

# Ein Adressbuch nach dem Model-View-Controller-Prinzip

Warum einfach, wenn es auch umständlich geht?

# Der View als grafische Benutzeroberfläche (GUI)



v:View

# Der View als grafische Benutzeroberfläche (GUI)

v:View

The screenshot shows a window titled "Adressbuch" with standard Windows window controls (minimize, maximize, close) in the top right corner. The window contains a form with the following elements:

- Nachname:** A text input field containing "Rau" and a "suchen" button to its right.
- Vorname:** A text input field containing "Thomas" and an "eintragen" button to its right.
- E-Mail:** A text input field containing "schule@herr-rau.de" and an "aktualisieren" button to its right.
- Telefon:** An empty text input field with a "Rückmeldung" button to its right.

At the bottom of the window, there is a status message: "Es gibt gerade 4 Einträge in der Adressverwaltung."

# Der View als grafische Benutzeroberfläche (GUI)

v:View

The screenshot shows a Java Swing window titled "Adressbuch" with standard window controls (minimize, maximize, close). The window contains the following components:

- Nachname:** A text field containing "Rau".
- Vorname:** A text field containing "Thomas".
- E-Mail:** A text field containing "schule@herr-rau.de".
- Telefon:** An empty text field.
- Buttons:** Three buttons labeled "suchen", "eintragen", and "aktualisieren" are arranged vertically on the right side.
- Status:** A text label at the bottom stating "Es gibt gerade 4 Einträge in der Adressverwaltung."

Red arrows and text labels identify the Swing classes used for these components:

- JTextField:** Points to the "Nachname" and "Vorname" text fields.
- JButtons:** Points to the "suchen", "eintragen", and "aktualisieren" buttons.
- JLabel:** Points to the "Telefon" text field and the status text "Es gibt gerade 4 Einträge in der Adressverwaltung."

# 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

The screenshot shows a window titled "Adressbuch" with standard window controls (minimize, maximize, close) in the top right corner. The form contains the following fields and buttons:

- Nachname:** A text input field containing "Rau" and a "suchen" button to its right.
- Vorname:** A text input field containing "Thomas" and an "eintragen" button to its right.
- E-Mail:** A text input field containing "schule@herr-rau.de" and an "aktualisieren" button to its right.
- Telefon:** An empty text input field.
- Rückmeldung:** A label positioned to the right of the "Telefon" field.

At the bottom of the window, a status message reads: "Es gibt gerade 4 Einträge in der Adressverwaltung."

# Der View als grafische Benutzeroberfläche (GUI)

v:View

The screenshot shows a window titled "Adressbuch" with standard Windows window controls (minimize, maximize, close) in the top right corner. The window contains a form with the following elements:

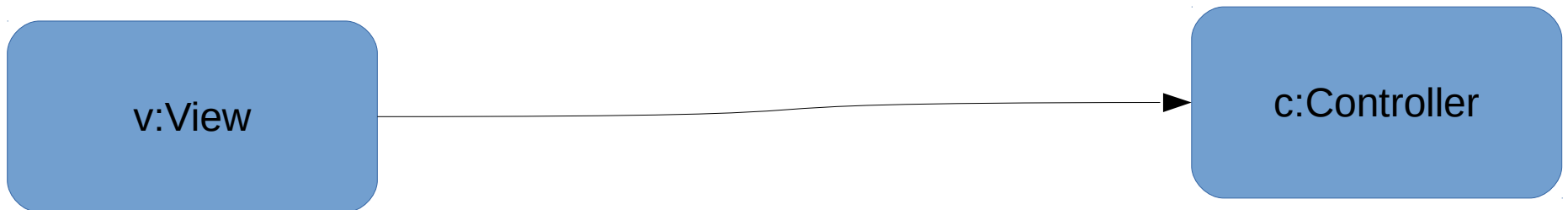
- Nachname:** A text input field containing "Rau" and a "suchen" button to its right.
- Vorname:** A text input field containing "Thomas" and an "eintragen" button to its right.
- E-Mail:** A text input field containing "schule@herr-rau.de" and an "aktualisieren" button to its right.
- Telefon:** An empty text input field with a "Rückmeldung" label to its right.

At the bottom of the window, there is a status message: "Es gibt gerade 4 Einträge in der Adressverwaltung."



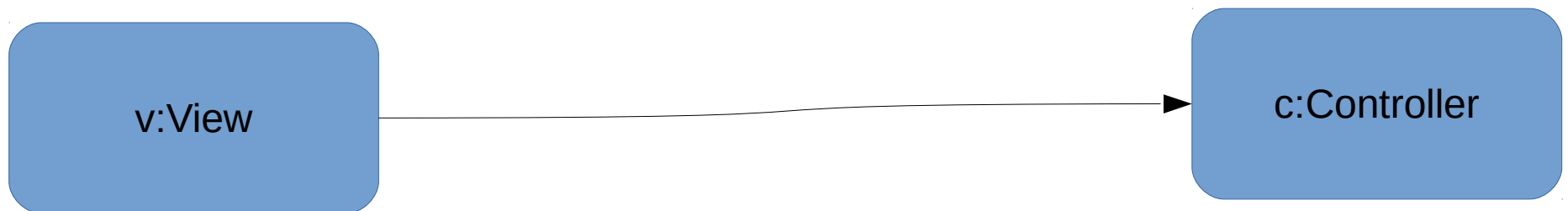
v:View

# Verbindung zum Controller (Referenzattribut)

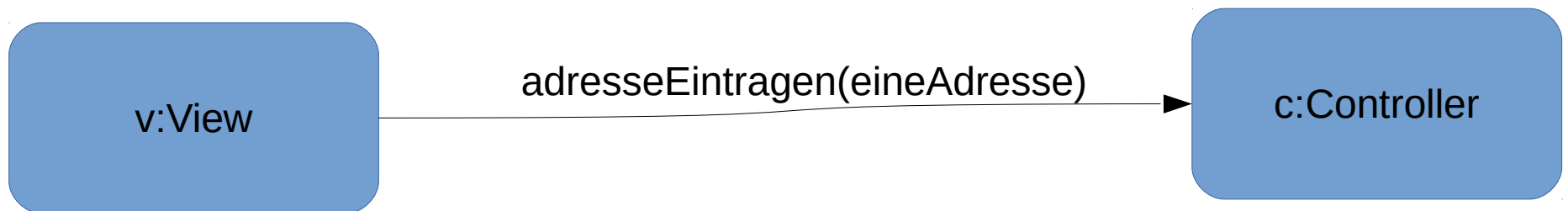




# Auftrag (Methodenaufruf) an den Controller

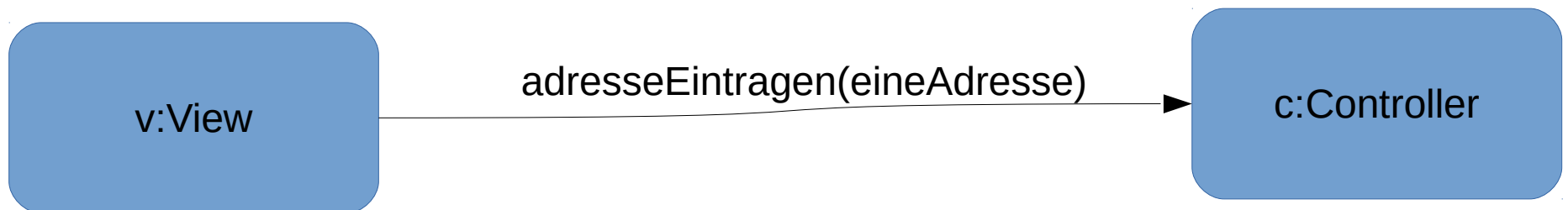


# Auftrag (Methodenaufruf) an den Controller

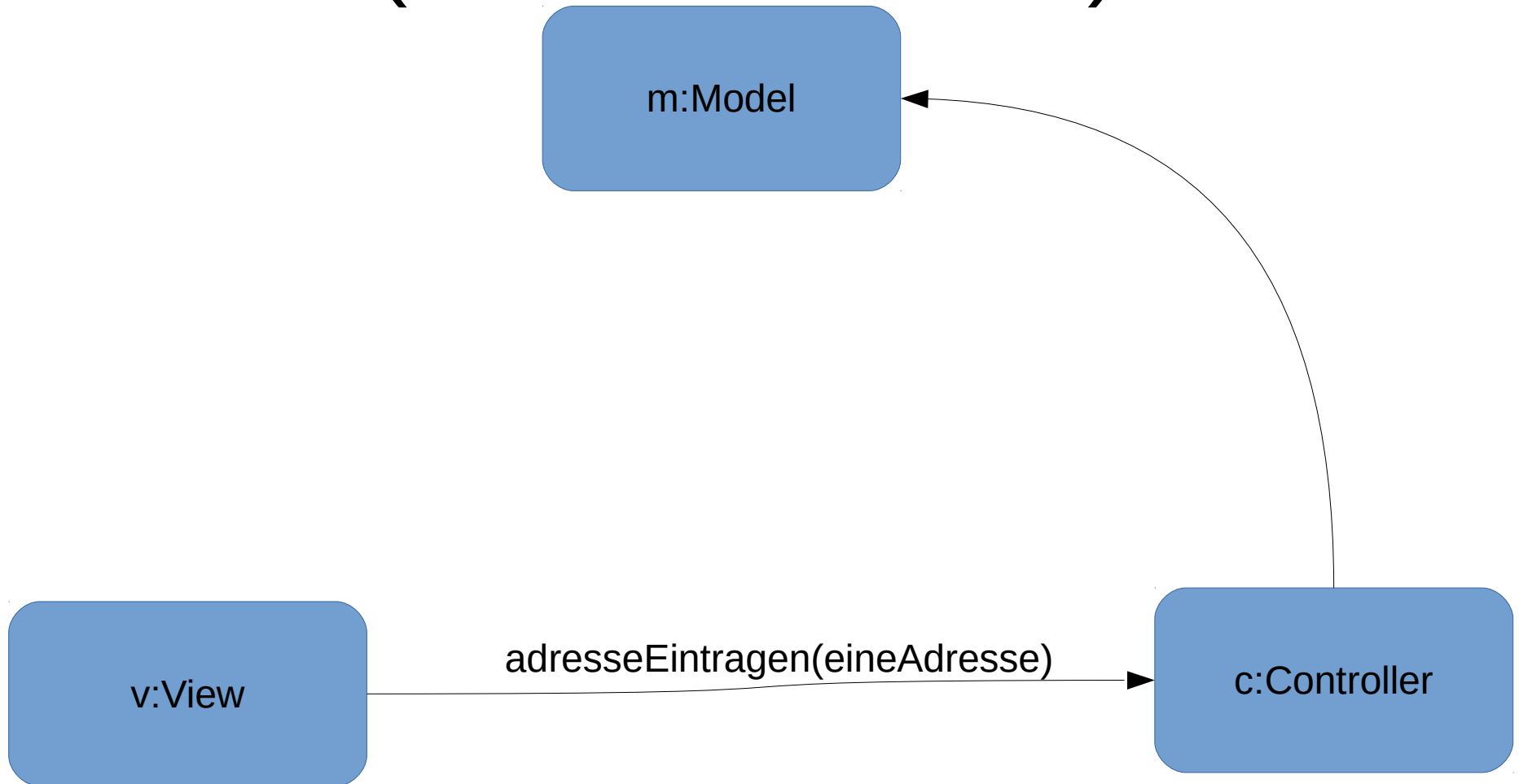


Der Controller überprüft zum Beispiel, ob die eingegebene Adresse schon vorhanden ist und ob die E-Mail eine korrekte Form hat.

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



# Verbindung zum Model (Referenzattribut)



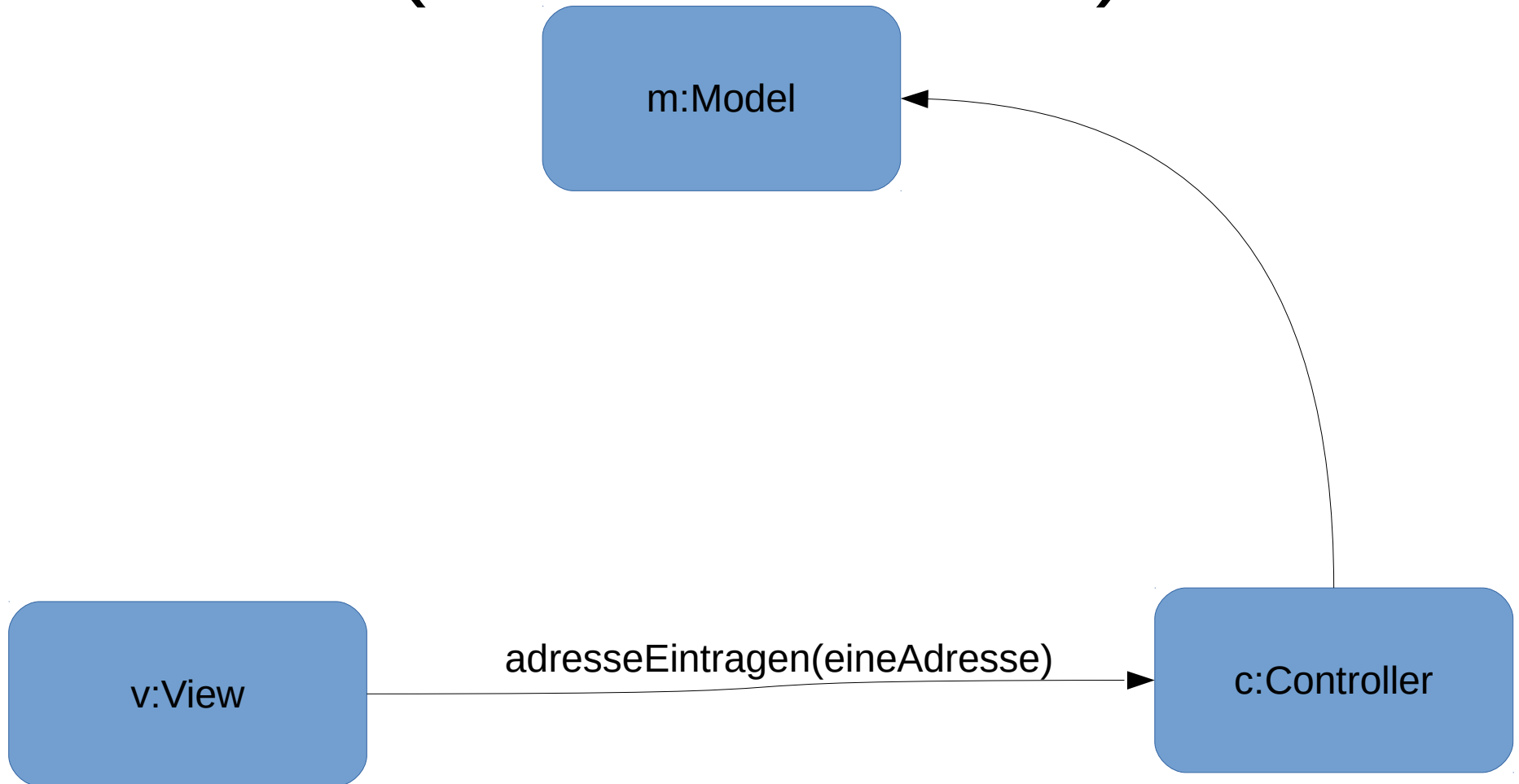
# Das Model: Hauptteil des Projekts



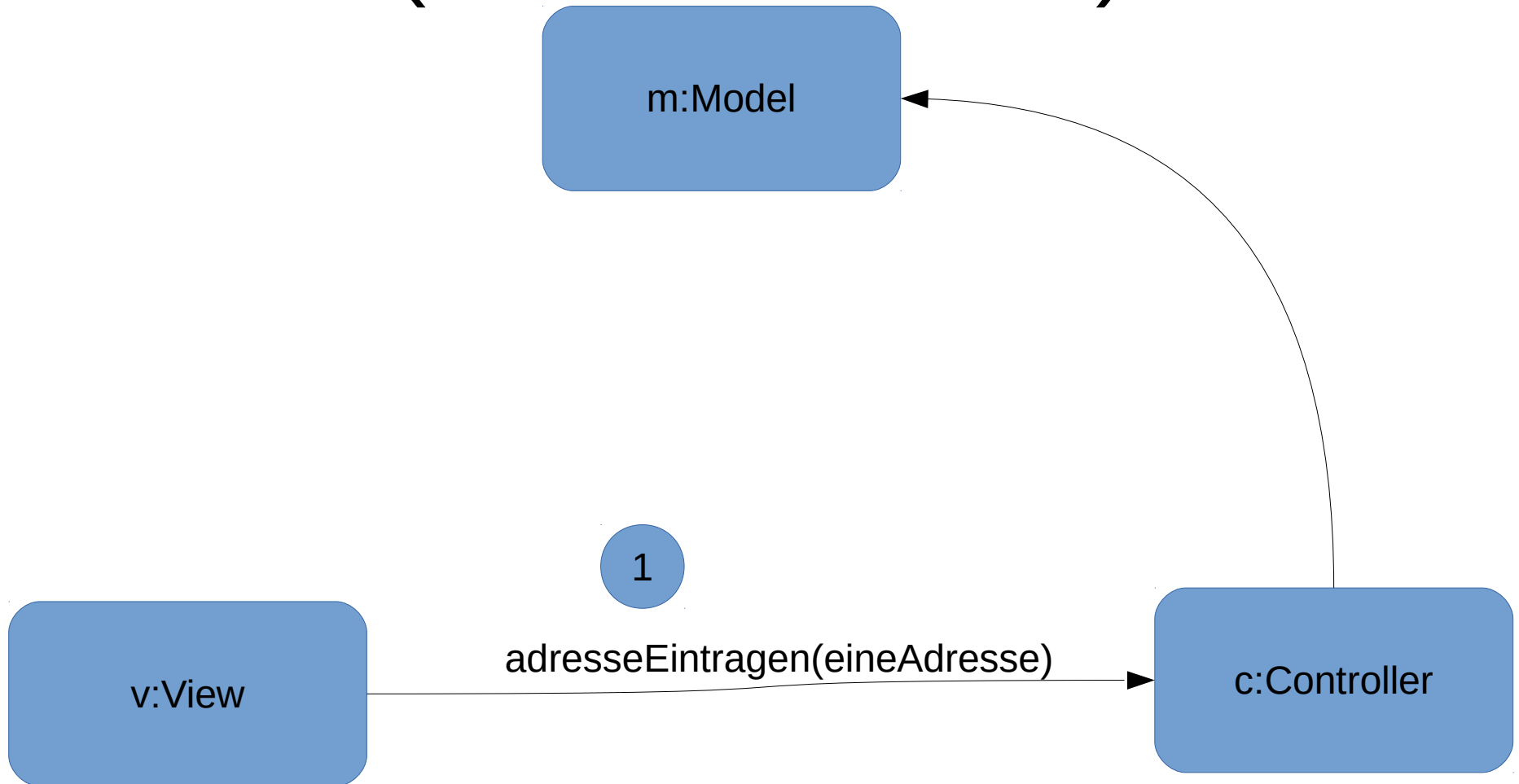
m:Model

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.

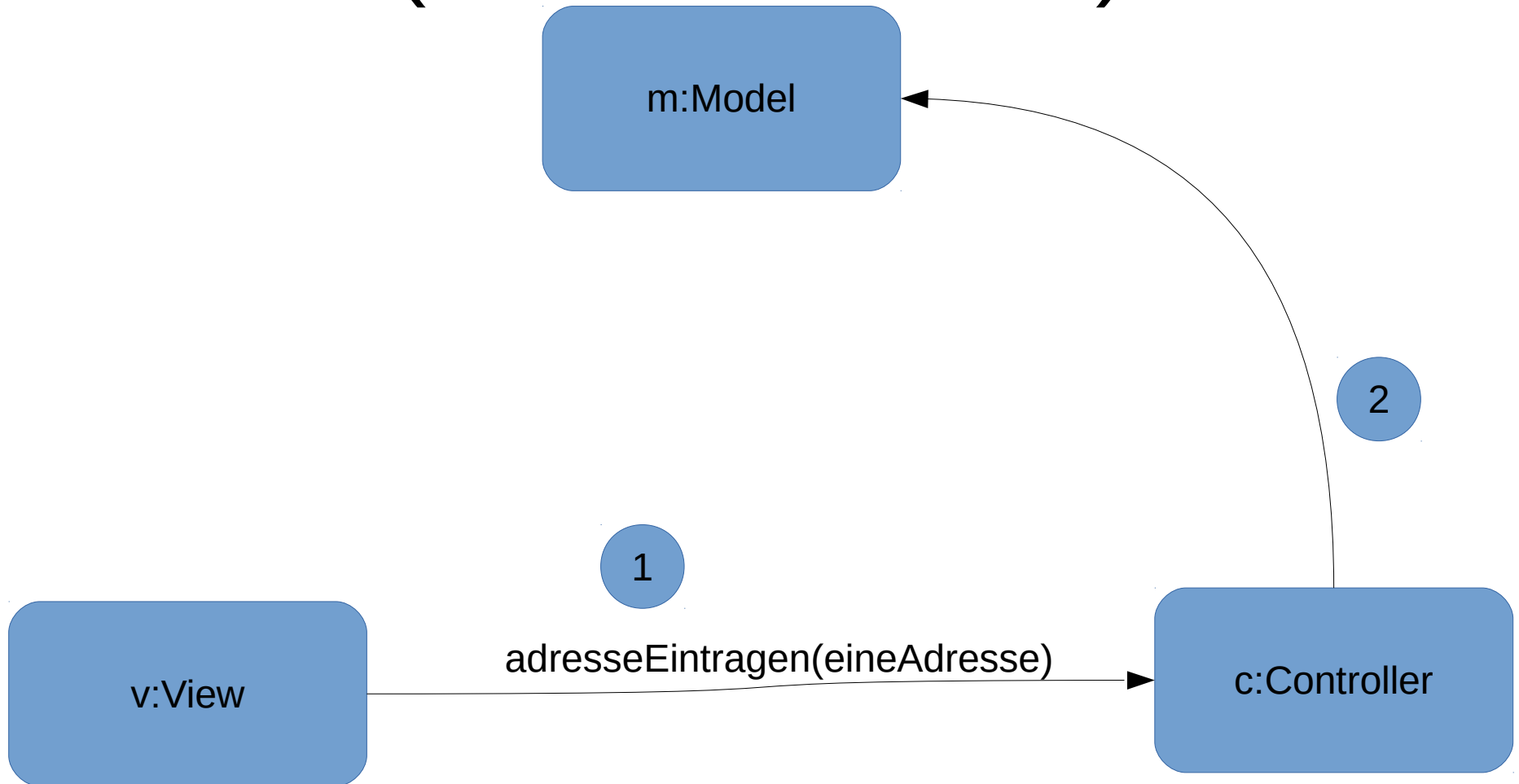
# Weitergabe des Auftrags an Model (Methodenaufruf)



# Weitergabe des Auftrags an Model (Methodenaufruf)

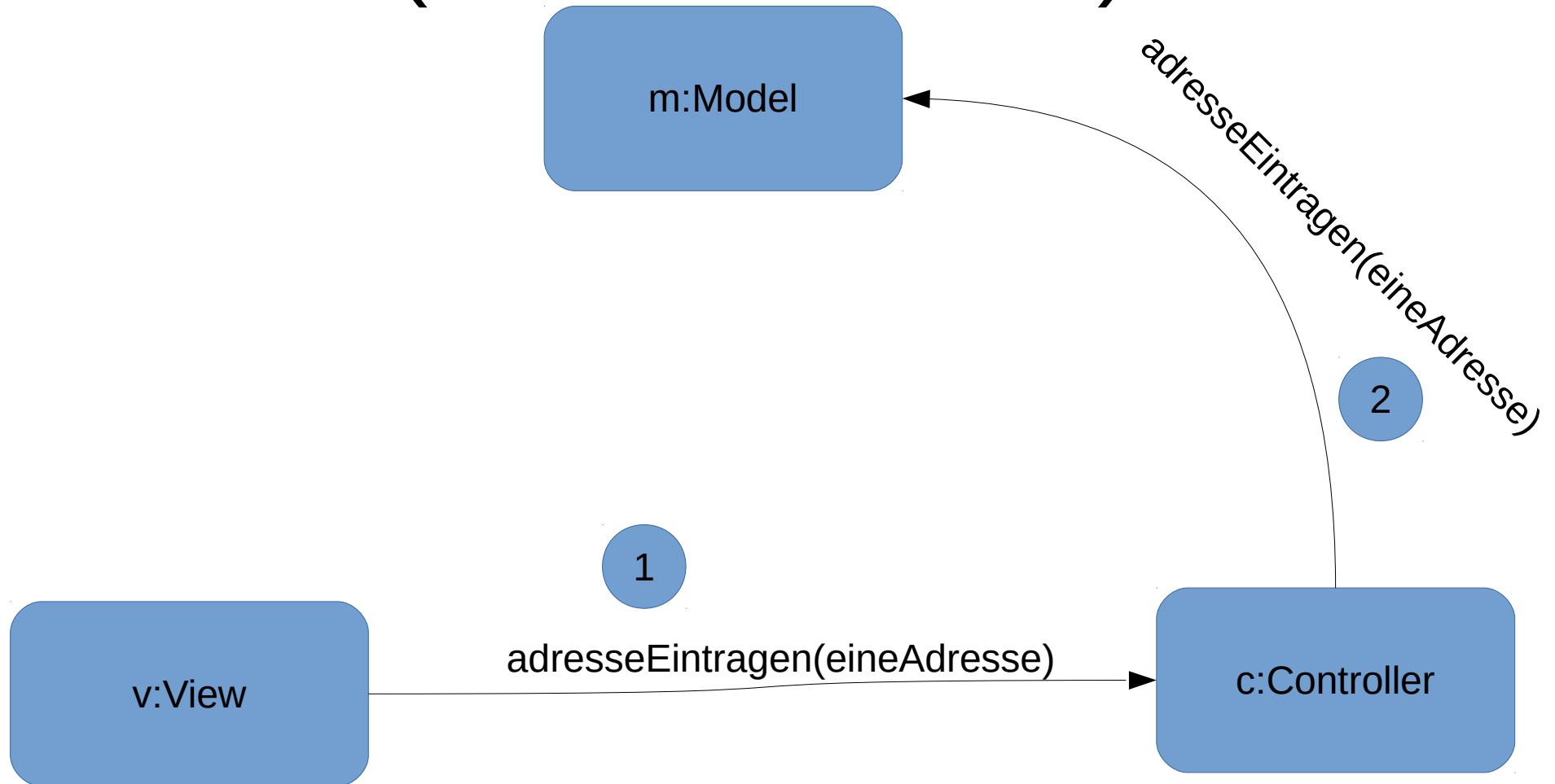


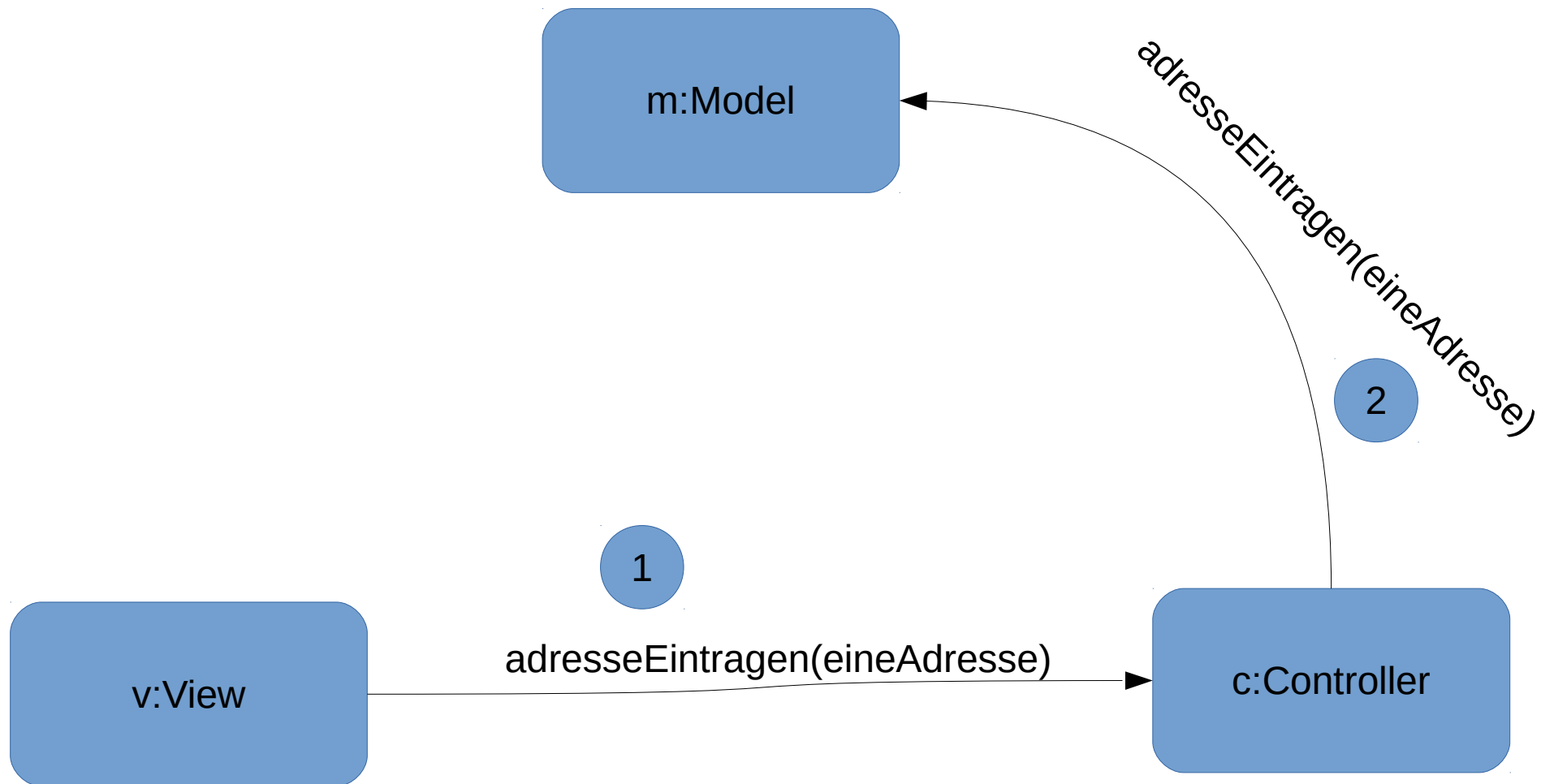
# Weitergabe des Auftrags an Model (Methodenaufruf)



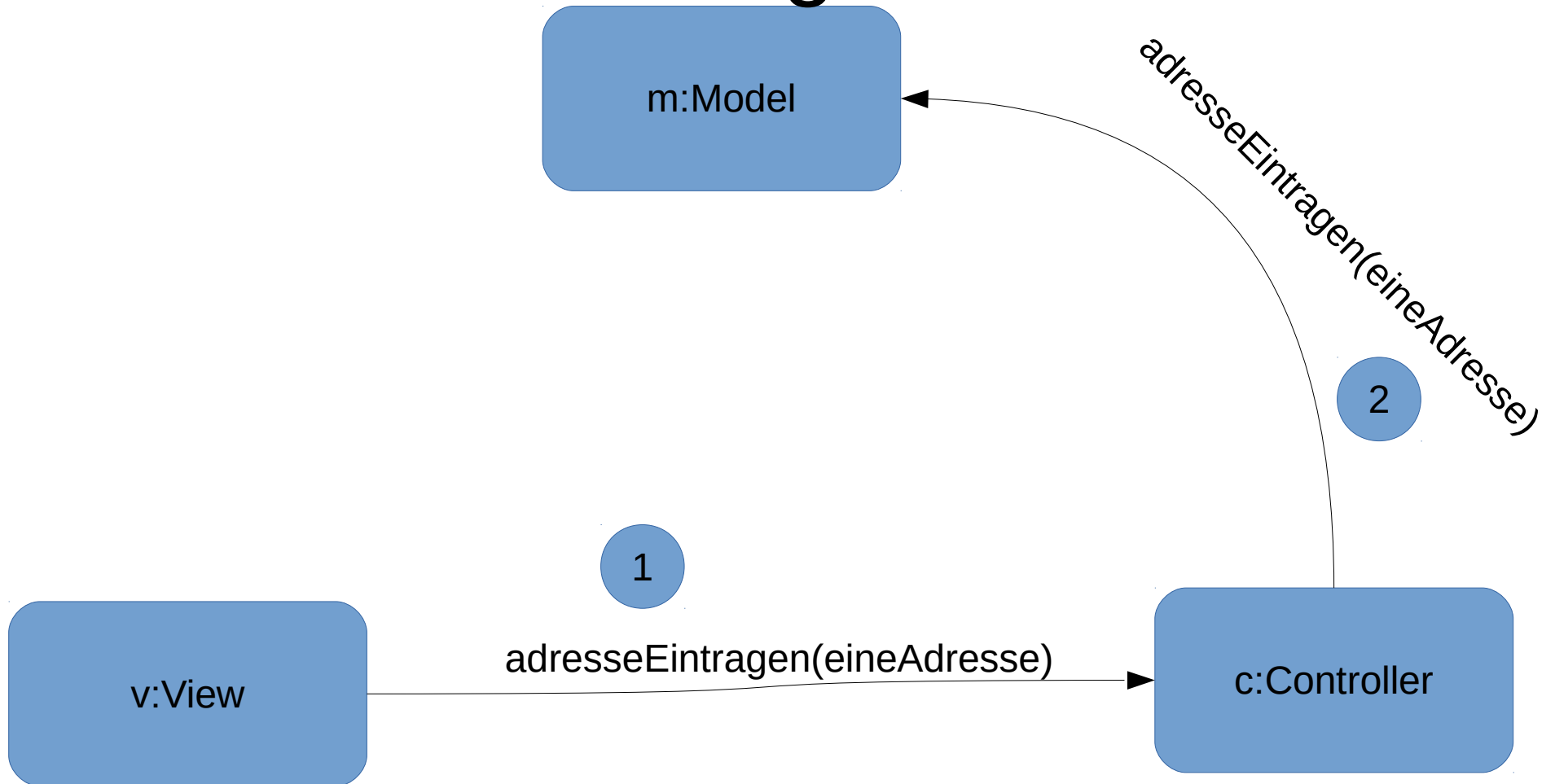


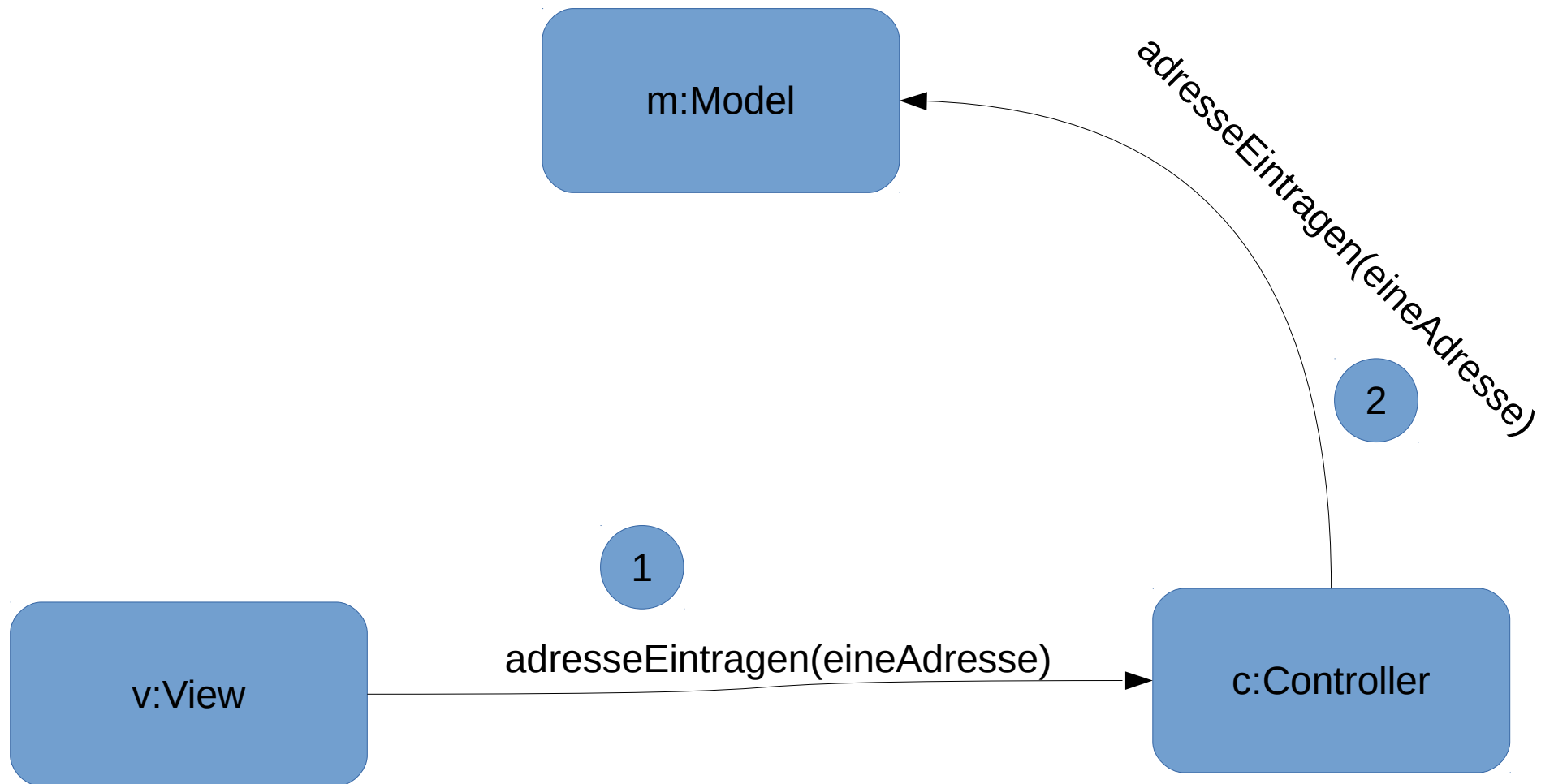
# Weitergabe des Auftrags an Model (Methodenaufruf)



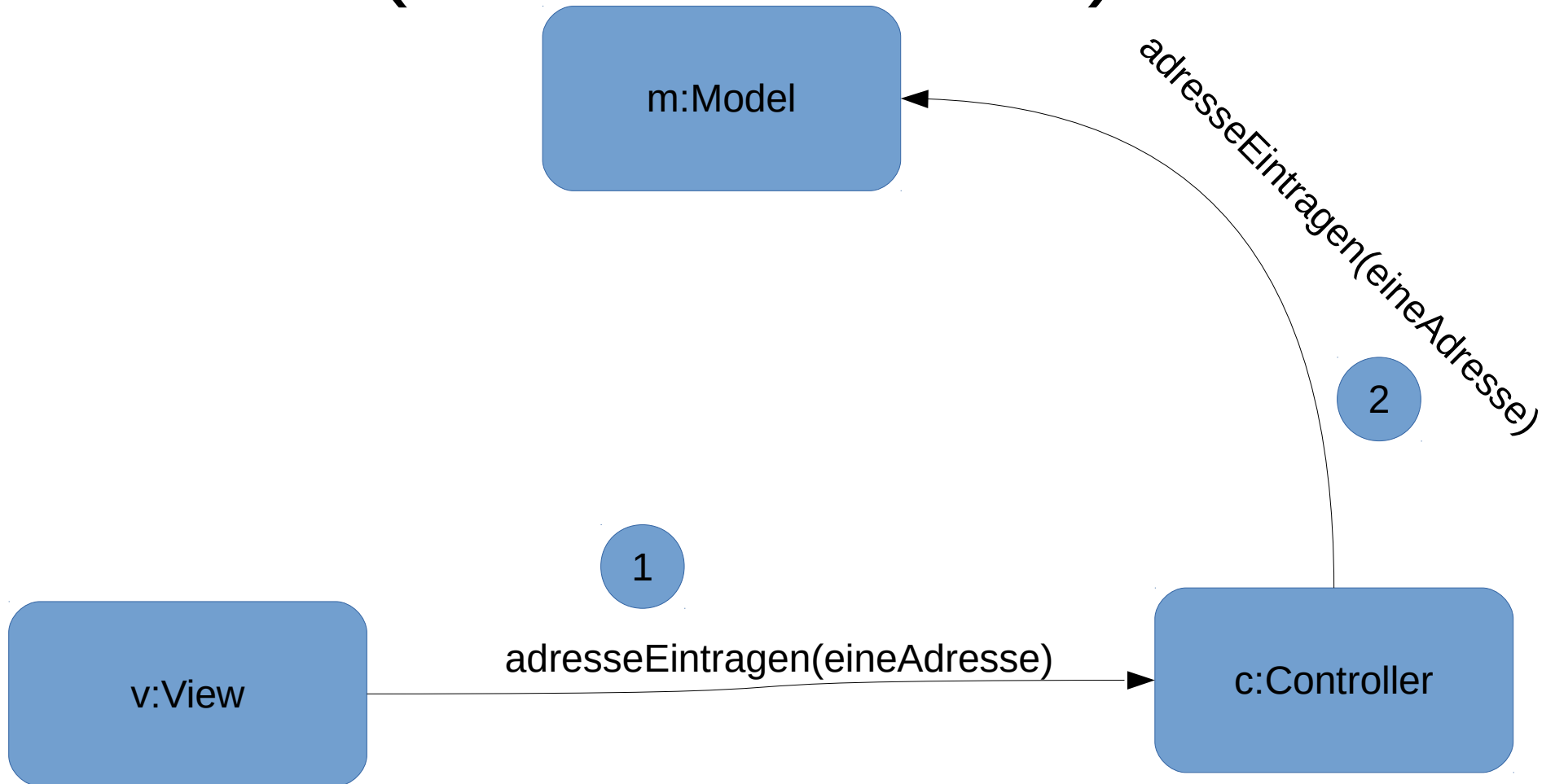


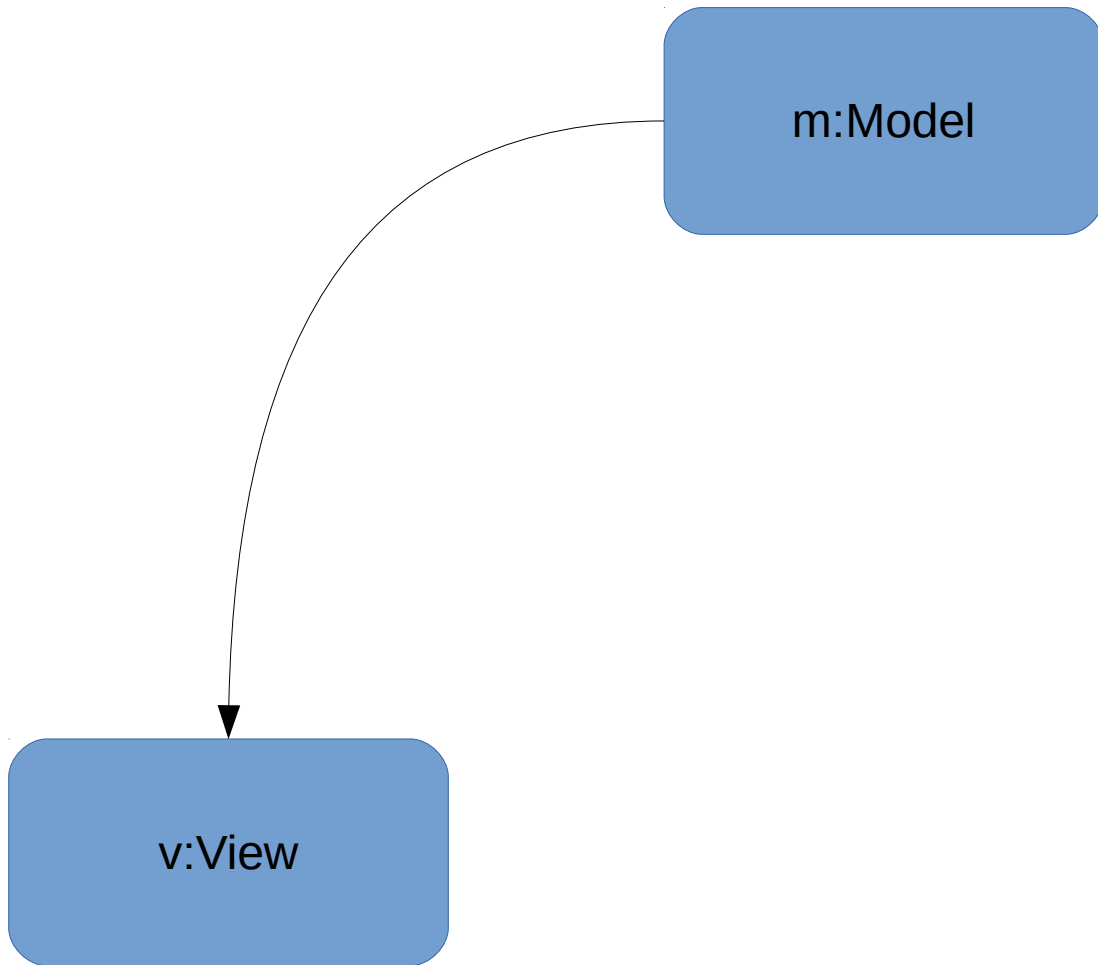
# Model nimmt die entsprechenden Änderungen vor

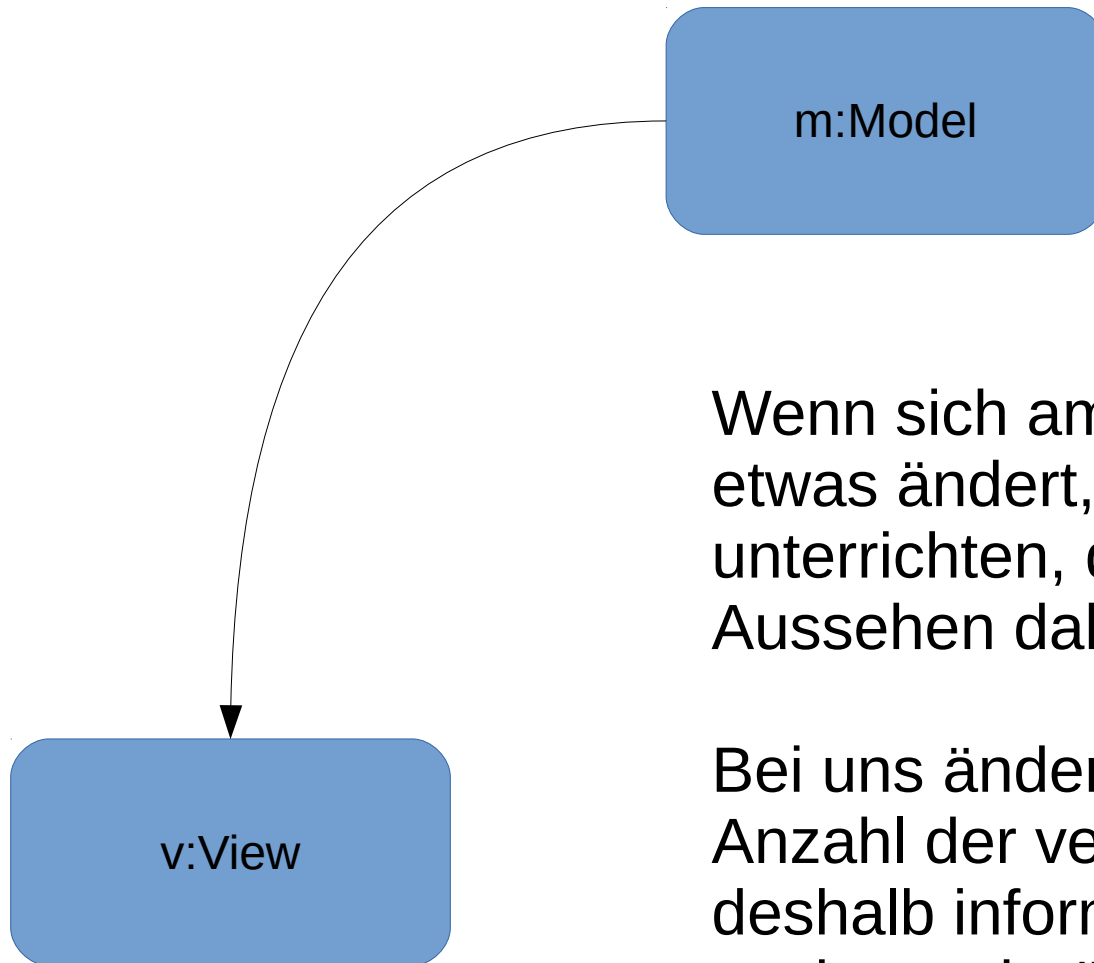




# Verbindung zum View (Referenzattribut)



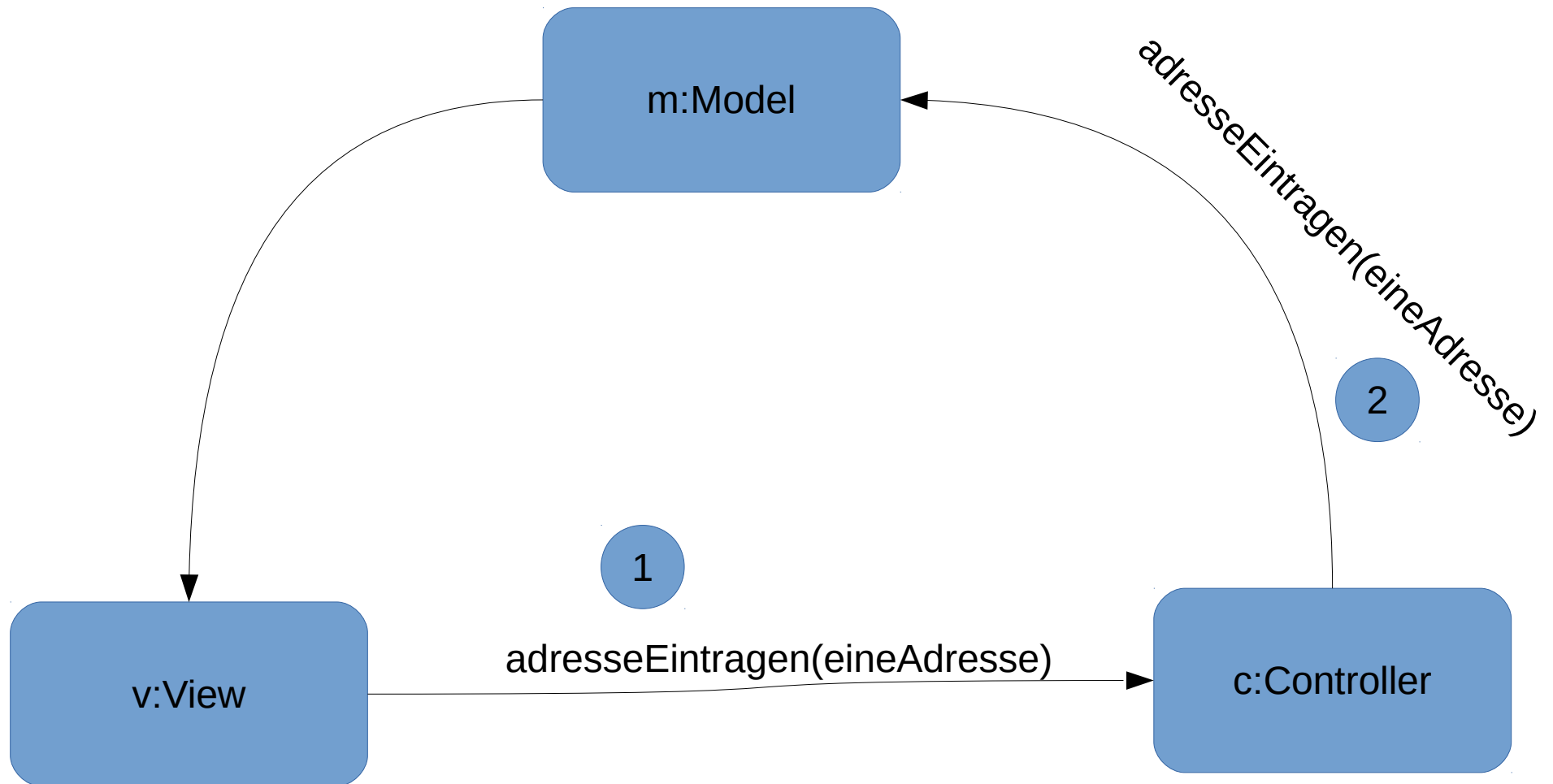




Wenn sich am Zustand des Models etwas ändert, kann es den View davon unterrichten, damit der View sein Aussehen dahingehend anpasst.

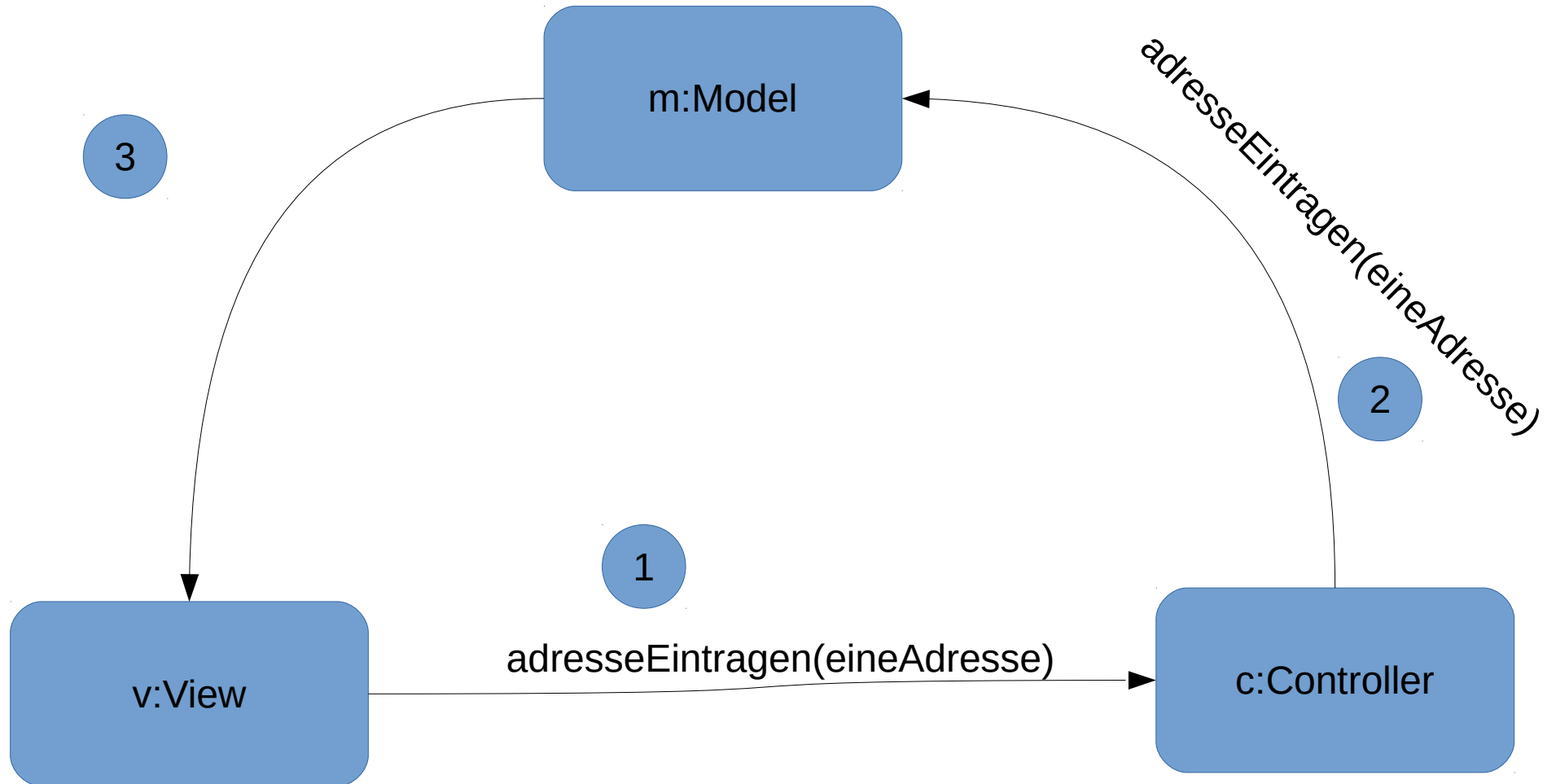
Bei uns ändert sich an eigentlich nur die Anzahl der verwalteten Einträge, und deshalb informiert das Model den View auch nur darüber.

# Model informiert View von Änderung

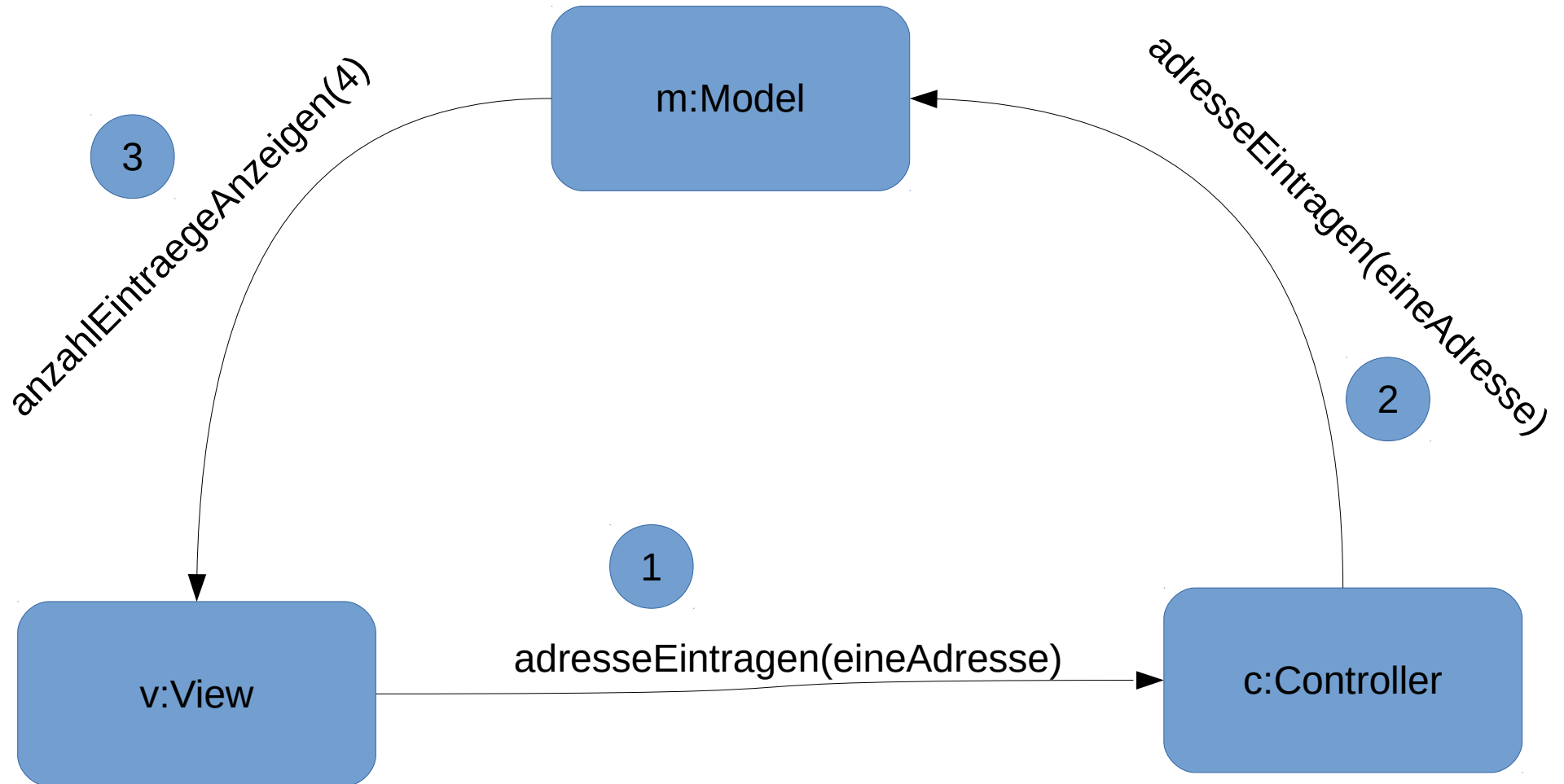




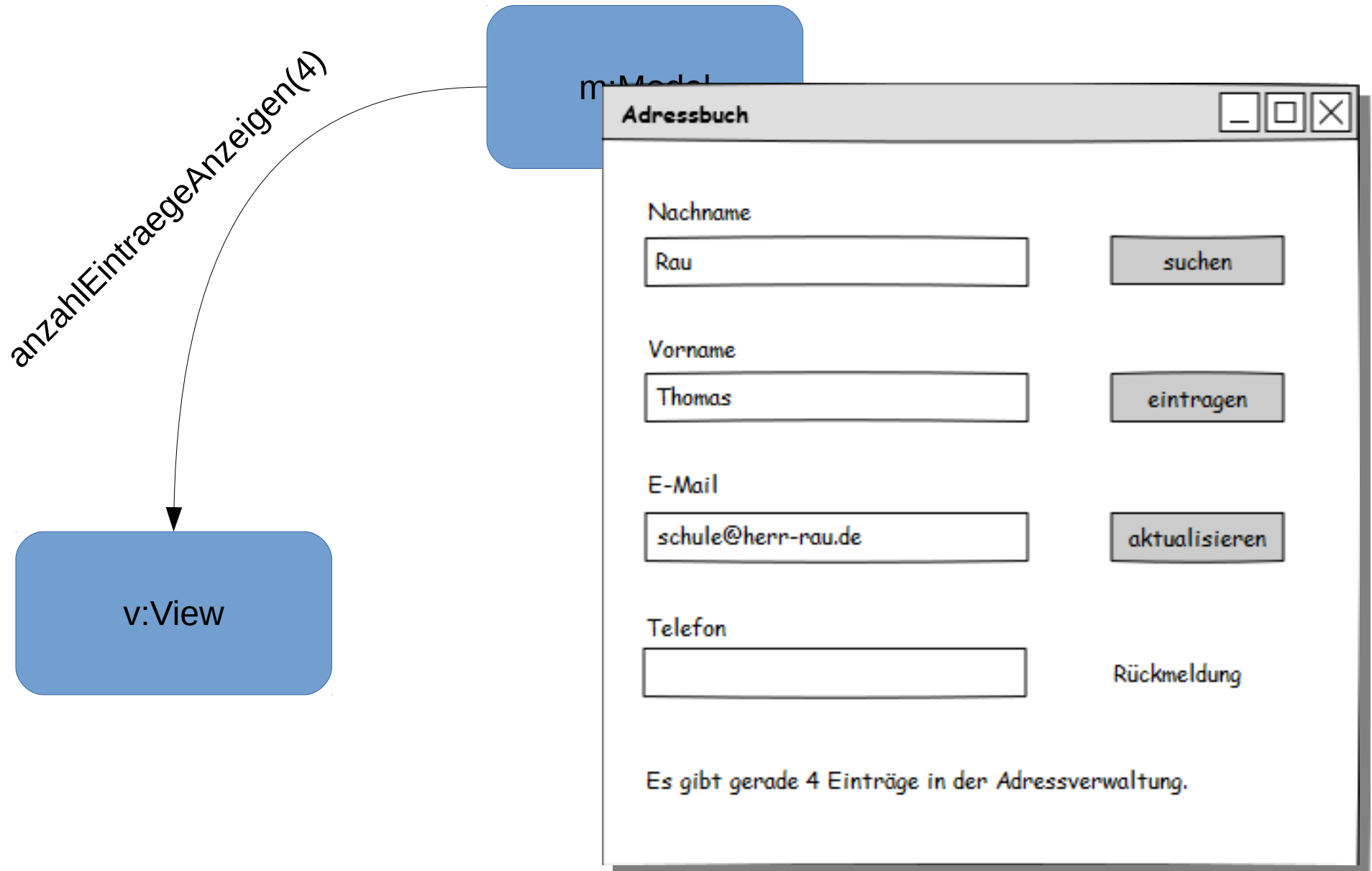
# Model informiert View von Änderung



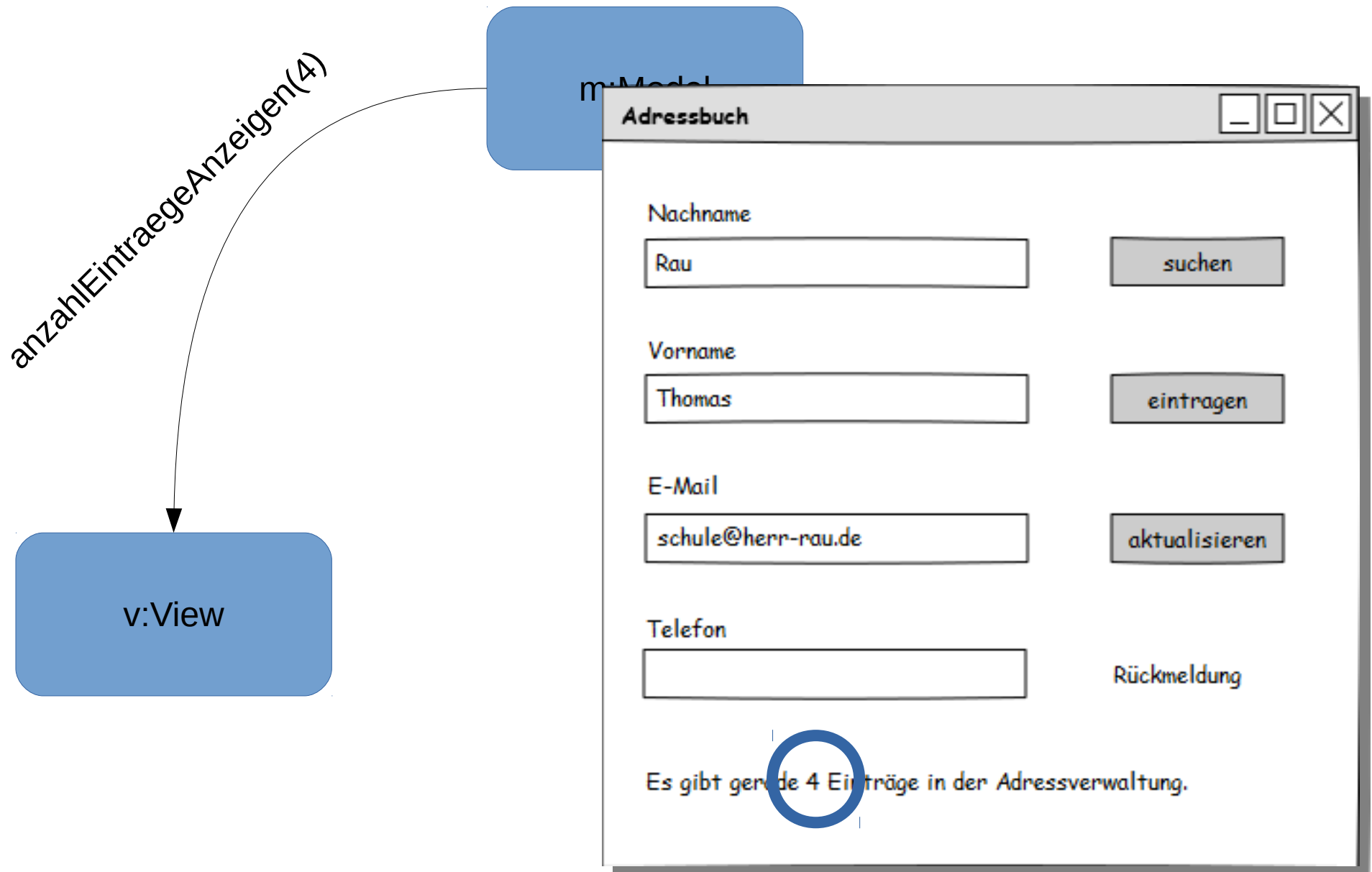
# Model informiert View von Änderung



# View ändert daraufhin Anzeige



# View ändert daraufhin Anzeige

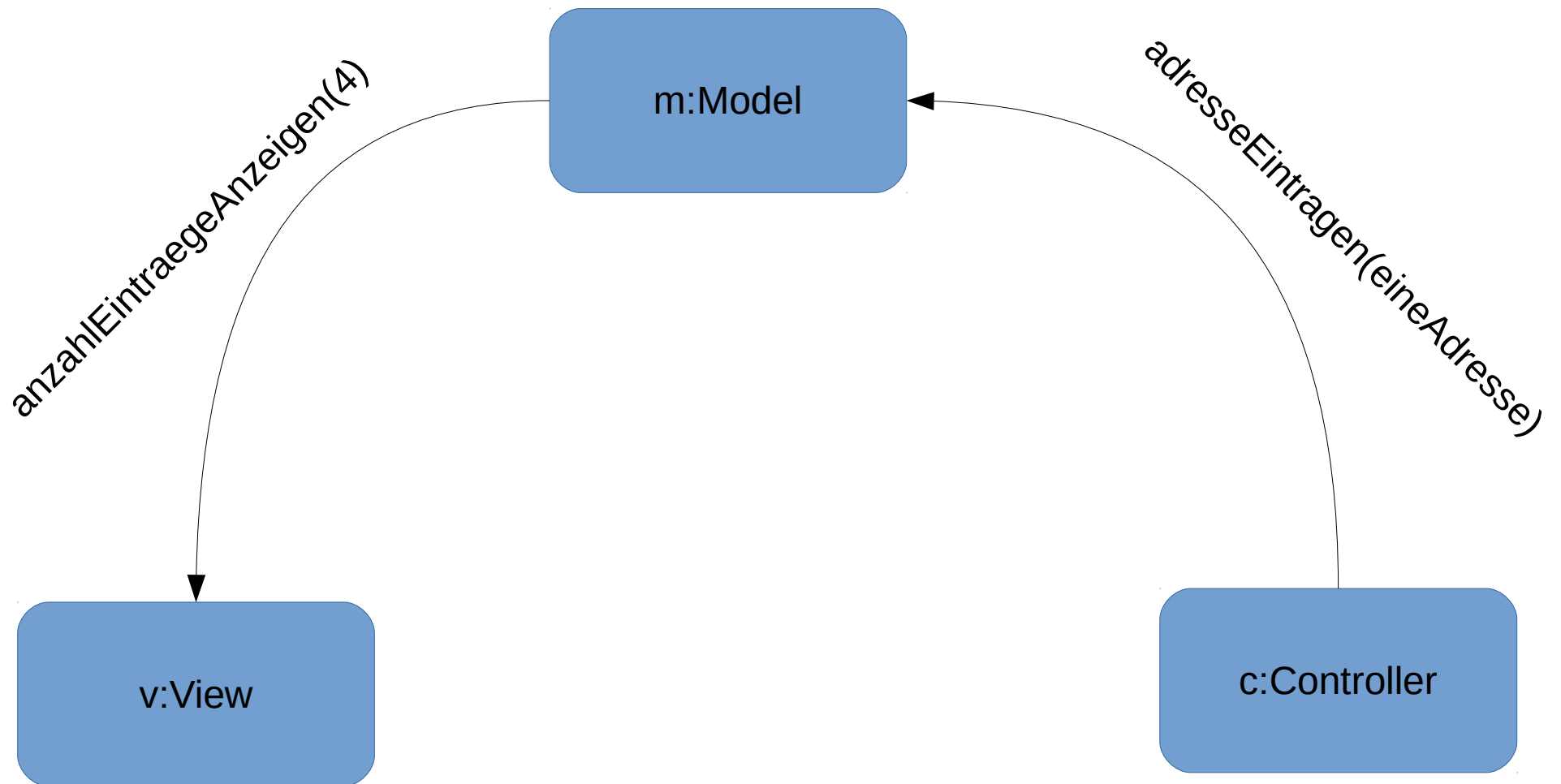


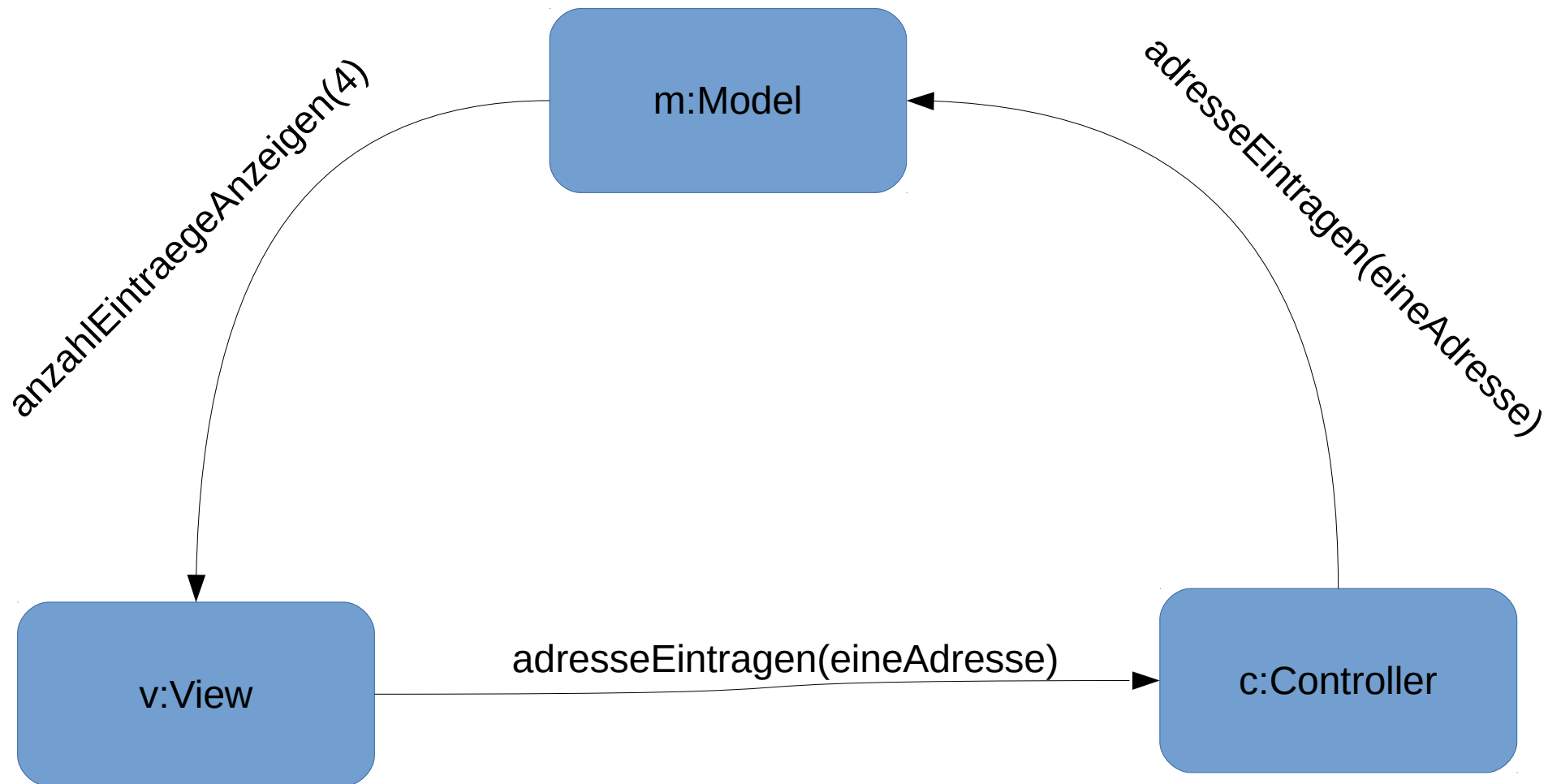
anzahlEintraegeAnzeigen(4)

m:Model

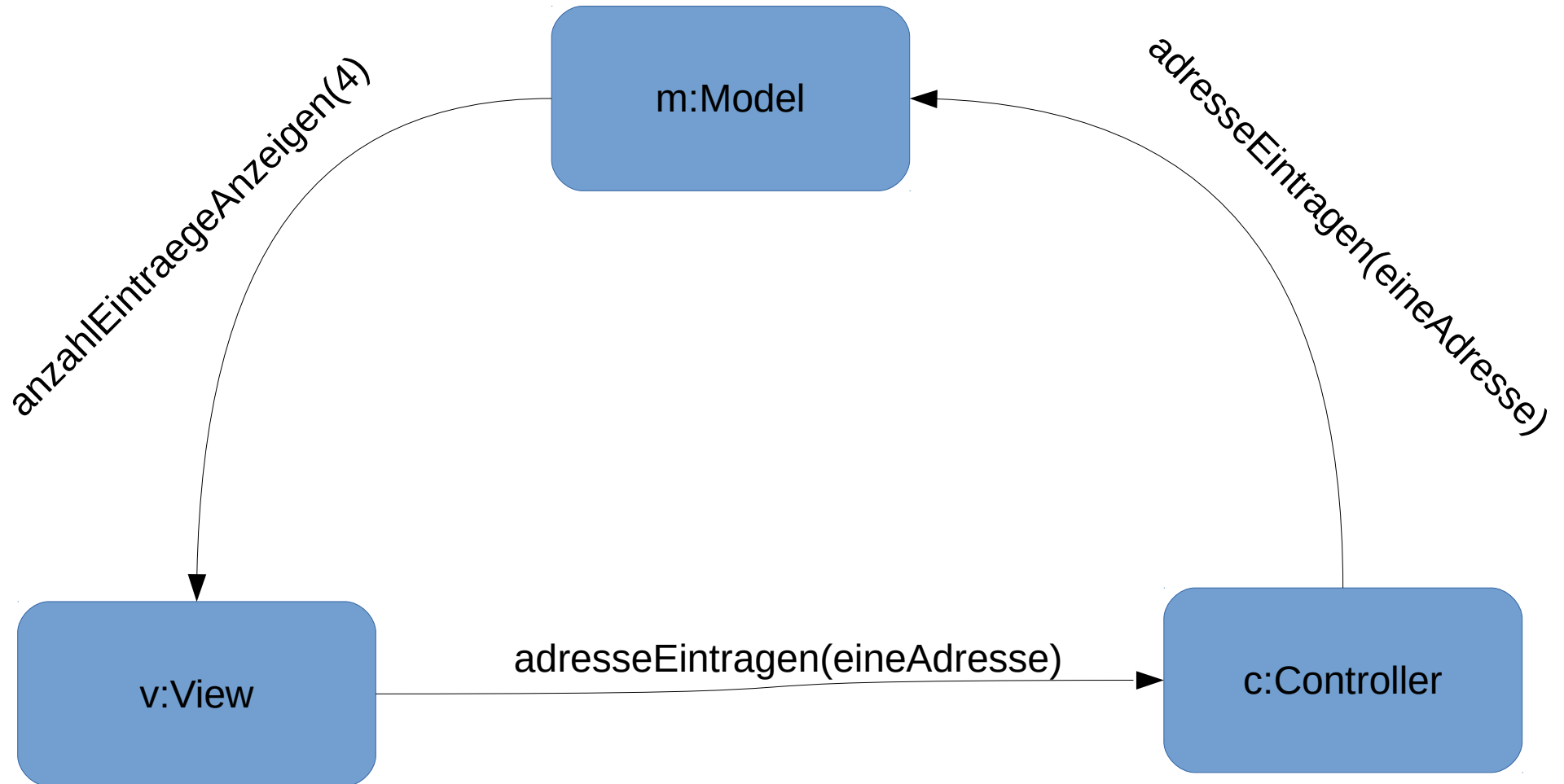
v:View



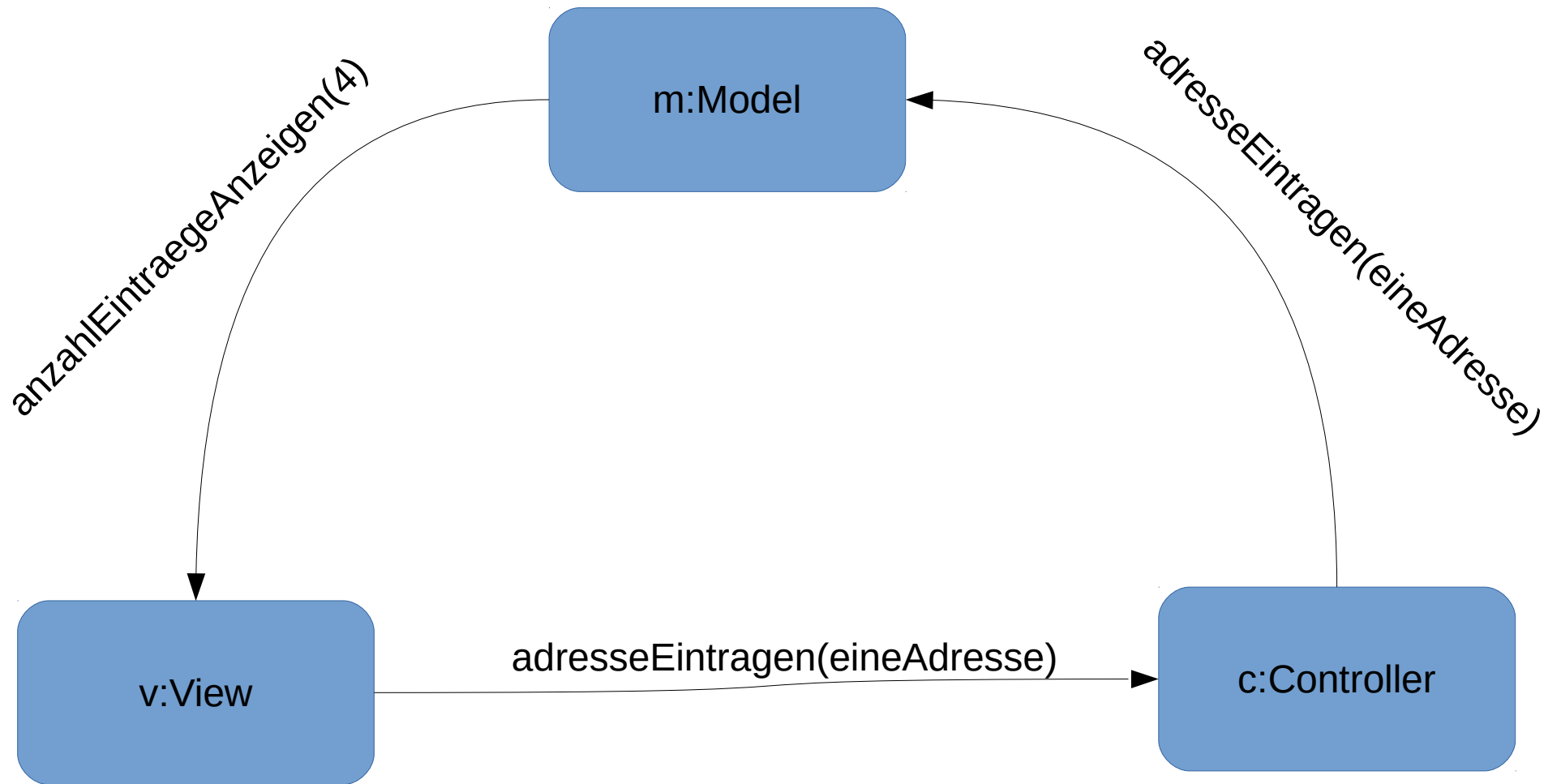


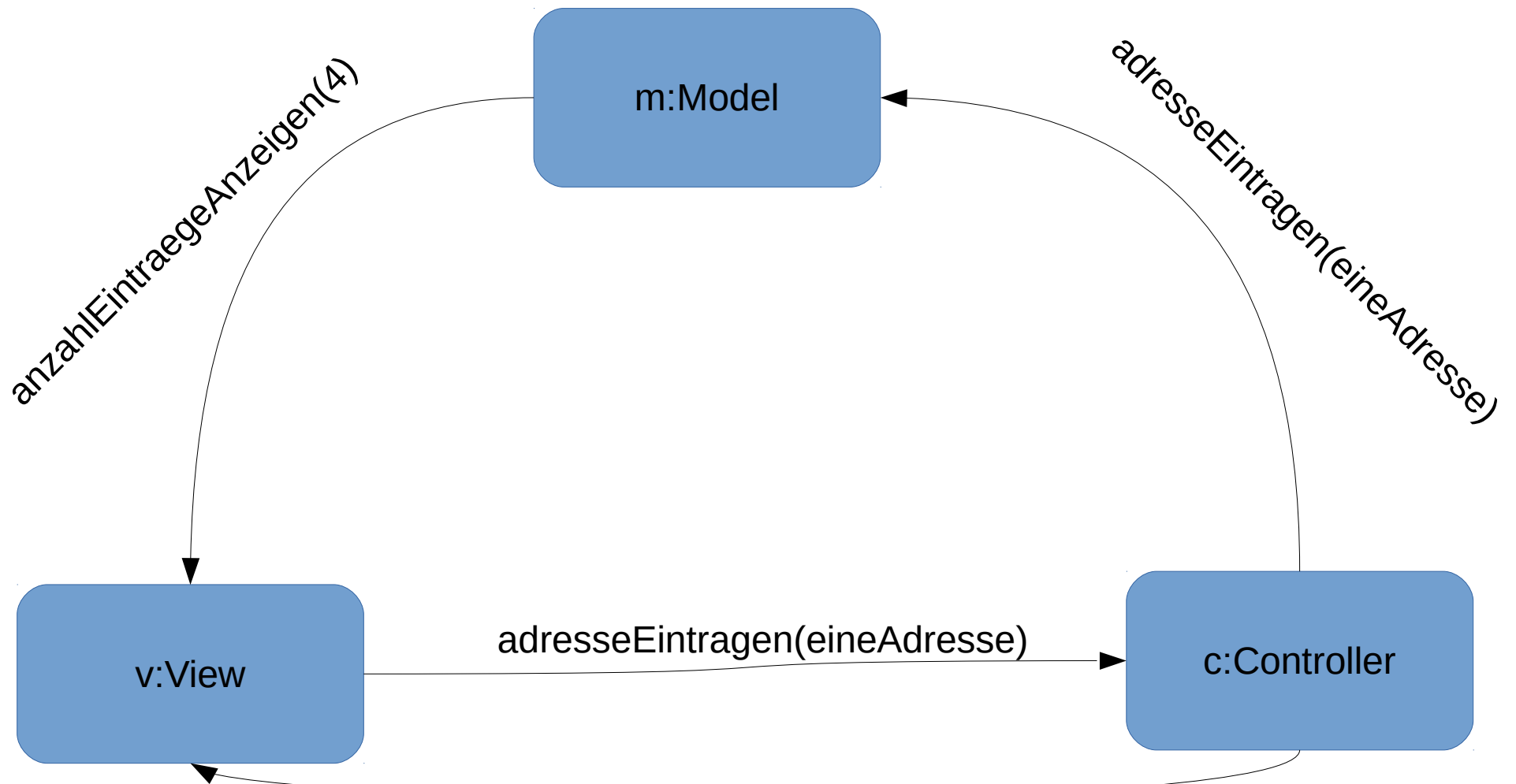


# Außerdem gibt es eine Verbindung Controller > View

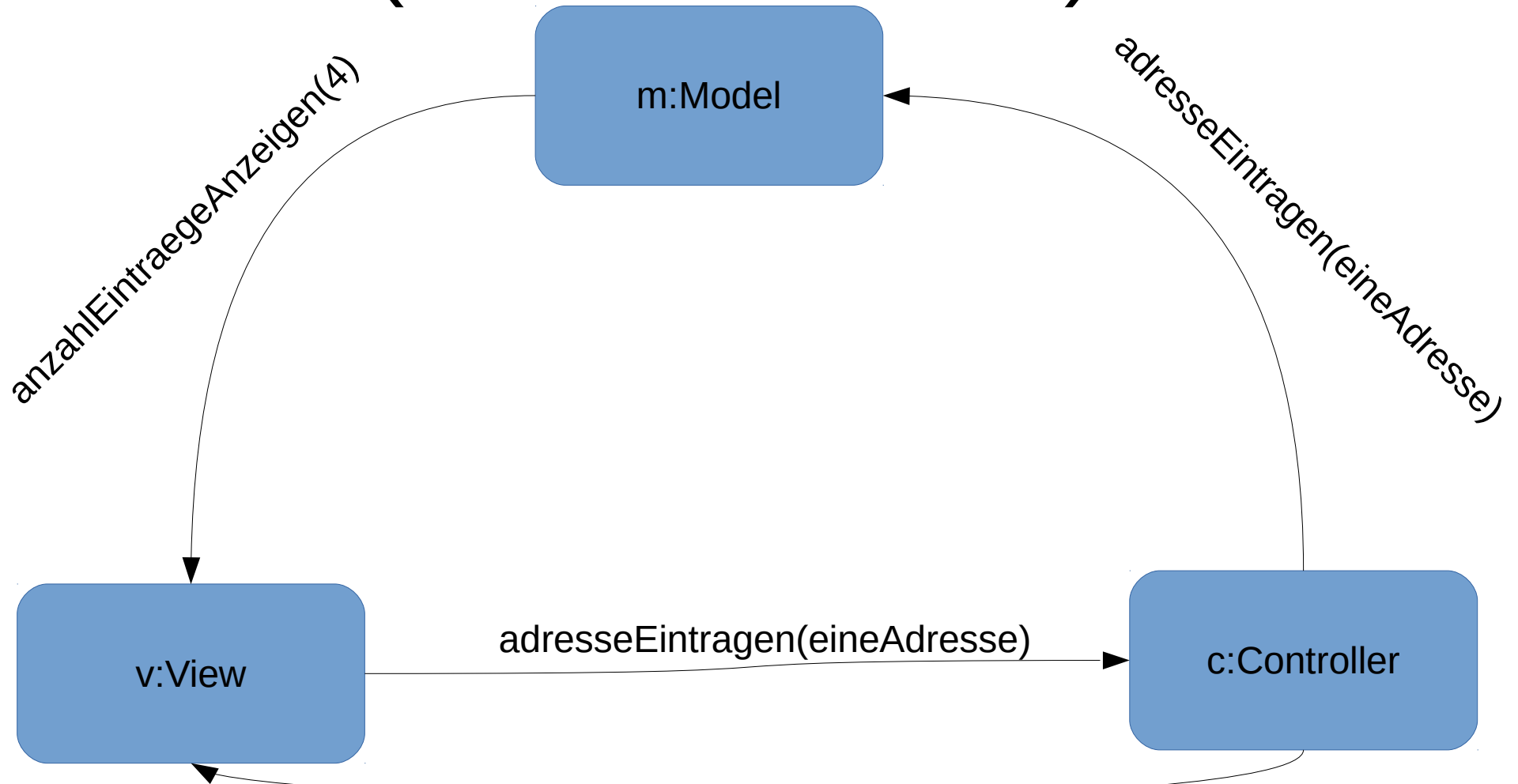




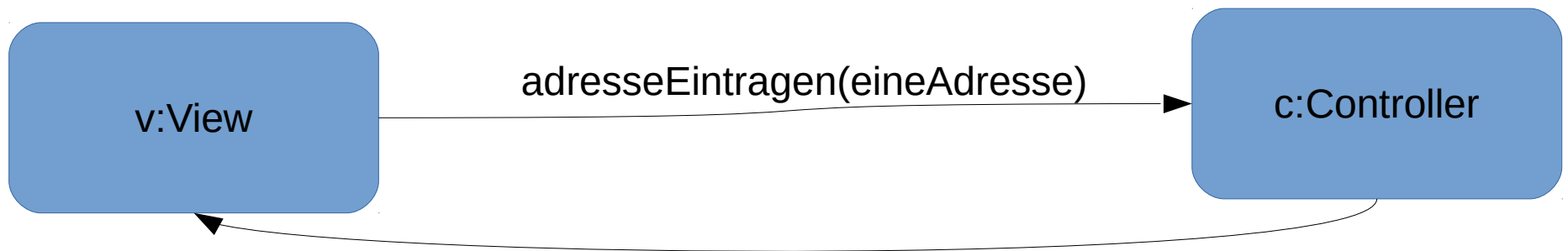




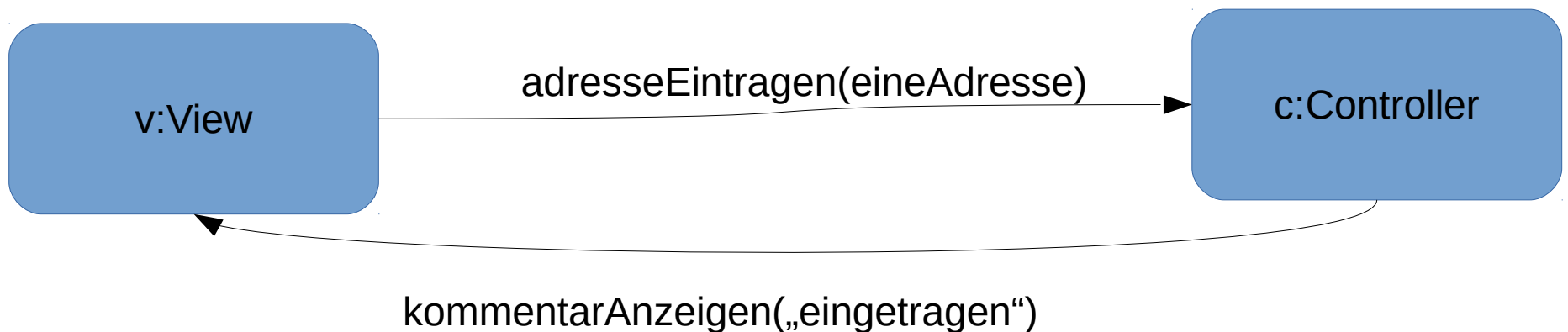
# Verbindung Controller > View (Referenzattribut)



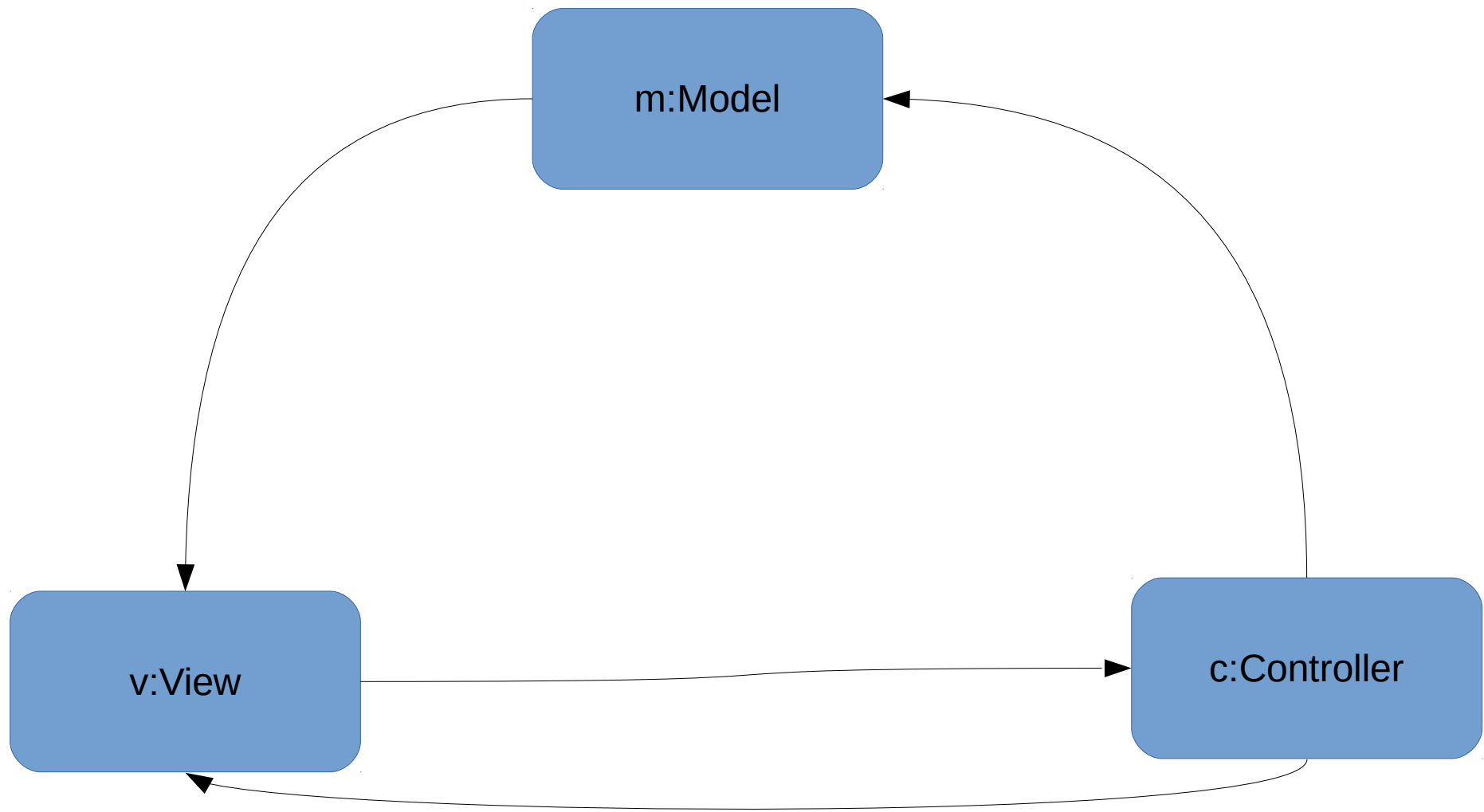
# Controller kann View über Erfolg/Misserfolg usw. informieren

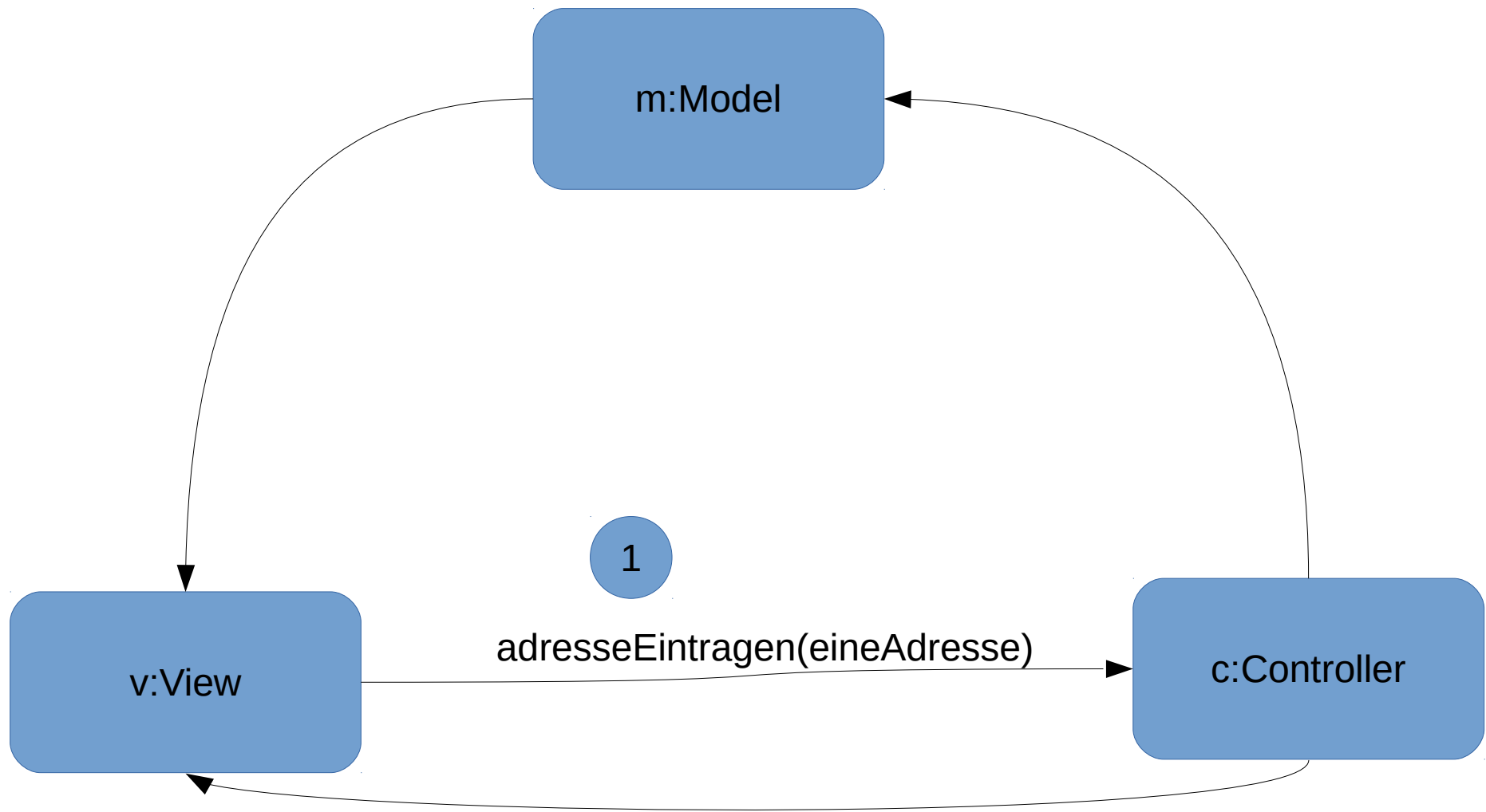


# Controller kann View über Erfolg/Misserfolg usw. informieren

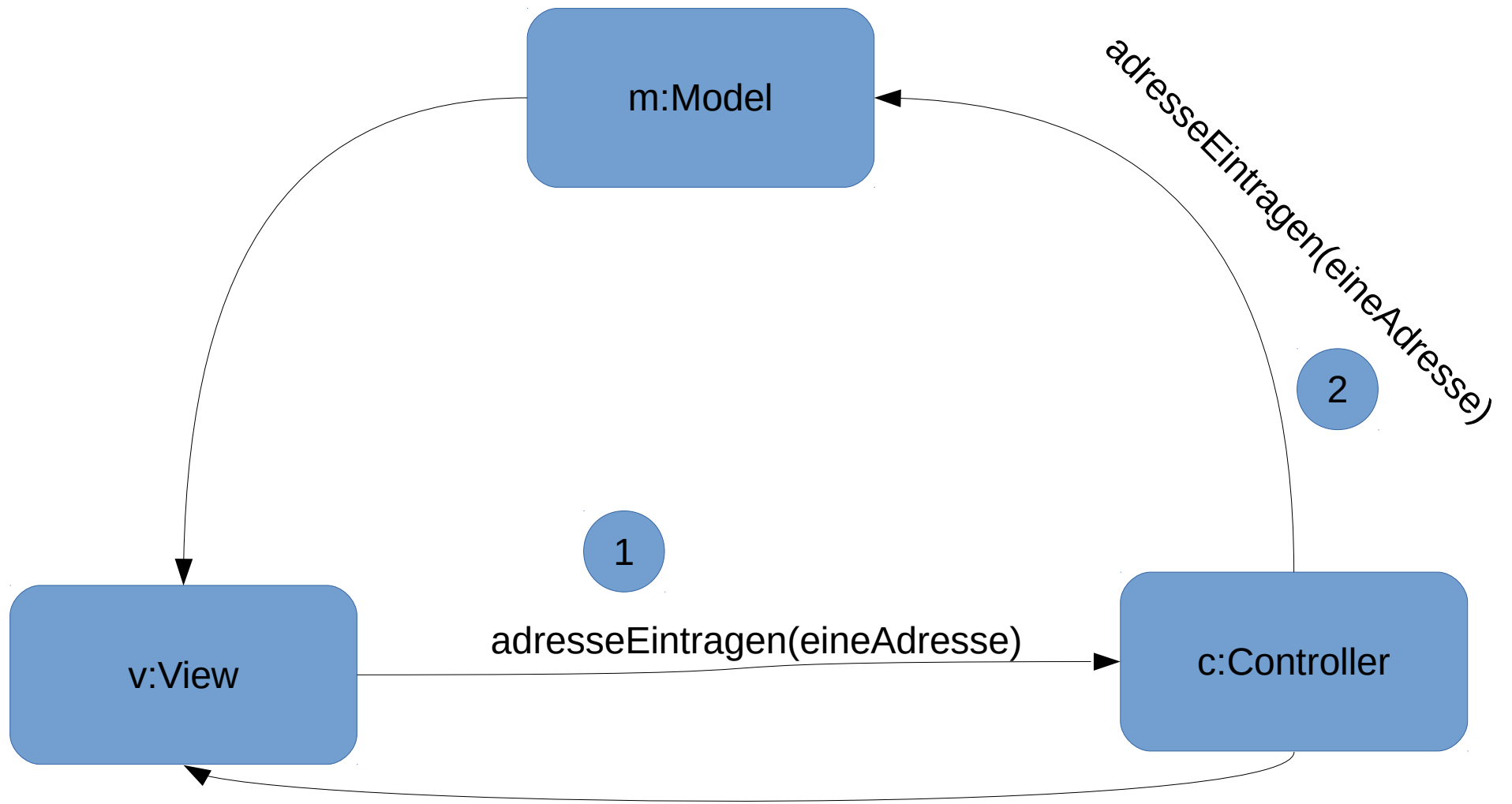


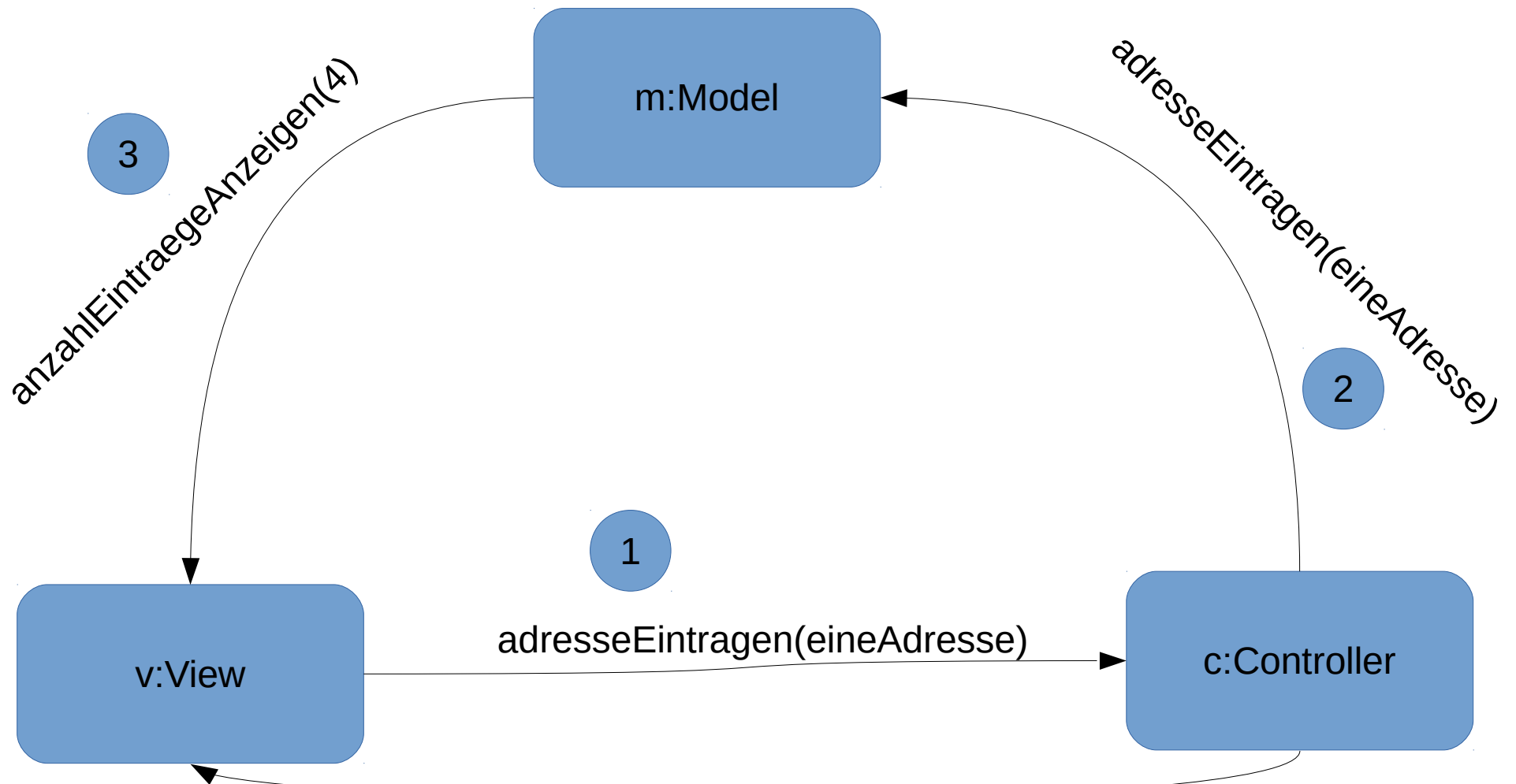
Und jetzt zusammen

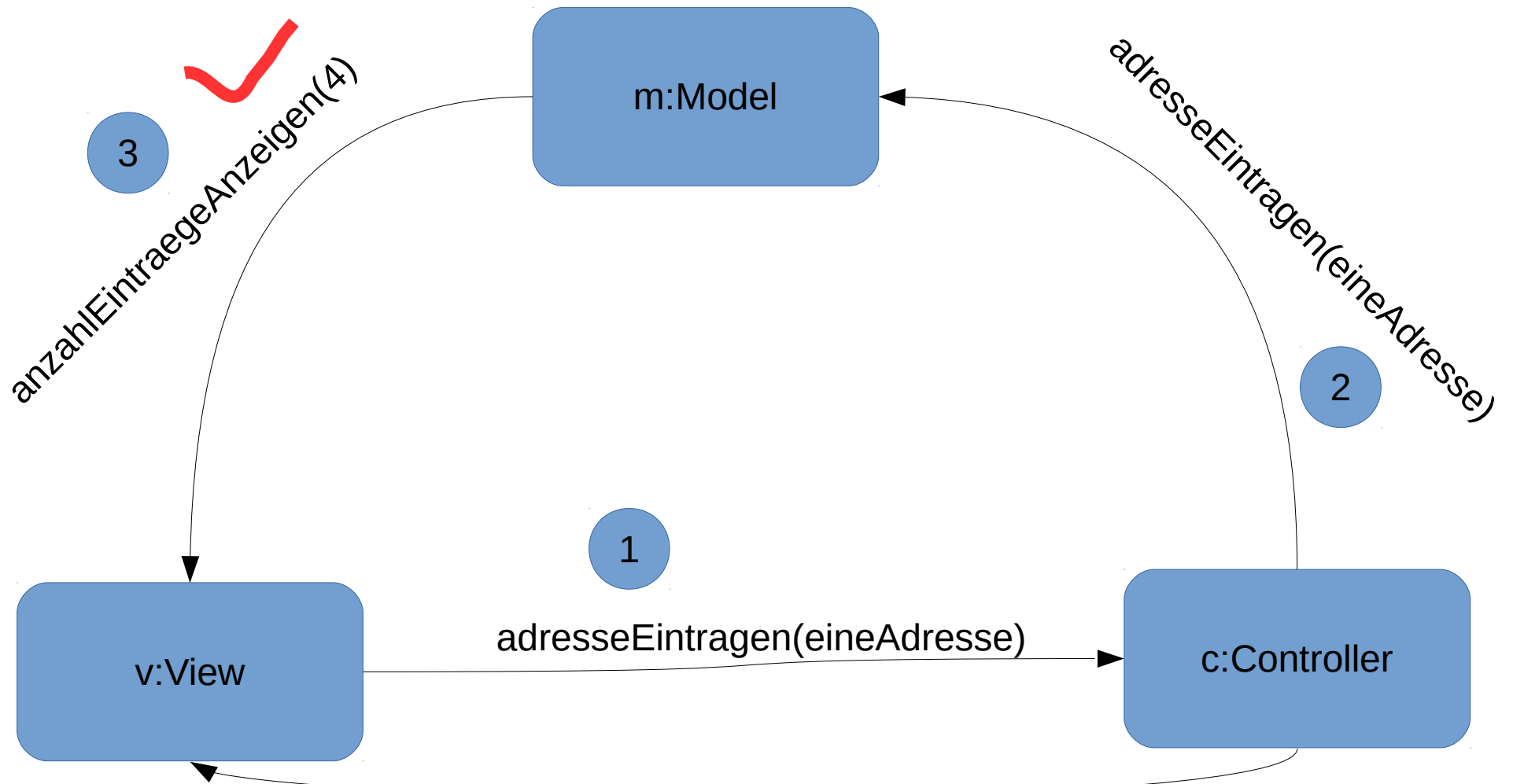


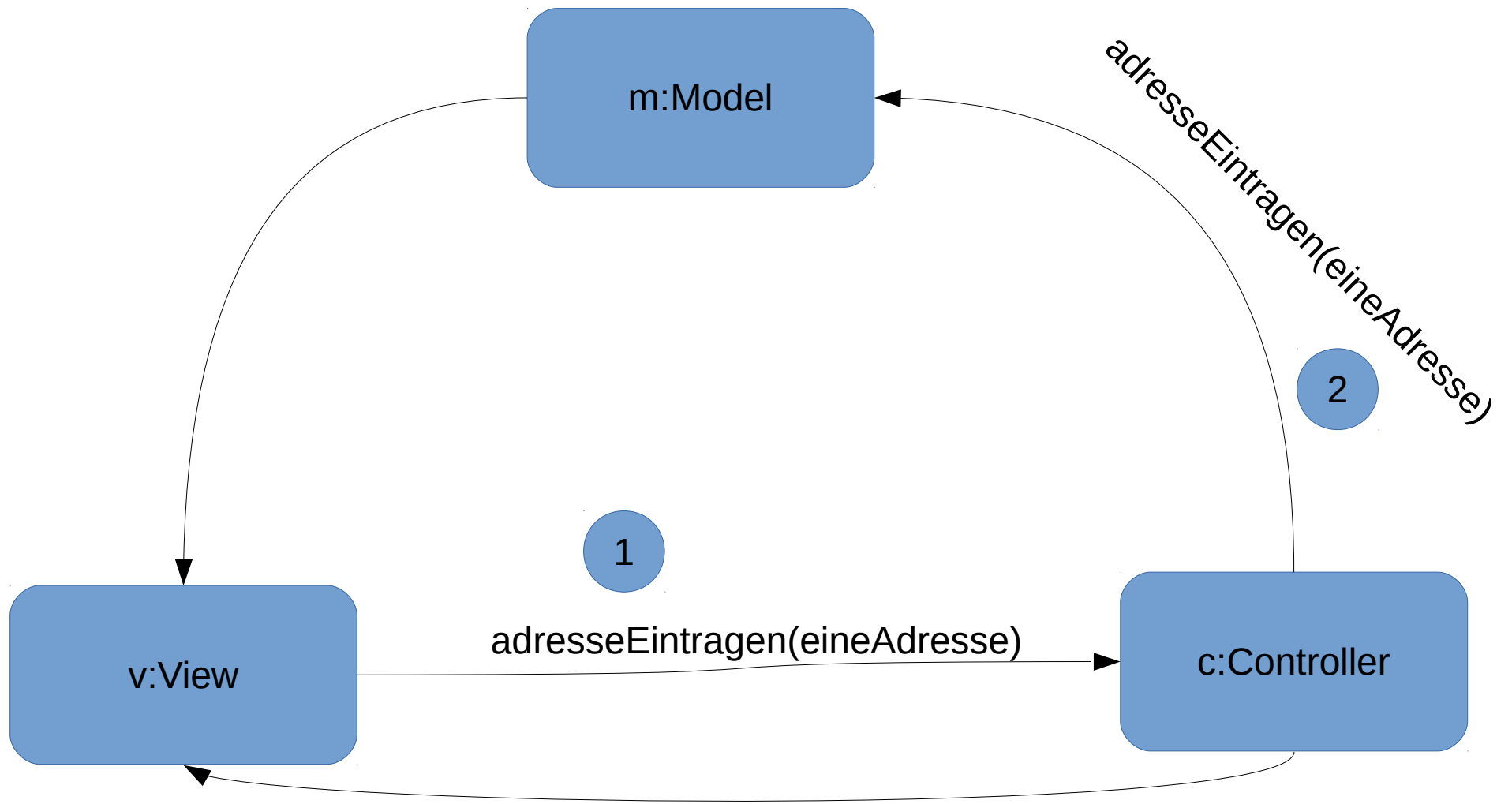


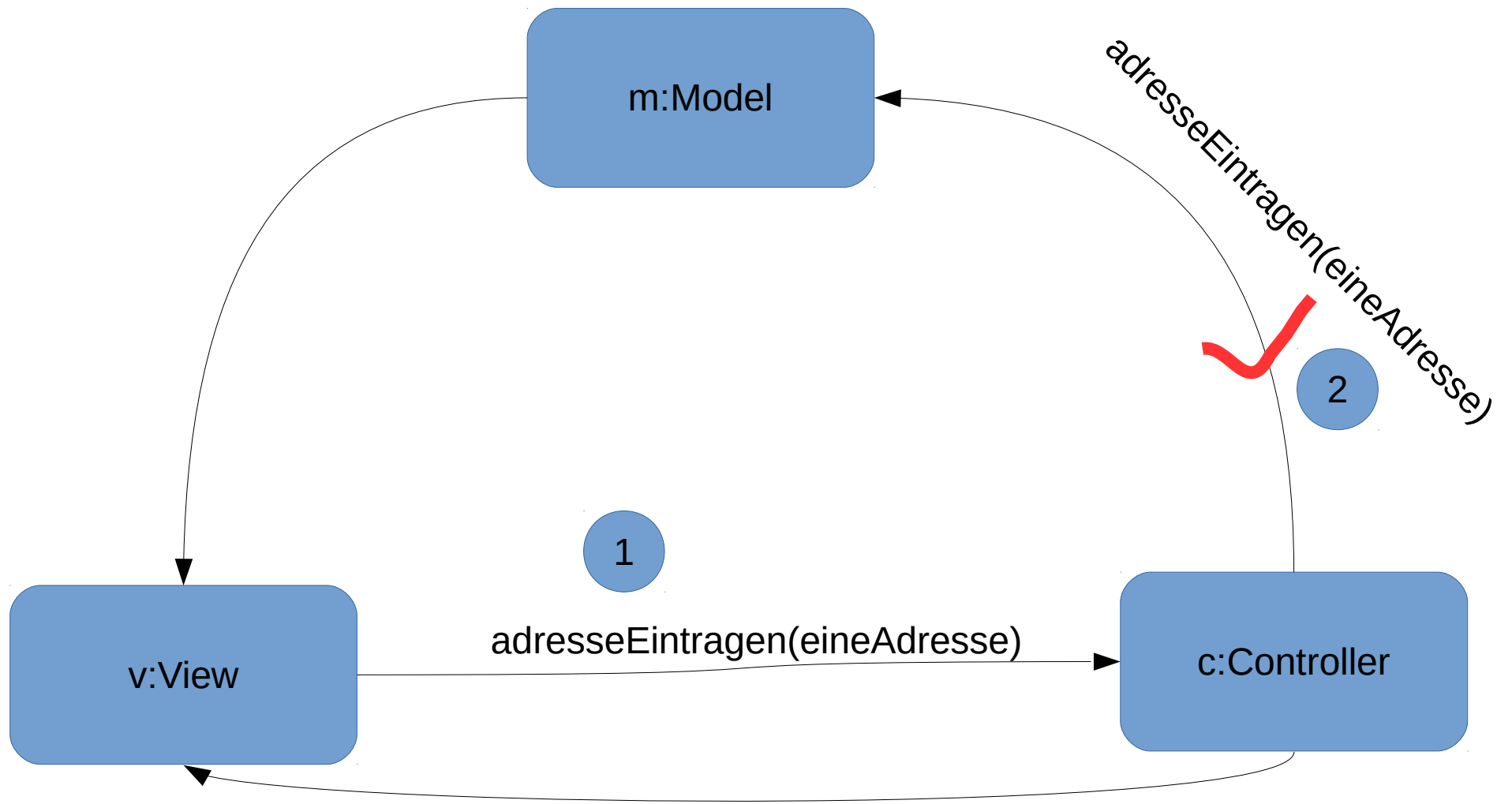


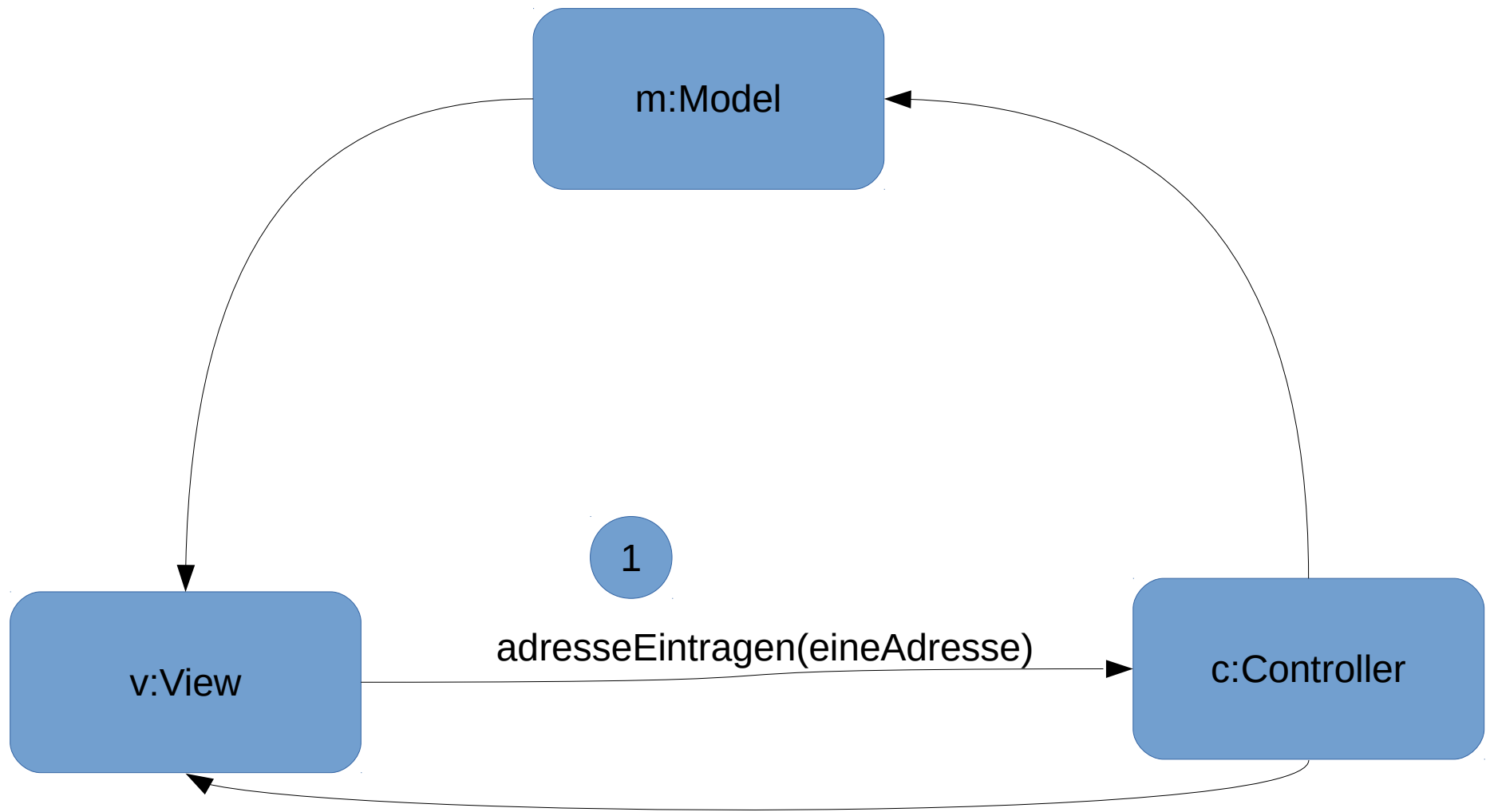


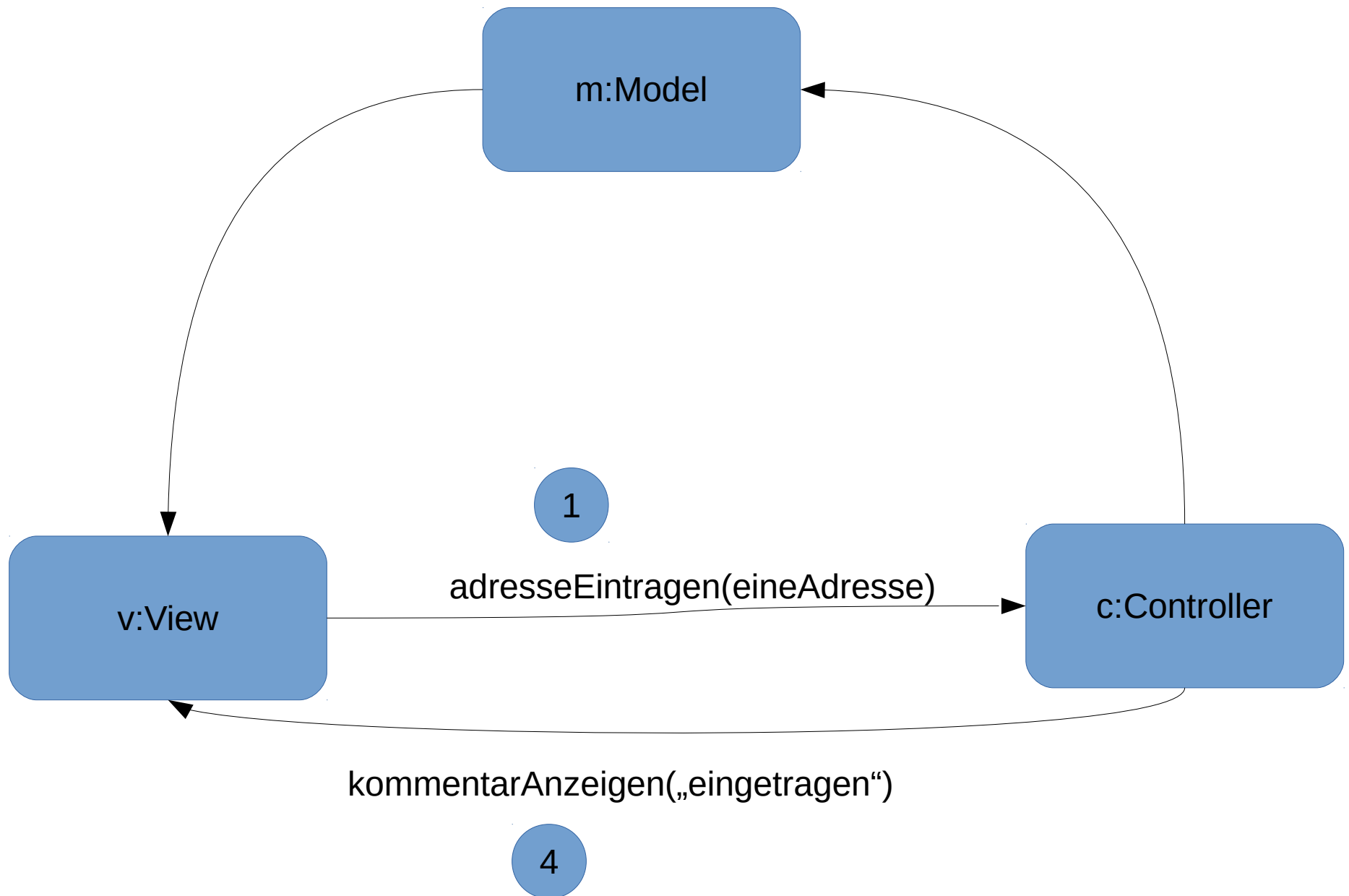


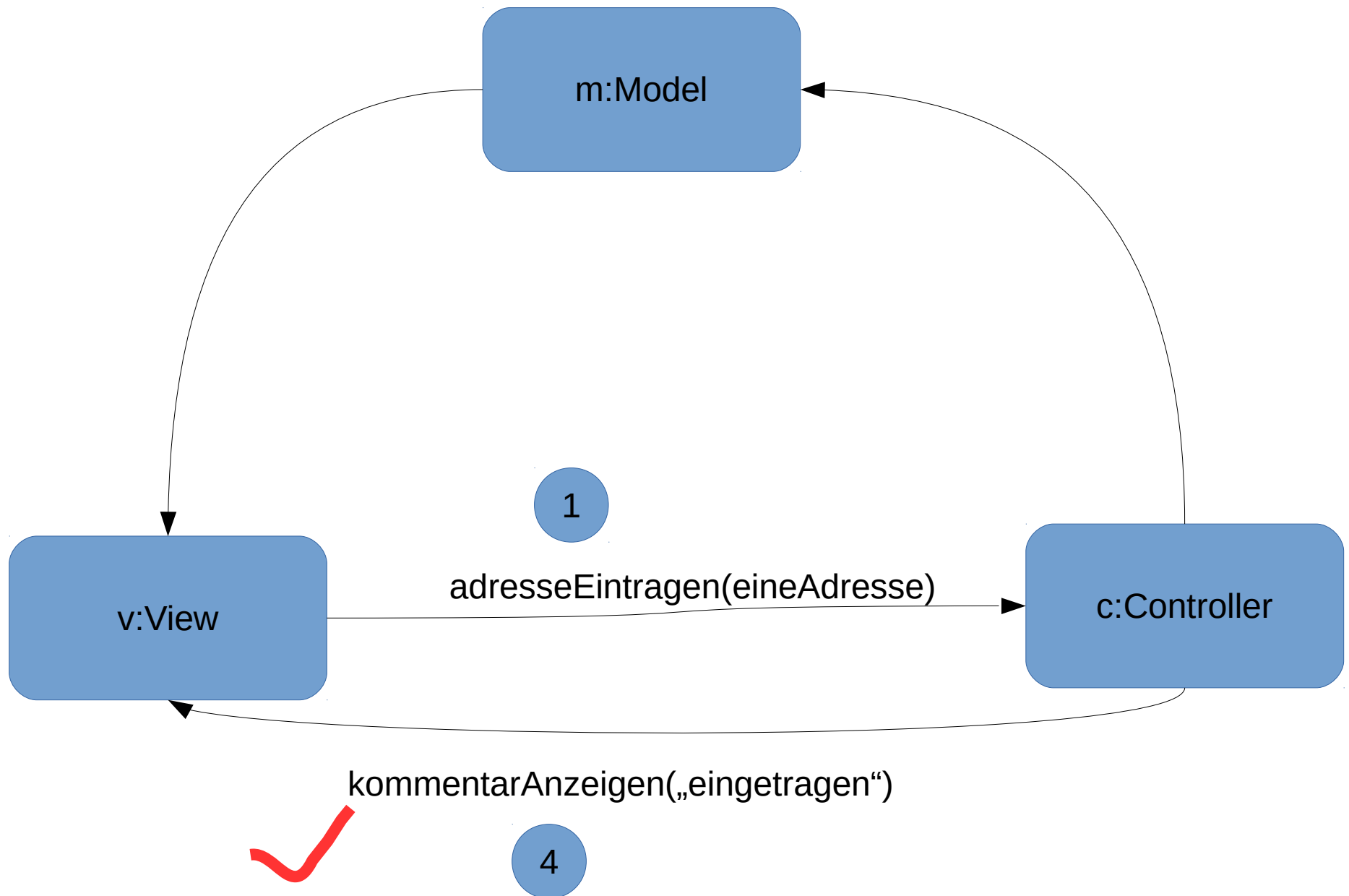




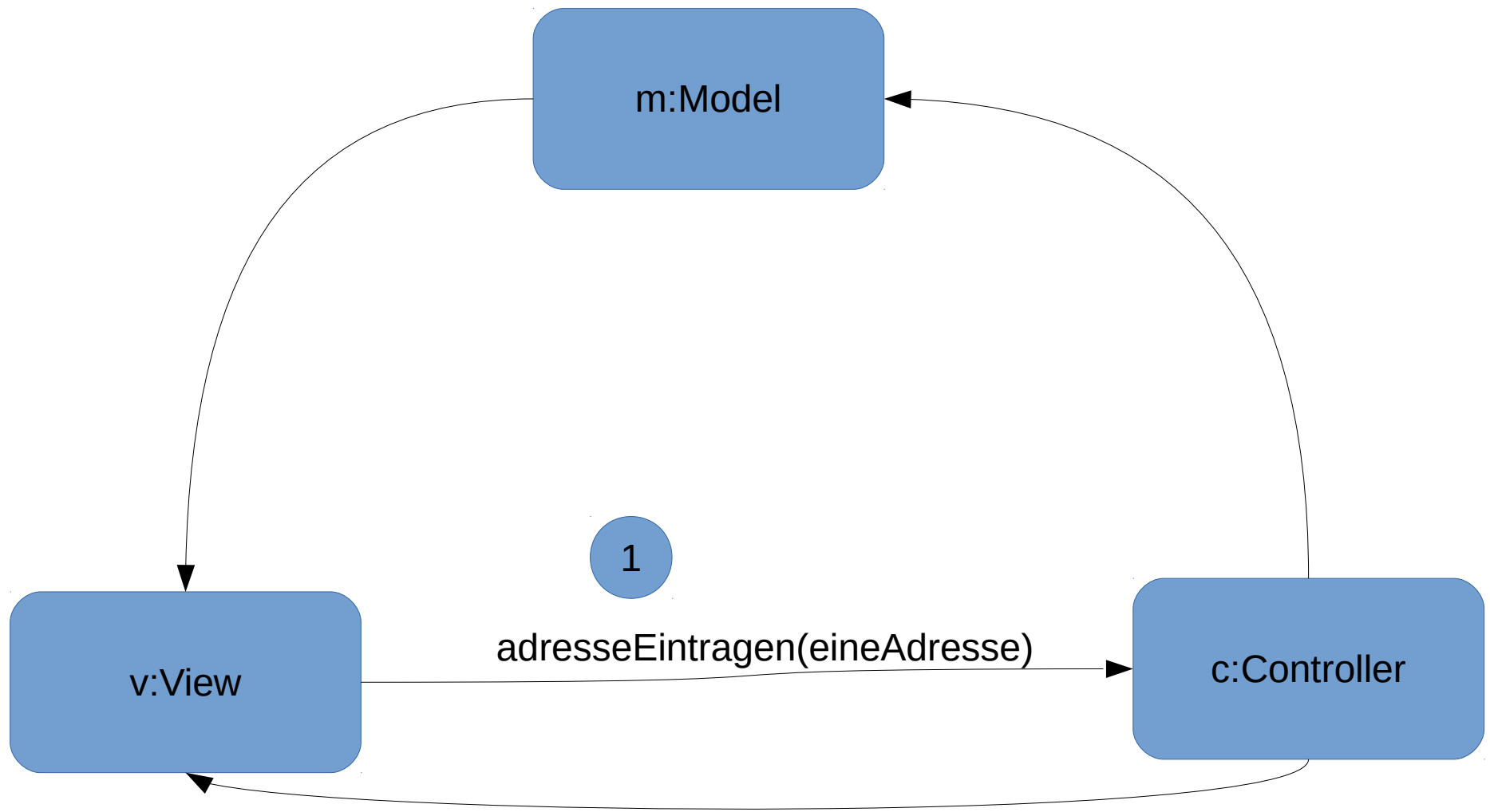


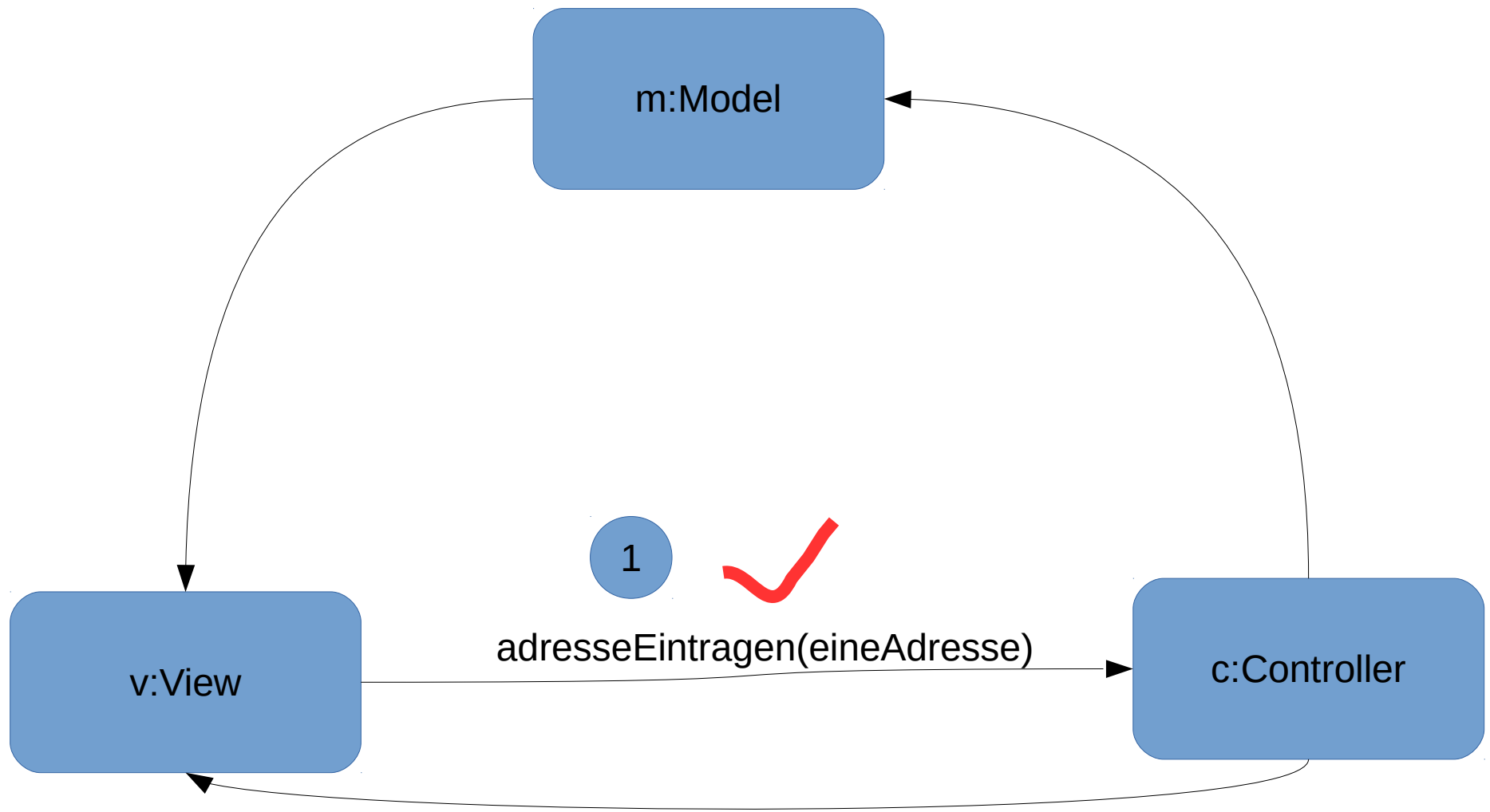


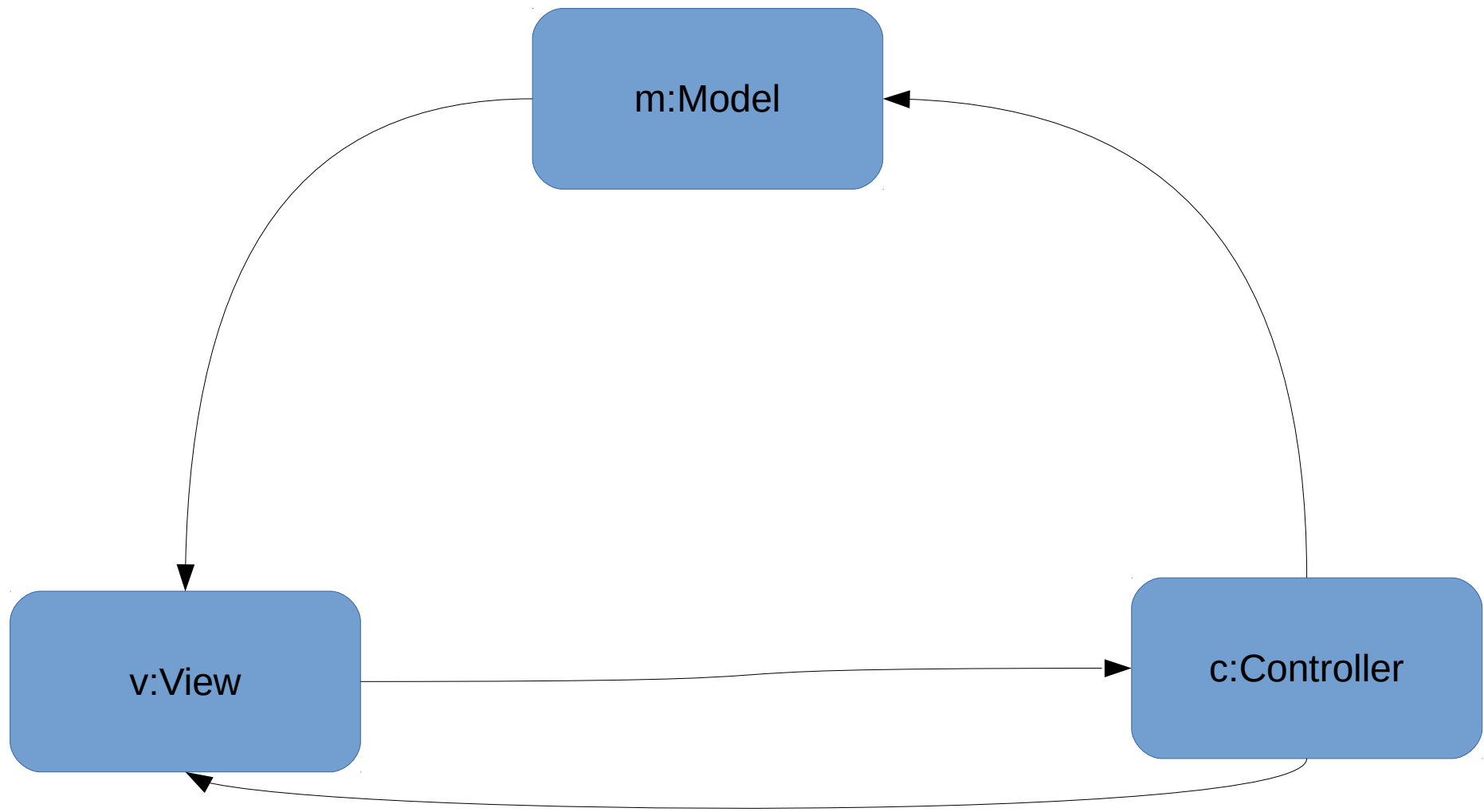








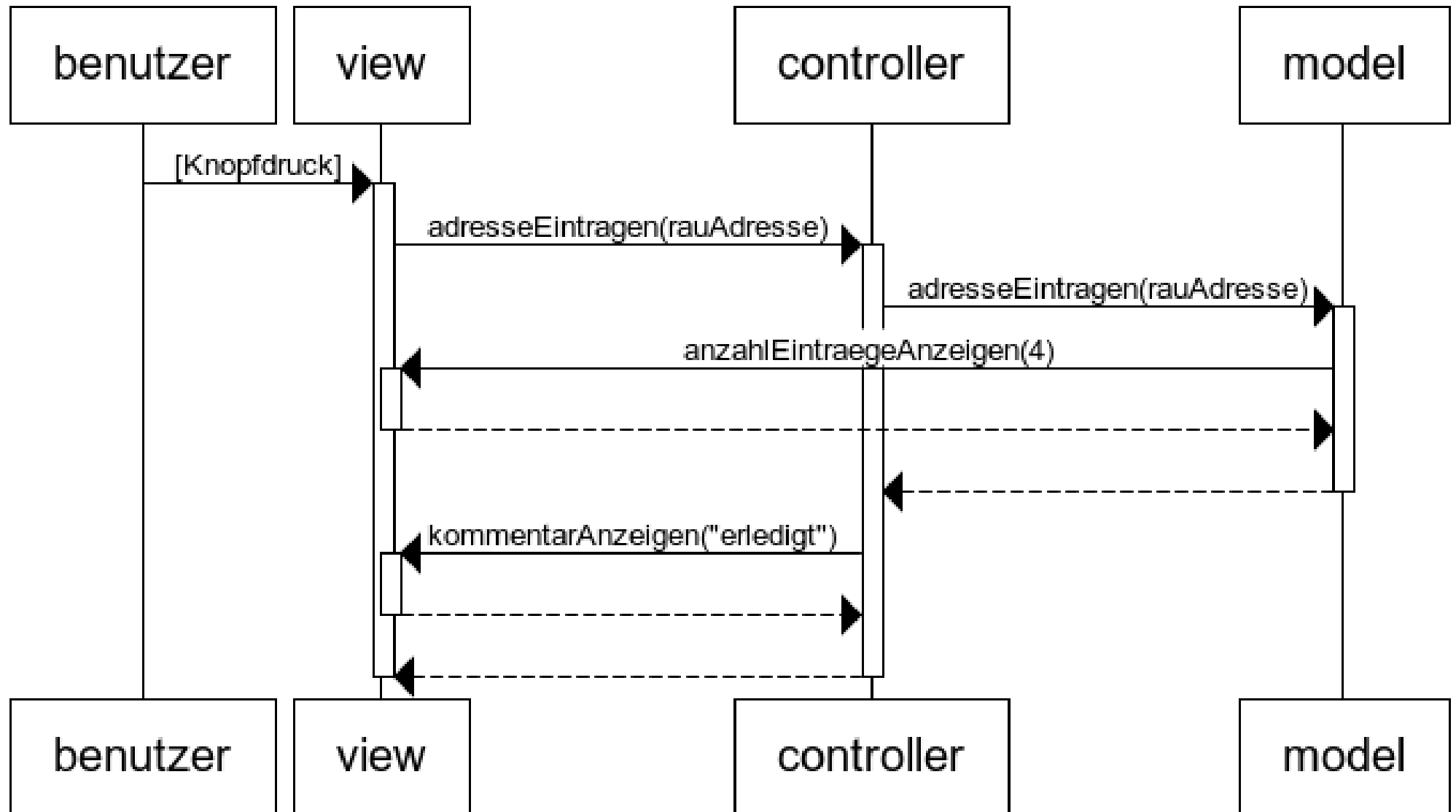




# Als Sequenzdiagramm

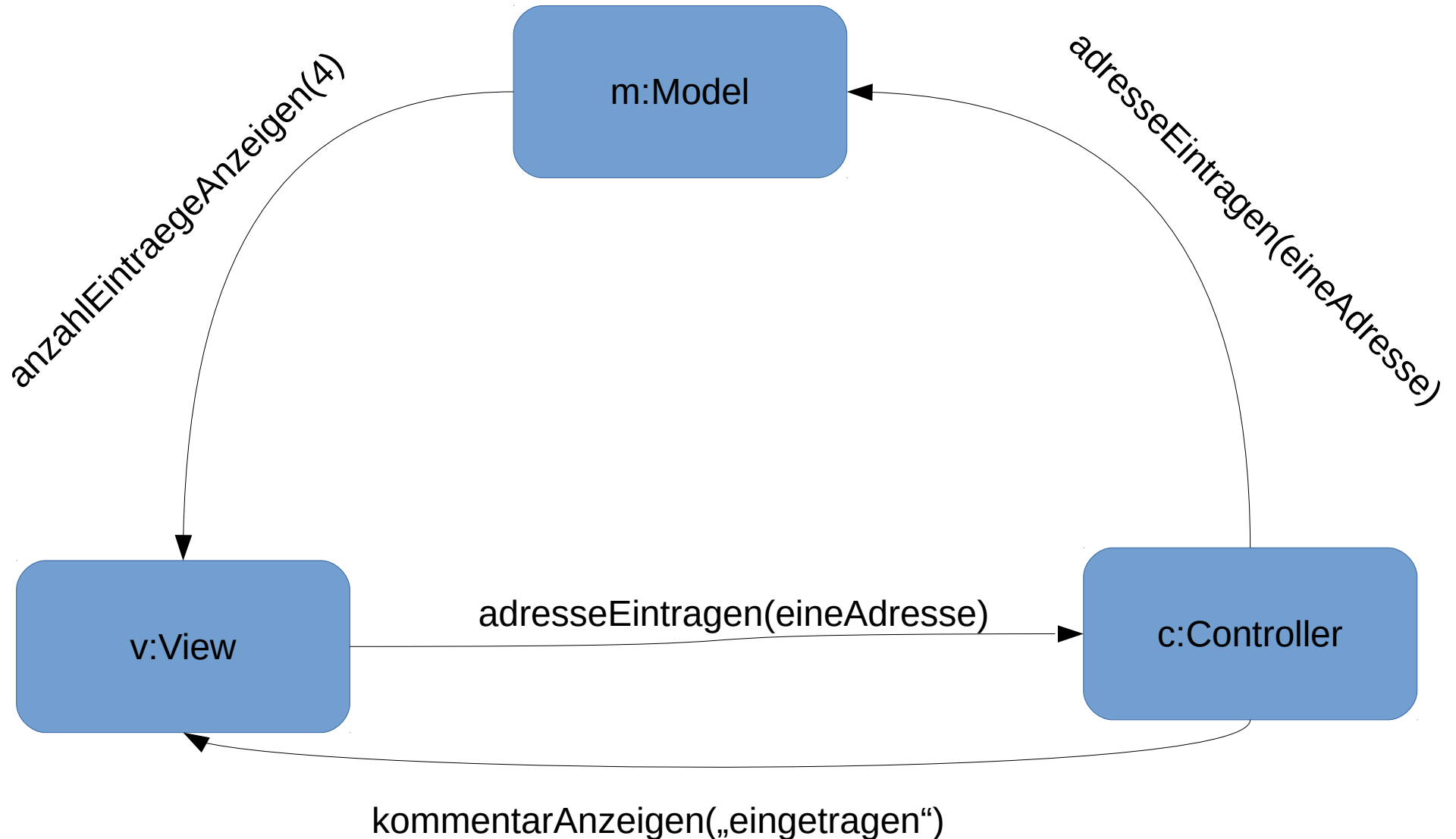
# Als Sequenzdiagramm

## Model-View-Controller-Kommunikation



# Notwendige Methoden und Attribute

# Notwendige Methoden und Attribute



# Notwendige Methoden und Attribute

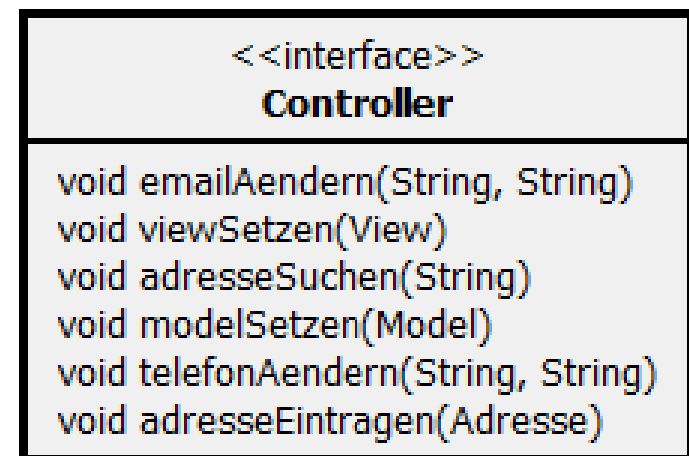
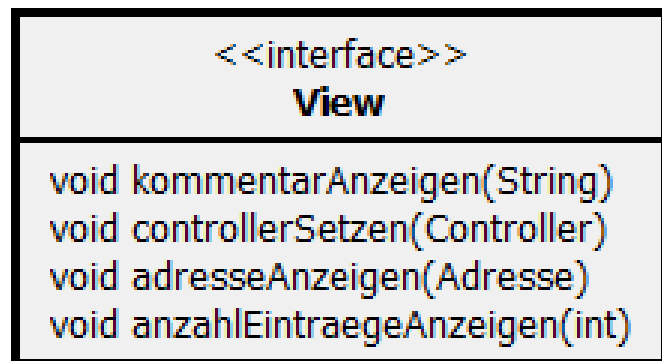
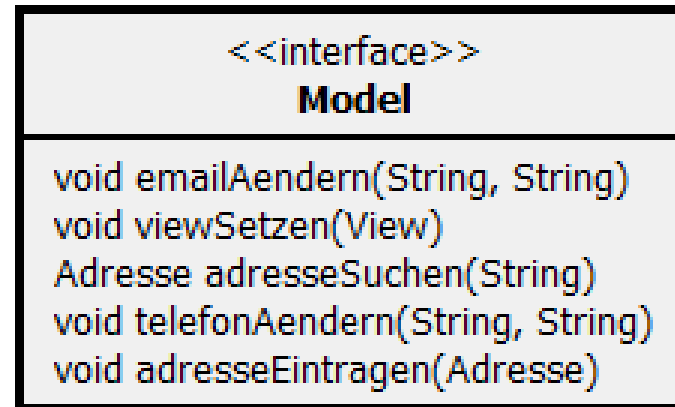
<b>Model</b>
View v
void adresseEintragen(Adresse a)

<b>View</b>
Controller c
void kommentarAnzeigen(String s) void anzahlEintraegeAnzeigen(int i)

<b>Controller</b>
Model m View v
void adresseEintragen(Adresse a)



# Notwendige Methoden und Attribute werden in Interfaces festgelegt



In BlueJ

