PM1/PT Ruby: Objekte und Klassen

Prof Dr. Birgit Wendholt

Prof. Dr. Bernd Kahlbrandt (Überarbeitung)

Konzepte

- Objekte
- Klassen
- Methoden
- Parameter

Objekte und Klassen

- Objekte einer Programmiersprache modellieren Objekte eines Anwendungsbereichs.
- Objekte repräsentieren individuelle Instanzen einer Klasse.

- Eine Klasse ist eine Vorlage für eine Menge gleichartiger Objekte.
- Eine Klasse beschreibt auf abstrakte Weise alle Objekte einer Art. (Metadaten)
- Objekte werden aus Klassen erzeugt.(Objektfabrik)
- Klassen sind eine besondere Art von Objekten.(„Metaklasse“)

Objekte und Klassen

- **Beispiel**: Verkehrssimulation
- Zentraler Begriff: „Auto“
- Ist „Auto“ ein **Objekt** oder eine **Klasse**?

- Fragen:
  - Welche Farbe hat ein Auto?
  - Was ist die Höchstgeschwindigkeit?
  - Auf welcher Straße fährt das Auto?
- Die Fragen lassen sich nur dann beantworten, wenn wir über ein bestimmtes Auto reden.

➔ Der Begriff „Auto“ bezieht sich auf die **Klasse** Auto.

Objekte und Klassen

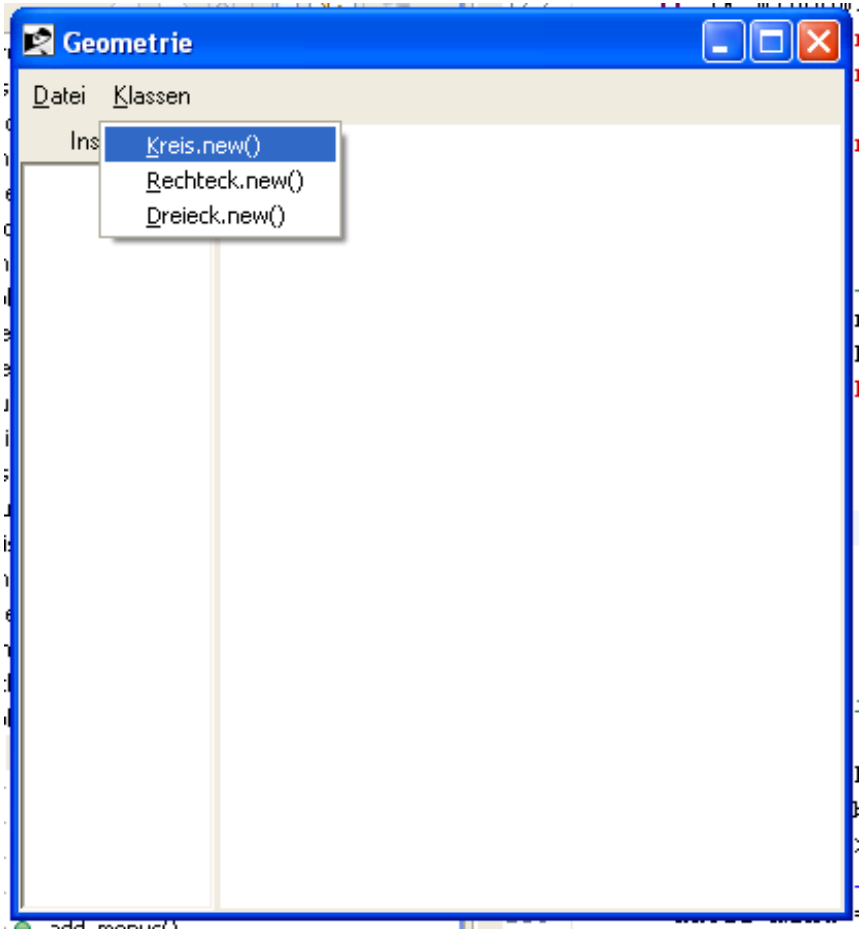
- Beispiel: Verkehrssimulation
- Zentraler Begriff: „Auto“
- Ist „Auto“ ein **Objekt** oder eine **Klasse**?

- Jetzt betrachten wir das Auto, das bei mir im Carport steht.
- Dann lassen sich die Fragen beantworten:
  - ➔ Das Auto ist silberfarben
  - ➔ Es fährt maximal 270 km/h
  - ➔ Es fährt nicht, sondern parkt zur Zeit an der HAW.
- Hier reden wir über ein **Objekt**, über ein spezielles Exemplar eines Autos.
- Ein Objekt heißt auch **Instanz** einer Klassen (von engl. ***instance***).

Instanzen von Klassen erzeugen

- **Beispiel**: Geometrische Figuren.
- Wir wollen Objekte zu den Klassen ***Kreis, Quadrat, Rechteck*** erzeugen.
- Öffnen Sie das Rubyprojekt ***v01_geometrie*** und starten Sie die ***Toolbox*** (Klasse im Projekt)
- Im oberen Bereich der Toolbox finden wir das Klassen Menu, das für die Klassen die Erzeuger-methode ***new*** enthält.

➔ Instanzen einer Klasse werden immer mit ***new()*** erzeugt.

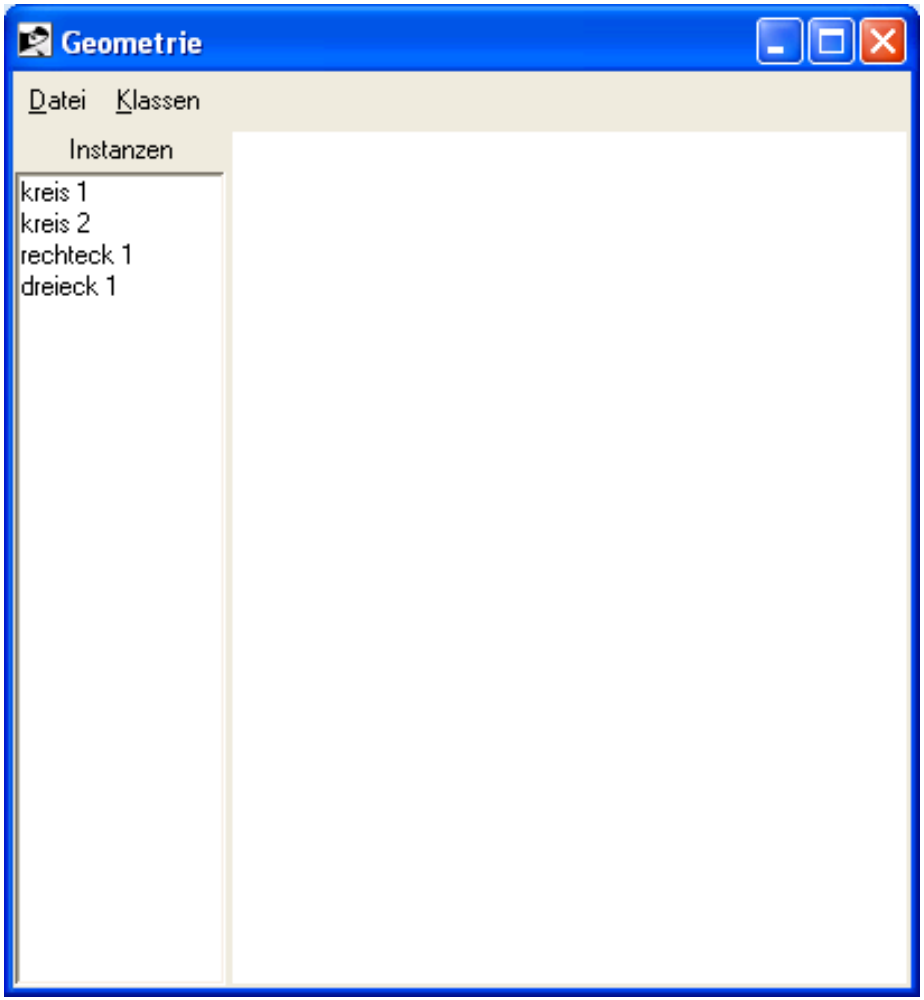


Instanzen von Klassen erzeugen

- Klicken wir jetzt z.B. auf den Eintrag ***Kreis.new(),*** dann erscheint in der Instanzenliste das Kreisobjekt ***kreis 1.***

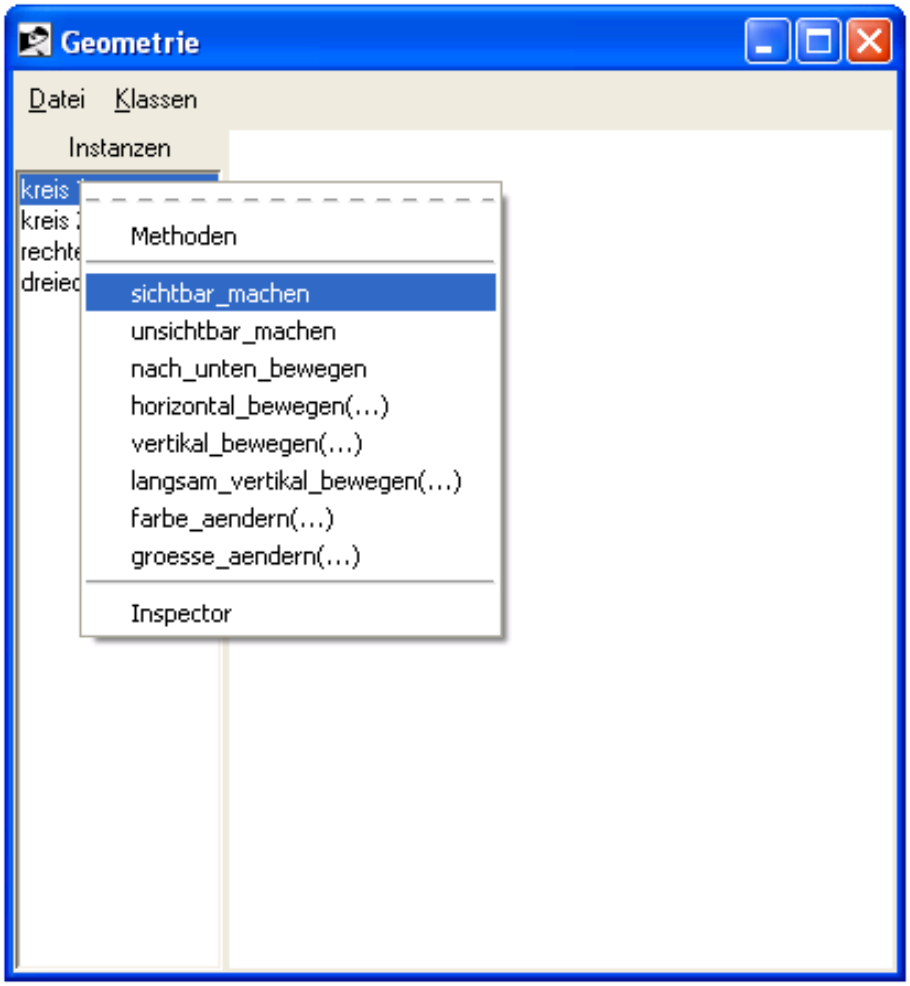
- **Konvention:**
  - **Klassennamen beginnen immer mit einem großen Buchstaben**
  - **Objektnamen beginnen immer mit einem kleinen Buchstaben**

- Wir fahren fort und erzeugen einen weiteren Kreis, ein Rechteck und ein Dreieck



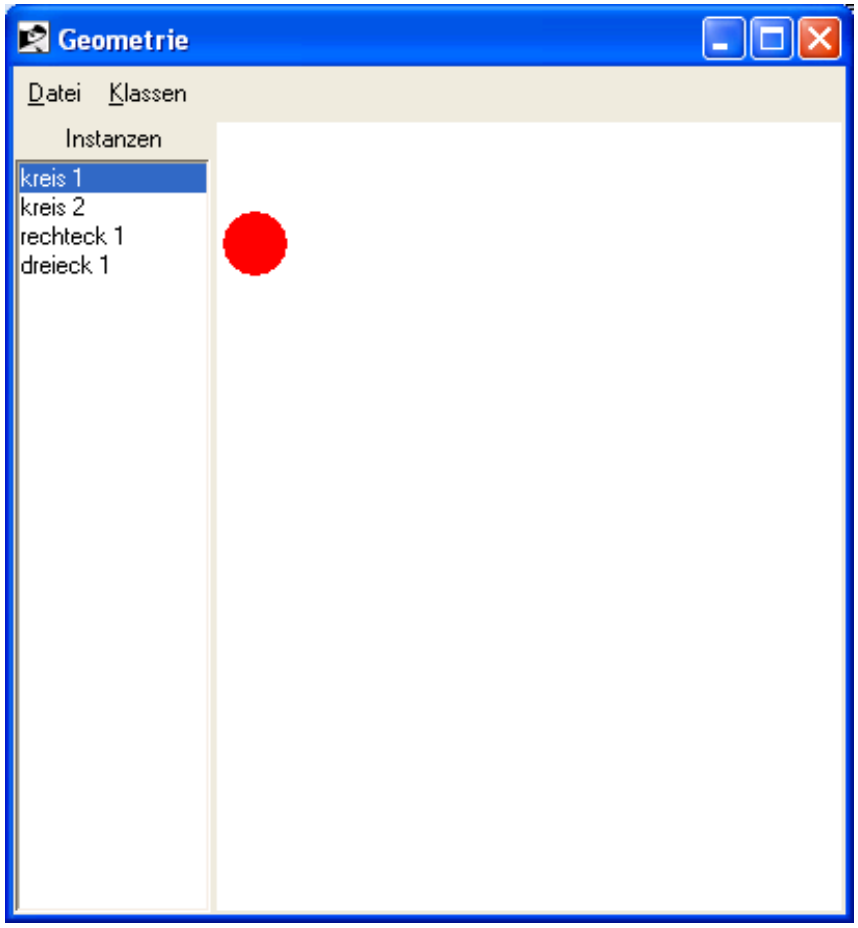
Methoden auf Instanzen aufrufen

- Wir aktivieren mit einem Rechtsklick das Kontextmenü für Instanzen.
- Jetzt erscheinen eine Reihe von „Funktionen“, die typisch für geometrische Figuren sind.
- Diese Funktionen heißen in der objektorientierten Sprache auch **Methoden**.
- Als erstes machen wir ***kreis1*** sichtbar, indem wir die Methode ***sichtbar_machen*** aufrufen.



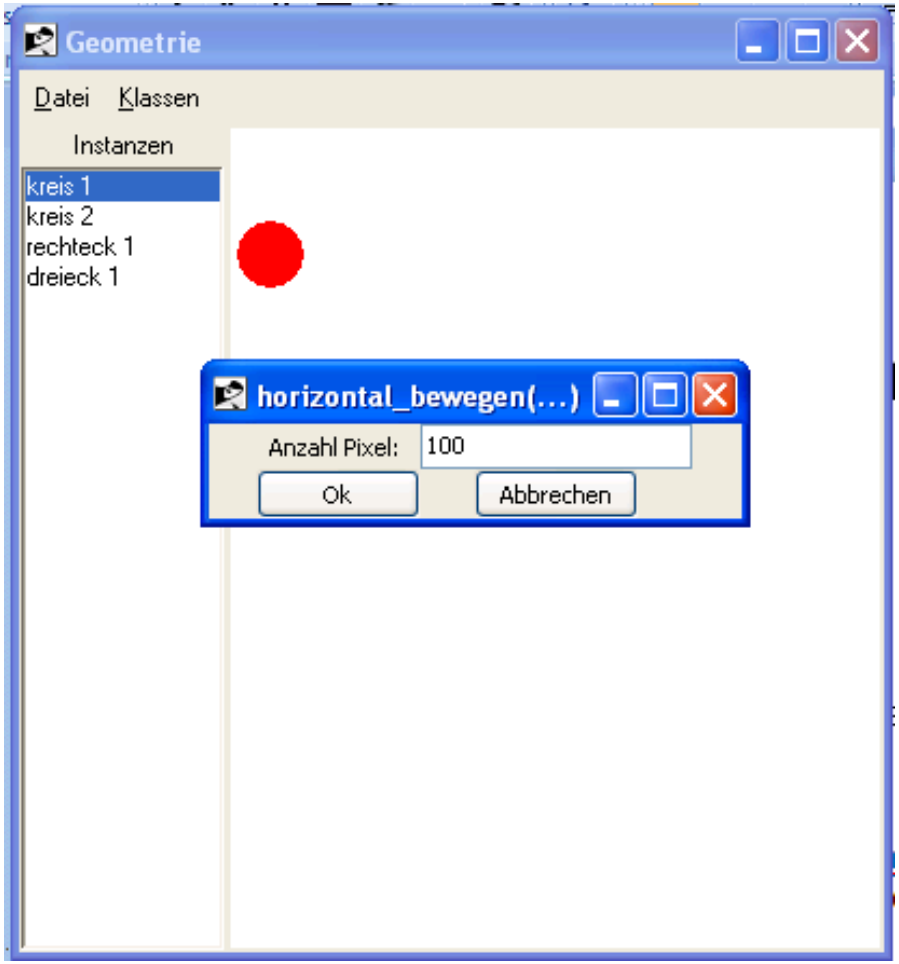
Methoden auf Instanzen aufrufen

- Die Methode ***sichtbar_machen*** stellt ***kreis1*** als roten Kreis auf der rechten Zeichenfläche dar.
- ***kreis1*** können wir jetzt mit den Methoden ***nach_unten_bewegen*** und dann ***unsichtbar_machen.***
- **Frage:** Was passiert, wenn Sie den unsichtbaren ***kreis1*** mehrfach nach unten bewegen und ihn dann darstellen?
- **Frage:** Was passiert, wenn Sie ***sichtbar_machen*** 2-mal nacheinander aufrufen



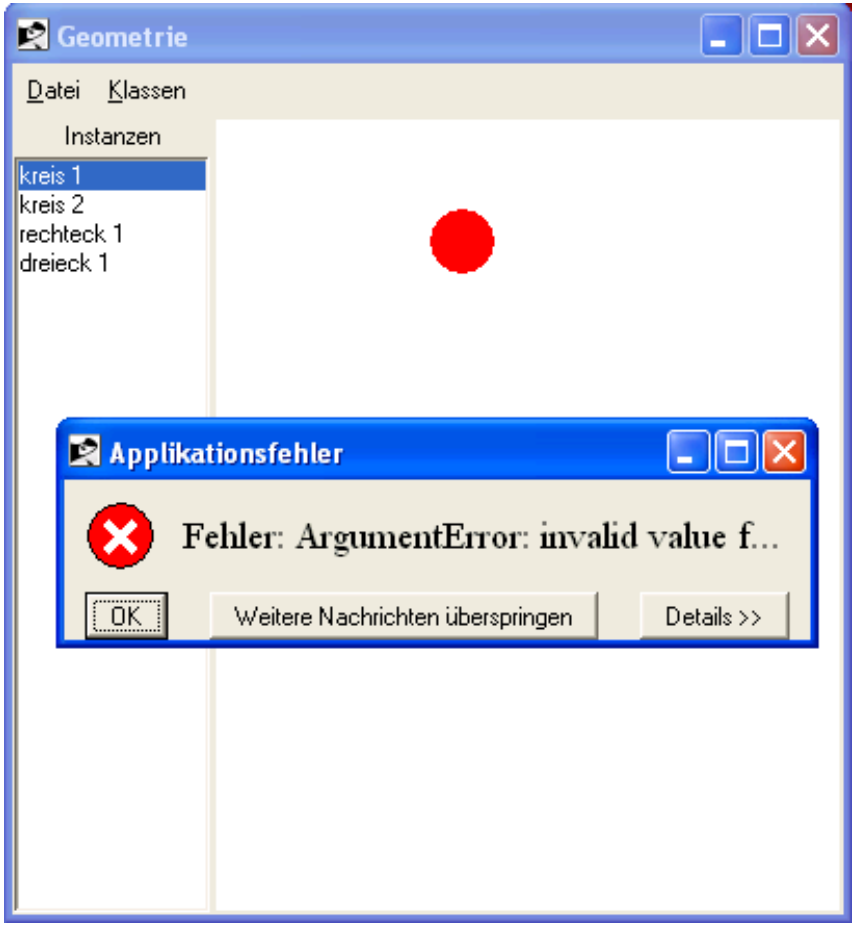
Methoden mit Parametern aufrufen

- Wir rufen nun die Methode ***horizontal_bewegen*** auf ***kreis1*** auf. Es erscheint ein Dialog, der eine Eingabe fordert.
- Die Methode ***horizontal_bewegen*** benötigt zusätzliche Information, um ihre Aufgabe erfüllen zu können.
- Die zusätzliche Informationen, die wir beim Aufruf der Methoden übergeben, heißen **Parameter** einer Methode.
- Sie ist flexibler als z.B. eine Methode ***nach_rechts_bewegen***, die Figuren nur um eine festgelegte Größe bewegt.
- **Demo:** Bevor wir weiter machen, werden wir ein wenig mit den Methoden ***vertikal_bewegen, langsam_vertikal_bewegen*** und ***groesse_aendern*** experimentieren.



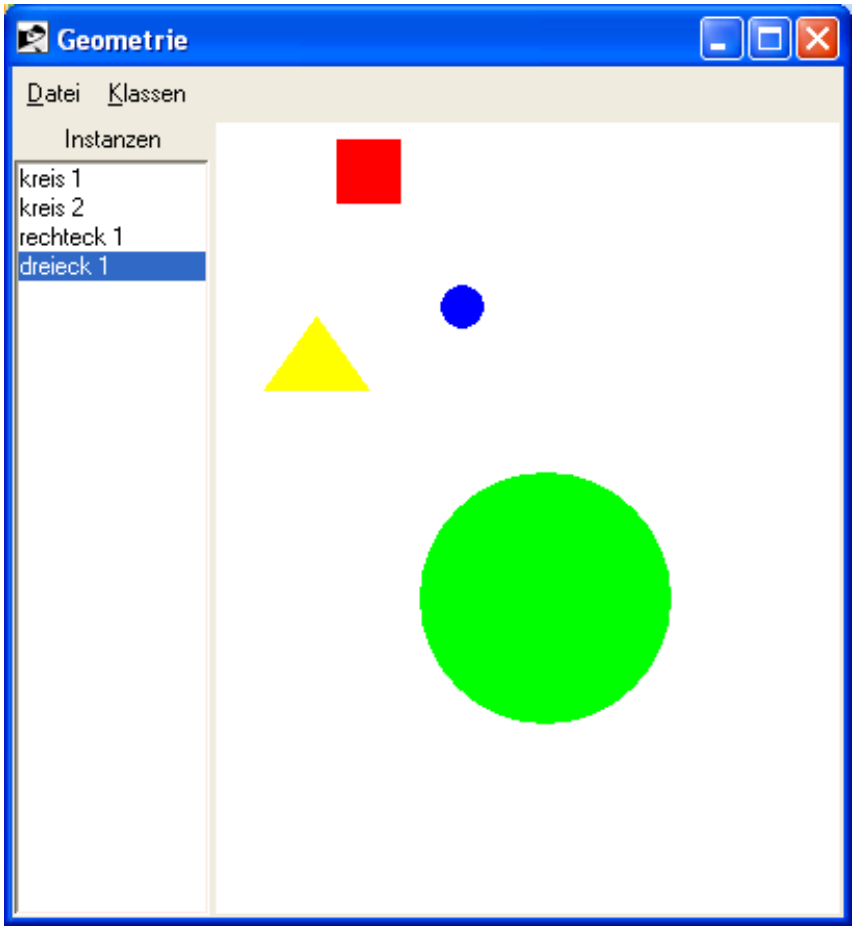
Datentypen

- Objekte haben einen Datentyp.
- Zunächst können wir als Faustregel festhalten, dass die Klasse der Datentyp eines Objektes ist.
- ***Wichtig:*** Der Datentyp bestimmt, welche Methoden auf dem Objekt aufgerufen werden können und welche Werte zulässig sind.
- Übergeben wir bspw. in der Methode ***horizontal_bewegen*** eine Zeichenkette ***"abc"***, dann erscheint eine Fehlermeldung, die besagt, dass aus dieser Zeichenkette keine Zahl entstehen kann.



Eine Klasse viele Instanzen

- **Konzept:** Von einer Klasse können viele gleichartige Instanzen erzeugt werden.
- **Übung**: Wir erzeugen mehrere Kreis-, Rechteck- und Dreiecksinstanzen, machen diese sichtbar, bewegen diese und ändern die Farbe. Wir machen eine Kreis groß und grün, den anderen klein und blau.
- **Hinweis**: Die Methode ***groesse_aendern*** eines Dreiecks benötigt 2 statt 1 Parameter. Generell kann eine Methode beliebig viele Parameter haben.



Zustand

- Die Menge der momentanen Werte der Attribute, die ein Objekt definieren (wie z.B. die Position, Farbe, Radius und die Sichtbarkeit eines Kreises) wird auch als **Zustand** eines Objekte bezeichnet.

- Der Zustand eines Objektes wird in den **Instanzvariablen** gespeichert.

- **Demo**: Wir können den Zustand eines Objektes untersuchen, indem wir im Kontextmenü den **Inspektor** aufrufen.

- Der Inspektor zeigt alle Instanzvariablen eines Objektes und deren momentanen Werte.

Das Innenleben eines Objektes

Instanzvariablen

- Instanzen einer Klasse, haben alle die gleichen **Instanzvariablen**: Anzahl und Namen der Variablen sind gleich.
- Die **Werte** der Instanzvariablen unterscheiden sich.
- Anzahl und Namen der Instanzvariablen werden durch die Klasse festgelegt.
- Werden Objekte einer Klasse erzeugt, so haben diese automatisch diese Instanz-variablen.
- Die Werte der Instanzvariablen werden in den Objekten gespeichert.

Methoden

- Methoden werden in der Klasse des Objektes **definiert**. Alle **Instanzen** einer Klasse **haben die gleichen Methoden**.
- Die Methoden werden **auf konkreten Instanzen aufgerufen**. So ist eindeutig, welches Objekt seinen Zustand ändern muss, wenn z.B. die Methode ***farbe_aendern*** aufgerufen wird.

Übung

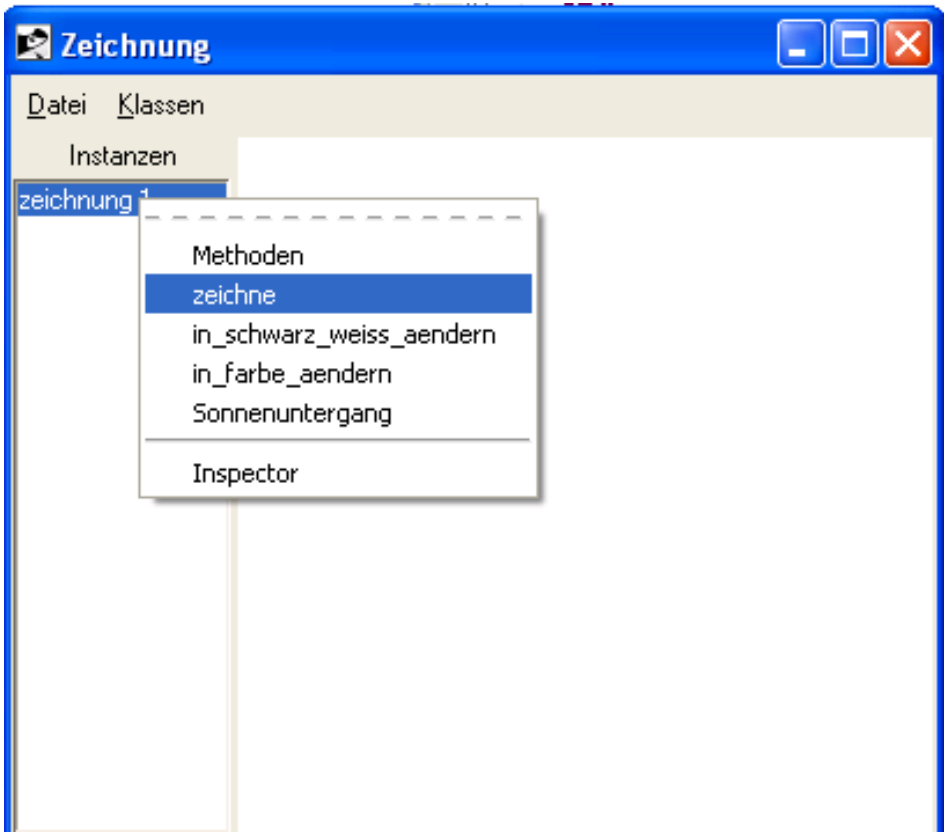
- Wir benutzen die Figuren aus dem Projekt ***v1-Geometrie***, um ein Bild mit einem Haus und einer Sonne zu zeichnen. Es sollte mindestens ein Fenster berücksichtigt werden.

- Wie gehen wir vor? Wie notieren wir die Schritte?

- Gibt es Alternativen zu dem ersten Vorgehen?

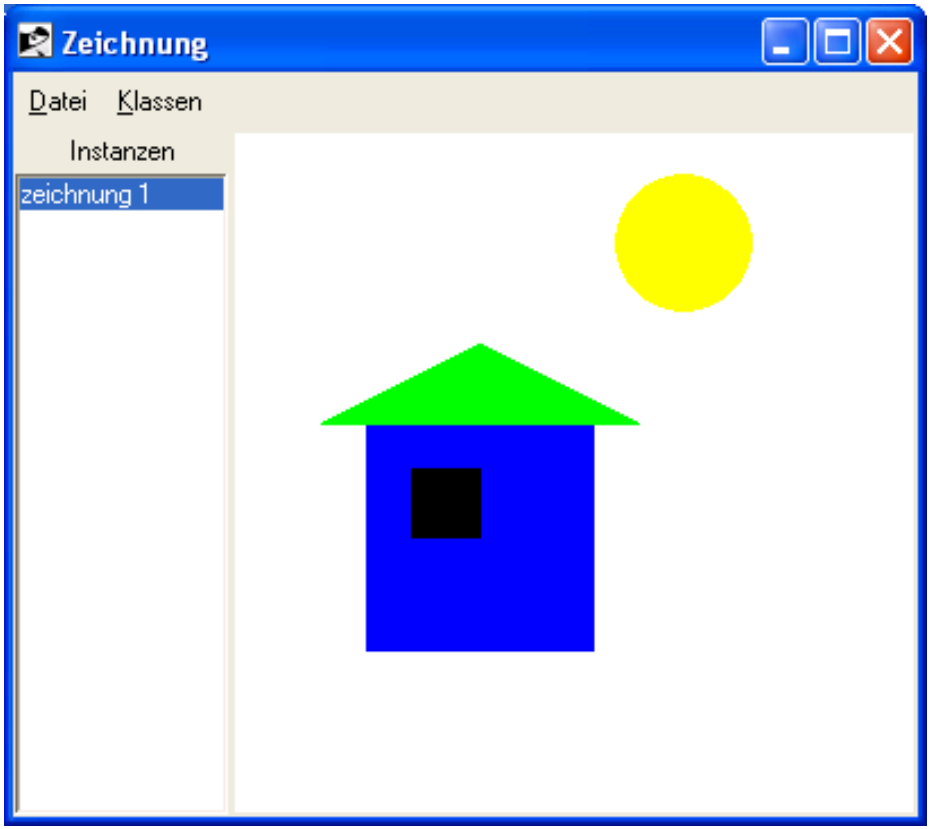
Objektinteraktion

- Wir öffnen das Projekt ***Zeichnung*** und starten die ***Toolbox***
- Wir erzeugen eine Instanz von ***Zeichnung*** und lassen die Zeichnung anzeigen.
- Wie arbeitet ein Objekt der Klasse ***Zeichnung***?
- Wenn wir in der Praxis eine Folge von Aufgaben zu erledigen haben, erzeugen wir eine Klasse, die genau diese Abfolge programmiert.



Objektinteraktion

- Das Objekt ***zeichnung1*** erzeugt 2 Rechteck-Instanzen, 1 Dreieck und einen Kreis.
- Es bewegt die Figureninstanzen, ändert deren Größe und Farbe, bis die Zeichnung rechts entsteht.
- ***Zeichnung*** enthält eine Konstruktionsvorschrift, einen **Algorithmus** für den Bau eines Hauses.
- **Entscheidend**: Dazu erzeugt ***Zeichnung*** andere Objekte und ruft deren Methoden auf.
- **Konzept**: Objekte können miteinander kommunizieren, indem sie gegenseitig ihre Methoden aufrufen



Quelltext (Source Code)

- Jede Klasse hat einen Quelltext, dieser enthält die Implementierung der Klasse.
- Der Quelltext einer Klasse legt die Struktur und das Verhalten (die Instanzvariablen und die Methoden) aller Instanzen einer Klasse fest.
- Die Implementierung ist in unserem Fall in der Programmiersprache Ruby formuliert.
- Die Kunst der objektorientierten Programmierung ist die Definition von Klassen.
- Um die Implementierung einer Klasse zu inspizieren, verwenden wir die Eclipse Umgebung und deren Editoren

```
class Zeichnung
  def initialize()
  ...
  end
  def sichtbar?()
      ...
  end
  def zeichne()
   ...
  end
  def in_schwarz_weiss_aendern()
    ...
  end
  def in_farbe_aendern()
    ...
  end
  def to_s
    return "zeichnung #@nummer"
  end
 end
```

Übung

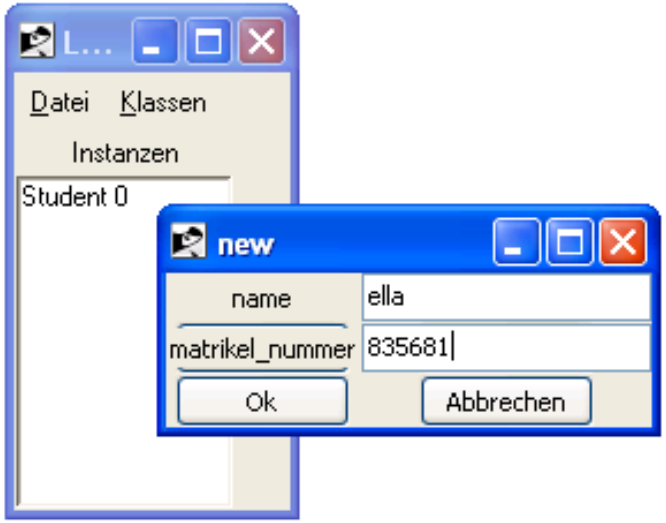
- Ändern Sie bitte die Implementierung der Klasse ***Zeichnung*** so, dass die Sonne rot wird.

- Die Zeichnung soll eine zweite Sonne erhalten. Dazu müssen die Instanz-variablen ergänzt werden und die Anweisungen für die zweite Sonne implementiert werden.

- Implementieren Sie bitte für die Zeichnung mit nur einer Sonne den Sonnenuntergang: Methode ***sonnenuntergang***.

Laborkursbeispiel

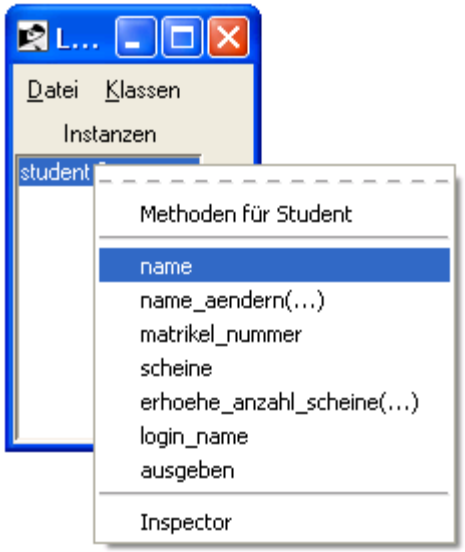
- Wir öffnen das Projekt ***v1-Laborkurs***.
- Wir erzeugen ein Objekt ***Student***.
- Jetzt werden wir nach Namen und Matrikelnummer bei der Erzeugung gefragt.
- Auch Objekterzeugung ist ein Methodenaufruf und kann Parameter haben.

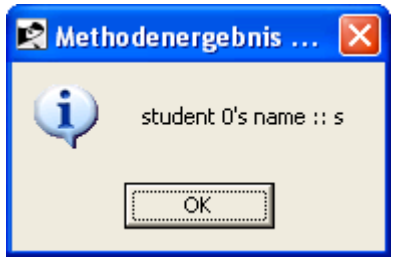


Aufrufergebnisse

- Wir rufen auf dem Studenten ***student1*** die Methode ***name*** auf.
- Eine Dialogbox zeigt das Ergebnis des Methodenaufrufs.

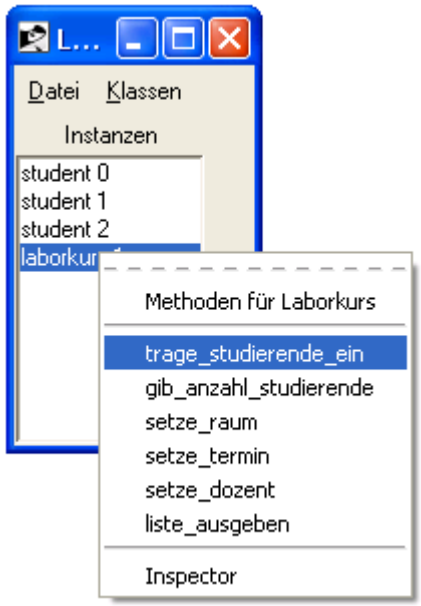
➔ Methodenaufrufe können Informationen über ein Objekt über einen Ergebniswert liefern.





Objekte als Parameter

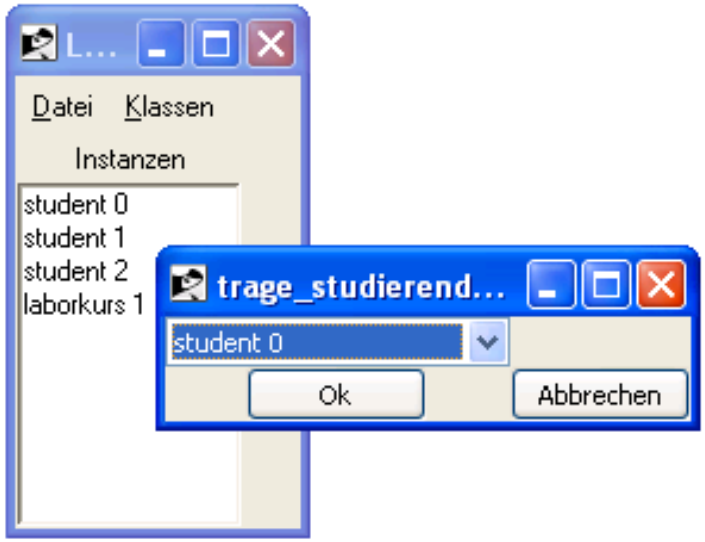
- Wir erzeugen einen ***Laborkurs*** mit einer festen Anzahl von Plätzen.
- Die anderen Instanzvariablen des Laborkurses ***laborkurs1*** setzen wir mit speziellen ***setze***-Methoden
- Wir erzeugen weitere Studierende.
- Wir melden Studierende mit der Methode ***trage_studierende_ein*** zum Kurs an.



Objekte als Parameter

- **Beobachtung**: Es ist möglich Objekte selbst definierter Klassen als Parameter zu übergeben.

- **Übung:**
  - Das Eintragen von Studierenden wiederholen wir mehrfach, bis der Kurs überbucht ist.
  - Wir inspizieren den Laborkurs.
  - Wir geben den Laborkurs auf der Konsole aus.



Zusammenfassung der Konzepte

- **Objekt**: Ruby Objekte modellieren Objekte eines Anwendungsbereiches.
- **Klasse**: Objekte werden durch Klassen erzeugt. Objekte repräsentieren individuelle Instanzen einer Klasse
- **Methode**: Wir können mit Objekten kommunizieren, indem wir ihre Methoden aufrufen.
- **Parameter**: Methoden können Parameter haben, mit denen zusätzliche Information für eine Aufgabe angegeben werden
- **Datentyp**: Objekte haben einen Datentyp. Fürs Erste: Die Klasse bestimmt den Datentyp. Der Datentyp legt fest, welche Methoden auf Objekten aufgerufen werden können.

Zusammenfassung

- **Eine Klasse, viele Instanzen**: Von einer Klasse können beliebig viele gleichartige Instanzen erzeugt werden.
- **Zustand**: Objekte haben einen Zustand. Dieser Zustand wird durch Werte repräsentiert, die in den Instanzvariablen gehalten wird. Der Zustand kann durch Methodenaufrufe modifiziert werden.
- **Methodenaufrufe**: Objekte können untereinander kommunizieren, indem sie gegenseitig Methoden aufrufen.
- **Quelltext**: Der Quelltext einer Klasse legt die Struktur und das Verhalten aller Instanzen einer Klasse fest.
- **Ergebnis**: Methoden können Informationen über ein Objekt durch einen Ergebniswert zurückliefern.