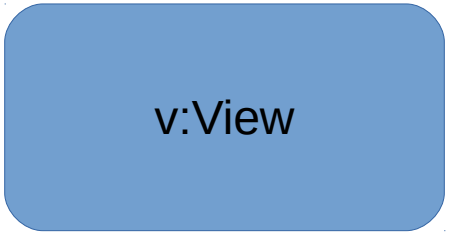
Ein Adressbuch nach dem Model-View-Controller-Prinzip

Warum einfach, wenn es auch umständlich geht?

Der View als grafische Benutzeroberfläche (GUI)

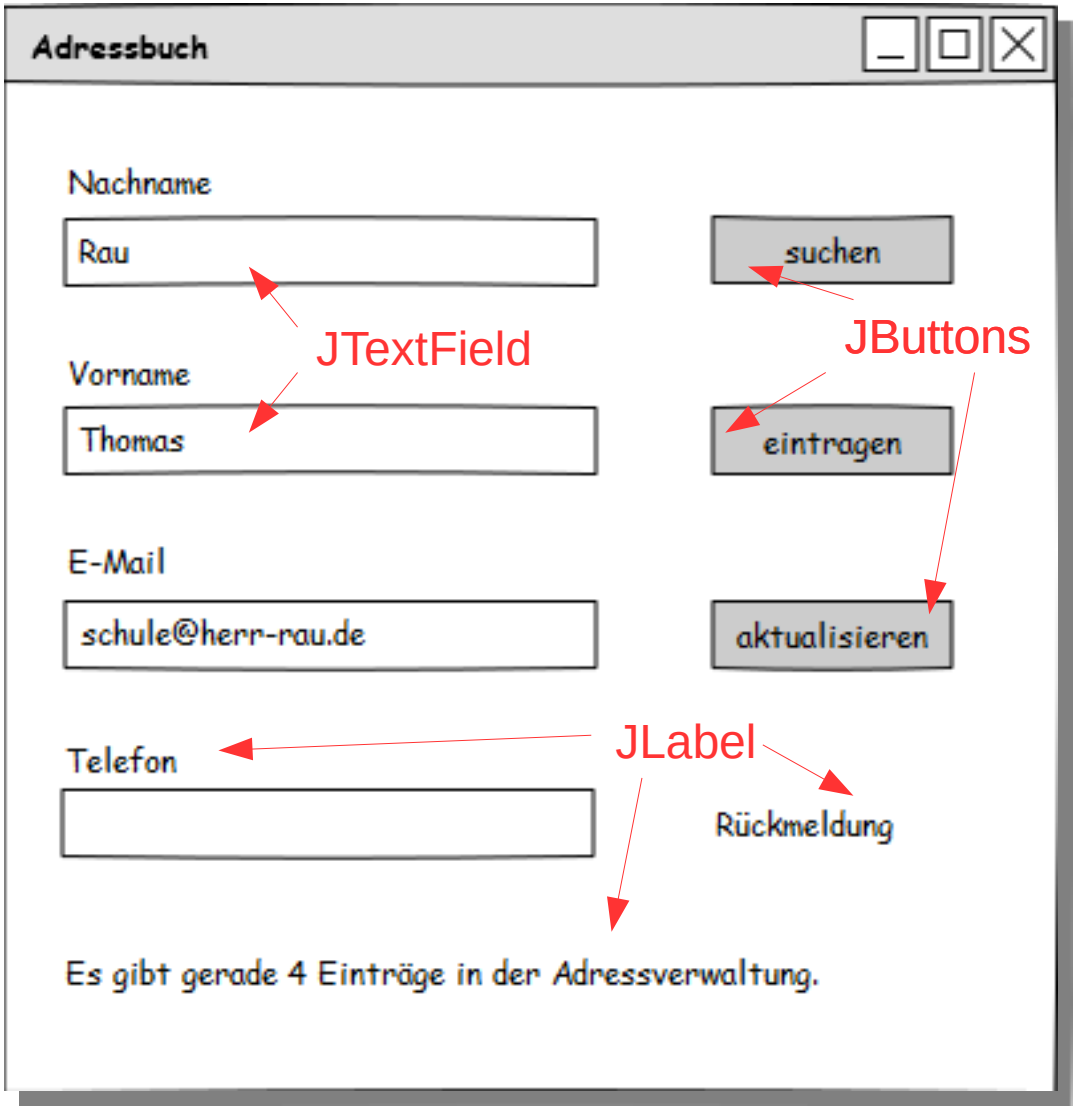


Der View als grafische Benutzeroberfläche (GUI)

v:View

Der View als grafische Benutzeroberfläche (GUI)

v:View



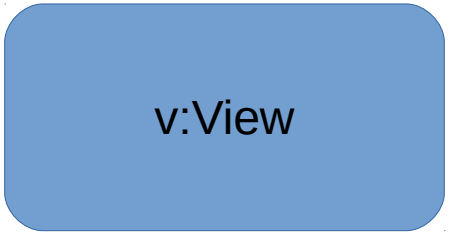
Der View als grafische Benutzeroberfläche (GUI)

- **Suchen:** Es wird nach „Rau, Thomas“ gesucht, das Ergebnis wird angezeigt
- **Eintragen:** Der Adressbucheintrag wird hinzugefügt; wenn er schon vorhanden ist, erscheint eine Fehlermeldung (da, wo „Rückmeldung“ steht)

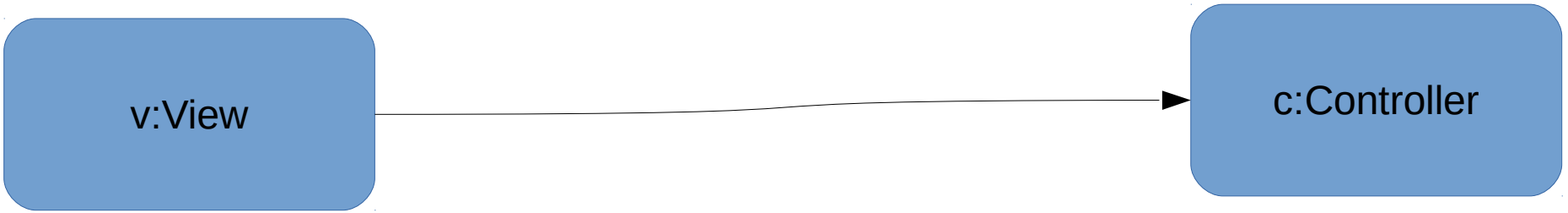
  Im Erfolgsfall wird der Satz mit der Anzahl der aktuellen Einträge aktualisiert.
- **Aktualisieren:** Der Adressbucheintrag „Rau, Thomas“ wird aktualisiert mit neuer E-Mail und Telefon

Der View als grafische Benutzeroberfläche (GUI)

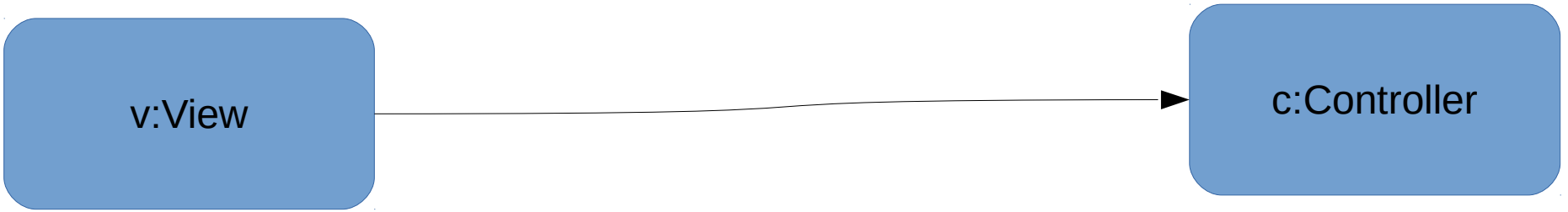
v:View


v:View

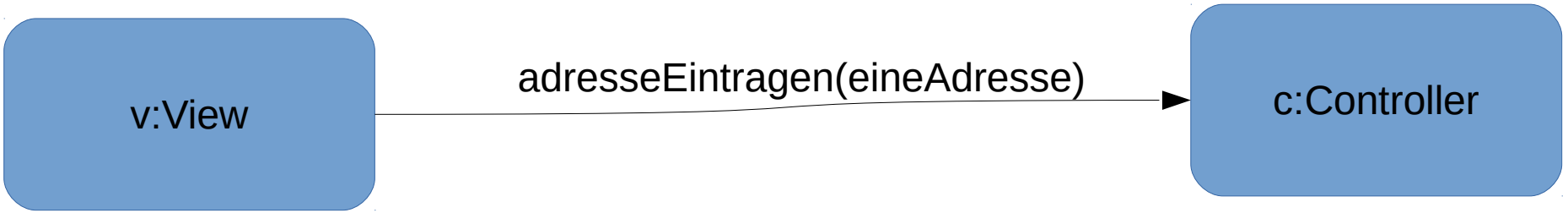
Verbindung zum Controller (Referenzattribut)



Auftrag (Methodenaufruf) an den Controller

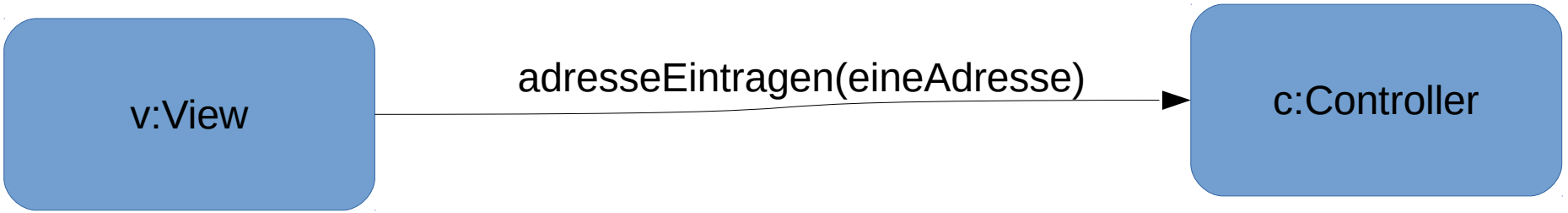


Auftrag (Methodenaufruf) an den Controller

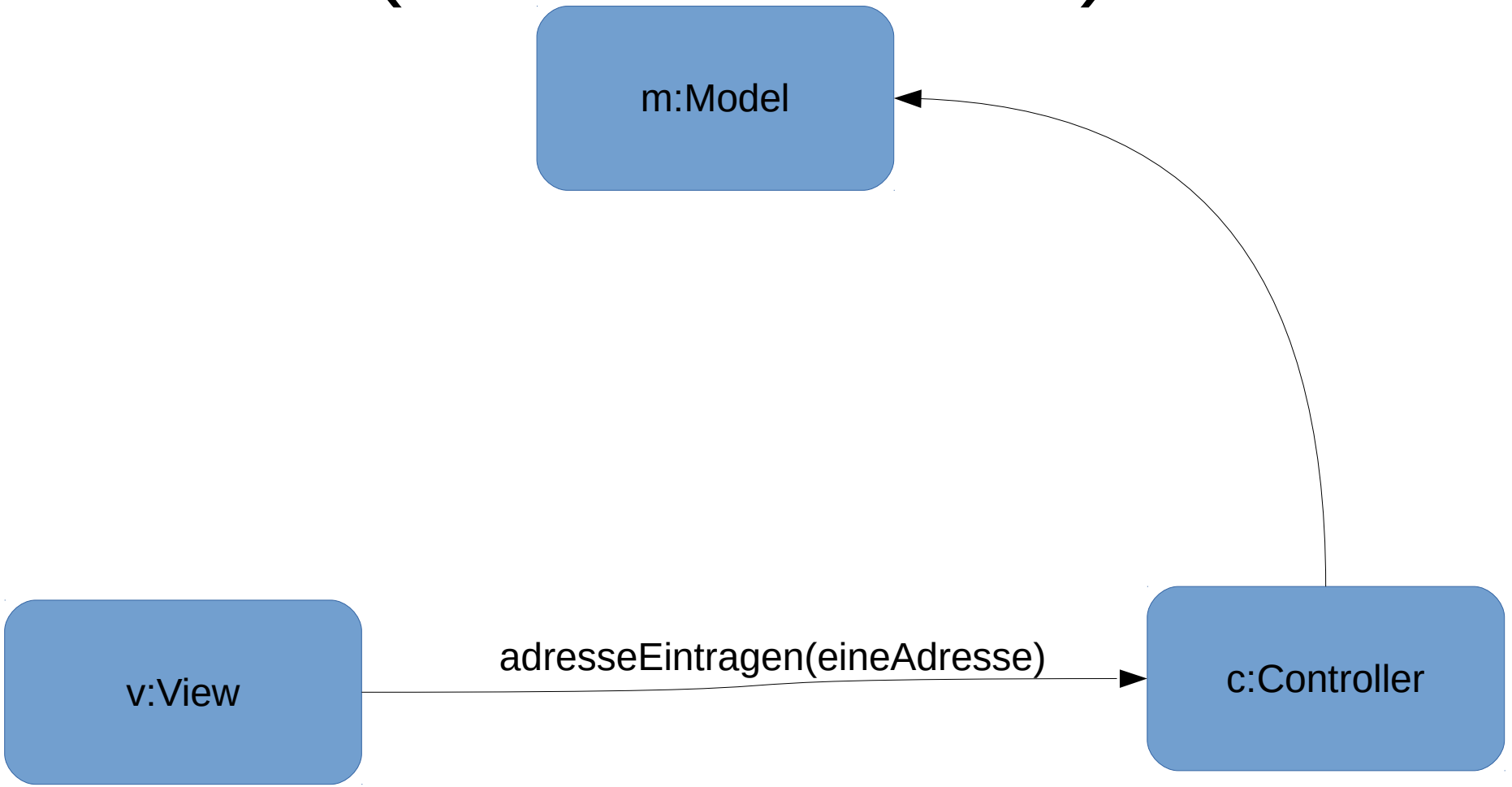


Der Controller überprüft zum Beispiel, ob die eingegebene Adresse schon vorhanden ist und ob die E-Mail eine korrekte Form hat.

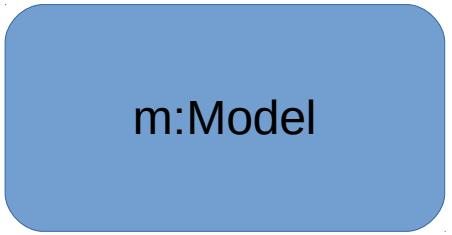
Wenn alles stimmt, gibt der Controller den Auftrag weiter an das **Model.**



Verbindung zum Model (Referenzattribut)

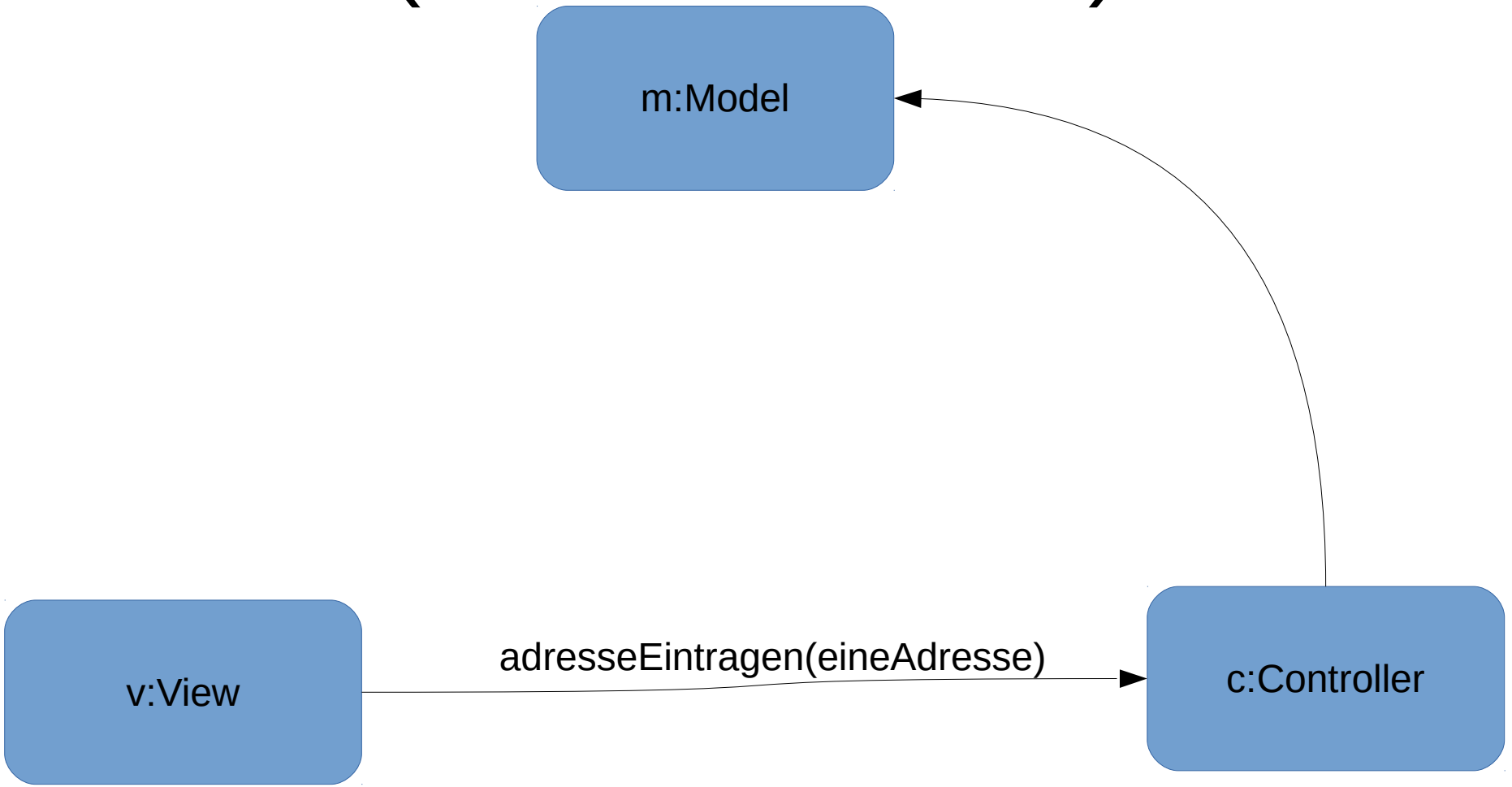


Das Model: Hauptteil des Projekts

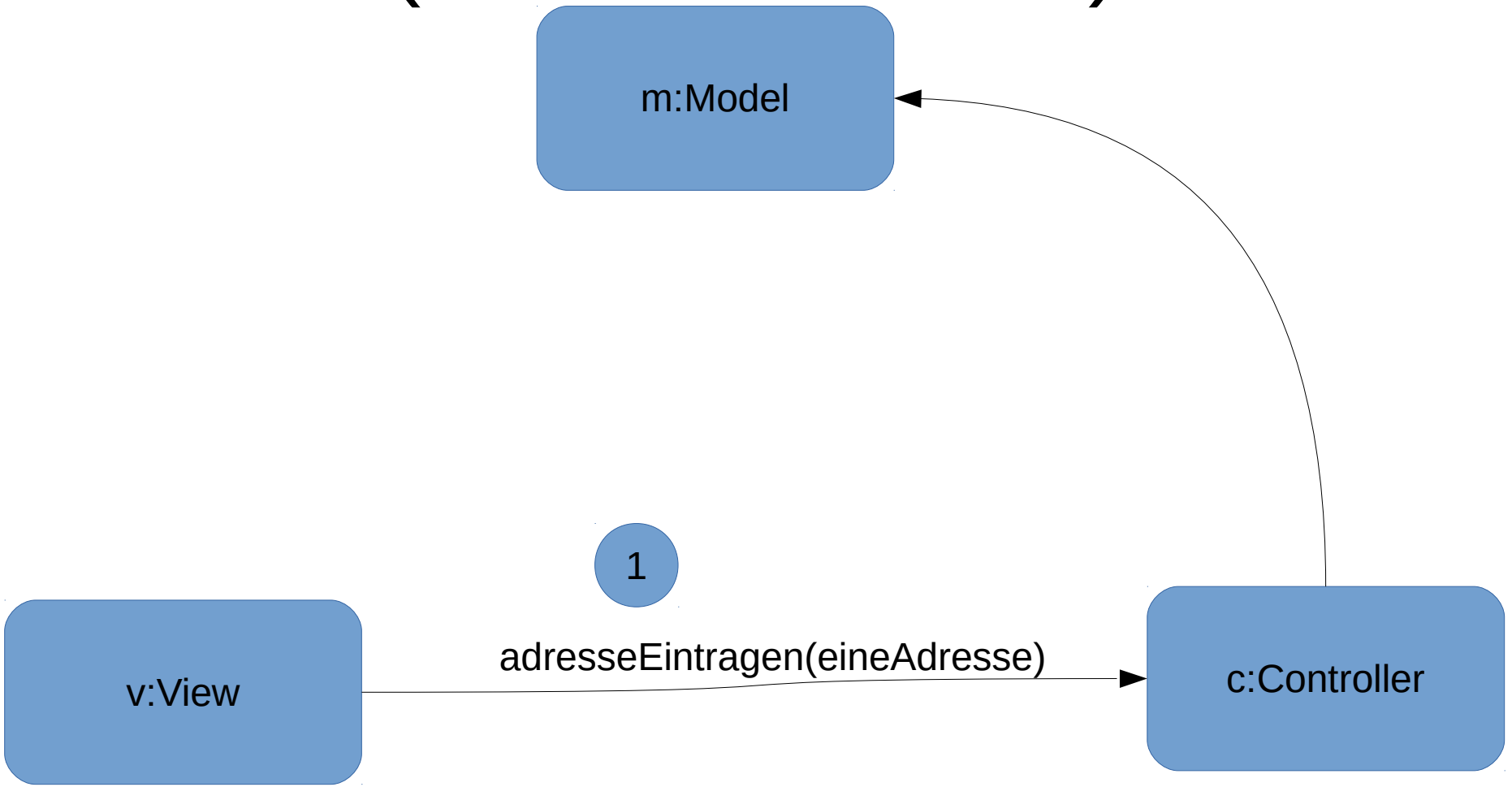


Wenn der Controller das Hirn des Projekts ist, dann ist das Modell der Körper. Er enthält die wichtigen Klassen und Informationen. Das Modell benutzt zum Beispiel einen Binärbaum, um Adressen zu speichern. Es stellt Methoden zur Verfügung, um Attributwerte zu ändern oder auszulesen.

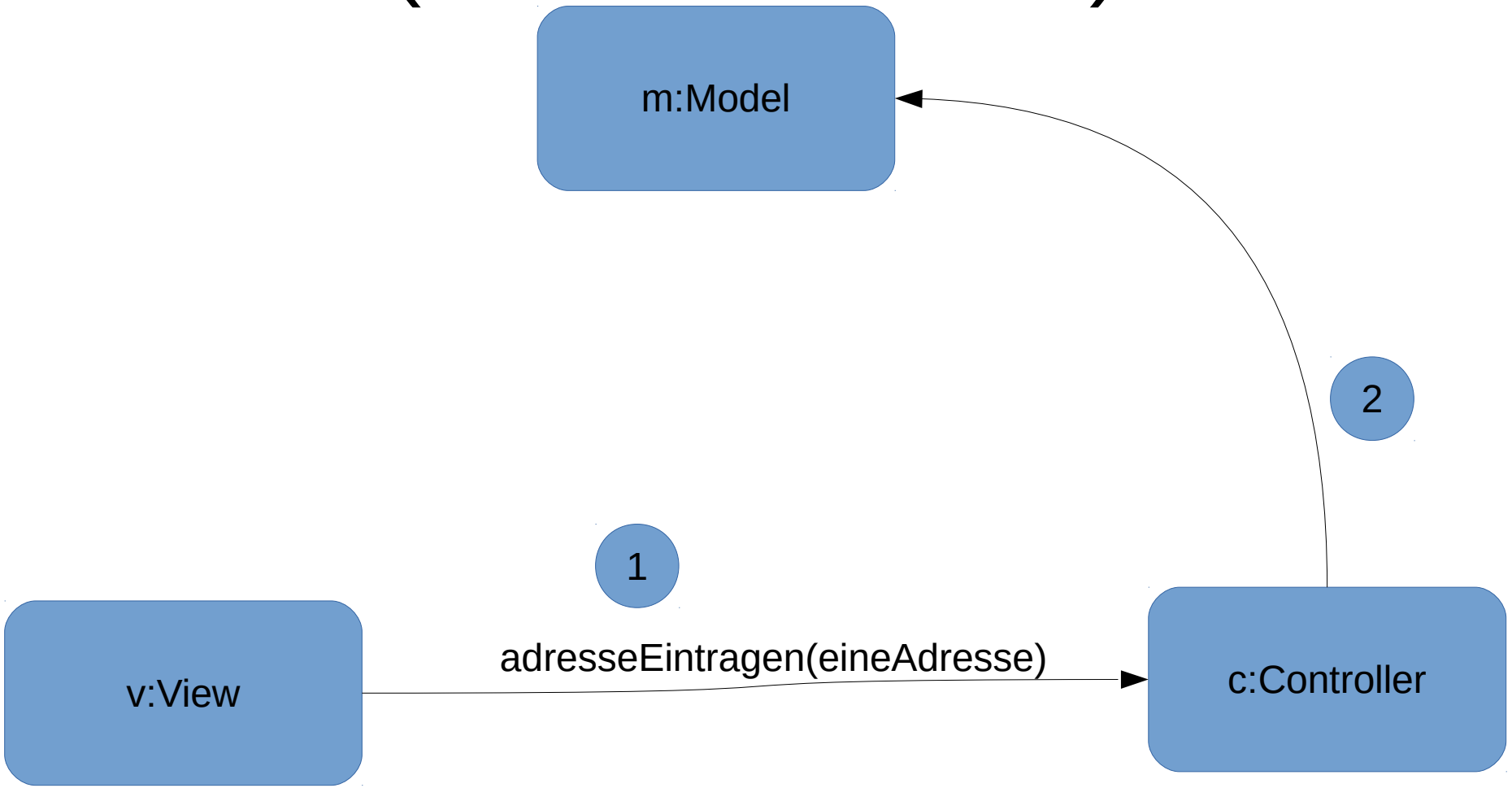
Weitergabe des Auftrags an Model (Methodenaufruf)



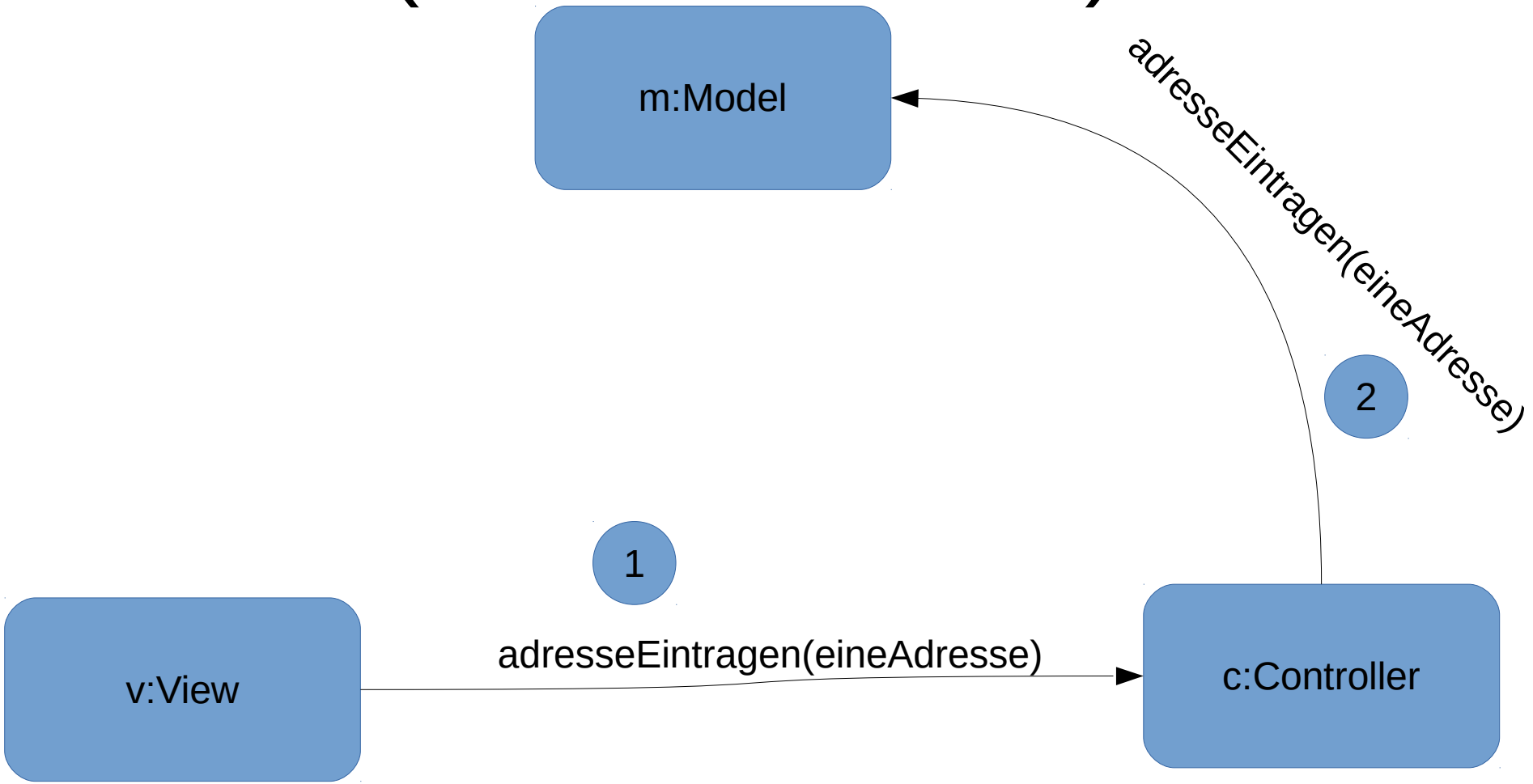
Weitergabe des Auftrags an Model (Methodenaufruf)

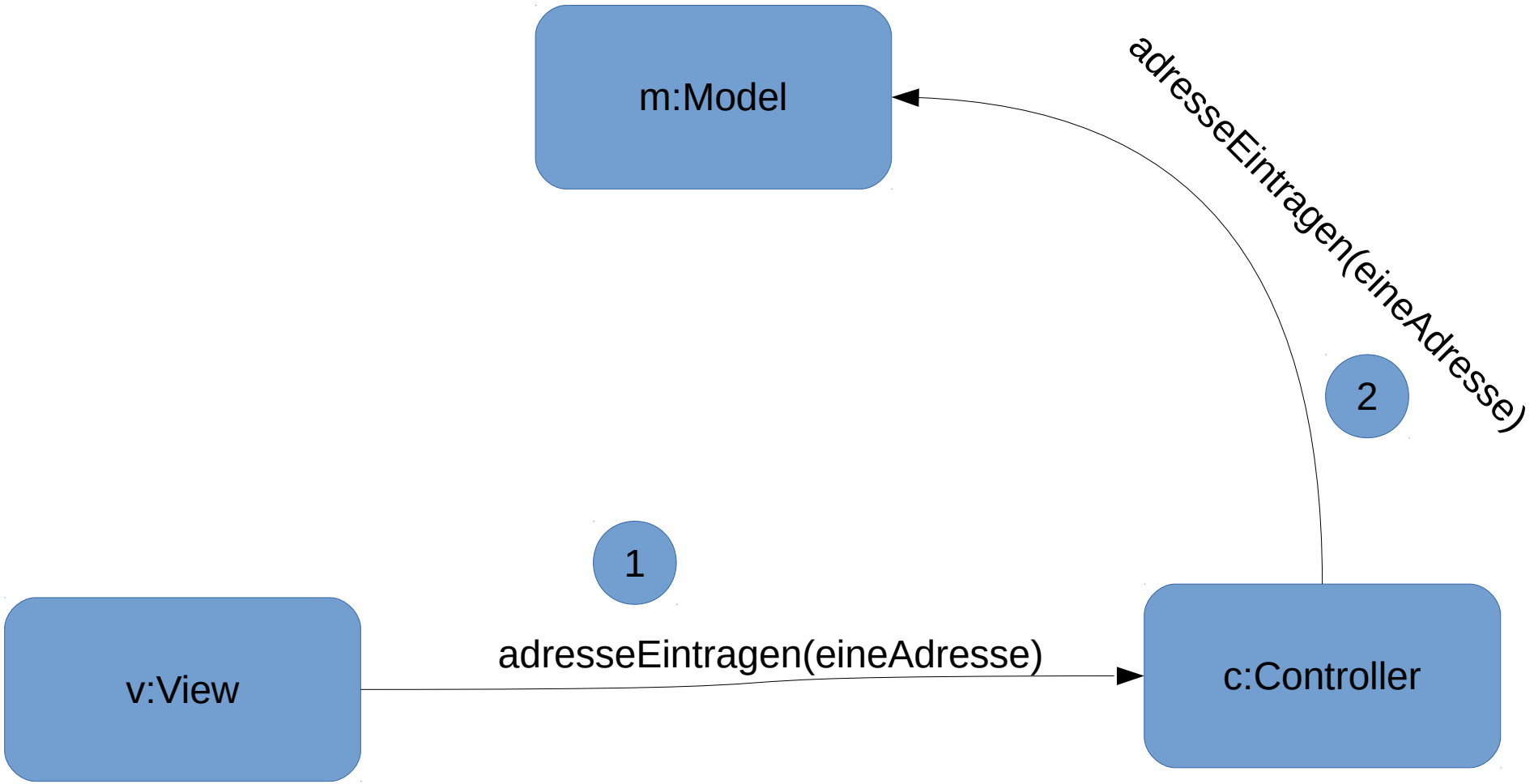


Weitergabe des Auftrags an Model (Methodenaufruf)

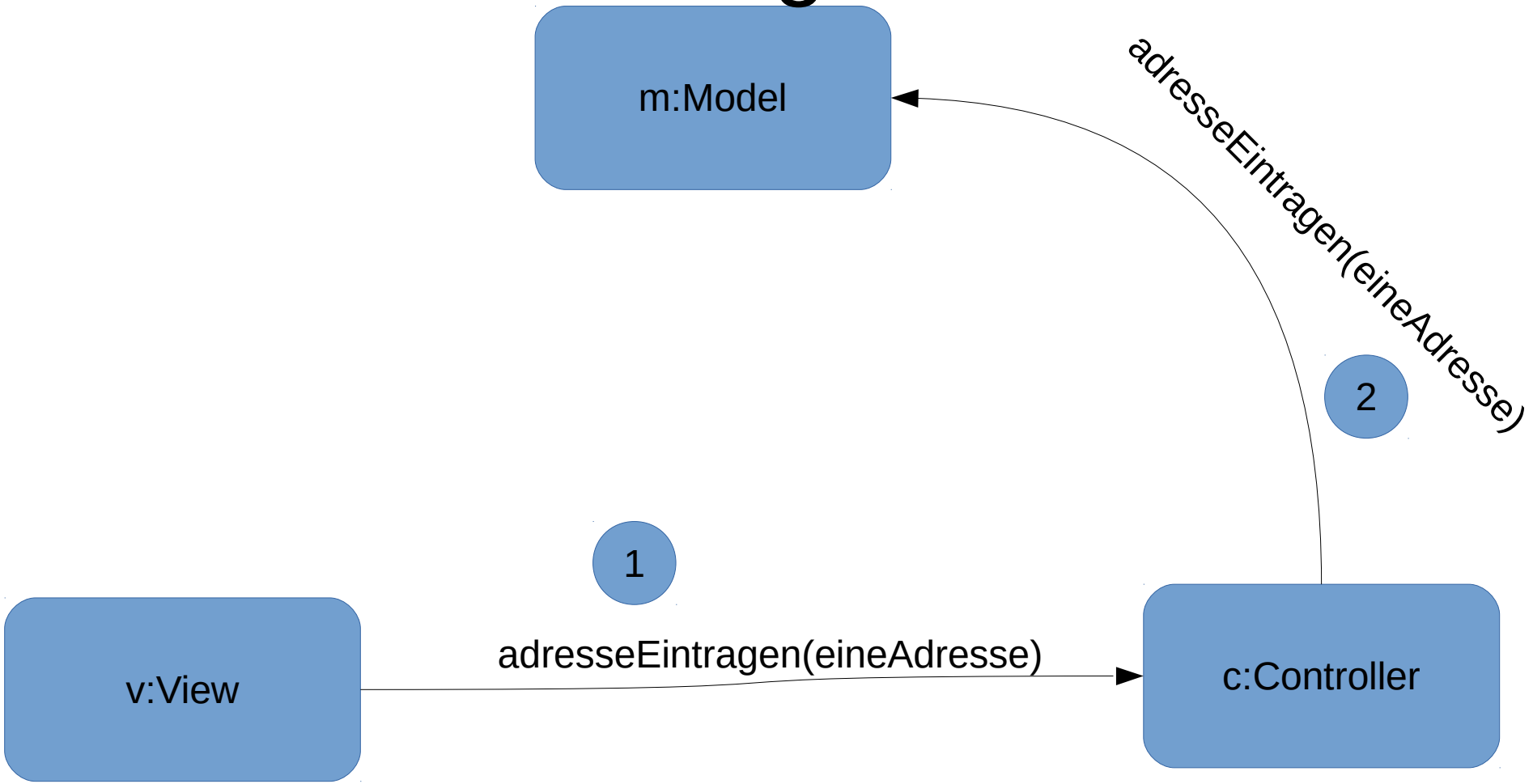


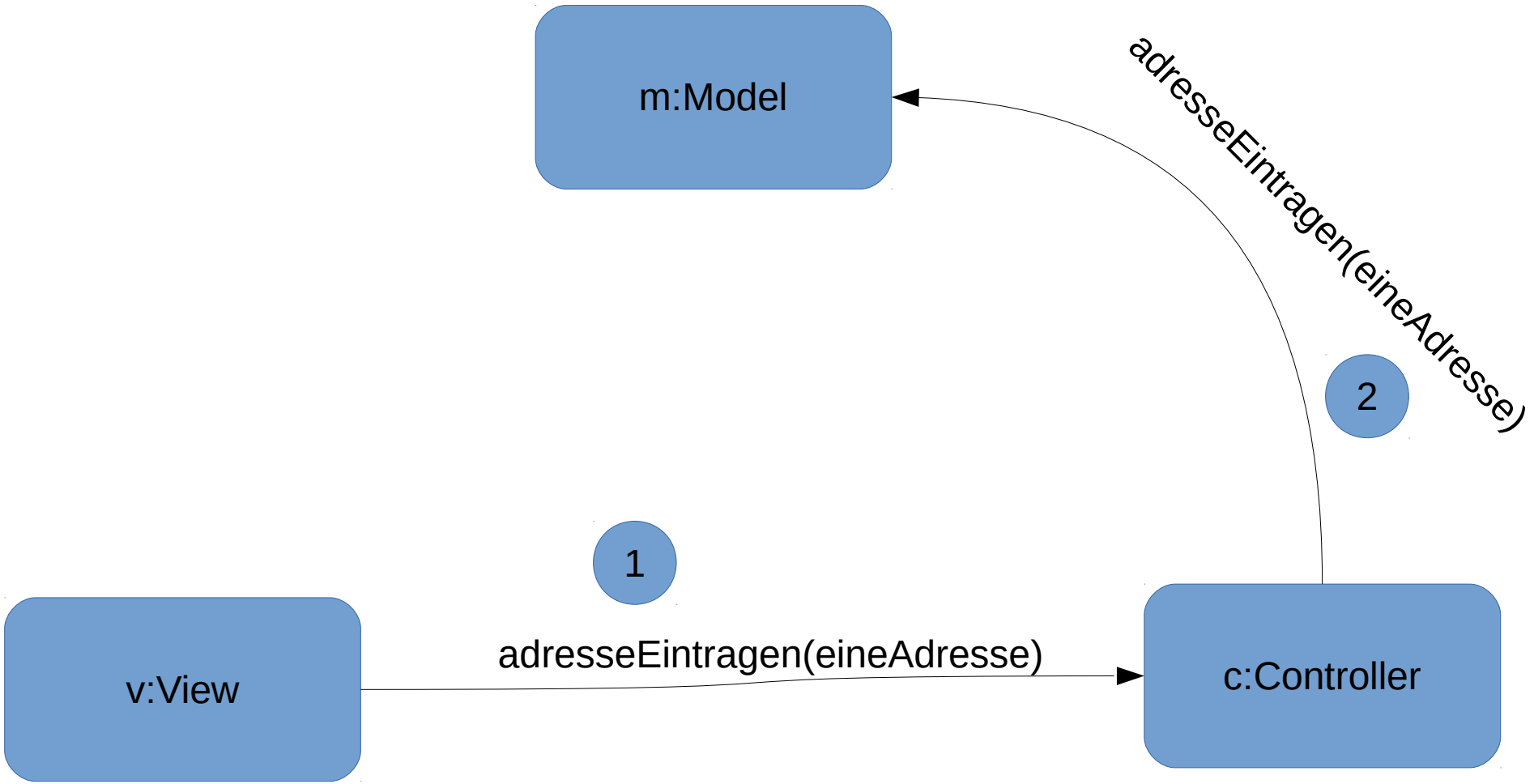
Weitergabe des Auftrags an Model (Methodenaufruf)



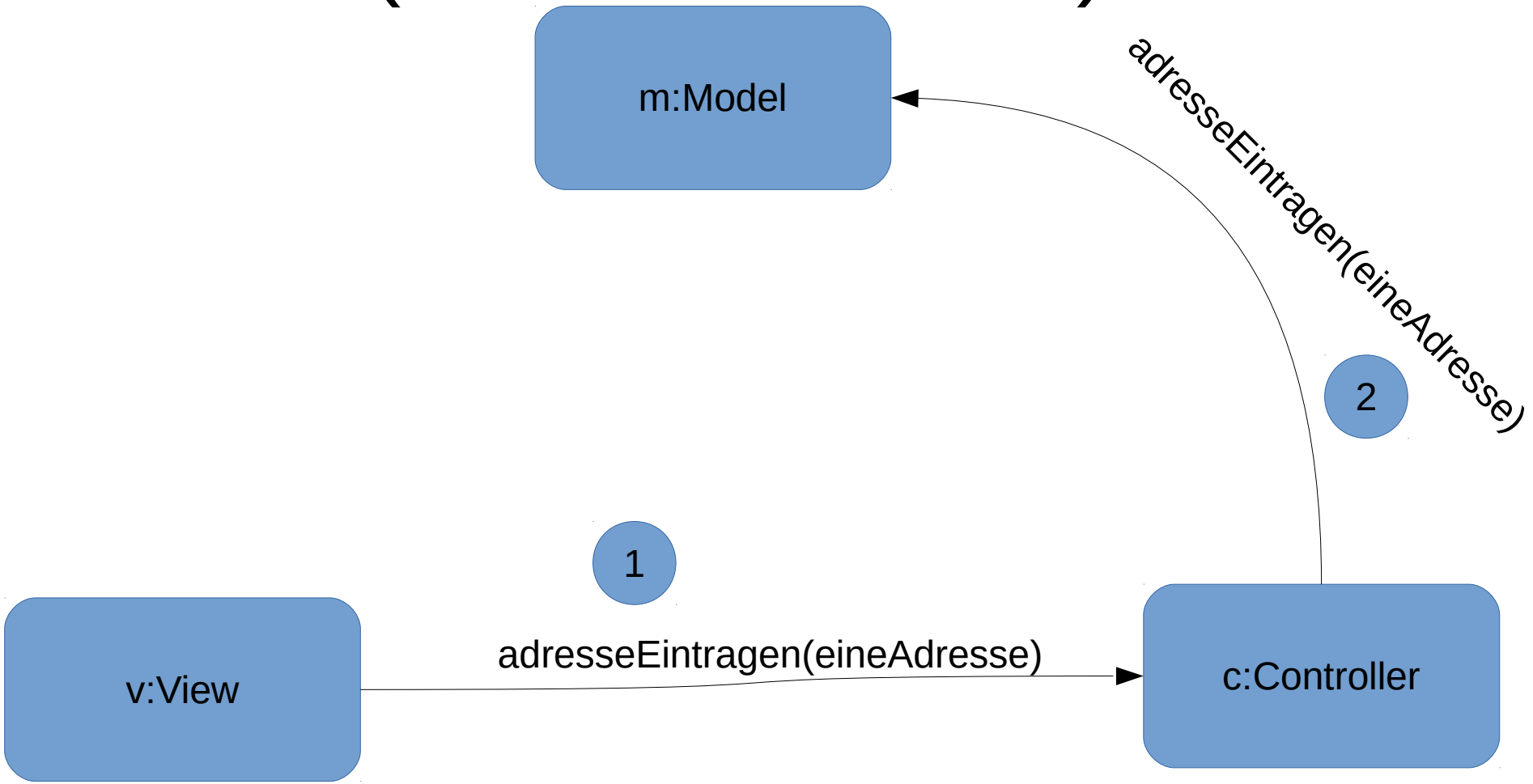

m:Model
adresseEintragen(eineAdresse)
2
1
v:View
adresseEintragen(eineAdresse)
c:Controller

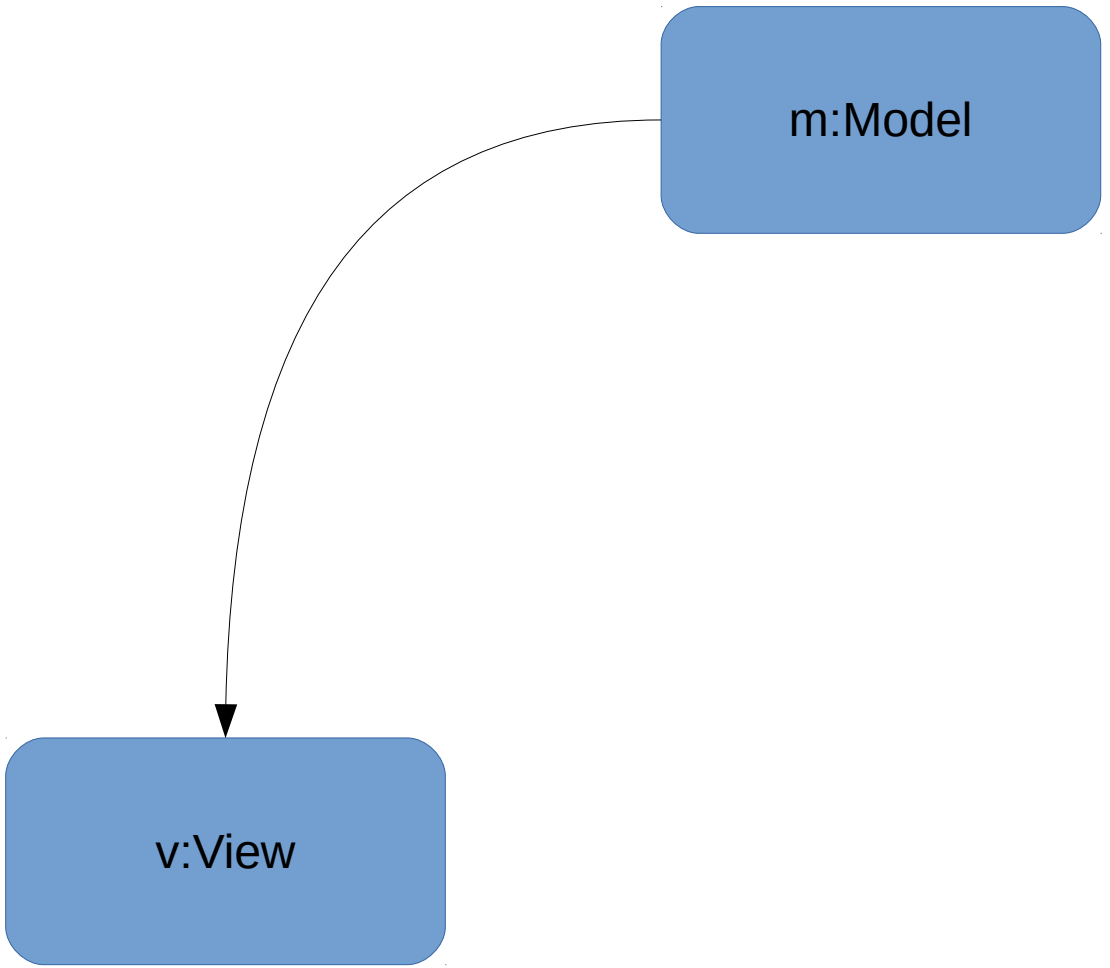
Model nimmt die entsprechenden Änderungen vor

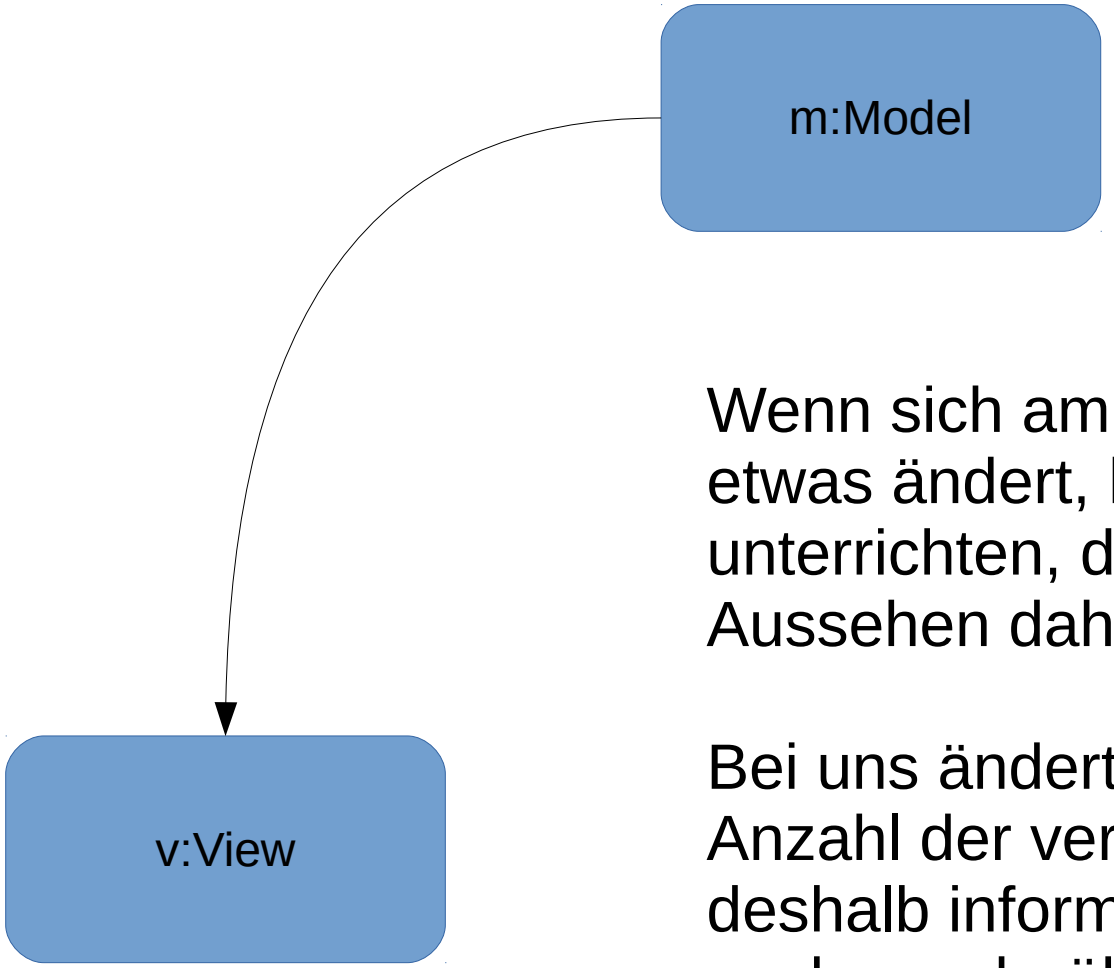



m:Model
adresseEintragen(eineAdresse)
2
1
v:View
adresseEintragen(eineAdresse)
c:Controller

Verbindung zum View (Referenzattribut)



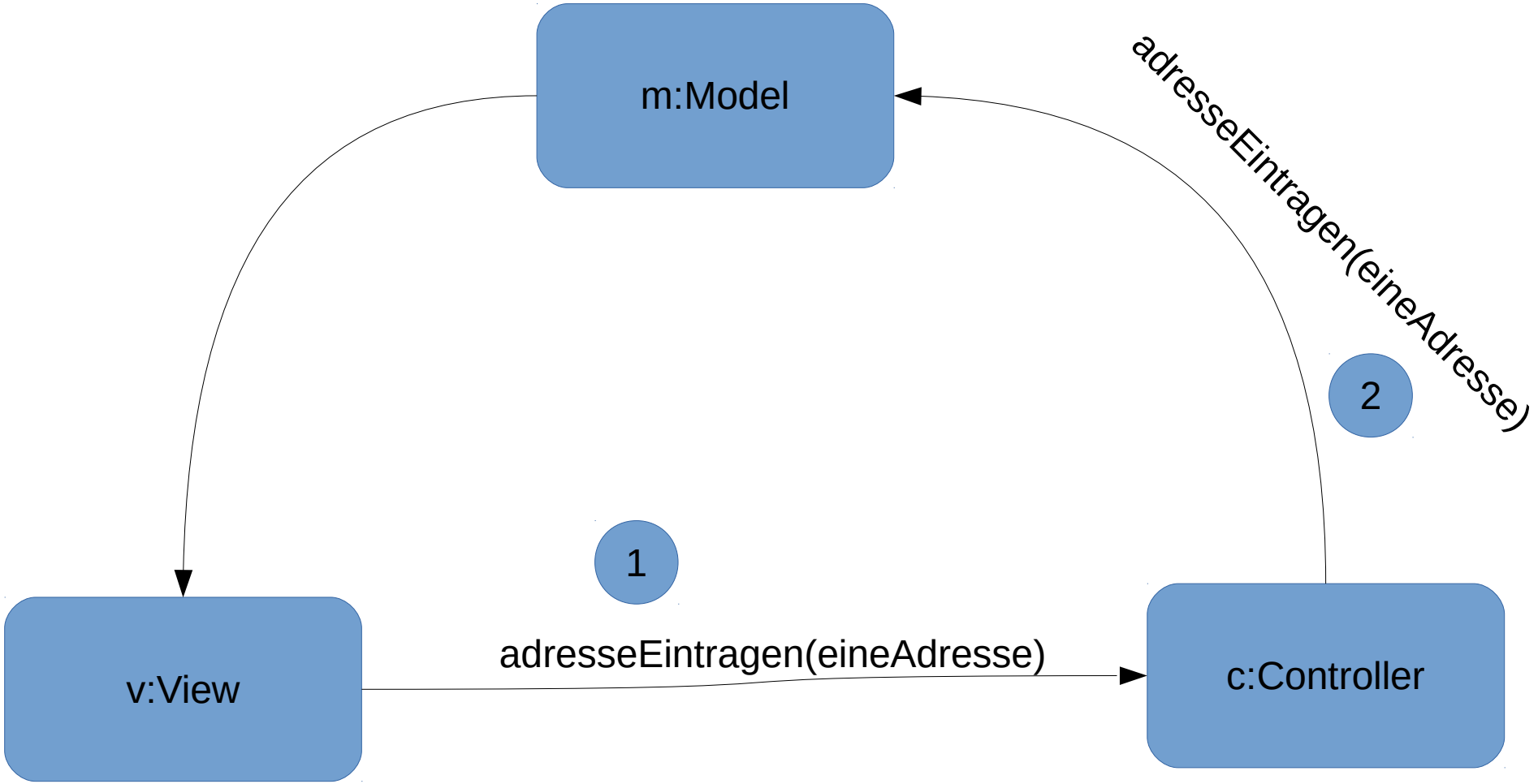

m:Model
v:View



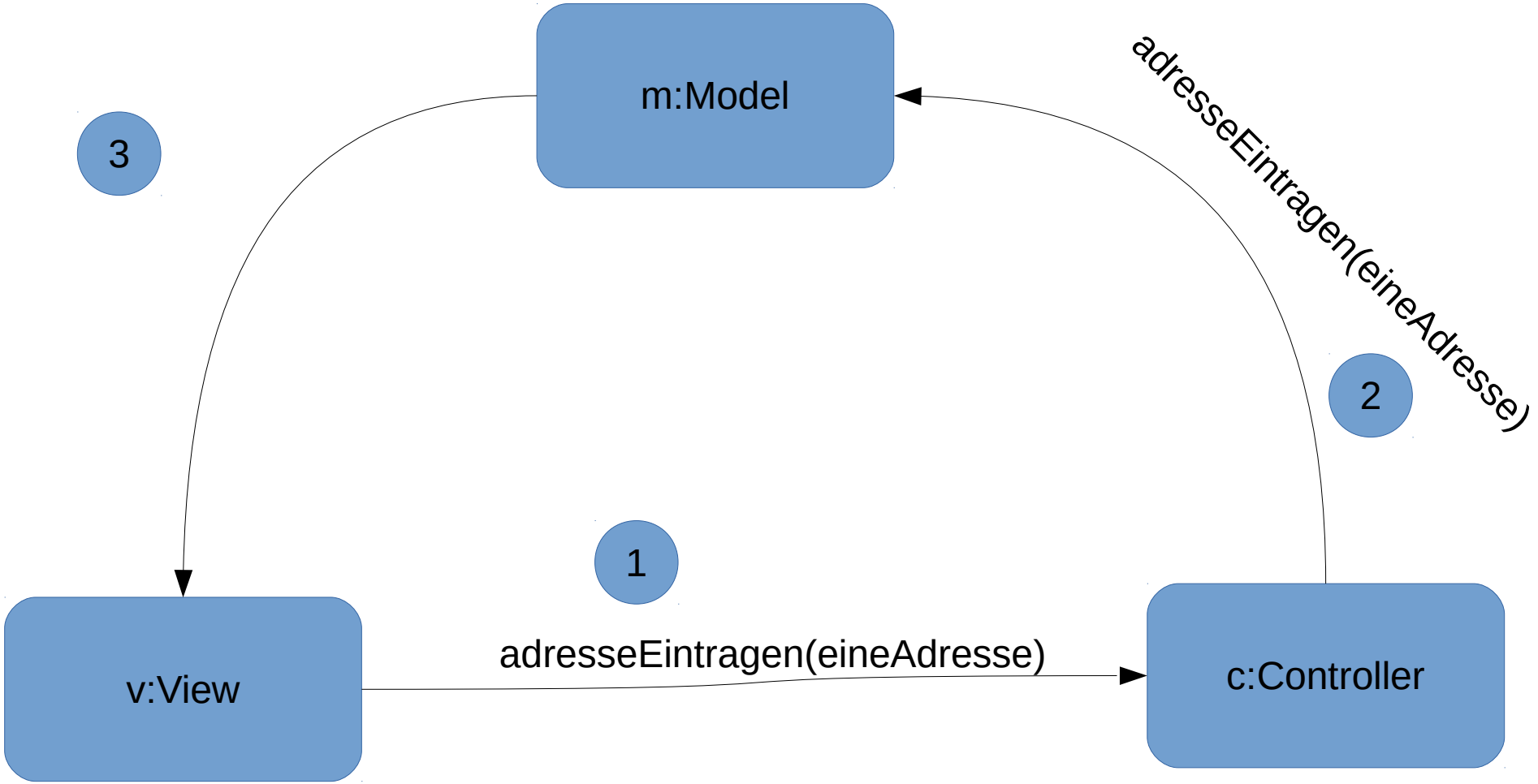
Wenn sich am Zustand des Models etwas ändert, kann es den View davon unterrichten, damit der View sein Aussehen dahingehend anpasst.

Bei uns ändert sich an eigentlich nur die Anzahl der verwalteten Einträge, und deshalb informiert das Model den View auch nur darüber.

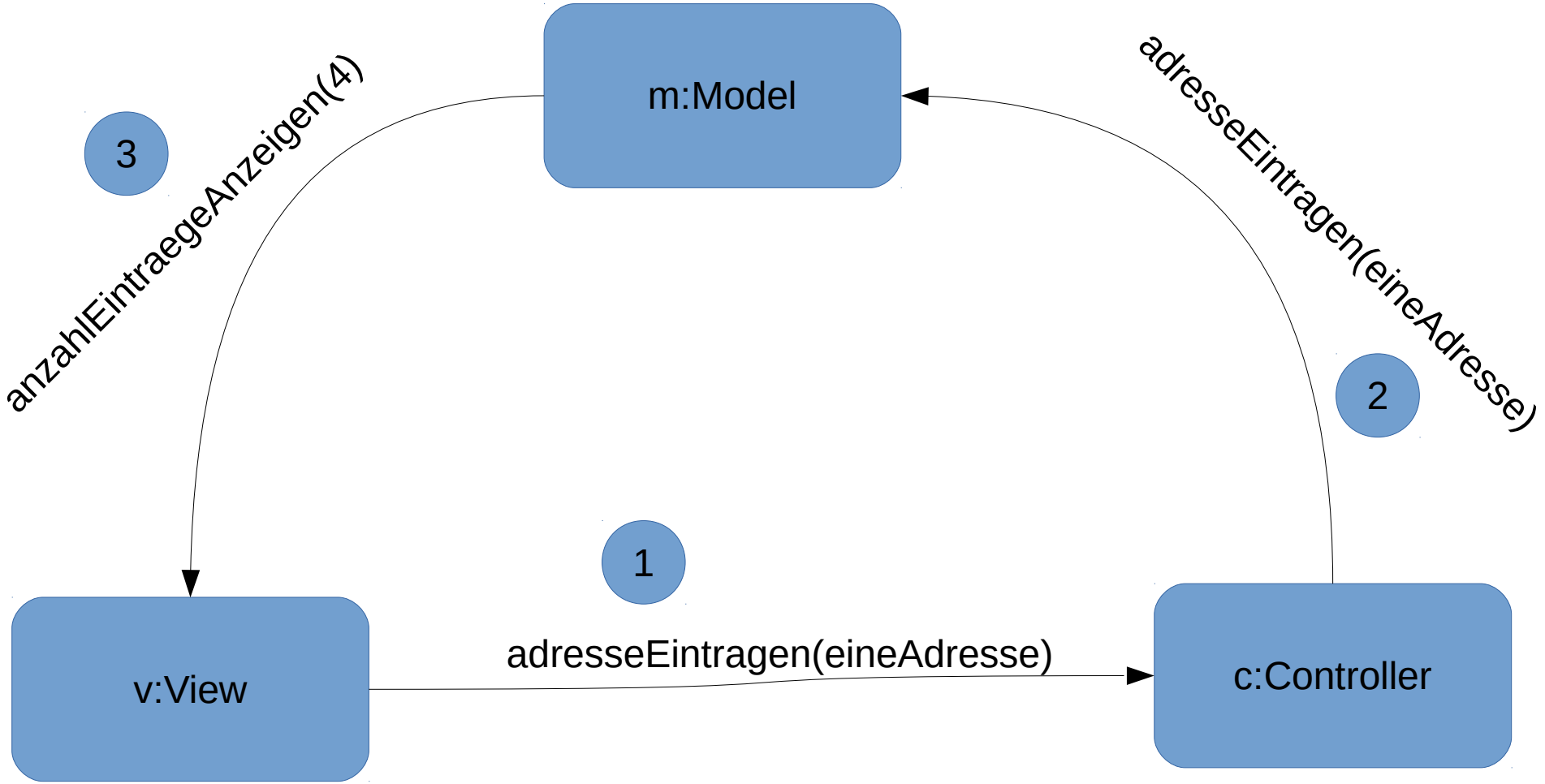
Model informiert View von Änderung



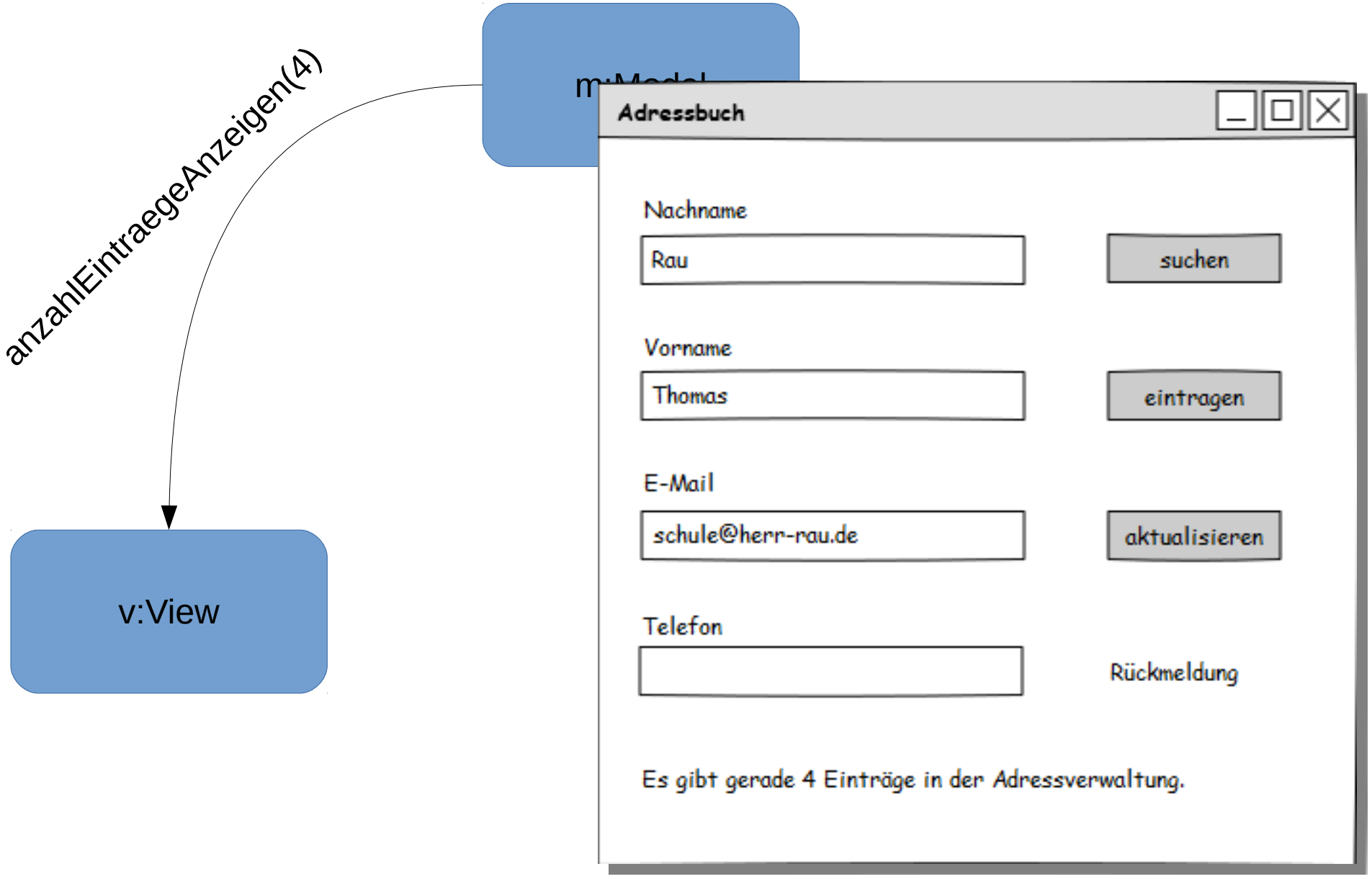
Model informiert View von Änderung



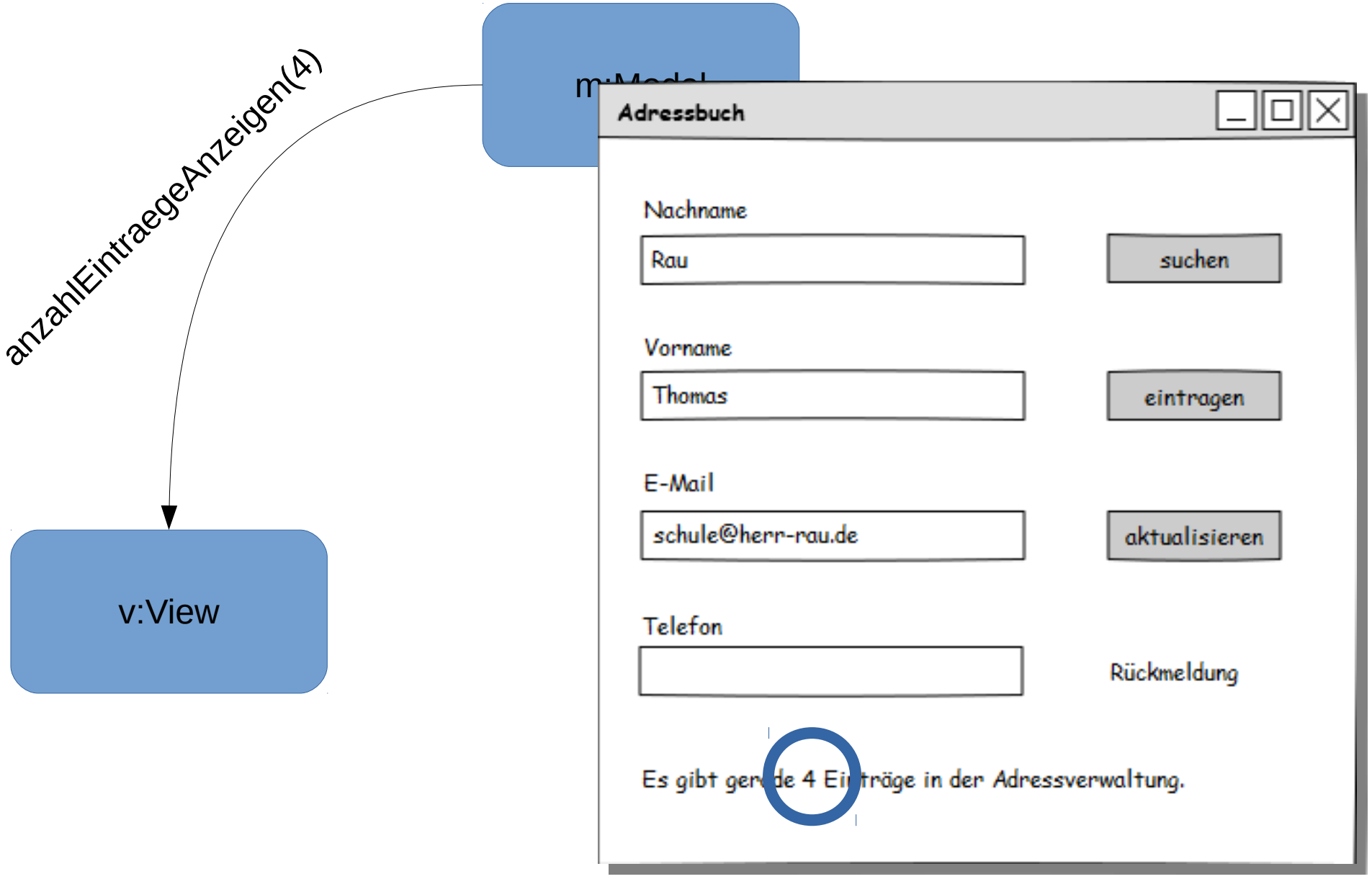
Model informiert View von Änderung

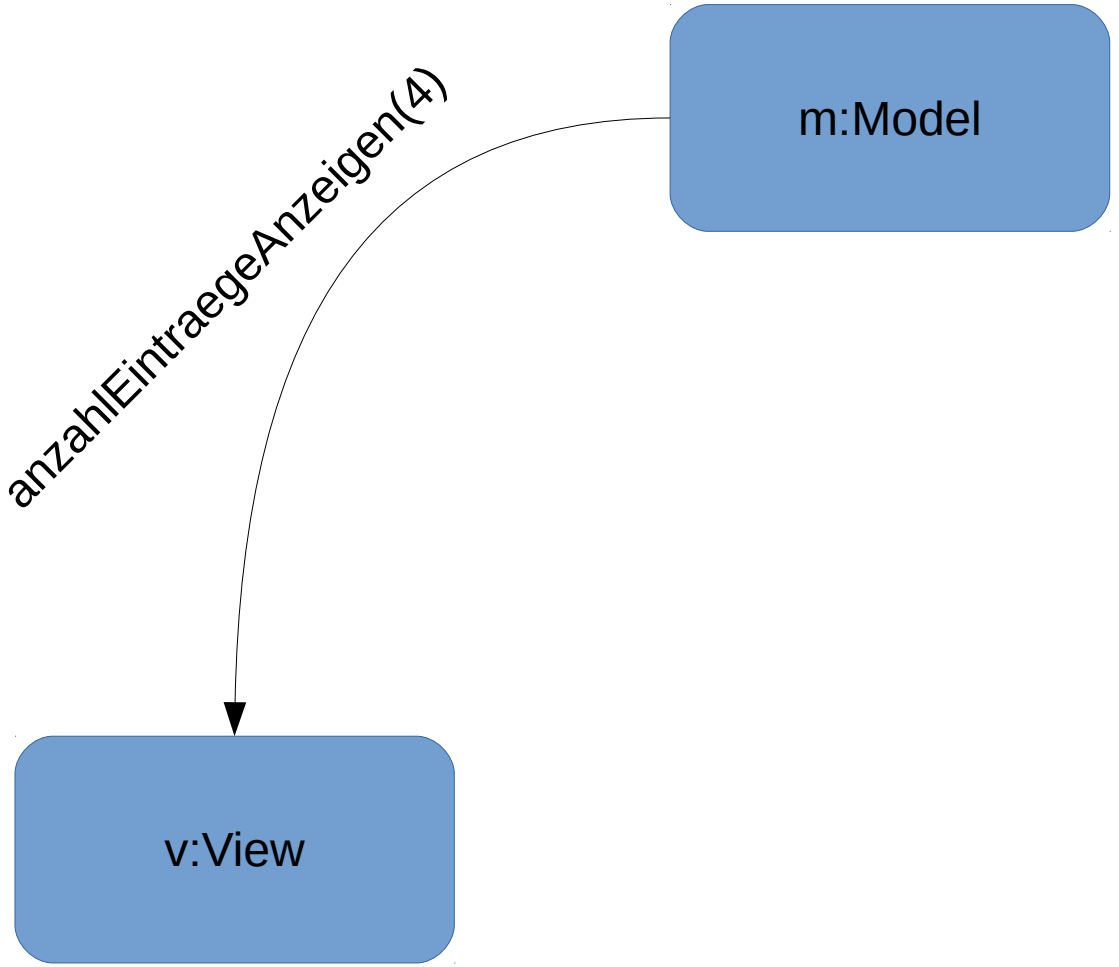


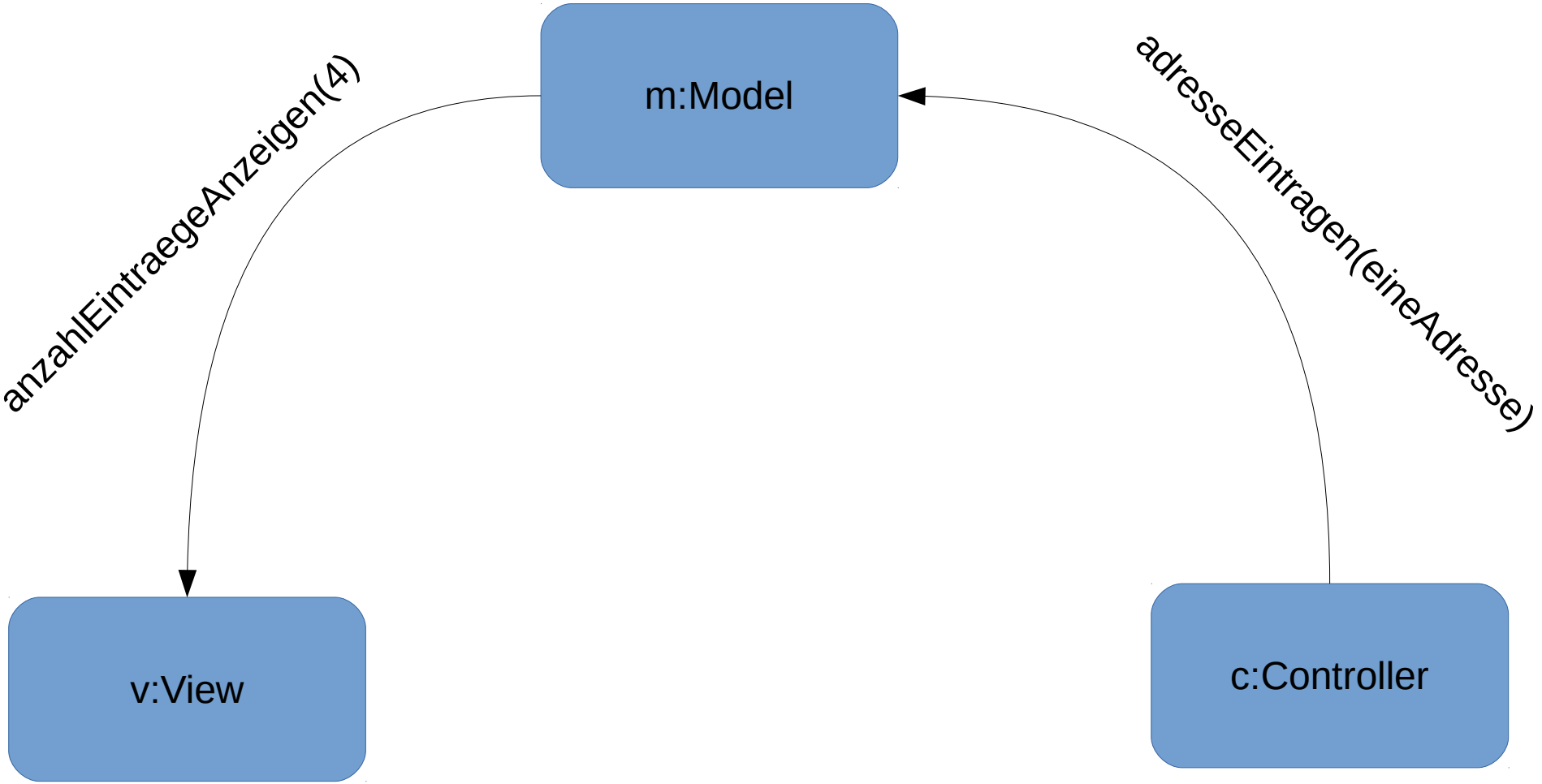
View ändert daraufhin Anzeige

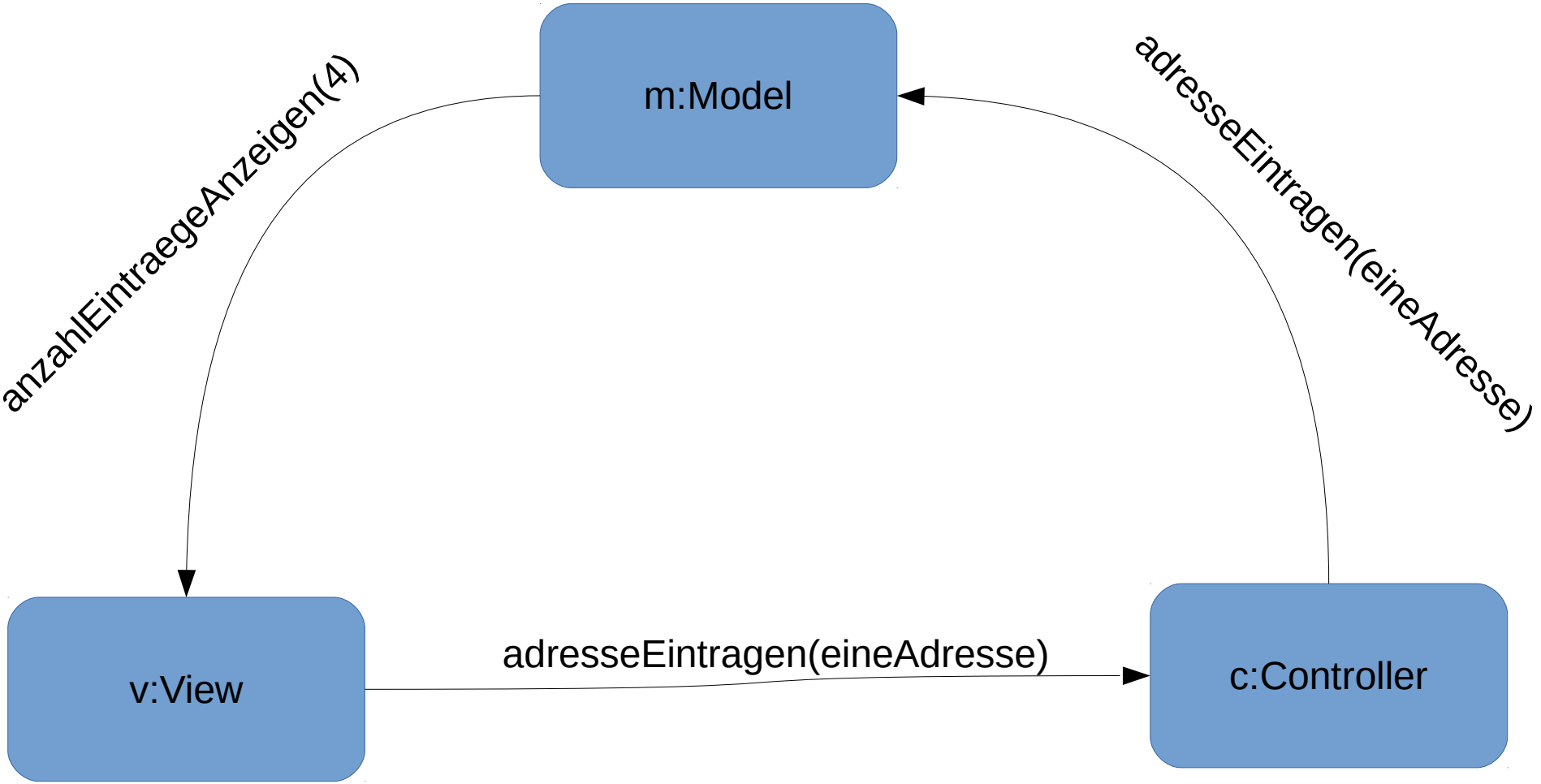


View ändert daraufhin Anzeige

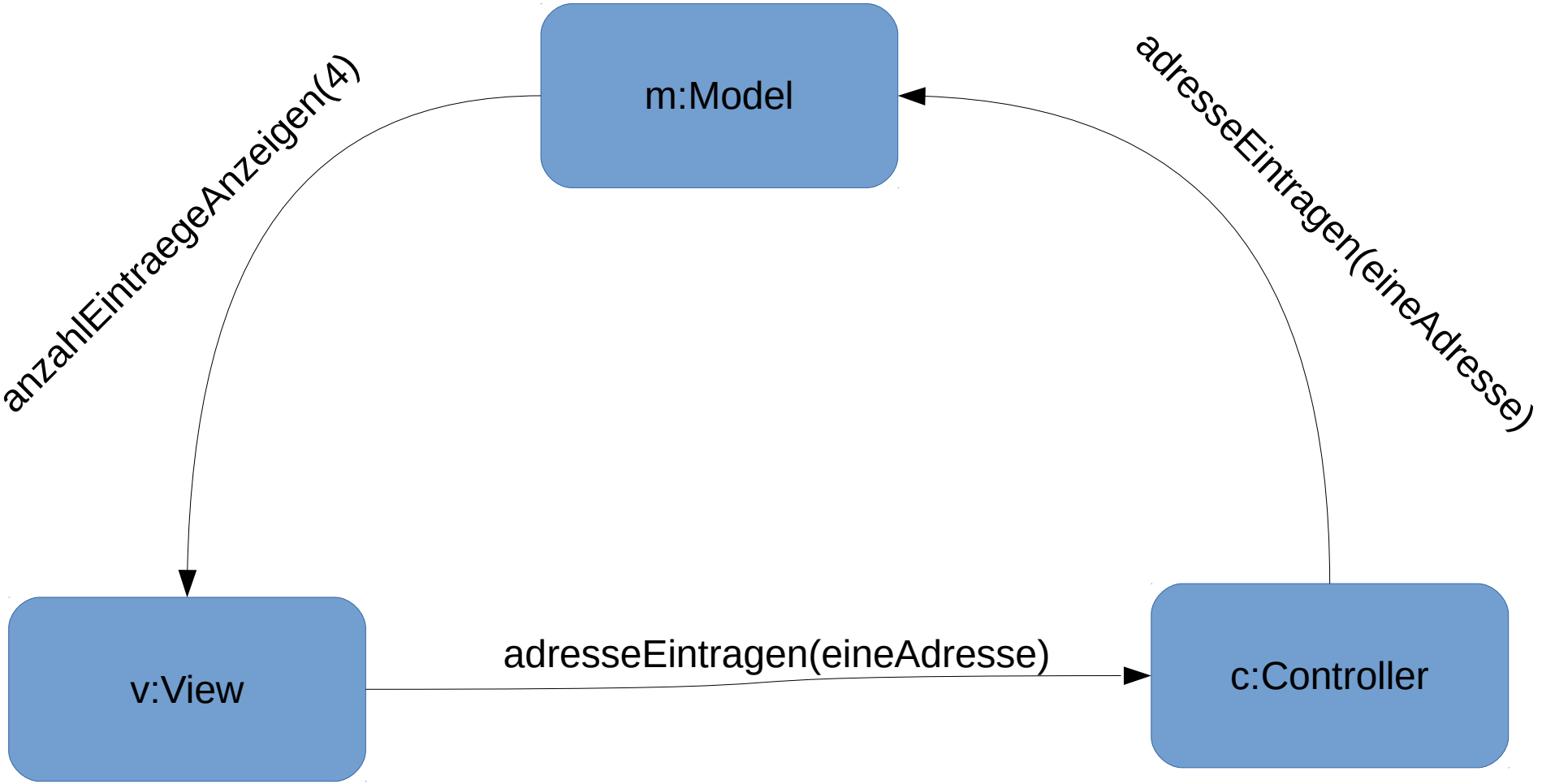


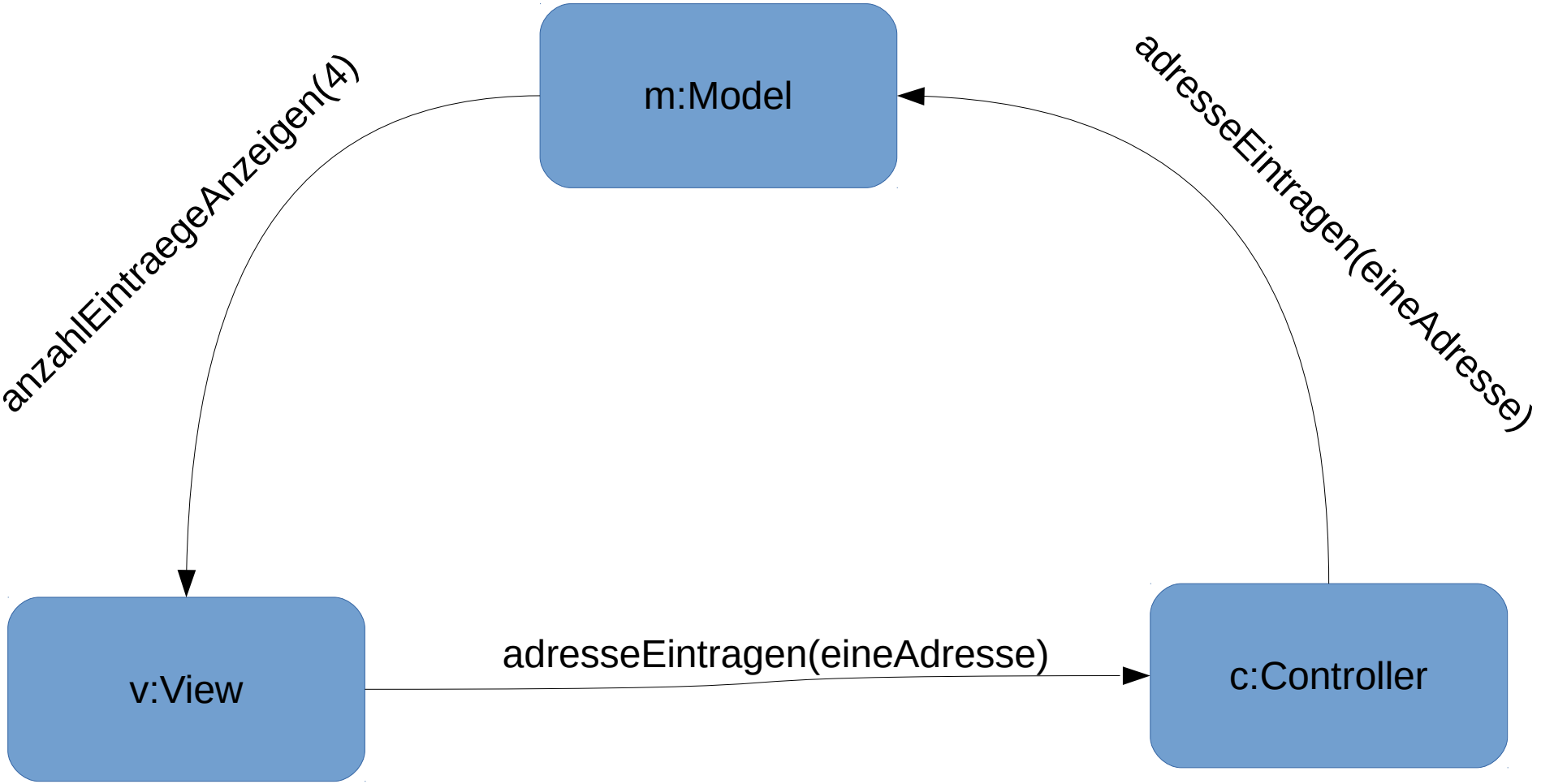

m:Model
anzahlEintraegeAnzeigen(4)
v:View

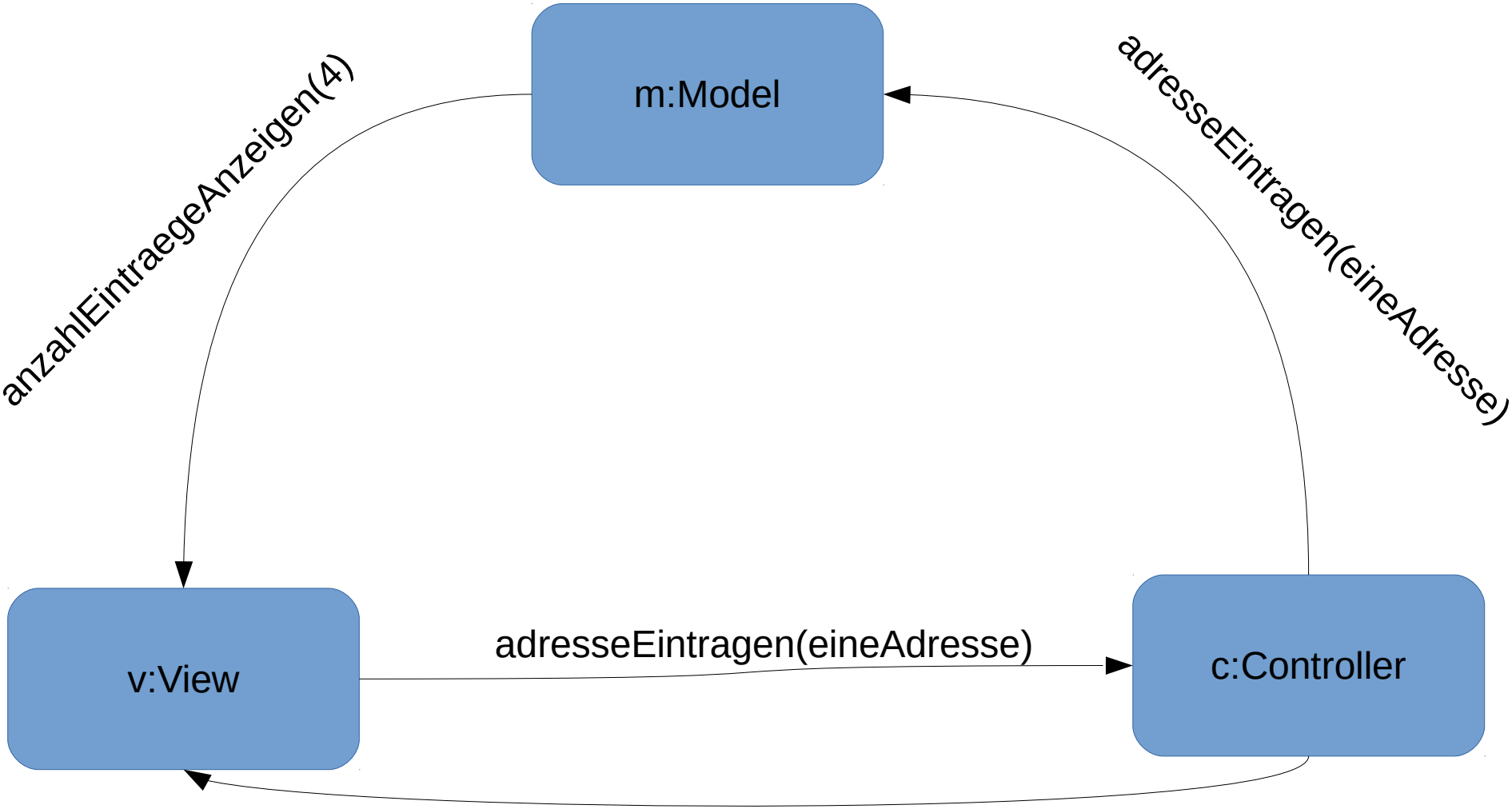

m:Model
anzahlEintraegeAnzeigen(4)
adresseEintragen(eineAdresse)
v:View
c:Controller


m:Model
anzahlEintraegeAnzeigen(4)
adresseEintragen(eineAdresse)
v:View
adresseEintragen(eineAdresse)
c:Controller

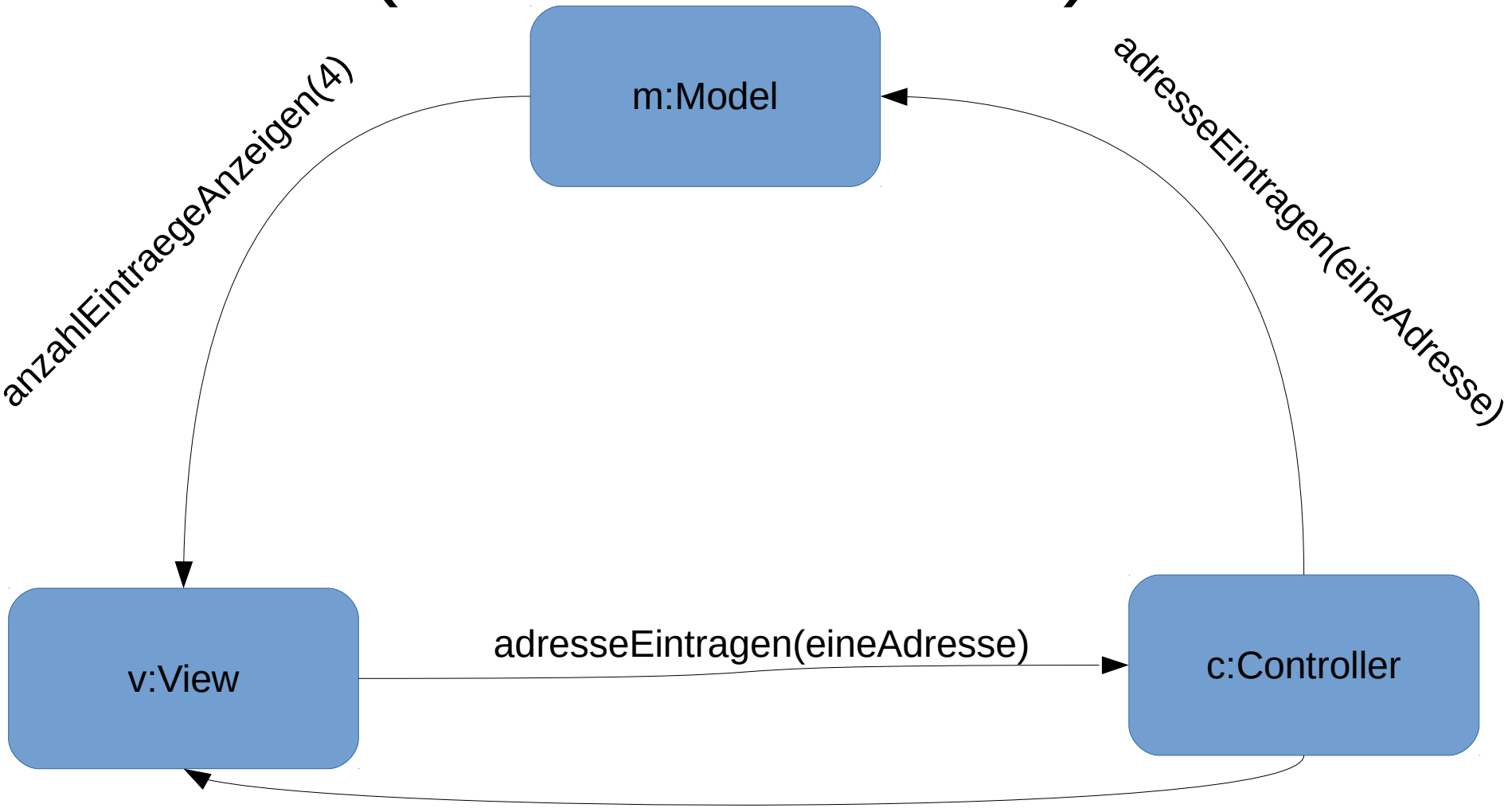
Außerdem gibt es eine Verbindung Controller > View



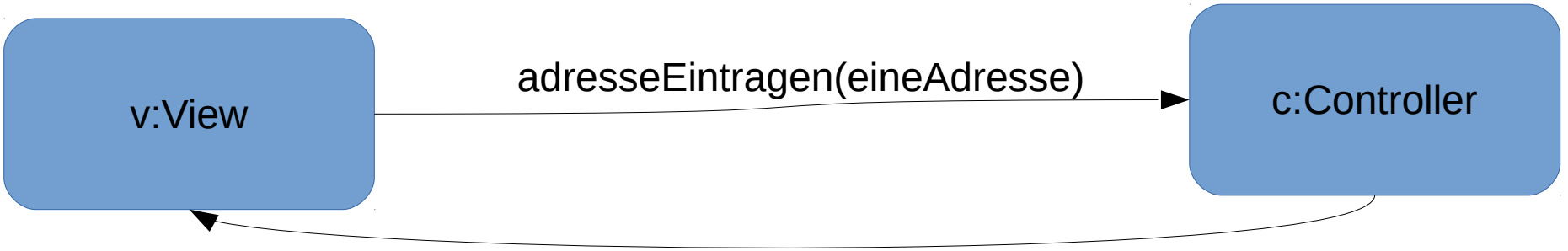

m:Model
anzahlEintraegeAnzeigen(4)
adresseEintragen(eineAdresse)
v:View
adresseEintragen(eineAdresse)
c:Controller


m:Model
anzahlEintraegeAnzeigen(4)
adresseEintragen(eineAdresse)
v:View
adresseEintragen(eineAdresse)
c:Controller

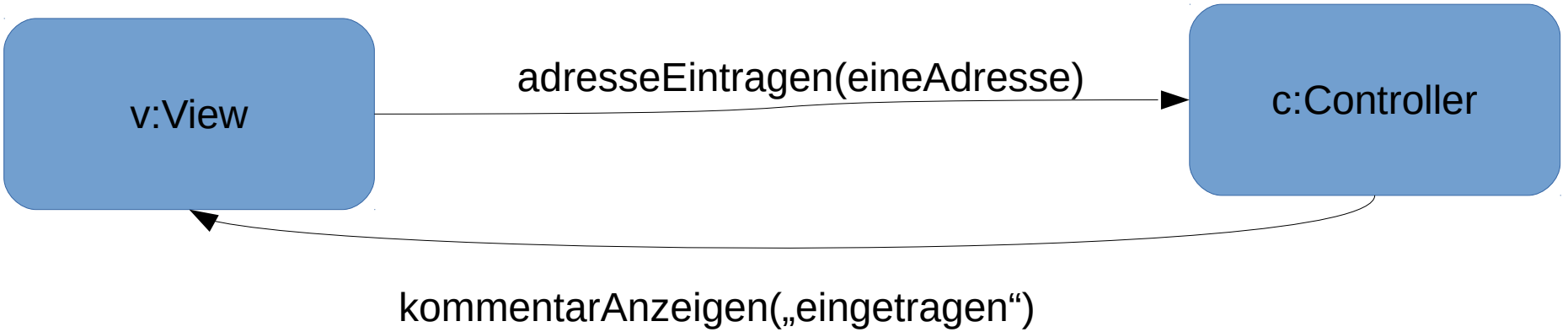
Verbindung Controller > View (Referenzattribut)



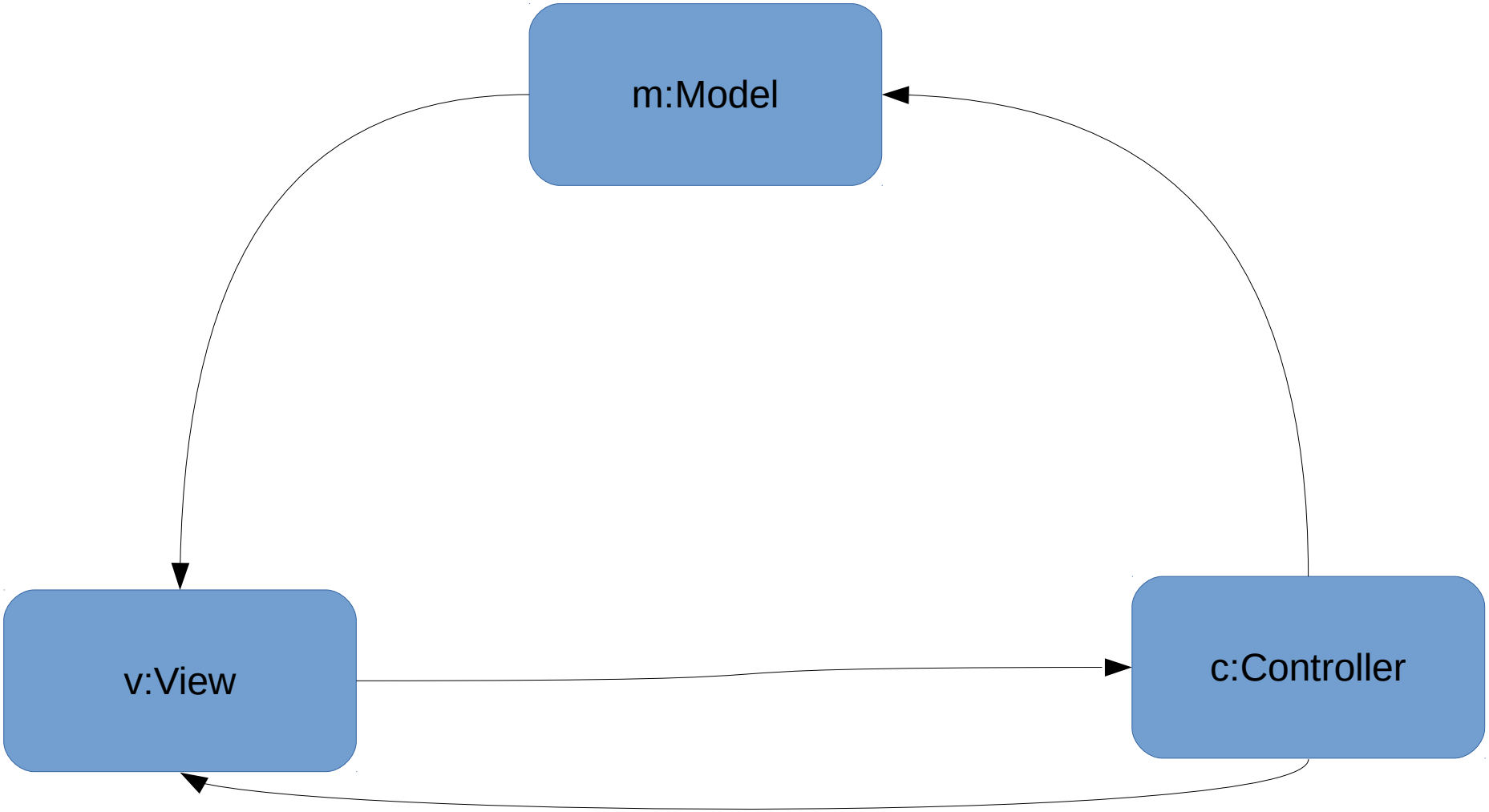
Controller kann View über Erfolg/Misserfolg usw. informieren

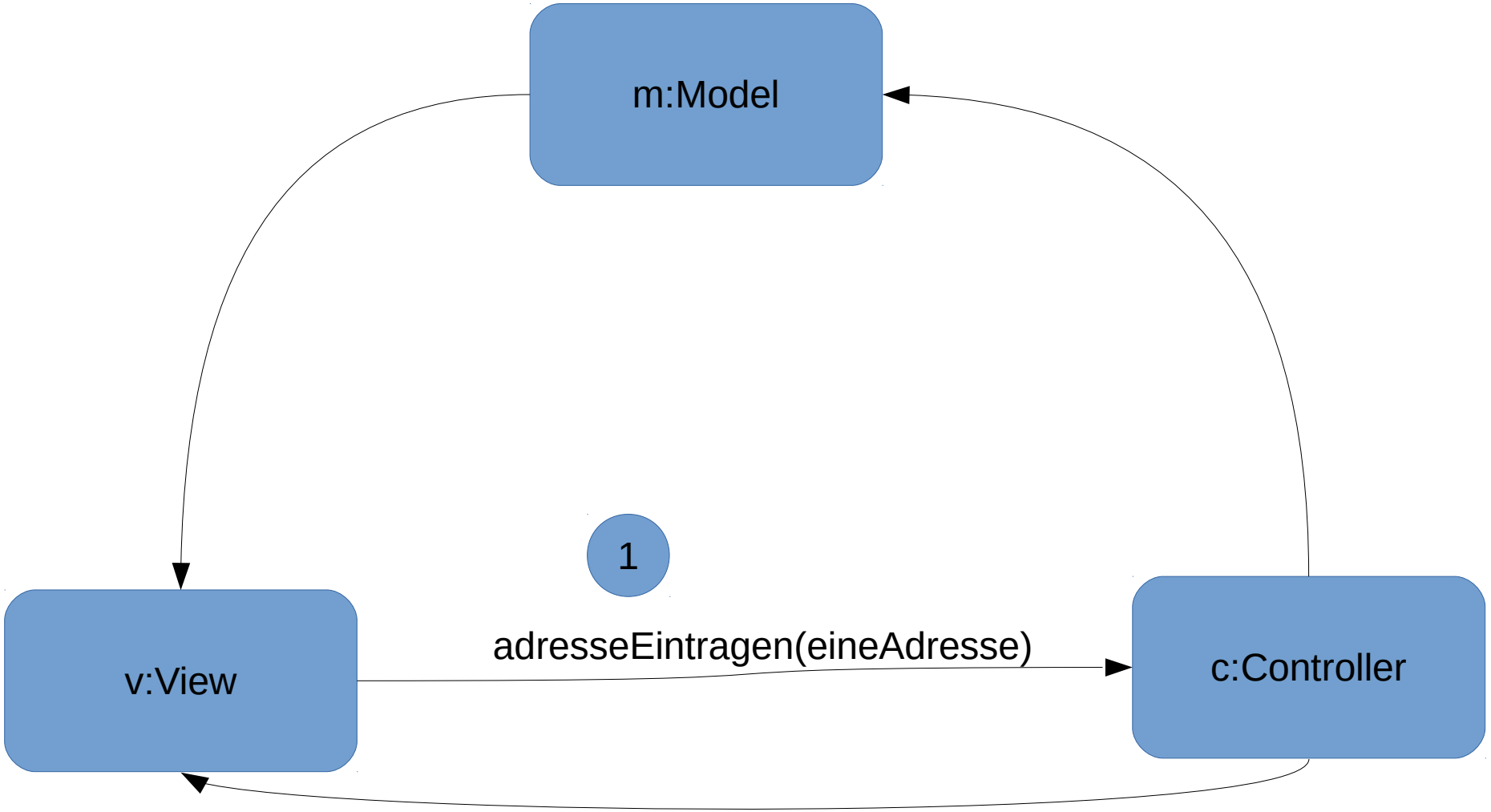


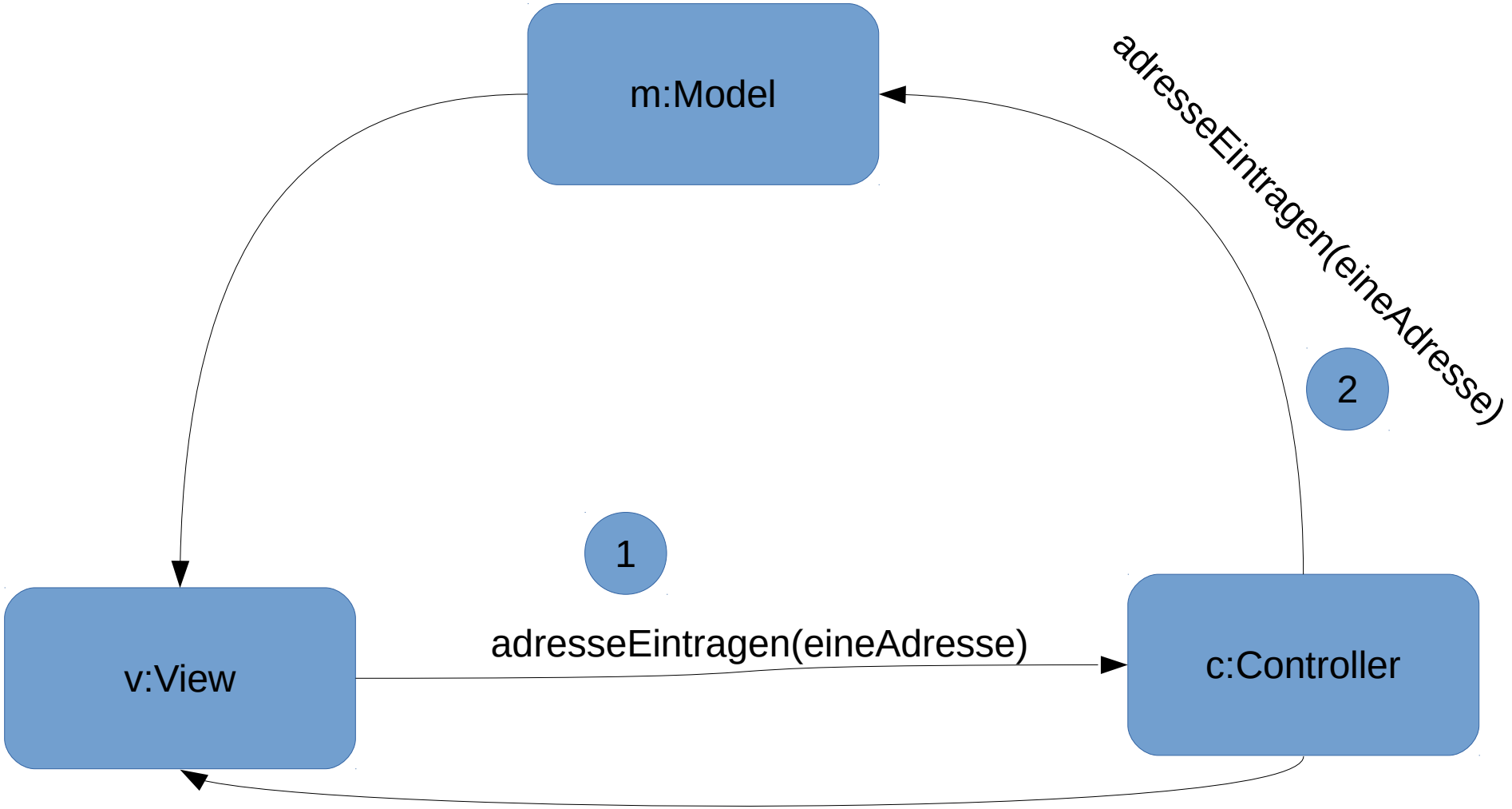
Controller kann View über Erfolg/Misserfolg usw. informieren

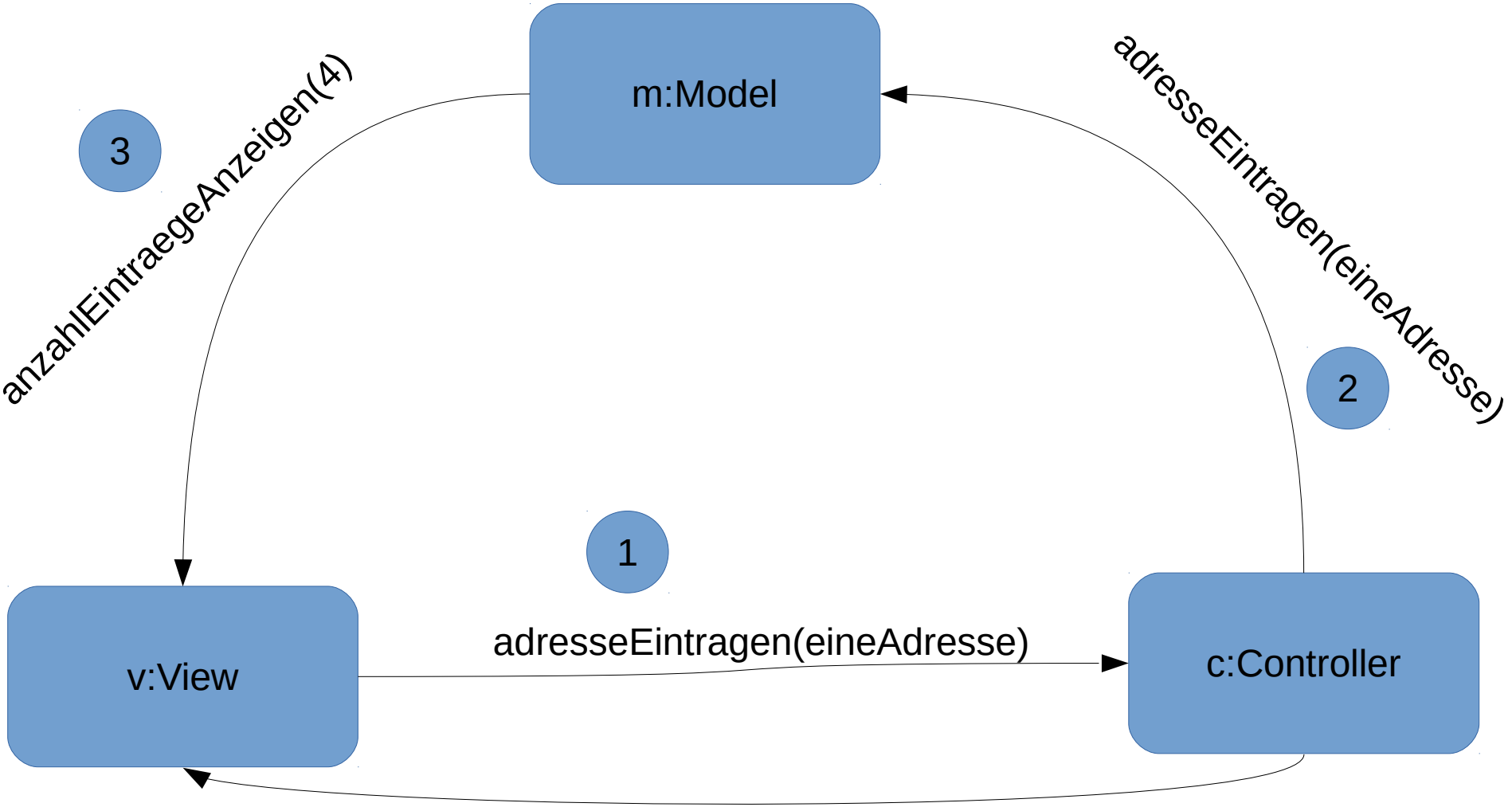


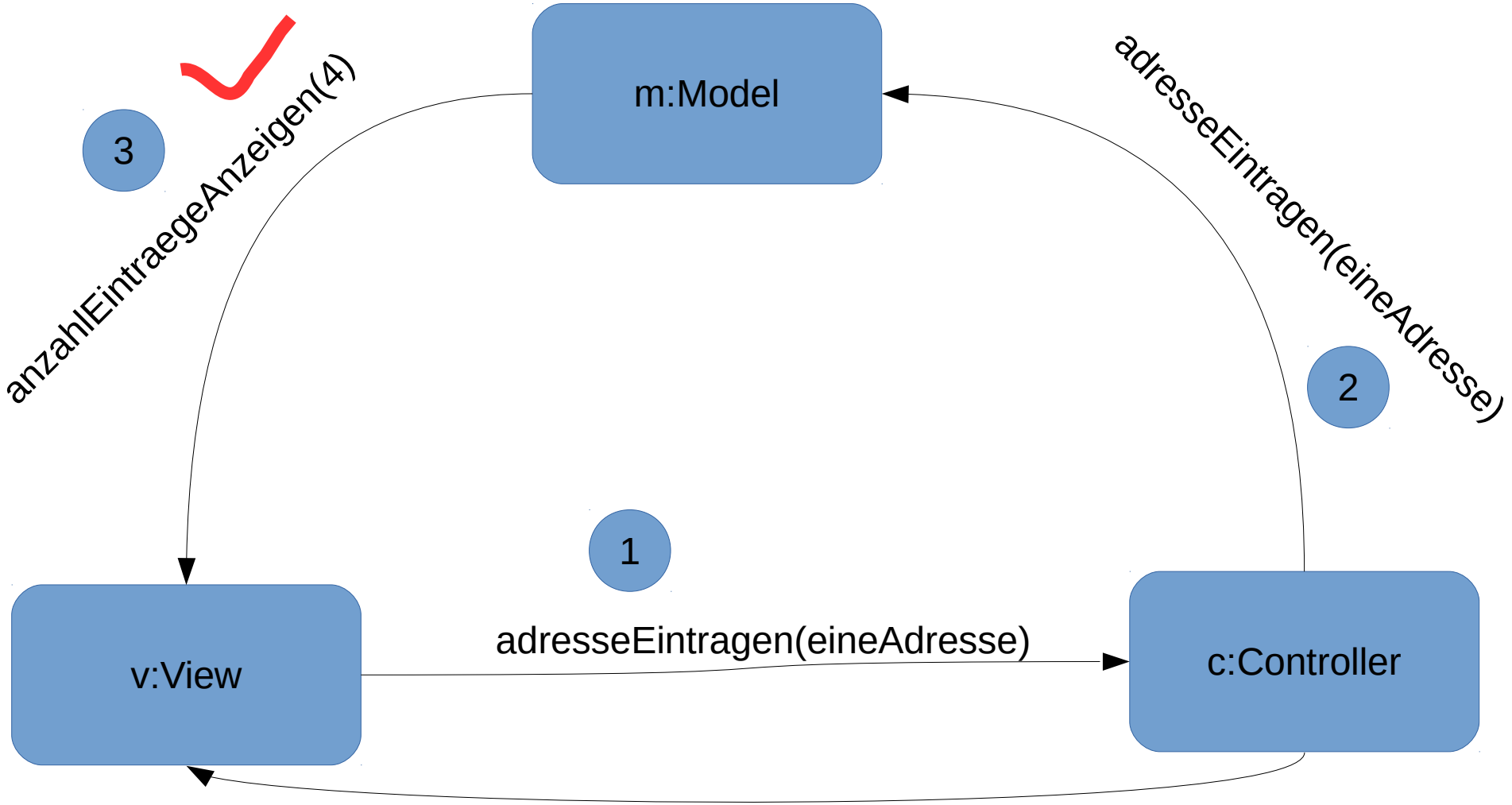
Und jetzt zusammen

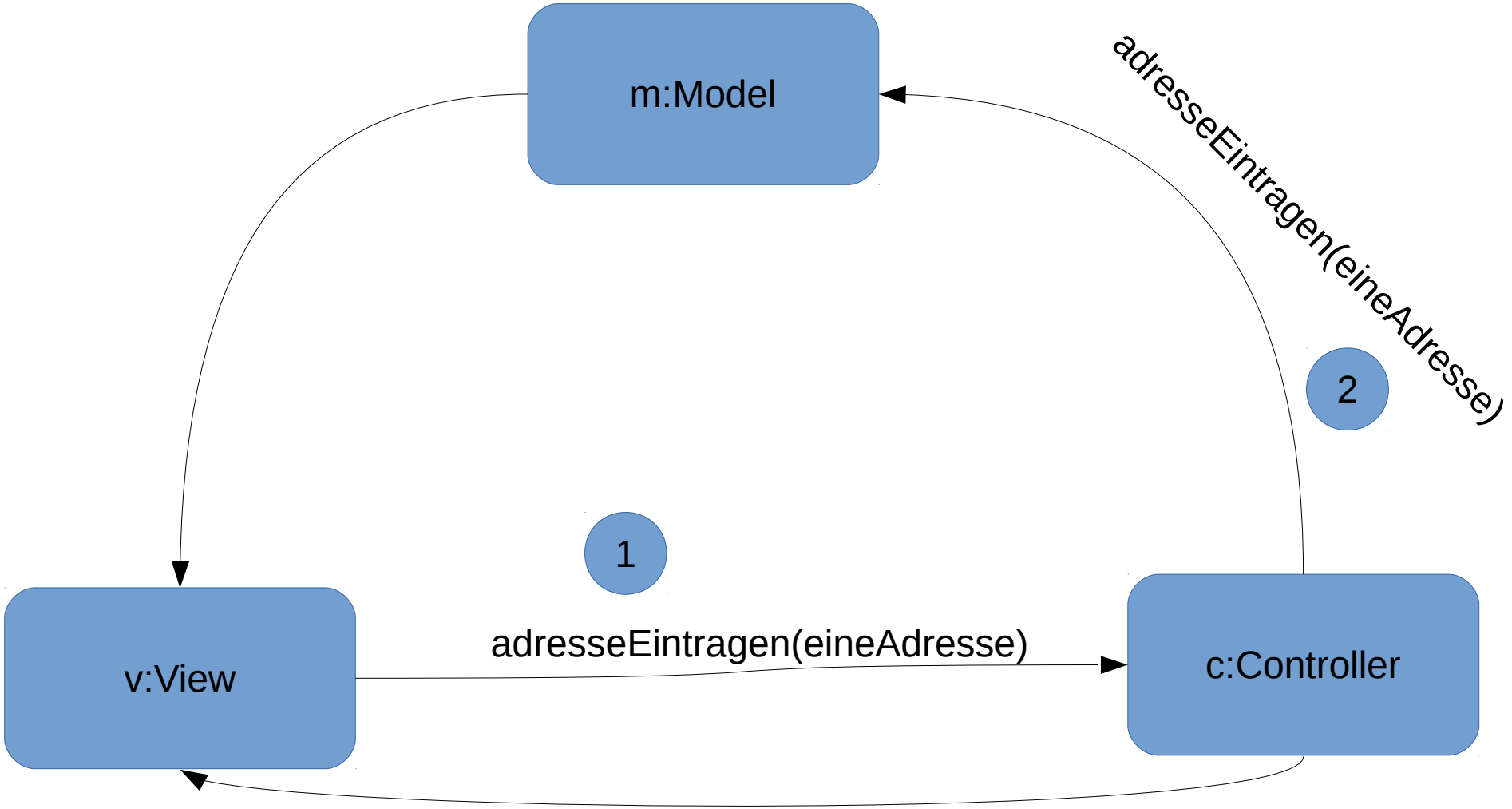

m:Model
v:View
c:Controller

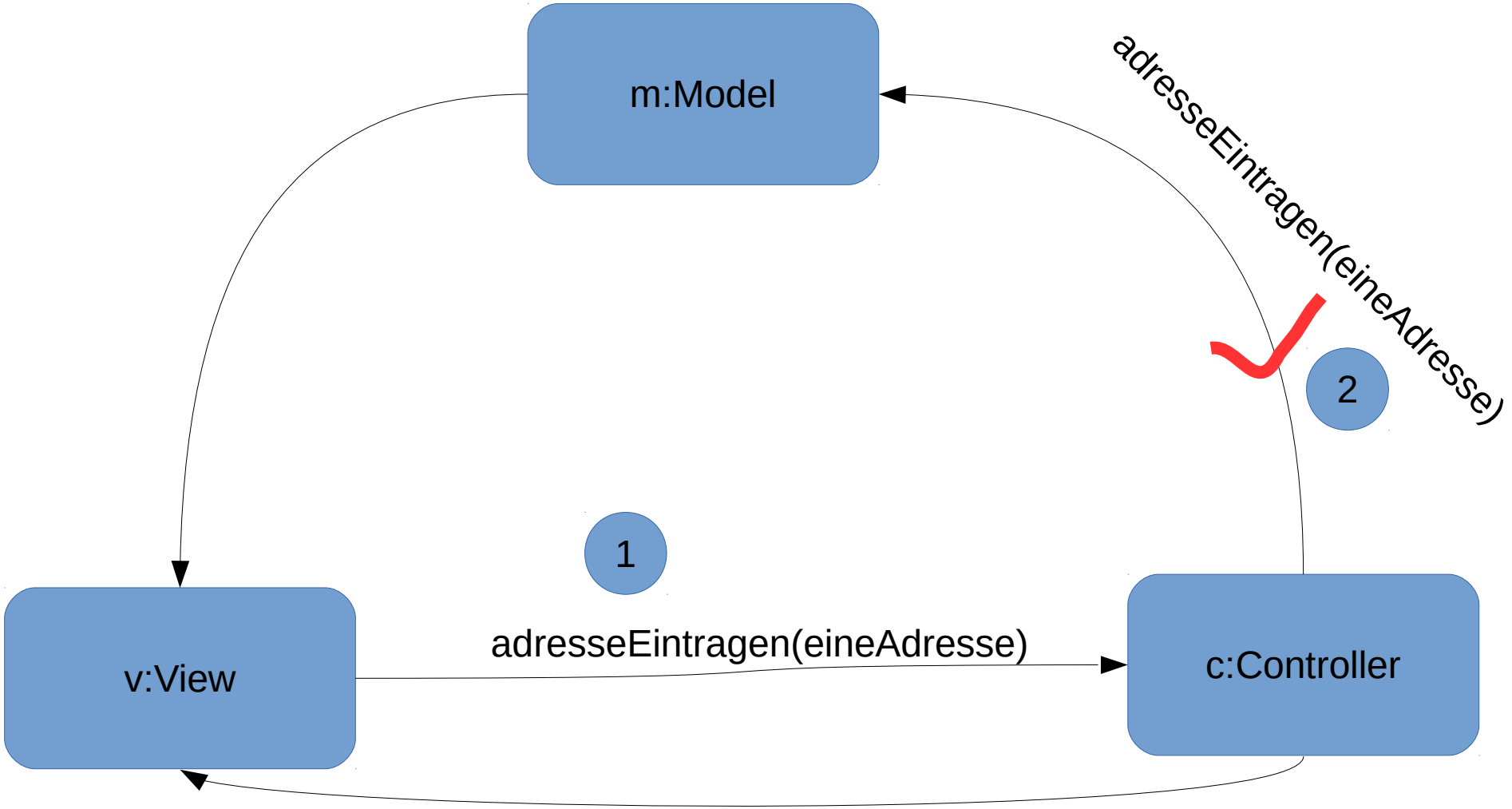

m:Model
1
v:View
adresseEintragen(eineAdresse)
c:Controller

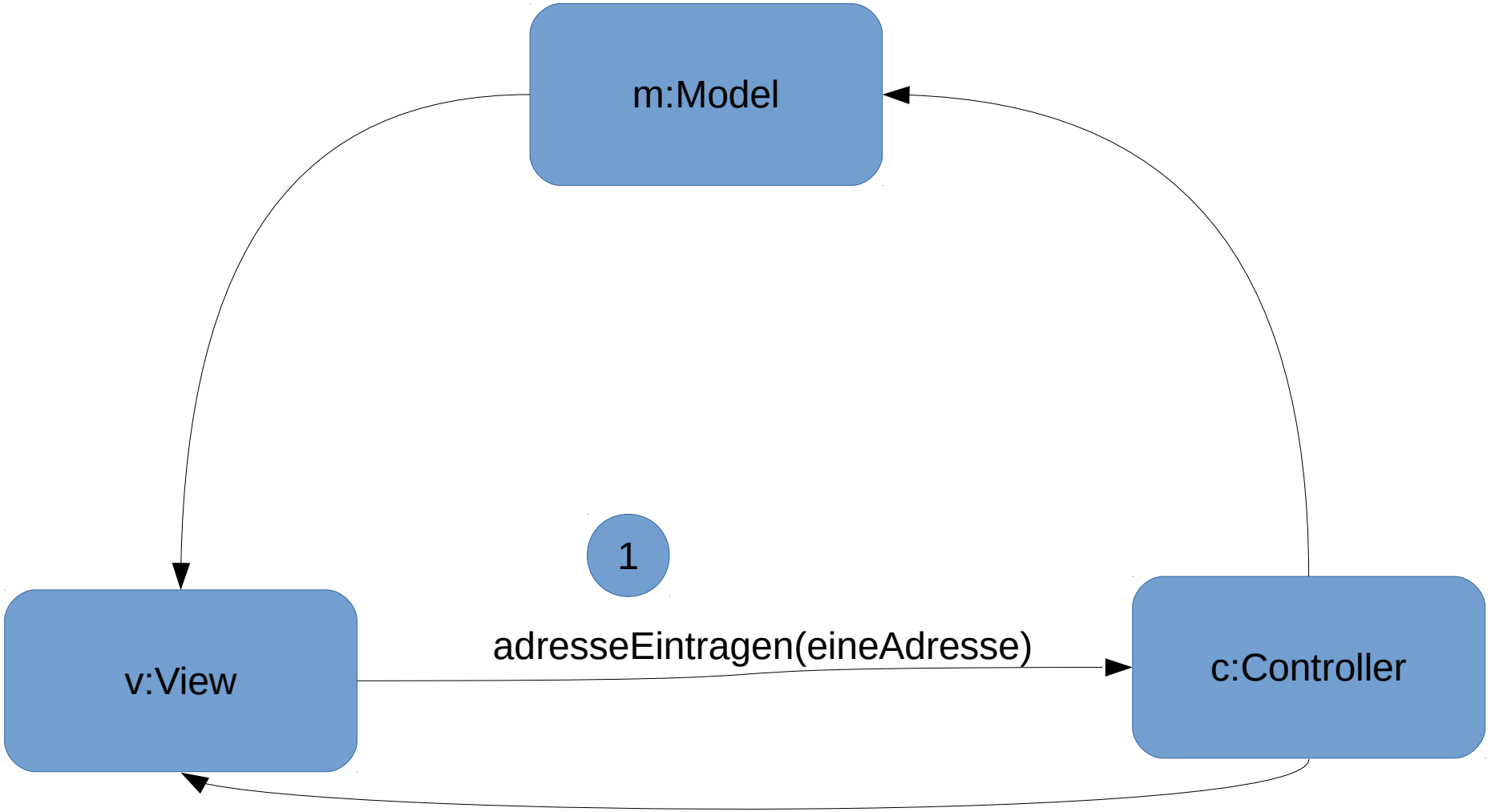

m:Model
adresseEintragen(eineAdresse)
2
1
v:View
adresseEintragen(eineAdresse)
c:Controller

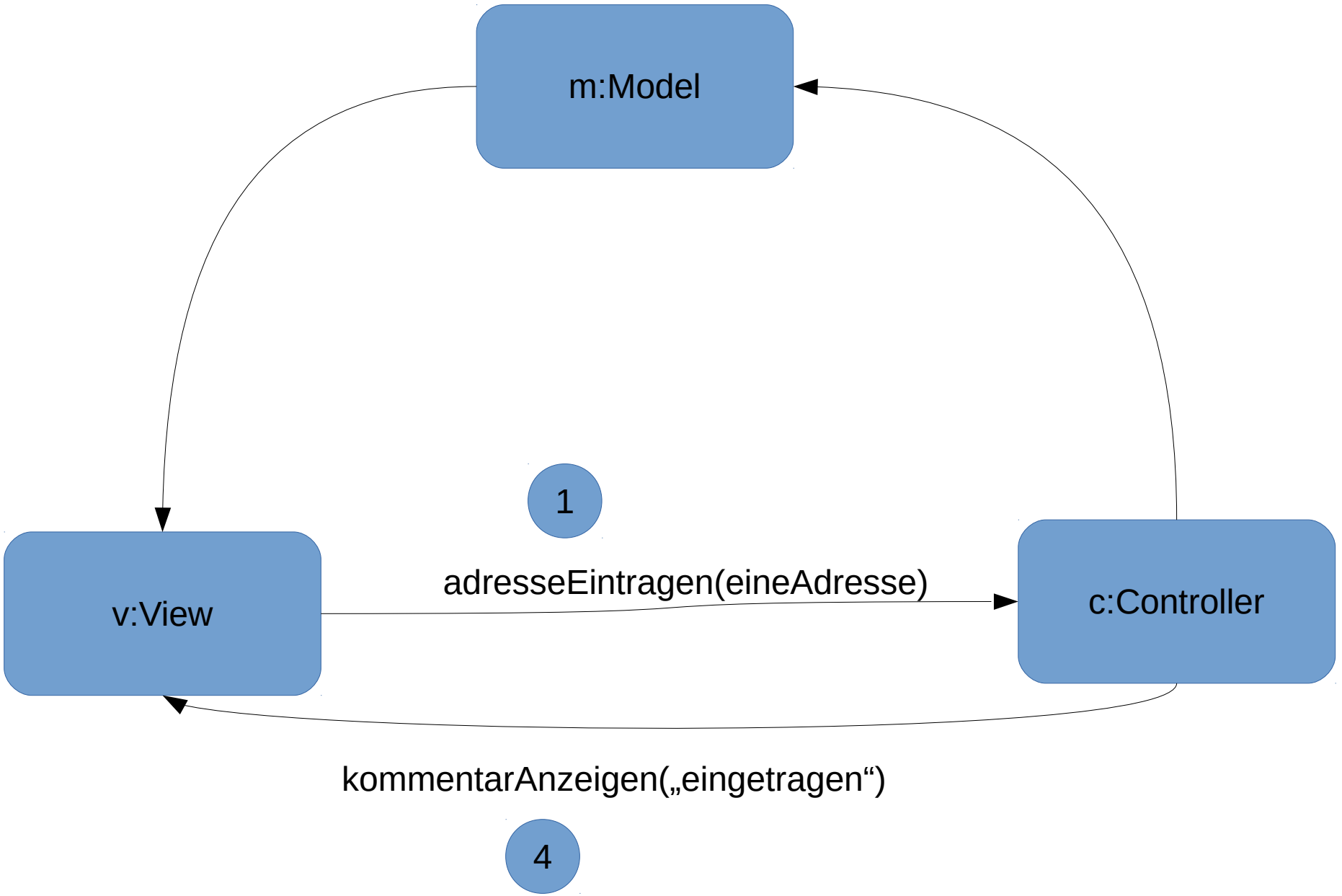

m:Model
v:View
c:Controller
1
adresseEintragen(eineAdresse)
2
adresseEintragen(eineAdresse)
3
anzahlEintraegeAnzeigen(4)

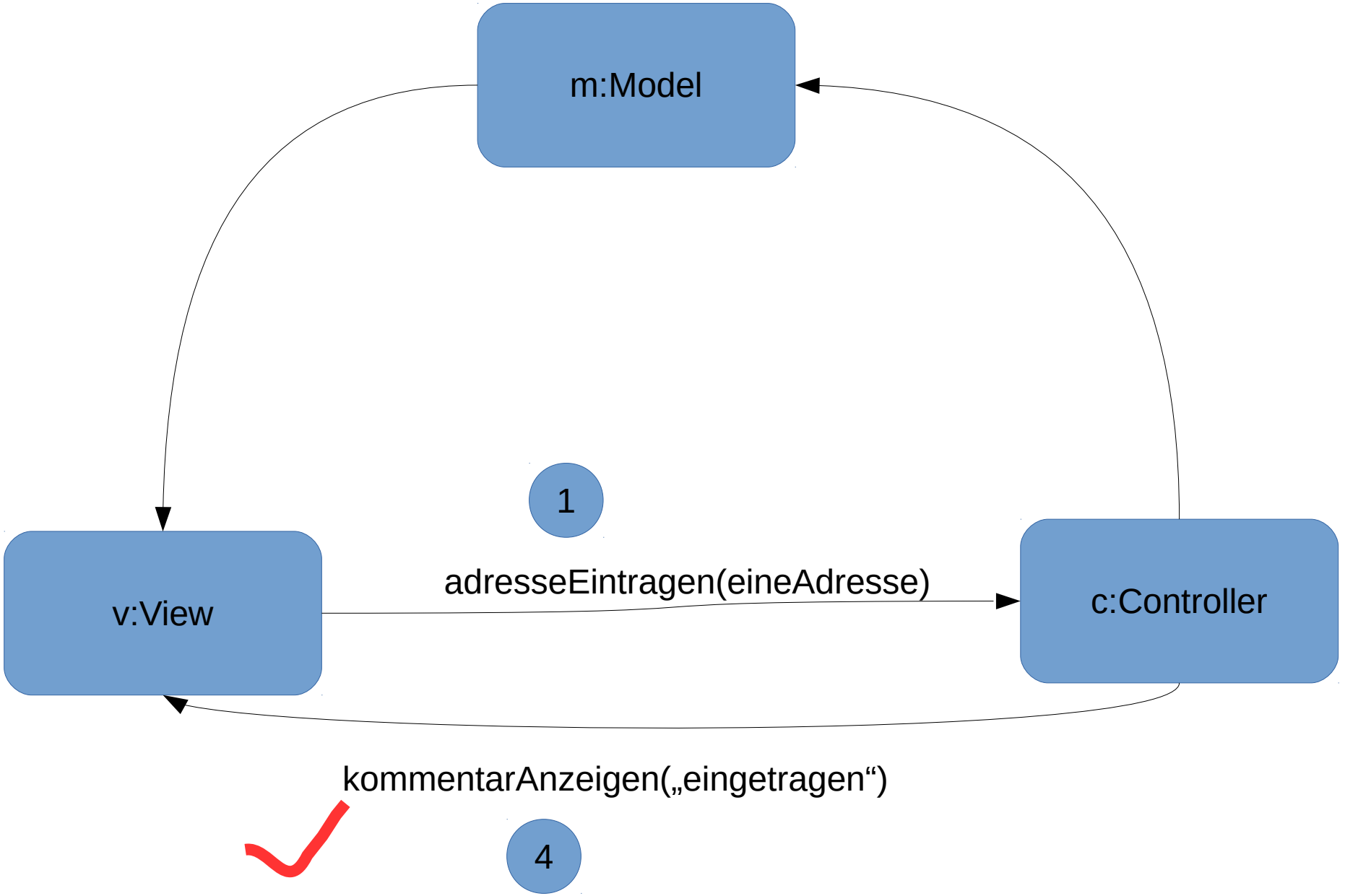

m:Model
v:View
c:Controller
1
adresseEintragen(eineAdresse)
2
adresseEintragen(eineAdresse)
3
anzahlEintraegeAnzeigen(4)

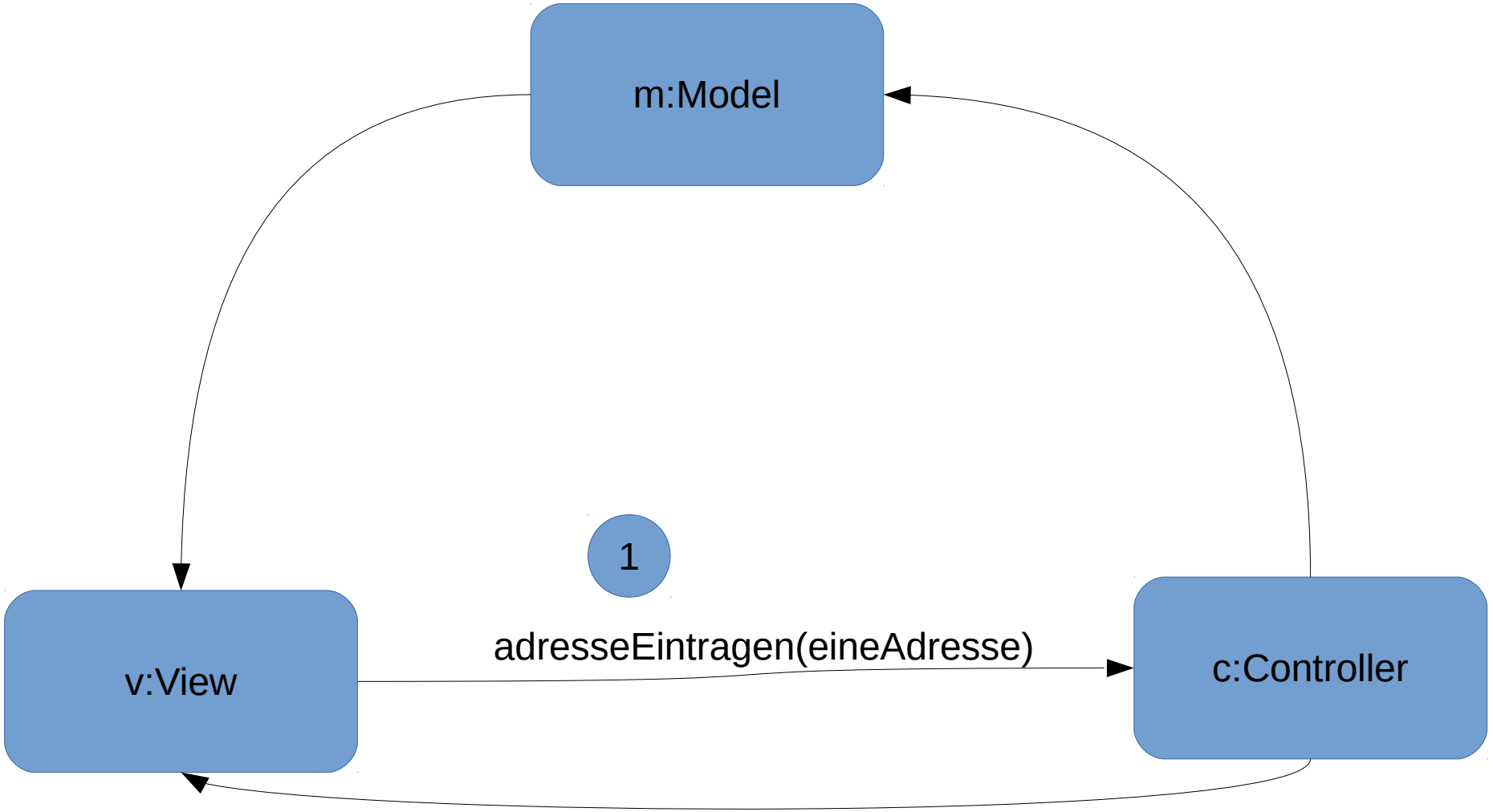

m:Model
adresseEintragen(eineAdresse)
2
1
v:View
adresseEintragen(eineAdresse)
c:Controller

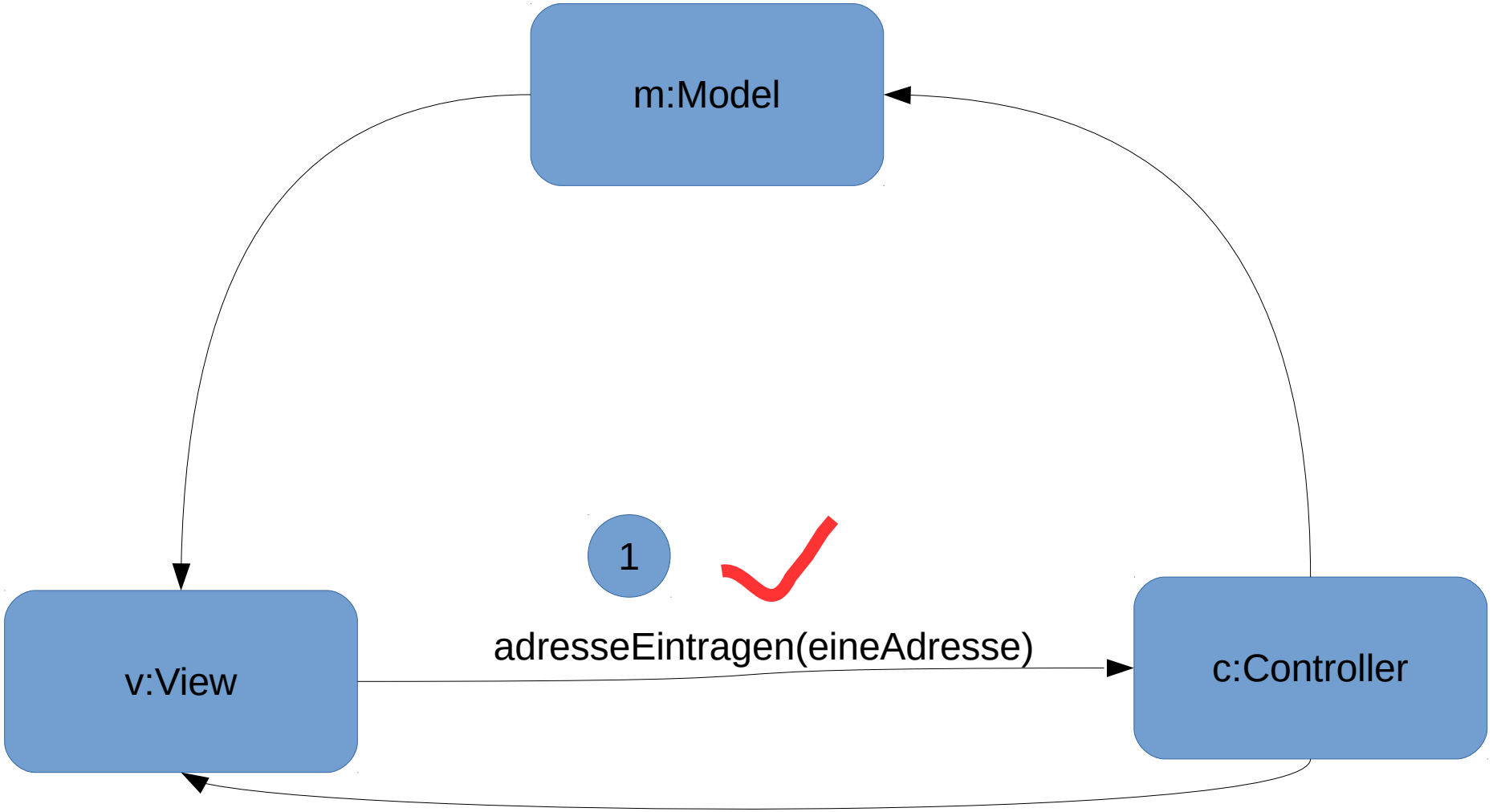

m:Model
adresseEintragen(eineAdresse)
2
1
v:View
adresseEintragen(eineAdresse)
c:Controller

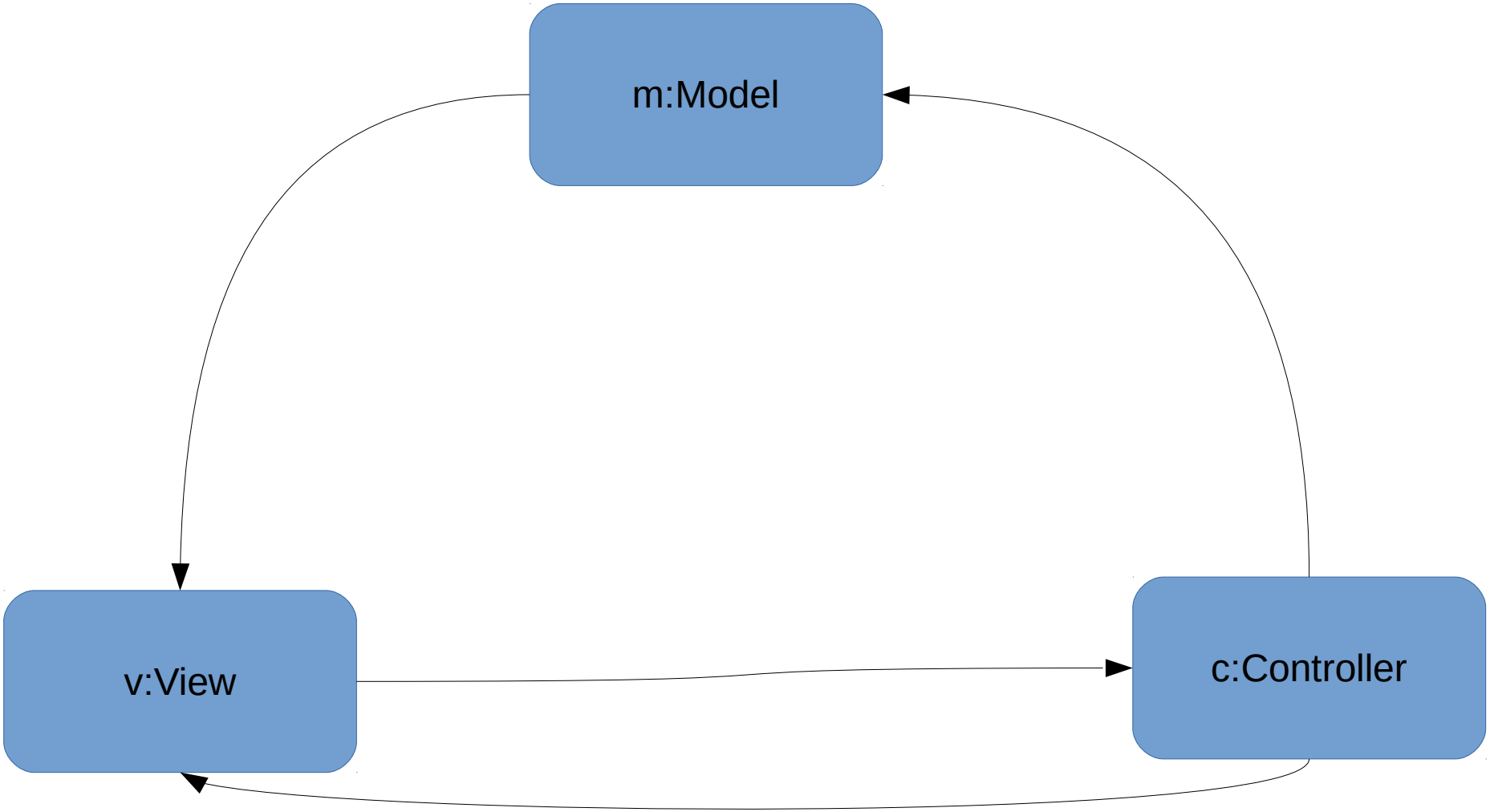

m:Model
1
v:View
adresseEintragen(eineAdresse)
c:Controller


m:Model
1
v:View
adresseEintragen(eineAdresse)
c:Controller
kommentarAnzeigen(„eingetragen“)
4


m:Model
1
v:View
adresseEintragen(eineAdresse)
c:Controller
kommentarAnzeigen(„eingetragen“)
4

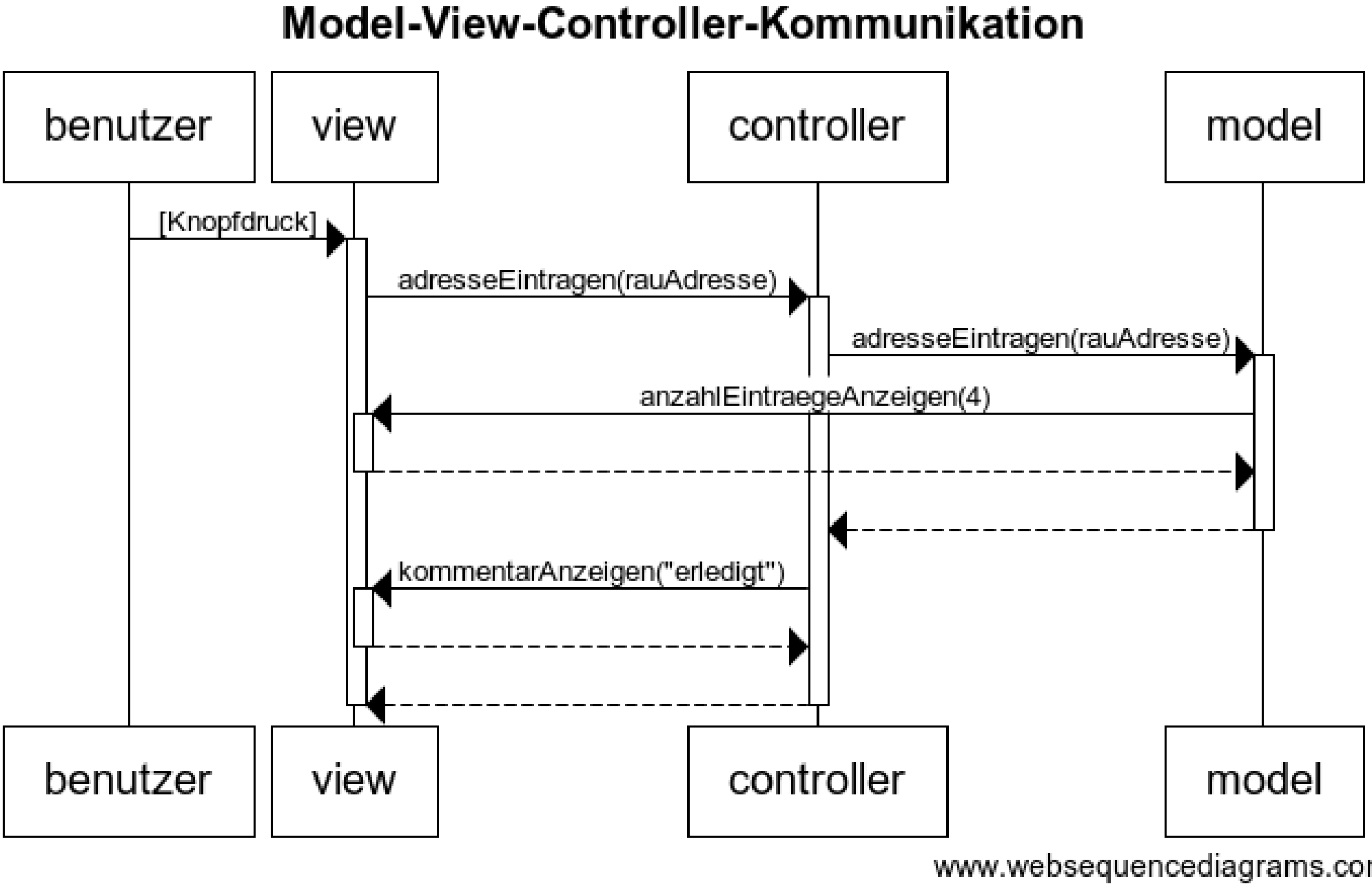

m:Model
1
v:View
adresseEintragen(eineAdresse)
c:Controller


m:Model
1
adresseEintragen(eineAdresse)
v:View
c:Controller


m:Model
v:View
c:Controller

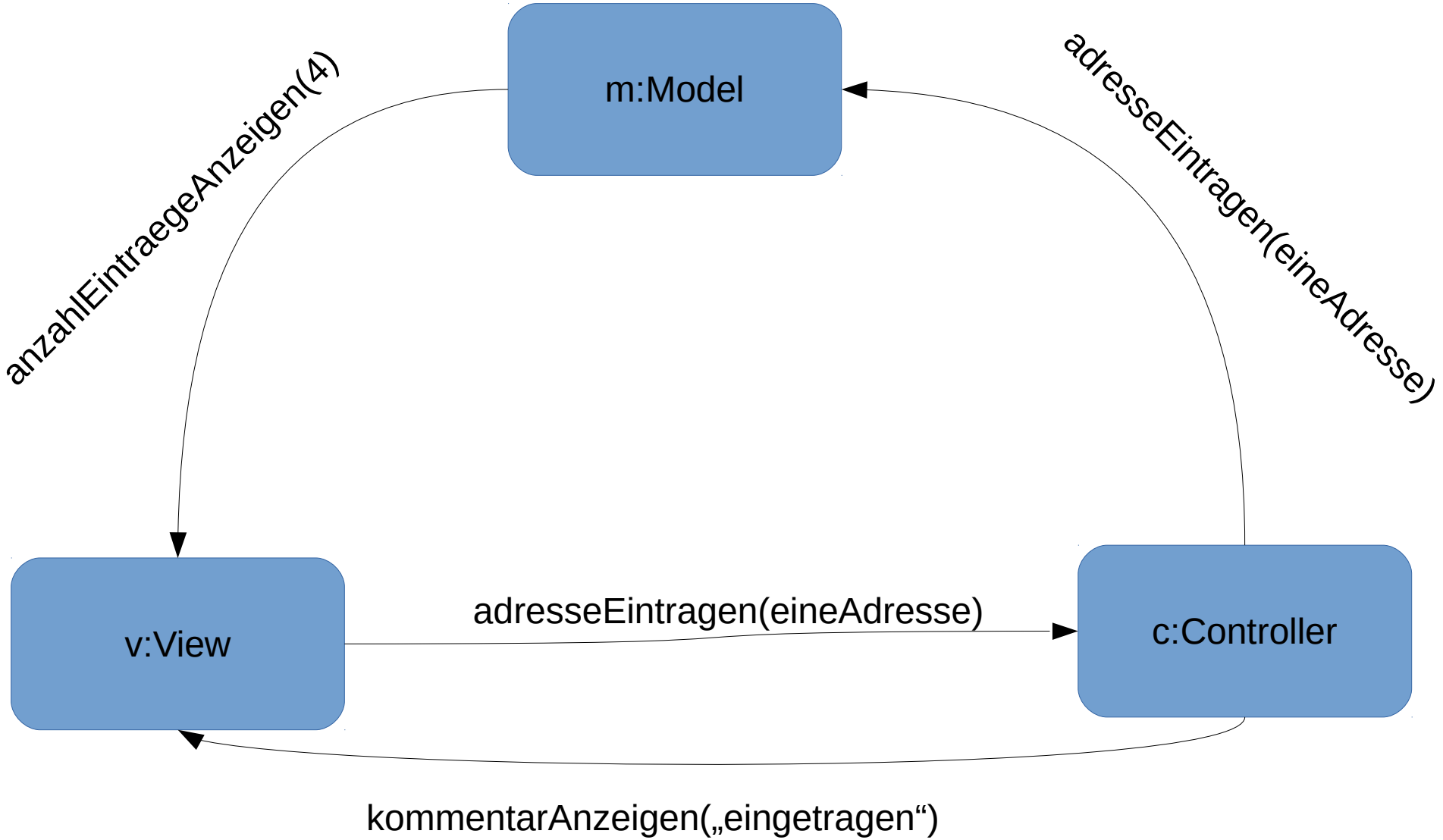
Als Sequenzdiagramm

Als Sequenzdiagramm

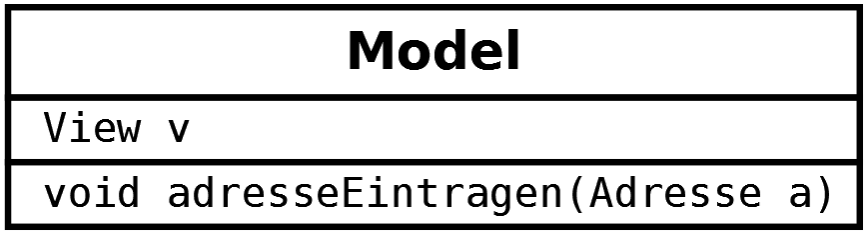


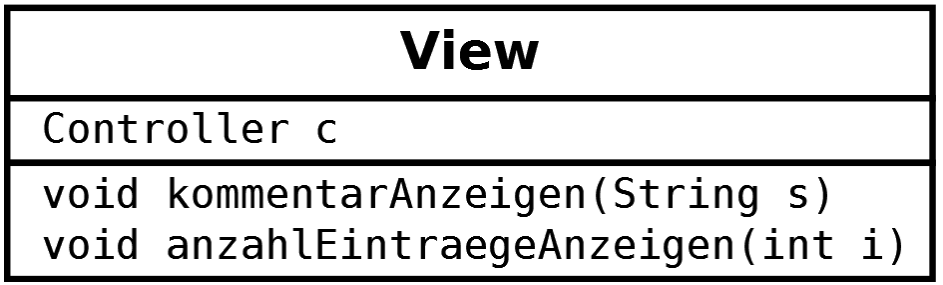
Notwendige Methoden und Attribute

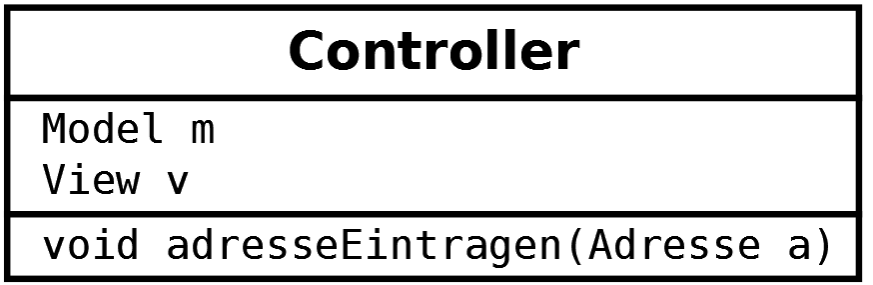
Notwendige Methoden und Attribute



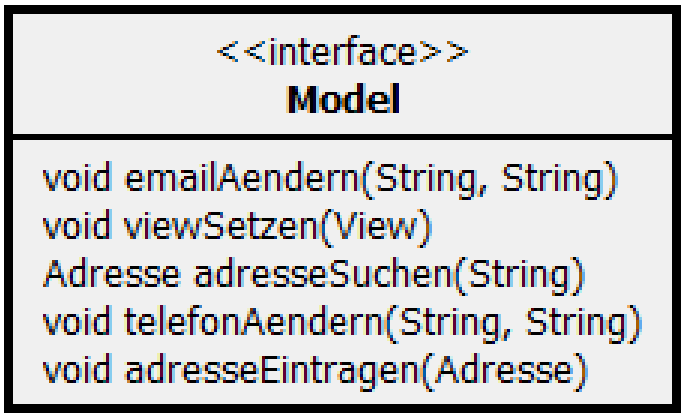
Notwendige Methoden und Attribute

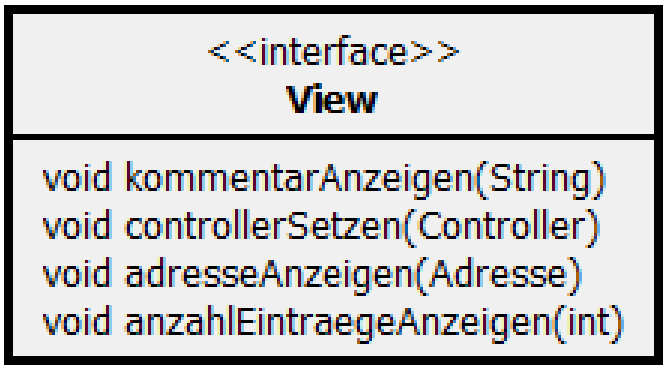


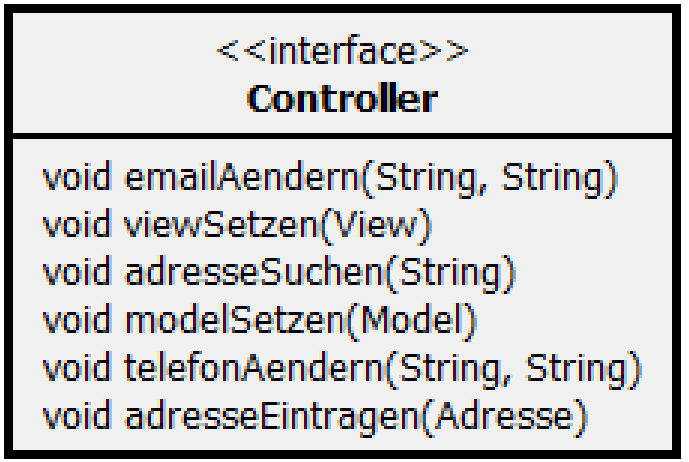




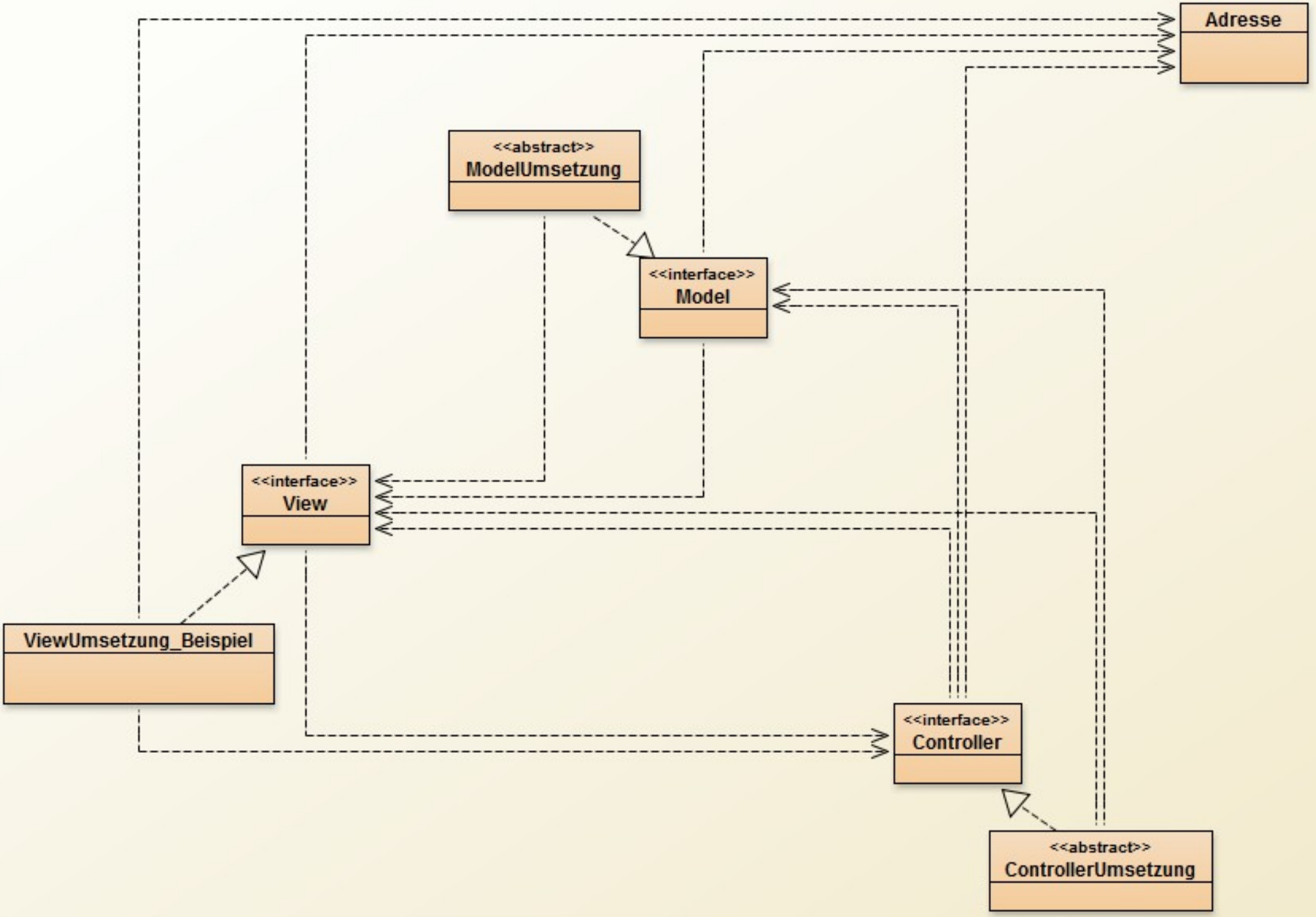
Notwendige Methoden und Attribute werden in Interfaces festgelegt

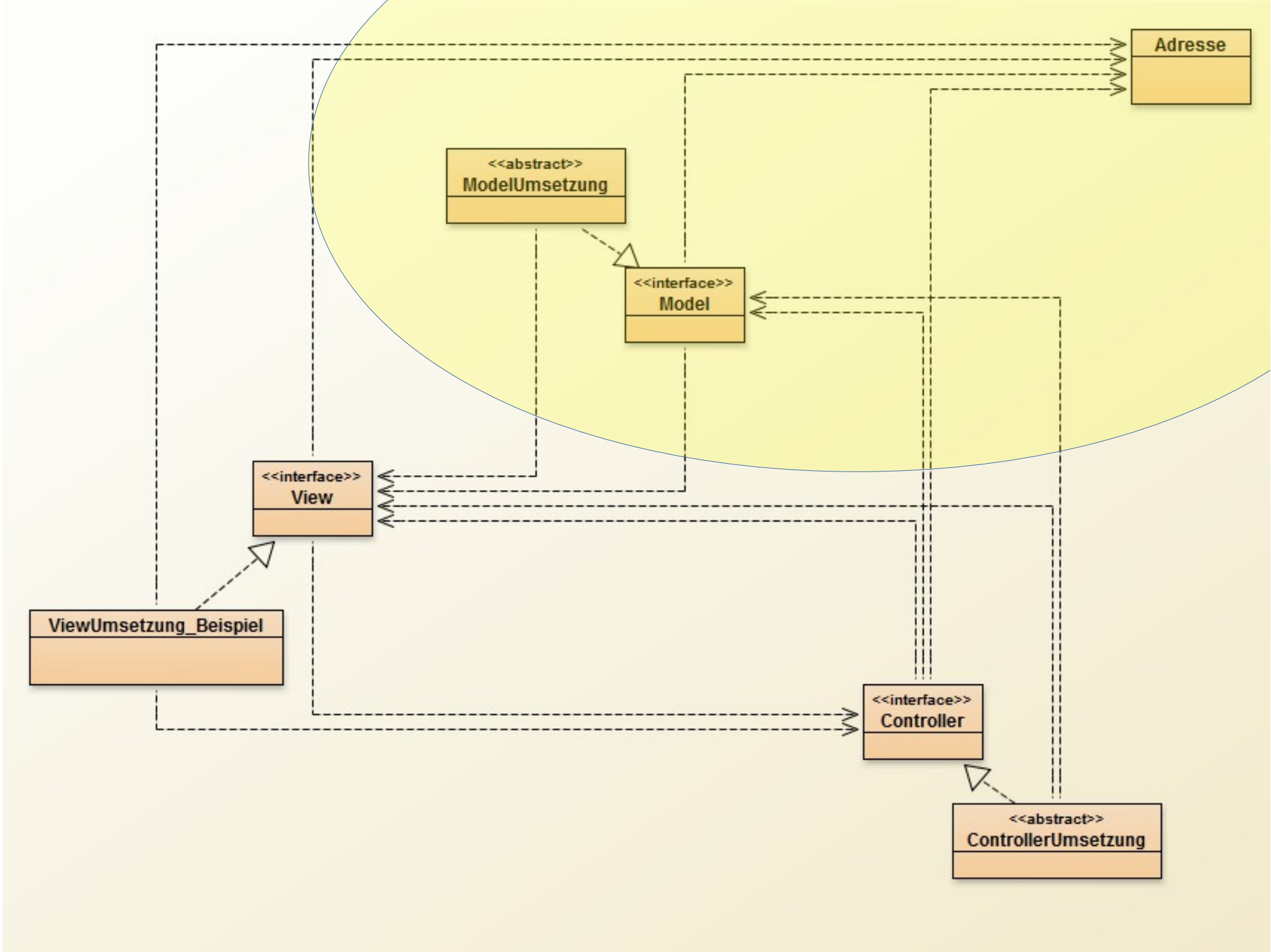


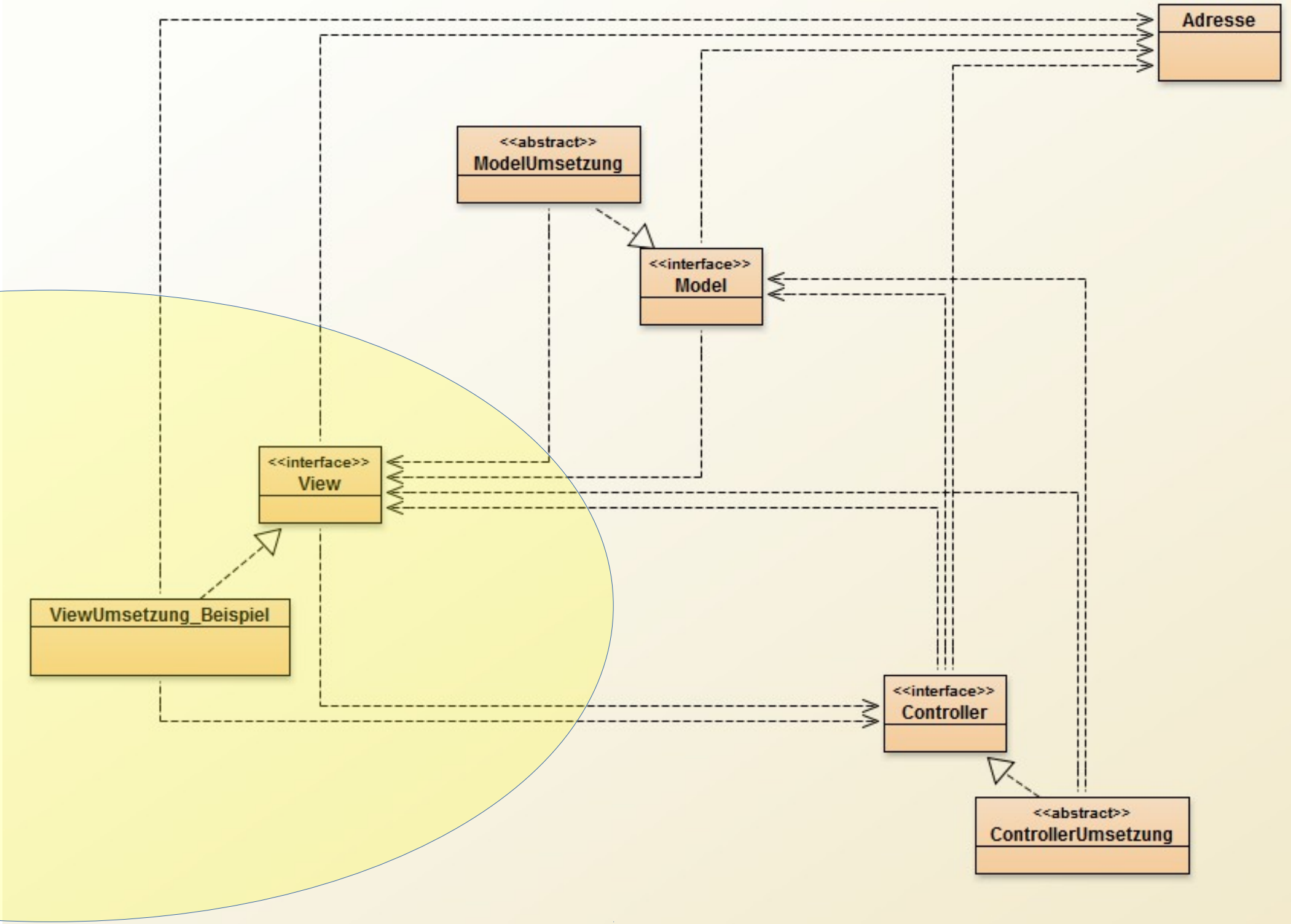


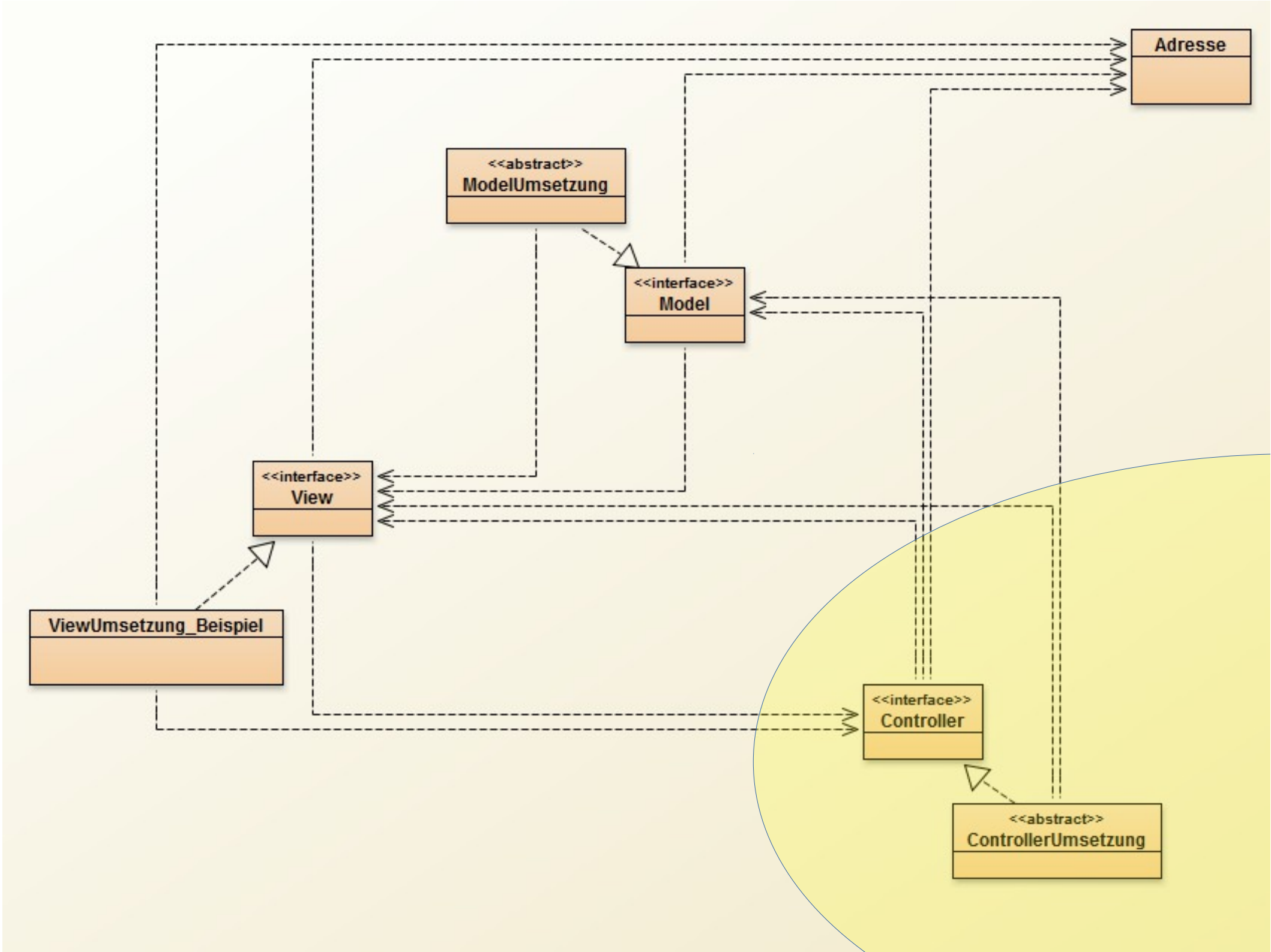


In BlueJ


Adresse
<<abstract>>
ModelUmsetzung
<<interface>>
Model
<<interface>>
View
ViewUmsetzung_Beispiel
<<interface>>
Controller
<<abstract>>
ControllerUmsetzung


Adresse
<<abstract>>
ModelUmsetzung
<<interface>>
Model
<<interface>>
View
ViewUmsetzung_Beispiel
<<interface>>
Controller
<<abstract>>
ControllerUmsetzung


Adresse
<<abstract>>
ModelUmsetzung
<<interface>>
Model
<<interface>>
View
ViewUmsetzung_Beispiel
<<interface>>
Controller
<<abstract>>
ControllerUmsetzung


Adresse
<<abstract>>
ModelUmsetzung
<<interface>>
Model
<<interface>>
View
ViewUmsetzung_Beispiel
<<interface>>
Controller
<<abstract>>
ControllerUmsetzung