

SE1, Aufgabenblatt 3

Softwareentwicklung I – Wintersemester 2014/15

Erster Kontakt mit Klassen und Exemplaren, Quelltext bearbeiten, Methoden, Variablen, Parameter

MIN-CommSy-URL: <https://www.mincommsy.uni-hamburg.de/>

Projektraum: SE1 CommSy WiSe 14/15

Ausgabedatum: 30. Oktober 2014

Kernbegriffe

Objekte (engl.: objects) sind *Exemplare* (engl.: instances) von *Klassen*. Eine Klasse (engl.: class) definiert für ihre Exemplare, welche *Methoden* (engl.: methods) an ihnen aufrufbar sind. Von einer Klasse können für gewöhnlich beliebig viele Exemplare erzeugt werden. Wir benutzen Exemplare, indem wir ihre Methoden aufrufen. Die Methoden, die an den Exemplaren einer Klasse aufrufbar sind, bilden die *Schnittstelle* (engl.: interface) der Klasse.

Methoden können *Parameter* (engl.: parameters) haben, mit denen zusätzliche Informationen für eine Aufgabe angegeben werden. Ein Parameter hat einen Namen und einen *Typ* (engl.: type) der festlegt, von welcher Art der Parameter ist. Beispiele für Typen sind die Ganzen Zahlen (in Java u.a. `int`), Zeichenketten (in Java `String`) und Wahrheitswerte (in Java `boolean`).

Ein Exemplar hat einen internen Zustand, welcher sich in den Belegungen seiner *Felder* (engl.: fields) widerspiegelt und von außen nicht unmittelbar einsehbar ist. Er kann aber über Methoden abgefragt und verändert werden. (Eine Besonderheit von BlueJ: der Zustand interaktiv erzeugter Exemplare lässt sich auch direkt anzeigen.) Welche Felder ein Exemplar hat, legt seine Klasse fest. Jedes Exemplar einer Klasse hat jeweils seinen eigenen, von anderen Exemplaren unabhängigen (konkreten) Zustand.

Den Programmtext einer Klasse nennen wir ihren *Quelltext* (engl.: source code) bzw. ihre *Klassendefinition* (engl.: class definition). Wir bearbeiten Quelltexte mit einem *Editor*. Um Objekte einer Klasse erzeugen zu können, muss der Quelltext der Klasse zuvor *übersetzt* (kompiliert, engl.: to compile) werden. Dabei wird der menschenlesbare Quelltext von einem *Compiler* in eine maschinenausführbare Form überführt.

Variablen (engl.: variables) sind Speicherplätze für Werte. Jede Variable in Java muss *deklariert* werden, durch Angabe eines Namens (*Bezeichner*, engl.: identifier) und eines Typs. In Java gibt es *primitive Typen* wie `int`, `float` und `boolean`, aber auch *Objekttypen* (wie `String` oder `Kreis`). *Lokale Variablen* werden innerhalb einer Methode deklariert. Eine lokale Variable existiert nur während der Ausführung der Methode und ist nur innerhalb der Methode zugreifbar. Eine lokale Variable muss initialisiert werden, bevor man lesend auf sie zugreifen darf.

Im Quelltext einer Klasse ist unter anderem beschrieben, wie die Methoden realisiert sind, d.h. was passieren soll, wenn eine Methode aufgerufen wird. Eine Methode besteht aus einem *Kopf* (engl.: header), welcher ihre Signatur definiert, und einem *Rumpf* (engl.: body). Der Rumpf enthält eine *Sequenz* von *Anweisungen* (engl.: statements) und *Deklarationen* (engl.: declarations). Anweisungen definieren die eigentlichen Aktionen eines Programms.

In Java wird im Kopf einer Methode auch ihre Aufrufbarkeit festgelegt. Die mit `public` deklarierten *öffentlichen Methoden* können als Dienstleistungen betrachtet werden, die durch *Klienten* aufrufbar sind, beispielsweise von Objekten anderer Klassen. Diese Methoden bilden die Schnittstelle, die in BlueJ auch interaktiv aufrufbar ist. Die mit `private` deklarierten *privaten Methoden* hingegen sind nur innerhalb der definierenden Klasse aufrufbar.

Die Signatur einer Methode legt u.a. ihre *Parameter* fest (genauer: deren Anzahl, Reihenfolge und jeweiligen Typ). Wir unterscheiden *formale Parameter* (engl.: formal parameters) und *aktuelle Parameter* (inzwischen etablierte, aber falsche Übersetzung des engl.: actual parameters). Formale Parameter sind Variablen, die im Methodenkopf deklariert werden und im Rumpf sichtbar sind. Die Anfangsbelegung eines formalen Parameters wird durch den jeweiligen aktuellen Parameter an der Aufrufstelle bestimmt. Wir sagen auch, dass bei einem Methodenaufruf die aktuellen Parameter des Aufrufers an die formalen Parameter der aufgerufenen Methode *gebunden* werden.

Als aktuelle Parameter können häufig *Literale* verwendet werden. Ein Literal ist eine Zeichenfolge im Quelltext (wie `42` oder `"gelb"`), die einen Wert repräsentiert und deren Struktur dem Compiler bekannt ist.

In einem objektorientierten System werden alle Aktionen durch Methodenaufrufe ausgelöst. Die Methoden eines Objektes werden üblicherweise in der *Punktnotation* (engl.: dot notation) aufgerufen: Zuerst wird das aufzurufende Objekt benannt, dann, getrennt durch einen Punkt, die aufzurufende Methode. Dann folgen innerhalb von runden Klammern die aktuellen Parameter. Falls eine Methode keine Parameter definiert, müssen in Java die runden Klammern trotzdem angegeben werden. Innerhalb einer Klassendefinition können die Methoden der eigenen Klasse auch direkt aufgerufen werden (ohne Objektname und Punkt).

Lernziele

BlueJ starten können; eigenes Verzeichnis zum Ablegen von Projekten kennen; Objekte interaktiv erzeugen können; Objekte interaktiv manipulieren können; einfache Parameter interaktiv übergeben können; Verhältnis zwischen Klasse und Objekt erklären können; Verhältnis zwischen Methoden und Schnittstelle erklären können.

Klassendefinition (Java-Quelltext) verstehen, verändern und übersetzen können; Fehlermeldungen des Compilers verstehen; sequenzielle Ausführung verstehen; Methodenaufrufe programmieren können; Parameter und lokale Variablen verstehen und anwenden können; Teile einer Methode in eine neue Methode auslagern können.

Aufgabe 3.1 Exemplare erzeugen und Methoden aufrufen



Erstellt in eurem *Home*-Verzeichnis (unter Windows als Laufwerk „Z:“ eingebunden) einen neuen Ordner SE1. Auf diesen könnt ihr von beliebigen Rechnern im Informatikum unter eurer RZ-Kennung zugreifen.

Öffnet einen Internet-Browser und besucht den SE1 Projektraum im MinCommSy. Die URL findet ihr im Kopf des Aufgabenblattes. Ladet die ZIP-Datei *Blatt03_Zeichnung.zip* in das eben angelegte Verzeichnis SE1 herunter. Ihr findet die Datei im CommSy unter „Materialien“ im Eintrag „Aufgabenblatt 03“.


Startet die Entwicklungsumgebung BlueJ. Unter Windows findet ihr den Programmeintrag im Start-Menü, unter Linux tippt ihr in einem Terminal „bluej&“ ein und drückt die Enter-Taste. Öffnet nun die ZIP-Datei, indem ihr den Menü-Eintrag „Projekt“ -> „Projekt öffnen“ auswählt. Ihr solltet nun ein Diagramm mit den fünf Klassen `Leinwand`, `Kreis`, `Dreieck`, `Quadrat` und `Zeichner` sehen.

- 3.1.1 Klassen können als *Baupläne* für die Erstellung von Exemplaren bezeichnet werden. Erzeugt interaktiv ein Exemplar von `Kreis`, indem ihr im Kontextmenü der Klasse `Kreis`, zu erreichen über die rechte Maustaste, den Eintrag `new Kreis()` auswählt (und den vorgegebenen Namen erst einmal akzeptiert).
Hinweis: Falls dies bei euch nicht klappt, müsst ihr wahrscheinlich einmal auf den Button „Übersetzen“ klicken. Danach sollte es funktionieren.
- 3.1.2 Unten auf der Objektleiste erscheint das gerade erzeugte Exemplar. Ruft die Methode `sichtbarMachen` für diesen Kreis auf, indem ihr im Kontextmenü den entsprechenden Eintrag auswählt. Wenn ihr das tut, wird die Methode `sichtbarMachen`, die in der Klasse definiert ist, auf diesem Kreisexemplar ausgeführt. Es sollte ein Fenster erscheinen (eventuell müsst ihr es in den Vordergrund holen), in welchem der Kreis sichtbar ist.
- 3.1.3 Führt weitere Methoden an diesem Kreis aus: verschiebt ihn z.B. oder ändert seine Farbe. Auf diese Weise ändert ihr den Zustand des Kreises. Es gibt Methoden, die aufgerufen werden können, ohne weitere Informationen zu benötigen. Andere Methoden brauchen zusätzliche Angaben als Parameter. Achtung: Zeichenketten (Typ `String`) müssen von doppelten Anführungszeichen umschlossen sein.
Beispiel: `"Hallo"`
- 3.1.4 Erzeugt mehrere grafische Objekte und zeichnet auf diese Weise ein Bild. Erklärt den Betreuern mit den Begriffen aus dem Einleitungstext, wie ihr dabei vorgegangen seid.
- 3.1.5 Vergleicht die Methoden der verschiedenen Figurentypen. Was sind die Gemeinsamkeiten, was die Unterschiede?
- 3.1.6 Was gehört neben der Farbe noch zum Kreiszustand? Verwendet die *Inspect*-Funktion von BlueJ (Doppelklick auf das Exemplar), um die Felder des Kreises anzuzeigen.


Aufgabe 3.2 Erster Kontakt mit Java-Quelltext

- 3.2.1 Erzeugt einen `Zeichner` und ruft die Methode `zeichne` daran auf.
- 3.2.2 Jetzt wird es ernst: Öffnet den Quelltext der Klasse `Zeichner` durch einen Doppelklick auf die Klasse. Im Editor könnt ihr die Implementierung der Methode `public void zeichne()` erkennen. Erklärt eurem Betreuer mit den Begriffen aus dem Einleitungstext, was ihr dort seht.
-  3.2.3 Nehmt ein beliebiges Semikolon weg und übersetzt die Klasse. Warum tritt ein Fehler auf, und wie wird er angezeigt? **Schreibt dies mit wenigen Worten nieder.**
- 3.2.4 Verändert die Implementierung von `zeichne`, so dass ein anderes Bild gezeichnet wird. Dazu könnt ihr die Methoden verwenden, die ihr bisher interaktiv aufgerufen habt. Hinweis: Nach jeder Veränderung muss der Quelltext neu übersetzt werden; dabei werden die alten Exemplare gelöscht, weil die „Blaupause“ zum Erzeugen neuer Exemplare geändert wurde.
-  3.2.5 Was ist eine Variablendeklaration, und aus welchen Teilen besteht sie? **Erklärt dies schriftlich** am Beispiel `String strasse;`.

Aufgabe 3.3 Verändern der Schnittstelle des Zeichners und Verbessern der internen Struktur

- 3.3.1 Öffnet das Projekt *Zeichnung* erneut und speichert eine Kopie als *Zeichnung2*. Arbeitet von nun an mit dem Projekt *Zeichnung2*.
- 3.3.2 Die Farbe der Hauswand soll nun interaktiv bei einem Methodenaufruf angegeben werden können. Hierzu müsst ihr in der Methode `zeichne` einen Parameter einbauen, der den übergebenen Farbnamen speichert und den ihr an passender Stelle für das Setzen der Wandfarbe verwendet.
Ein Beispiel für eine Methode mit Parameter ist `horizontalBewegen` in der Klasse `Kreis`. Wer möchte, kann zusätzliche Parameter definieren, etwa für die Fensterfarbe.
- 3.3.3 Ihr habt durch Einführung des formalen Parameters die Schnittstelle der Klasse `Zeichner` verändert. Dokumentiert dies, indem ihr *Schnittstellenkommentare* schreibt. Beispiele für dokumentierte Methoden mit Parametern findet ihr in der Klasse `Kreis`. Werden eure Kommentare bei interaktiven Aufrufen angezeigt?
- 3.3.4 Die Methode `zeichne` ist inzwischen vermutlich recht unübersichtlich und lang geworden. Sie soll jetzt übersichtlicher werden, indem ihr die Funktionalität auf weitere Methoden aufteilt. Definiert dazu Methoden wie `zeichneDach` und/oder `zeichneWand` und ruft diese von der zentralen `zeichne`-Methode aus auf. Ist es nötig, diese Methoden als `public` zu deklarieren? Wenn ja, warum? Wenn nein, weshalb wäre es dann sinnvoll, sie als `private` zu deklarieren?
-  3.3.5 Was ist der Unterschied zwischen formalen und aktuellen Parametern? **Erläutert ihn schriftlich** mit euren eigenen Worten.

Aufgabe 3.4 Unterschiede zwischen Schnittstelle und Implementation

- 3.4.1 Experimentiert mit der Entwicklungsumgebung BlueJ. Wie kann man z.B. im Editor die Zeilennummern anzeigen lassen? Was bedeutet es, wenn manche Klasse gestreift erscheinen? Wie lassen sich alle Klassen auf einmal übersetzen? Was ist der Unterschied zwischen „Übersetzen“ und „Paket neu übersetzen“?
- 3.4.2 Im Editorfenster von BlueJ könnt ihr zwischen „*Implementierung* bzw. *Quelltext*“ und „*Schnittstelle* bzw. *Dokumentation*“ wählen (Auswahlbox rechts oben). Die Schnittstellensicht wird von einem Programm namens *javadoc* erzeugt, das den Quelltext analysiert und die relevanten Informationen extrahiert und in HTML-Darstellung aufbereitet. Javadoc ist Bestandteil des JDK (Java Development Kit) und nichts BlueJ-spezifisches. Welche Elemente der Klasse werden im Schnittstellen-Modus angezeigt? Was ist verborgen?
-  3.4.3 **Schreibt auf**, wann man die Schnittstellenansicht benötigt und warum sie hilfreich ist.