

Attribute sollten moeglichst einfach gehalten werden (moeglichst simpel)

Sobald ein Attribut eigenschaften haben sollte bzw. diese Beschreibt ist eine eigene Klasse sinnvoll!

#### Communication Diagram

Pfeil zeigt die Richtung der Nachricht;

Angabe des Rueckgabewertes moeglich(inklusive Datentyp);

1:\*[i=1...10]

\* ... steht fuer eine Iteration

Der gesamte Ausdruck bedeutet optional 1 bis 10

Creat wird verwendet, da das Diagramm sprachunabhaengig ist; gemeint ist z.B.: unter Java ein new

[new item]

Bedeutet Guard, d.h. eine Vorraussetzung, hier das es ein neues Item sein muss

Bei Use Casees mit GUIs ist es ev. sinnvoll allen buttons einen useCase zuzuweisen.

#### Expert Pattern in Design Phase

Welche Klasse soll welche Verantwortung haben?

Welche Klasse kann worauf zugreifen? Welche Aufgaben muss sie dadurch uebernehmen?

#### Creator Pattern in Design Phase:

Wer ist verantwortlich um Objekte zu erzeugen?

#### Controller Pattern in Design Phase:

Wer ist verantwortlich fuer den Input in das System?

Es gibt nach aussen hin eine Klasse, die Fassade. Sie faengt alle Inputs von aussen ab und reicht sie weiter (fascade controller) die einfachste Moeglichkeit; es wird das gesamte System nach aussen gekapselt

Einzelne Rollen im System (z.B.: geld abheben) erhalten einzelne Klassen, die fuer ihre Aufgabe die Inputs entgegen nehmen (Role Controller)

Fuer jeden UseCase gibt es eine Klasse die seine Methoden entgegen nimmt und den ablauf des UseCases steuert (usecase controller); fuer Systeme mit sehr vielen use cases!

z.B.: das http Protokoll bei Kommunikation mit einem Webserver stellt einen useCase dar;