SE1, Aufgabenblatt 6

Softwareentwicklung I – Wintersemester 2011/12

Schleifen, Vertiefung syntaktische Strukturen

MIN-CommSy-URL: ..... https://www.mincommsy.uni-hamburg.de/
CommSy-Projektraum ..... SE1 CommSy WiSe 11/12
Ausgabewoche ..... 24. November 2011

Kernbegriffe

Neben den Kontrollstrukturen *Sequenz* und *Auswahl* gibt es noch die *Wiederholung*. Diese wird in Java üblicherweise durch *Schleifenkonstrukte* (engl.: loop constructs) realisiert, die ermöglichen, dass eine Reihe von Anweisungen mehrfach, nur einmal oder gar nicht ausgeführt wird. Grundlegend lassen sich u.a. *Zählschleifen* (in Java mit `for` möglich) und *bedingte Schleifen* (in Java mit `while` und `do-while` möglich) unterscheiden. Die Wiederholung durch Schleifenkonstrukte wird auch *Iteration* genannt.

Jeder Variablen (Exemplarvariable, formaler Parameter, lokale Variable) in einem Java-Programm ist ein *Sichtbarkeitsbereich* (engl.: scope) zugeordnet, in dem sie im Quelltext benutzt werden kann. Er entspricht der Programmeinheit, in der die Variable deklariert ist.

Der Sichtbarkeitsbereich...

- ... einer Exemplarvariablen ist die gesamte Klassendefinition;
- ... eines formalen Parameters ist seine definierende Methode;
- ... einer lokalen Variablen beginnt nach ihrer Deklaration und endet mit dem Ende des Blocks, in dem sie definiert wurde.

Sichtbarkeitsbereiche können ineinander geschachtelt sein, dabei sind Variablen aus umgebenden Bereichen auch in geschachtelten sichtbar; beispielsweise sind die Exemplarvariablen einer Klasse in allen Methoden der Klasse sichtbar, weil Sichtbarkeitsbereiche von Methoden in denen von Klassen geschachtelt sind.

Die *Lebensdauer* (engl.: lifetime) einer Variablen oder eines Objektes ist die Zeit *während der Ausführung eines Programms*, in der der Variablen oder dem Objekt Speicher zugeteilt ist. Lokale Variablen werden beispielsweise auf dem Aufruf-Stack abgelegt und nach Ausführung ihrer Methode automatisch wieder entfernt, sie leben also maximal für die Dauer einer Methodenausführung. Die Lebensdauer von Referenzvariablen muss nicht gleich der Lebensdauer der referenzierten Objekte sein.

Lernziele

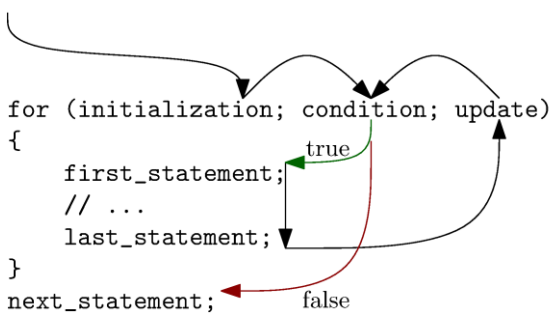
Programmgesteuerte wiederholte Ausführung verstehen; Unterschiede zwischen den verschiedenen Schleifenkonstrukten kennen; Debugger einsetzen; syntaktische Elemente im Quelltext zuordnen; logische von formalen Fehlern unterscheiden können; korrekte von lediglich syntaktisch korrekten Quelltexten unterscheiden können.

Aufgabe 6.1 Iterative Schleifenkonstrukte

6.1.1 Öffnet das Projekt *Iteration* und schaut euch die Klasse `Schleifendreher` an. Dort findet ihr Beispiele für die verschiedenen Schleifentypen in Java. Führt die Beispiele aus, um euch mit den Schleifen vertraut zu machen. Beachtet dabei die Ausgaben auf der Konsole.

Lest euch die Methodenrümpfe gründlich durch. Führt sie anschließend im Debug-Modus von BlueJ aus: Öffnet dazu die Klasse im BlueJ-Editor und klickt links in die weiße Leiste neben eine Anweisung in der Methode; es erscheint ein kleines rotes Stoppschild. Dies ist ein Haltepunkt. Wenn ihr nun wie gewohnt die Methoden aufruft, öffnet sich der Debugger automatisch und ihr könnt den Programmablauf Schritt für Schritt steuern, indem ihr auf den Knopf „Schritt über“ (engl. step) klickt. Erklärt euren Betreuern, warum die Beispiele zu diesen Ausgaben führen. Konsultiert im Zweifel den zweiten Teil des Skripts.

6.1.2 Das folgende Diagramm zeigt, wie der Kontrollfluss durch eine `for`-Schleife wandert:



Zeichnet entsprechende Diagramme für die `do-while`-Schleife und die `while`-Schleife. **Schriftlich**.

6.1.3 Schaut euch die Klasse `TextAnalyse` des Projektes *Iteration* an. Dort gibt es eine vorgegebene Methode `istFrage(String text)`, die demonstriert, wie man die Länge eines Strings erhält und wie man auf einzelne Zeichen eines Strings zugreift. Probiert diese Methode interaktiv aus, indem ihr ein Exemplar von `TextAnalyse` erstellt und dann `istFrage` z.B. mit dem aktuellen Parameter "Wie geht's?" aufruft.

Vergleicht die Methoden `istFrage` und `istFrageKompakt`. Worin unterscheiden sie sich?

6.1.4 Schreibt nun eine eigene Methode `int zaehleVokale(String text)`, die für einen gegebenen Text als Ergebnis liefern soll, wie viele Vokale er enthält. Für den String "hallo" soll die Methode beispielsweise eine 2 zurückgeben. Verwendet in der Implementierung einen Schleifenzähler, der bei 0 beginnt und alle Positionen des Strings durchläuft. Die eigentliche Prüfung auf einen Vokal lässt sich am elegantesten mit der switch-Kontrollstruktur lösen - schaut euch im Zweifel den zweiten Skriptteil an, wenn ihr nicht genau wisst, wie das switch funktioniert.

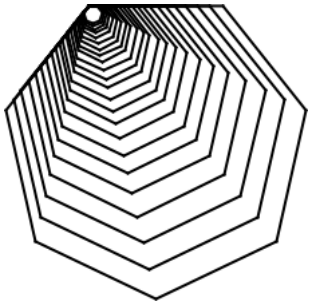
6.1.5 Schreibt eine weitere Methode `boolean istPalindrom(String text)`, die nur für Palindrome wie anna, otto, regallager oder axa *true* liefert. Vergleicht dazu die passenden Zeichen innerhalb des Strings.

Verwendet die Methode `toLowerCase()` aus der Klasse `String`, um den Unterschied zwischen Groß- und Kleinschreibung zu ignorieren, damit auch Anna, Otto und Regallager als Palindrome erkannt werden.

6.1.6 *Zusatzaufgabe*: Was passiert, wenn ihr der Methode `istFrage` den leeren String als aktuellen Parameter übergebt, also `istFrage("")`? Wie könnt ihr diesen Fall sinnvoll behandeln?

Aufgabe 6.2 Turtle-Graphics

Kopiert das Projekt `TurtleGraphics` in euer Arbeitsverzeichnis und öffnet es in BlueJ. Das Projekt enthält zwei Klassen `Turtle` und `Dompteur`. Die Klasse `Turtle` stellt Methoden zur Verfügung, mit denen sehr einfach eine Turtle „bewegt“ werden kann; diese Bewegungen werden auf einer Zeichenfläche aufgezeichnet. Die Klasse `Dompteur` enthält eine Methode `start`, in der beispielhaft die Verwendung einer `Turtle` dargestellt ist. Mit Hilfe des Beispiels und der Dokumentation der `Turtle`-Schnittstelle sollen die folgenden Aufgaben gelöst werden.



6.2.1 Implementiert eine Methode in `Dompteur`, die mit Hilfe von `Turtle` ein n-Eck zeichnet. Die Anzahl der Ecken und die Kantenlänge sollen als Parameter übergeben werden.

6.2.2 Erweitert nun die Methode so, dass es möglich wird, die Position und Farbe des n-Ecks festzulegen.

6.2.3 Schreibt eine Methode in `Dompteur`, die kleiner werdende, ineinander geschachtelte n-Ecke zeichnet. Über Parameter soll festgelegt werden können,

- wie viele Ecken die n-Ecke haben sollen,
- wie groß die Kantenlänge des ersten n-Ecks ist.

Aufgabe 6.3 Syntax und Korrektheit

6.3.1 Für diese Aufgabe steht der Ausdruck einer Klasse zur Verfügung. Bearbeitet die dort beschriebene Aufgabe **innerhalb von 45 Minuten**. Die Direkteingabe kann bei der Lösung einiger Aufgaben helfen.

Aufgabe 6.4 Syntaktische Konstrukte und Sichtbarkeitsbereiche erkennen

6.4.1 Für diese Aufgabe liegen 3 ausgedruckte Java-Quelltexte aus. Bearbeitet die dort beschriebenen Aufgaben. Es ist nicht wichtig, dass ihr versteht, was die jeweiligen Klassen tun. Es geht lediglich darum, syntaktische Konstrukte erkennen zu können.
Nehmt euch auch für diese Aufgabe maximal 45 Minuten lang Zeit.