

PR1, Aufgabenblatt 1

Programmieren 1 – Wintersemester 2021/22

Erster Kontakt mit Klassen und Exemplaren; Quelltext, Methoden, Parameter

Ausgabedatum: 11. Oktober 2021

Allgemeines vorab: Aufgabenblätter und Kernbegriffe

Für das Praktikum zu PR1 wirst du im Laufe dieses Semesters sieben Aufgabenblätter bearbeiten. Auf jedem Aufgabenblatt sind eingangs die **Lernziele** genannt, die wir mit den Aufgaben erreichen wollen. Dies kann dir helfen, explizit festzustellen, ob die Lernziele für dich persönlich erreicht wurden.

Neben den Aufgaben stellen wir auf jedem Blatt die **Kernbegriffe** in einem zusammenhängenden Text zusammen, die auf dem jeweiligen Blatt behandelt werden. Wie sollst du mit den Kernbegriffen umgehen?

- Vor Bearbeiten der zugehörigen Aufgaben: zumindest lesen.
- Nach dem Bearbeiten: verstanden haben, für alle folgenden Blätter!

Wir erwarten **nicht**, dass du den Text **vor** dem Bearbeiten der Aufgaben vollständig **verstanden** hast. Aber wir erwarten, dass du den Text vorab vollständig **gelesen** hast. Das kann etwas trocken sein, denn die Kernbegriffstexte sind keine Lehrbuchtexte; aber es wird dir helfen. Mache dir beim Lesen vorab klar, welche Begriffe du noch nicht verstanden hast, und versuche diese Begriffe mit Hilfe der Materialien zur Veranstaltung zu klären. Bearbeite dann die Aufgaben. Anschließend sollten die Kernbegriffe deutlich klarer sein.

Wenn wir feststellen, dass Kernbegriffe der vorherigen Blätter nicht gut verstanden sind, werden wir individuell das Pflegen eines **Glossars** fordern. Ein Glossar liefert wie ein Lexikon zu jedem Begriff einen Eintrag, der ihn erläutert.

Lernziele

Termin 1 – objektorientierte Grundlagen: 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.

Termin 2 – imperative Grundlagen: Klassendefinition (Java-Quelltext) verstehen, verändern und übersetzen können; Fehlermeldungen des Compilers verstehen; sequenzielle Ausführung in Methodenrümpfen verstehen; Methodenaufrufe programmieren können; Parameter und lokale Variablen verstehen und anwenden können; Teile einer Methode in eine neue Methode auslagern können.

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. Wir produzieren Exemplare durch *Exemplarerzeugungen* (engl.: instance creations) und benutzen sie, 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 in Java hat immer 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.

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) 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.

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 im Rumpf der Methode zugreifbar. Eine lokale Variable muss initialisiert werden, bevor man lesend auf sie zugreifen darf.

In Java wird im Kopf einer Methode 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 Aufrufbarkeit bestimmt also, *ob* ein Klient eine Methode aufrufen kann.

Wie eine Methode aufzurufen ist, wird über ihre *Signatur* (engl.: signature) beschrieben. In Java legt die Signatur einer Methode nur ihren *Namen* und ihre *Parametertypen* fest. Die Signatur einer Methode

```
boolean istAuszahlenMoeglich(int euro, int cent)
```

lautet demnach:

```
istAuszahlenMoeglich(int, int)
```

Die Signatur enthält also nicht den Ergebnistyp und für die Parameter nur deren Anzahl, Reihenfolge und jeweiligen Typ, aber nicht ihre Namen.

Wir differenzieren Parameter in *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 benutzbar 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).

Vorbereitung

Erstelle in einem Verzeichnis, auf das Du sowohl zuhause als auch an der Hochschule zugreifen kannst, ein eigenes Verzeichnis PR1 für das Praktikum.


Starte die Entwicklungsumgebung BlueJ, unter Windows findest du den Programmeintrag im Start-Menü. **Stelle als allererstes die Oberfläche von BlueJ auf Deutsch um**, falls nicht bereits geschehen: den Menü-Eintrag „Tools“ -> „Preferences...“ auswählen und im erscheinenden Fenster auf dem Reiter „Interface“ unter „Language selection“ auf „German“ umstellen, dann BlueJ einmal beenden und neu starten.

Lade aus dem pub-Verzeichnis zu PR1 die ZIP-Datei *Blatt01_Figuren.zip* aus dem Ordner „Aufgabenblatt01“ in das eben angelegte Verzeichnis herunter. Öffne die Datei, indem du den Menü-Eintrag „Projekt“ -> „Open ZIP/JAR“ auswählst. Du solltest nun ein Diagramm mit den vier Klassen `Kreis`, `Quadrat`, `Dreieck` und `Leinwand` sehen.

Aufgabe 1.1 Exemplare erzeugen und Methoden aufrufen (Termin 1)

- 1.1.1 Klassen können als *Baupläne* für die Erstellung von Exemplaren bezeichnet werden. Erzeuge interaktiv ein Exemplar von `Kreis`, indem du im Kontextmenü der Klasse `Kreis`, zu erreichen über die rechte Maustaste, den Eintrag `new Kreis()` auswählst (und den vorgegebenen Namen erst einmal akzeptierst).
Hinweis: Falls dies bei dir nicht klappt, musst du wahrscheinlich einmal auf den Button „Übersetzen“ klicken. Danach sollte es funktionieren.
- 1.1.2 Unten auf der Objektleiste erscheint das gerade erzeugte Exemplar. Ruf nun die Methode `sichtbarMachen` für diesen Kreis auf, indem du im Kontextmenü den entsprechenden Eintrag auswählst. Wenn du das tust, wird die Methode `sichtbarMachen`, die in der Klasse definiert ist, von diesem Kreisexemplar ausgeführt. Es sollte ein Fenster erscheinen (das immer im Vordergrund bleibt): das ist die „Leinwand“, auf der sich der Kreis gezeichnet hat.
- 1.1.3 Führe weitere Methoden an diesem Kreis aus: verschiebe ihn z.B. oder ändere seine Farbe. Auf diese Weise änderst du 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`) mit doppelten Anführungszeichen umschließen! Beispiel: `"Hallo"`
- 1.1.4 Erzeuge mehrere grafische Objekte und zeichnet auf diese Weise ein Bild. Erkläre den Betreuern mit den Begriffen aus dem Einleitungstext, wie du dabei vorgegangen bist. (Hinweis: Wenn du komplett von vorne beginnen willst, brauchst du nur auf die Schaltfläche mit dem gebogenen Pfeil unten rechts im BlueJ-Fenster zu klicken: Die Leinwand und alle erzeugten Objekte verschwinden dann.)
- 1.1.5 Vergleiche die Methoden der verschiedenen Figurentypen. Was sind die Gemeinsamkeiten, was die Unterschiede?
- 1.1.6 Was gehört neben der Farbe noch zum Kreiszustand? Verwende die *Inspect*-Funktion von BlueJ (Doppelklick auf das Exemplar), um die Felder des Kreises anzuzeigen. Was unterscheidet zwei Objekte derselben Klasse?

Aufgabe 1.2 Erster Kontakt mit Quelltext (Termin 2)

- 1.2.1 Öffne das Projekt *Blatt01_Zeichnung*. Erzeuge einen `Zeichner` und rufe die Methode `zeichne` daran auf.
- 1.2.2 Jetzt wird es ernst: Öffne den Quelltext der Klasse `Zeichner` durch einen Doppelklick auf die Klasse. Im Editor kannst du die Implementation der Methode `public void zeichne()` erkennen. Erkläre bei der Abnahme mit Hilfe der Kernbegriffe, was du dort siehst.
- 1.2.3 Verändere die Implementation von `zeichne`, so dass ein anderes Bild gezeichnet wird. Dazu kannst du die Methoden verwenden, die du bisher interaktiv aufgerufen hast. 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.
-  1.2.4 Was ist eine Variablendeklaration, und aus welchen Teilen besteht sie? **Erkläre dies schriftlich** am Beispiel `String strasse;`.

Aufgabe 1.3 Verändern der Schnittstelle des Zeichners und Verbessern der internen Struktur (Termin 2)

1.3.1 Öffne das Projekt *Zeichnung* erneut und speichere eine Kopie als *Zeichnung2*. Arbeite von nun an mit dem Projekt *Zeichnung2*.

1.3.2 Die Farbe der Hauswand soll nun interaktiv beim Methodenaufruf von `zeichne` angegeben werden können. Hierzu musst du 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 `farbeAendern` in der Klasse `Kreis`. Weitere Beispiele findest du in den Kernbegriffen. Wer möchte, kann zusätzliche Parameter definieren, etwa für die Fensterfarbe.

1.3.3 Du hast durch Einführung des formalen Parameters die Schnittstelle der Klasse `Zeichner` verändert. Dokumentiere dies, indem du *Schnittstellenkommentare* schreibst. Beispiele für dokumentierte Methoden mit Parametern findest du in der Klasse `Kreis`. Werden deine Kommentare bei interaktiven Aufrufen angezeigt?

1.3.4 Die Methode `zeichne` ist recht unübersichtlich und lang. Sie soll jetzt übersichtlicher werden, indem du die Funktionalität auf weitere Methoden aufteilst. Definiere dazu Methoden wie `zeichneDach` und/oder `zeichneWand` und rufe 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?



1.3.5 Was ist der Unterschied zwischen formalen und aktuellen Parametern? **Erläutere ihn schriftlich** mit deinen eigenen Worten.



1.3.6 *Zusatzaufgabe:* Was versteht man unter dem Kopf einer Methode? Was ist die Signatur einer Methode? Worin besteht der Unterschied? Wofür benötigt man die Signatur einer Methode? **Schriftlich.**

Zusatzaufgabe 1.4 Redundanten Quelltext vermeiden (Termin 2)

1.4.1 Schreib eine **parameterlose** Methode `zeichneHaeuser`, die (neben der Sonne, eventuell einem Himmel etc.) drei oder mehr Häuser gleicher „Architektur“ mit unterschiedlichen Wandfarben und an verschiedenen Stellen zeichnet. Versuche durch eine Aufteilung in kleinere Methoden und die Übergabe von Parametern Arbeit zu sparen und Wiederholungen im Quelltext zu reduzieren.