SE1, Aufgabenblatt 10

Softwareentwicklung I – Wintersemester 2012/13

Arrays, Klassen als Objekte

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

CommSy-Projektraum ..... SE1 CommSy WiSe 12/13

Ausgabewoche ..... 20. Dezember 2012

Kernbegriffe

Arrays sind eine spezielle Form von Sammlungen gleichartiger Elemente. Sie werden von den meisten imperativen Sprachen angeboten, üblicherweise unterstützt durch eine spezielle Syntax. Der Zugriff auf ein Element erfolgt, wie bei einer Liste, über einen Index. Da Arrays jedoch direkt auf den zugrunde liegenden Speicher abgebildet werden, kann mit den Mitteln der unterliegenden Rechnerarchitektur (mit Indexregistern o.ä.) ein sehr schneller wahlfreier Zugriff gewährleistet werden. Dafür sind Arrays jedoch in den meisten Sprachen statisch in ihrer Größe festgelegt, entweder bereits in ihrer Deklaration (wie in der Sprache Pascal) oder spätestens bei ihrer Erzeugung zur Laufzeit (wie in Java).

In Java können die Elemente eines Arrays sowohl Werte der Basistypen als auch Referenzen auf Objekte sein. Der Index ist in Java eine natürliche Zahl zwischen 0 und (Größe des Arrays)-1. Er wird in eckigen Klammern direkt hinter dem Bezeichner des Arrays verwendet (`a[0]` beispielsweise bezeichnet das erste Element des Arrays `a`).

Ein Array selbst ist in Java ein Objekt, eine Array-Variable ist daher immer eine Referenzvariable. Der Typ dieser Variablen wird als *Array von <Elementtyp>* festgelegt. Arrays müssen, genauso wie Exemplare von Klassen, mit einer *new-Anweisung* erzeugt werden. Erst bei der Erzeugung eines konkreten Arrays wird festgelegt, wie viele Elemente dieses Array aufnehmen kann. Diese *Größe* (oder *Länge*) dieses Arrays ist dann festgelegt und kann nicht mehr verändert werden. Eine Array-Variable kann jedoch zu verschiedenen Zeitpunkten auf Arrays unterschiedlicher Größe verweisen.

In Java kann auch eine Klasse als ein Objekt angesehen werden, das zur Laufzeit einen Zustand hat und Operationen anbietet. Die Operationen des Klassenobjektes werden mit dem Schlüsselwort `static` deklariert, ebenso wie die Datenfelder für den Zustand des Klassenobjektes.

Lernziele

Einfache und mehrdimensionale Arrays verstehen und anwenden können, Schleifen über Arrays verwenden können, Java-Programme ohne BlueJ mit Hilfe der `main`–Methode von der Kommandozeile aufrufen können, Kommandozeilenparameter übergeben können.

Aufgabe 10.1 Strings in der Kommandozeile analysieren

In dieser Aufgabe wollen wir eine Java-Methode einmal nicht innerhalb von BlueJ aufrufen, sondern von der Kommandozeile des jeweiligen Betriebssystems. In Java ist für diesen Zweck eine spezielle Operation definiert worden: `public static void main(String[] args)`. Wenn eine Klasse eine Methode mit genau dieser Signatur definiert, dann kann diese Methode aus der Laufzeitumgebung der plattformabhängigen Java Virtual Machine aufgerufen werden.

10.1.1 Erstellt eine Klasse, die eine solche `main`-Methode enthält. Im Rumpf der Methode soll vorläufig lediglich eine beliebige Meldung mit `System.out.println` auf der Konsole ausgegeben werden. Ruft diese Methode in BlueJ auf.

10.1.2 Versucht nun, diese Methode von der Kommandozeile aus aufzurufen. Dazu müsst ihr zuerst ein Fenster öffnen, in dem ihr Kommandos eingeben könnt. (*WindowsXP: Start → Ausführen → cmd; Linux: Terminalfenster*) Wechselt in das Projekt-Verzeichnis von BlueJ, in dem eure Klasse (*Klassenname*.class) liegt. Dann könnt ihr folgende Zeile eingeben:

java *<Klassenname>*

10.1.3 In der Signatur der Methode `main` seht ihr, dass diese Parameter vom Typ String in Form eines String-Arrays entgegennimmt. Findet heraus, wie man aktuelle Parameter in der Konsole bei einem Aufruf der Methode übergeben kann. Ändert nun eure `main`-Methode so ab, dass die übergebenen Strings nacheinander mit `System.out.println` ausgegeben werden. Testet diese Änderung, indem ihr von der Kommandozeile aus eure `main`-Methode mit Parametern aufruft.

10.1.4 Schreibt in eurer Klasse eine Methode `analysiereText(String text)`, die für einen übergebenen String erfasst, wie häufig die 26 Buchstaben des Alphabets (ohne Umlaute, nur Kleinbuchstaben) darin vorkommen. Verwendet hierzu ein `int`-Array der Länge 26. Wendet eure Methode auf die Parameter der `main`-Methode an. Anschließend soll das *Gesamtergebnis* für alle Parameter mit `System.out.println`

ausgegeben werden. Testet erneut, entweder von der Kommandozeile aus oder in BlueJ. Tipp: Für diese Aufgabe müssen Buchstaben auf Array-Positionen abgebildet werden. Dabei soll die Position 0 für den Buchstaben 'a' stehen, die Position 25 für den Buchstaben 'z'. Die korrekte Array-Position für einen Buchstaben erhaltet ihr, indem ihr 'a' subtrahiert.

10.1.5 *Zusatzaufgabe*: Was passiert, wenn ihr aus einer Klassenmethode (z.B. der `public static void main(String[] args)`) auf eine Exemplarvariable zugreifen wollt? Gebt euren Betreuern eine Erklärung für das auftretende Verhalten.

Aufgabe 10.2 TicTacToe-Spielfeld als Array

Zweidimensionale Spielfelder sind ideale Kandidaten für die Verwendung von Arrays. In dieser Aufgabe implementieren wir das bereits bekannte Interface `Spielfeld` mit einer Klasse, die Arrays verwendet.

10.2.1 Zweidimensionale Arrays sind in Java als Arrays von Arrays realisiert. **Skizziert eine Grafik,** die diesen Sachverhalt anhand eines TicTacToe-Spielfels verdeutlicht (denkt an die Referenzen).

10.2.2 Öffnet das neue Projekt *TicTacToe*. Es ist nicht übersetzbar, weil eine Implementation für das Interface `Spielfeld` fehlt. Schaut euch dieses Interface an. Schreibt anschließend eine Klasse `SpielfeldArray`, die das Interface mit Hilfe eines zweidimensionalen Arrays implementiert. Mithilfe der vorgegebenen Testklasse könnt ihr die fachliche Korrektheit überprüfen.

10.2.3 Schreibt Implementationskommentare an den Stellen im Quelltext, an denen a) ein Array deklariert, b) ein Array erzeugt, c) lesend und d) schreibend auf ein Array zugegriffen wird.

10.2.4 *Zusatzaufgabe:* die Methode `istVoll()` in der Klasse `SpielfeldArray` lässt sich sehr elegant realisieren, indem man zwei verschachtelte erweiterte for-Schleifen einsetzt. Falls ihr dies bisher nicht so programmiert habt, versucht dies nun wie vorgeschlagen umzusetzen.

Aufgabe 10.3 Bildbearbeitung

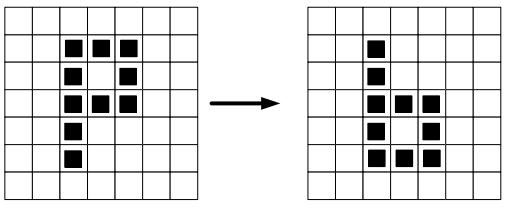
In dieser Aufgabe betrachten wir mehrdimensionale Arrays von elementaren Datentypen. Als Beispiel verwenden wir das Projekt *Bildbearbeitung*. Es enthält drei Klassen, von denen wir uns jedoch nur mit einer beschäftigen: `SWBild`. Die Klasse `BildEinleser` dient dazu, Bilder einzulesen, die Klasse `Leinwand` wird genutzt, um die Bilder anzuzeigen. Beide Klassen wollen wir nicht näher betrachten.

10.3.1 Öffnet das Projekt in BlueJ und erzeugt ein Exemplar von `SWBild`, eine Erläuterung findet sich in der Projektdokumentation. Wenn alles klappt, wird ein Bild in einem eigenen Fenster angezeigt.

Mit der Operation `dunkler(int delta)` könnt ihr dieses Bild abdunkeln. Probiert es aus. Schaut euch dann die Implementierung der Operation an. Wie wird das Array verwendet? Was befindet sich in dem Array? **Skizziert in einer Zeichnung** die implementierte Objektstruktur des Bilddaten-Arrays.

10.3.2 Implementiert die Operation `heller(int delta)` in der Klasse `SWBild`. Mit ihr soll das Bild um den Wert von `delta` aufgehellt werden.

10.3.3 Implementiert in `SWBild` die Operation `horizontalSpiegeln`, die das Bild an der horizontalen Achse spiegelt. Die folgende Darstellung soll die Aufgabe verdeutlichen:



Vor dem Spiegeln nach dem Spiegeln

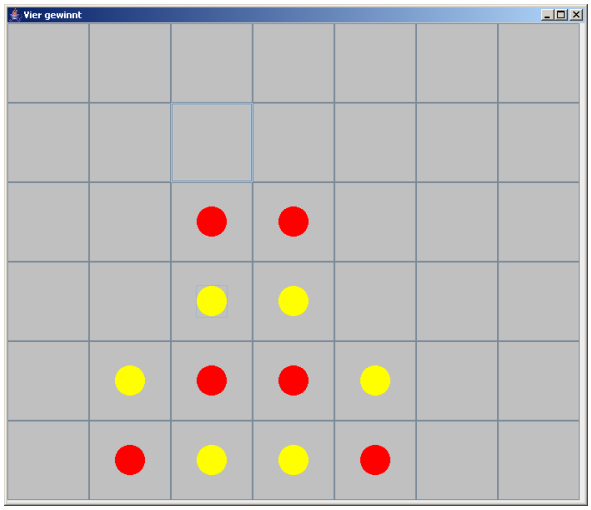
10.3.4 *Zusatzaufgabe*: Implementiert die Operation `weichzeichnen`. Das Weichzeichnen wird erreicht, indem der Mittelwert der 8 umgebenden Bildpunkte errechnet und anschließend für den gerade betrachteten Bildpunkt verwendet wird.

10.3.5 *Zusatzaufgabe*: Implementiert die Operation `punktSpiegeln`, die alle Punkte am Mittelpunkt des Bildes spiegelt.

10.3.5 *Zusatzaufgabe*: Implementiert die Operation `spot`, die ein Scheinwerferlicht projiziert.

Aufgabe 10.4 Vier Gewinnt

Wie in Aufgabe 10.2 implementieren wir ein Spielfeld, diesmal jedoch für das Spiel *Vier Gewinnt*. Zur Erinnerung: Anders als bei Tic Tac Toe müssen vier Felder in einer Reihe (vertikal, horizontal oder diagonal) durch einen Spieler besetzt sein, damit er das Spiel gewinnt. Felder können nicht beliebig besetzt werden, sondern es kann nur in eine noch nicht vollständig gefüllte Spalte ein Spielstein „eingeworfen“ werden, der auf die bereits in der Spalte vorhanden Steine „herunter fällt“.



Beispiel einer Spielsituation bei „Vier Gewinnt“

10.4.1 Öffnet das Projekt *VierGewinnt*. Es ist nicht übersetzbar, weil eine Implementation für das Interface `Spielfeld` fehlt. Schaut euch dieses Interface genau an. **Diskutiert schriftlich** mindestens zwei verschiedene mögliche Implementationen.

10.4.2 Schreibt eine Klasse `SpielfeldArray`, die das Interface implementiert. **Bedenkt dabei, dass Zeile 0 die unterste Zeile ist**. Die mitgelieferte Testklasse sollte euch helfen, eventuelle Fehler aufzudecken.

10.4.3 *Zusatzaufgabe*: Ergänzt die Testklasse um weitere sinnvolle Testfälle, die euch einfallen.