der Superklasse je nach Verwendung entschieden werden, welches Objekt hier erstellt werden soll. Durch dieses Designkonzept kann wieder viel Logik in Klassen gepackt und so viele Stellen im Programm sehr simpel und sauber gehalten. Zusätzlich muss im Code nicht ständig verschiedene Fälle unterschieden werden, da das Objekt selber entscheiden kann, welches Objekt in einer factory zurückgeben wird. Etwas komplizierter, aber im Prinzip gleich, funktioniert eine „abstract-factory“. Hier wird in einer Interface-Klasse eine Abstrakte-faktory erstellt. Diese wiederum entscheidet auf welche abstrakte Unterklasse verwiesen werden soll, um dort wiederum die dort beinhaltenden factorys dann wieder entscheiden, was für ein Objekt gebildet wird. Dies erleichtert das Programmieren sehr, da die Komplexität und Logik auf einer sehr abstrahierten oberen Schicht schon eingebaut ist, welches Objekt erzeugt wird und zurückgegeben wird. Dadurch wird die Implementierung und das Nutzen solcher Objekte einfacher, da hier einfach nur auf die factory verwiesen wird und diese dann, das richtige Objekt zurückgibt. Da diese verschachtelte Abstraktion doch eher selten benötigt wird, ist diese auch nur selten anzutreffen. Da solch ein Programm schon ein hohes Maß an Komplexität mitbringen muss, um eine solche Verschachtelung zu benötigen. Ein ähnliches Konzept wie bei einer Abstrakten Factory ist die Builder Architektur. Bei dem Builder Design wird eine Funktion übergeben, die einer vordefinierten Builderklasse ein Objekt zurückgeben kann. Vorteil hier ist das hier keine Abhängigkeit bestehen muss wie bei einer factory. Ein weiterer Vorteil ist, dass durch die Builderklasse bei sehr umfangreichen Objekten, so definiert werden das Attribut die sich selten ändern oder gewisse werter benötigen, gleich vergeben werden als Standardwerte.

Kontrolliert am: _____ Unterschrift :_____