Javakurs2007/Übungspool

Aus FreitagsrundenWiki

Wir haben jeder Aufgabe eine Einschätzung ihrer Schwierigkeit auf einer Skala von 1 bis 10 vorangestellt. Bei dieser Einschätzung sind wir von einem Gehirn ausgegangen, für welches das jeweilige Themengebiet völlig neu ist, das jedoch schon ein ganz gutes Verständnis dafür hat wie eine Programmiersprache denn so funktioniert. Entsprechend kann eine schwer bewertete Aufgabe aus einem frühen Themengebiet für manch einen sehr einfach oder eine leicht bewertete aus einem späten Themengebiet für einen anderen sehr schwer sein. Es ist nur ein Wert, der euch helfen soll eure nächste Aufgabe zu wählen. Einen kleinen Überblick zu den Schwierigkeitsgraden findet ihr ganz unten auf dieser Seite.

Falls ihr Anmerkungen zu den Aufgaben habt, könnt ihr die Diskussionsseiten der jeweiligen Aufgaben nutzen.

Aufgabe	Schwierigkeit	Vorwissen	Lerneffekte
Vom Arbeiten mit der Konsole bis "Hallo Welt"	2	1. VL	Einfaches Arbeiten mit der Kommandozeile Kompilieren und Ausführen von Javacode
Variablen und Zuweisungen	1	1. VL	Deklarieren und definieren von Variablen grundlegende Regeln der Codeformatierung
Cowsay oder "Wie macht die Kuh?"	3	1. VL	etwas mehr Bildschirmausgabe schöne Graphiken
Erstellen eines Kassenbons	2	1. VL	Arbeiten mit Variablen Fallunterscheidungen
Ein eigenes Menü	3	1. VL	Lesen von Benutzereingaben über die Konsole Fallunterscheidungen
Glücksspiel	3	1. VL	Anwenden von Fallunterscheidungen
Quickies Arrays	2	2. VL	Verwenden von Arrays
Verschachtelte Schleifen	3	2. VL	Umgang mit mehreren verschachtelten Schleifen
Primzahlen	2-5	2. VL	Geschicktes Einsetzen von Schleifen und Arrays
Wetterstation	4	2. VL	Schleifen und Arrays
Geometrie	4	2. VL	Methoden schreiben schreiben mathematischer Funktionen
Zweierpotenzen	5	2. VL	Schleifen und Arrays
Schleifen und Modulo	5	2. VL	Verwendung von Schleifen Benutzung des Modulo-Operators

Quickies zu Methoden	2	3. VL	Verwenden von Methoden
Eigene Methoden	3	3. VL	Schreiben eigener Methoden
Schaltjahr	3	3. VL	Schreiben von Methoden Fallunterscheidungen Benutzerinteraktion
Crack the Password	3	2/3. VL	Passwörter knacken ;-)
Rekursion vs. Iteration	5	3. VL	Fehler finden
Lineare Funktionen	5	3. VL	Verwendung von Methoden Schleifen und Arrays
Fibonaccizahlen	5	3. VL	Schleifen und Arrays
Brainfuck vs Turingmaschine	5	3. VL	Arbeiten mit Arrays Allgemeinwissen :-)
Gauß-Algorithmus	6	3. VL	Arrays und Matrizen
Zahlenumrechner	6	3. VL	Arbeiten mit Methoden mathematisches Denken
Der Fehler im System	3	3. VL	Fehler finden
Spaß mit Quersummen	5 - 7	3. VL	Benutzung des Modulo-Operators Auslagern von Funktionalität in Methoden
Appleman und Fraktale	7	3. VL	Schleifen, Umgang mit Fließkommazahlen
Diffusion Limited Aggregation	8	3. VL	2-dimensionale Arrays, Methoden, Schleifen
Palindrome	3	4. VL	Entwurf von Programmen Parameterübergabe beim Programmaufruf
Hangman	5	4. VL	Entwurf von Programmen Arbeiten mit Methoden
Selection Sort	5	4. VL	Verstehen von komplexeren Algorithmen Schleifen und Arrays (Hilfs-)Methoden
Chiffrierung/Dechiffrierung (Cäsar)	9	4. VL	eigenständiges Problemlösen. Alle Konzepte des Kurses
Abzählspiel	7	4. VL	Analyse von Aufgabenstellungen Entwurf eines komplexen Programms von Grund auf
Schneckenrennen	4	5. VL	Objektorientierte Programmierung
Studentendatenbank	4	5. VL	Vertiefung in objektorientiertes Programmieren
Mehrdimensionale Arrays	10	5. VL	Methoden, Schleifen, Arrays Objektorientierung

Virtuelles Canvas mit 3D-Objekten	5	6. VL	Arbeiten mit Methoden mathematisches Denken
ASCII DOOM	8	6. VL	Vertiefung in objektorientierte Programmierung Benutzung der Java-API
Texte lesen und analysieren	7	6. VL	eigenständiges Problemlösen

Aufgaben als PDF

Alle Aufgaben in einer PDF-Datei gibt es hier: http://www.freitagsrunde.org/~kai/aufgaben_javakurs2007.pdf (Stand: 10.04.2007 07:00, 79 Seiten, 309kB)

Schwierigkeitsgrade

1. Reines Tutorial. Man muss nicht viel selbst denken
2. ...
3. Leichte Übungsaufgabe. Studies, die noch eher unsicher sind werden hier gut begleitet.
4. ...
5. Typische Aufgabe. Man muss den Kopf benutzen, bekommt an schwierigen Stellen aber Hilfestellungen.
6. ...
7. Herausfordernd, eher etwas für Studies die etwas fitter sind.
8. ...
9. Auch erfahrene Programmierer weden hier ein bis zwei Übungen beschäftigt sein
10. Tja, die Idee war da. Wir finden sie auch gut, aber es könnte wirklich anstrengend werden.

Javakurs2007/Von der Console zum HelloWorld

Aus FreitagsrundenWiki

- Melde dich an einer der Sun-Rays in den Rechnerräumen mit deinem CS-Account an. Solltest du selbst keinen CS-Account haben, wende dich an einen der Tutoren.
- Sollte das Arbeiten unter Unix neu für dich sein, dann nimm dir etwas Zeit, um dich mit den elementaren Konsolenkommandos vertraut zu machen. Erste Hilfe zu den Kommandos gibt es im Wiki unter https://wiki.freitagsrunde.org/Javakurs2007/Kommandohilfe.
- Lege dir ein Verzeichnis "Javakurs" an und kopiere die Vorgaben von http://docs.freitagsrunde.org/javakurs/2007/vorgaben/VonDerConsoleZumHelloWorld.tar an diesen Ort.
- Entpacke die Vorgaben, die sich nun in deinem Verzeichnis befinden. (Mit dem Befehl `tar -xf VonDerConsoleZumHelloWorld.tar` in einer Shell)
- Lösche im entstandenen Verzeichnis VonDerConsoleZumHelloWorld/ die Datei WegMitDieserDatei und das Verzeichnis WegMitDiesemVerzeichnis.
- Benenne im selben Verzeichnis die Datei GibMirEinenNamen.txt in CompiliereMich.java um.
- Compiliere die soeben entstandene Datei CompiliereMich.java mit Hilfe von javac.
- Führe das Programm in der entstandenen Datei CompiliereMich.class mit Hilfe von java aus.
- Öffne einen Texteditor deiner Wahl. Schreibe ein Programm, das "HalloWelt" auf der Konsole ausgibt und teste es.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

hallo alle freitagsrunde Team,

diese lehrveranstaltung finde ich persönlich super megageil, aber ich wünsche,dass die Musterlösungen von solchen schweren Aufgaben im Netz gestellt werden.

ich wünsche Euch viel Erfolg und weiterso...:-)

Javakurs2007/Variablen und Zuweisungen

Aus FreitagsrundenWiki

Hinweis: Alle in diese Übung verwendeten Klassen findest du auch als Vorgabedateien unter http://docs.freitagsrunde.org/javakurs/2007/vorgaben/VariablenUndZuweisungen.tar

(Die Datei wird mit dem Kommando tar -xf entpackt)

Inhaltsverzeichnis

Deklaration und Initialisierung

- An welcher Stelle im folgenden Programm wird die Variable *aWholeNumber* deklariert?
- An welcher Stelle im folgenden Programm wird die Variable *aWholeNumber* initialisiert?
- Beantworte diese Fragen auch für die Variablen: *notAWholeNumber*, *aString*, *anotherWholeNumber*, *anotherNotSoWholeNumber*, *anotherString*

- Erkläre deine Meinung einem Tutor oder diskutiere sie mit der Person am Rechner nebenan.

```
class DeklarationUndInitialisierung {
        public static void main(String [] ignored) {
                int aWholeNumber;
                double notAWholeNumber;
                String aString;

                aWholeNumber = 0;
                notAWholeNumber = 0.0;
                aString = "";
                System.out.println(aWholeNumber);
                System.out.println(notAWholeNumber);
                System.out.println(aString);

                int anotherWholeNumber = 0;
                double anotherNotSoWholeNumber = 0;
                String anotherString = "";
                System.out.println(anotherWholeNumber);
                System.out.println(anotherNotSoWholeNumber);
                System.out.println(anotherString);
        }
}
```

Selber Deklarationen und Definitionen durchführen

- Realisiere die Kommentare im folgenden Codestück.
- Teste deinen Code.

```
class SelbstDeklarierenUndDefinieren {
	public static void main(String [] ignored) {
		// deklariere eine Variable "birnen"  vom Typ int

		// weise ihr den Wert drei zu

//		System.out.println("3 erwartet: " + birnen);
		// addiere fuenf zu dem Wert

//		System.out.println("8 erwartet: " + birnen);
		// deklariere eine Variable "aepfel" und weise ihr den Wert fuenf zu

//		System.out.println("5 erwartet: " + aepfel);
		// subtrahiere zwei von dem Wert

//		System.out.println("3 erwartet: " + aepfel);


	}
}
```

Zuweisung an Variablen

- Überlege dir, was für eine Ausgabe der folgende Code erzeugen würde.
- Compiliere den Code, führe ihn aus und vergleiche das Resultat mit deinen Erwartungen.
- Was sind die Unterschiede?
- Diskutiere die Unterschiede mit der Person am Rechner nebenan, oder einem Tutor.

```
class ZuweisungenAnVariablen {
	public static void main(String [] ignored) {
		System.out.println(0);
		int zero = 0;
		System.out.println(zero);
		int one = 1;
		int whatsThis = zero;
		System.out.println(whatsThis);
		whatsThis = one;
		System.out.println(whatsThis);
	}
}
```

Variablen ändern ihre Werte

- Überlege dir, was für eine Ausgabe der folgende Code erzeugen würde.
- Compiliere den Code, führe ihn aus und vergleiche das Resultat mit deinen Erwartungen.
- Was sind die Unterschiede?
- Diskutiere die Unterschiede mit der Person am Rechner nebenan, oder einem Tutor.

```
class VariablenAendernIhreWerte {
        public static void main(String [] ignored) {
                int number = 1;
                System.out.println(number);
                number = 1;
                System.out.println(number);
                number = 2;
                System.out.println(number);
                number = 3;
                System.out.println(number);
                number = 1;
                System.out.println(number);
                number = 17;
                System.out.println(number);
        }
}
```

Formatierung und Namen

- Wieso haben wir die Code-Beispiele nach den geschweiften Klammern eingerückt und die Variablen so benannt, wie wir es taten und nicht so:

```
class _{static long _
(long __,long  ___) {
return __==0 ?___+ 1:
___==0?_(__-1,1):_(__
-1,_(__, ___-1)) ;  }
static  {int _=2 ,___
= 2;System.out.print(
"a("+_+',’+___+ ")="+
_ (_,  ___) ) ;System
.exit(1);}}//(C) Ulli
```

- Etwas weniger Extrem: Was ist im folgenden Beispiel bei der Formatierung schief gelaufen?

```
class Formatierung1 {public static void main
        (String [] ignored) {
        System.out.println(0); int zero = 0;
                        System.out.println(zero);
                int one = 1;
                int whatsThis = zero;
                        System.out.
                println(whatsThis);
        whatsThis = one;                System.out.println(whatsThis)
;}}
```

- Warum ist der folgende Code so schlecht lesbar?
- Woran erinnert er dich?

```
class b {
	public static void main(String [] i) {
		System.out.println(0);
		int a = 0;
		System.out.println(a);
		int b = 1;
		int c = a;
		System.out.println(c);
		c = b;
		System.out.println(c);
	}
}
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

hallo alle freitagsrunde Team,

diese lehrveranstaltung finde ich persönlich super megageil, aber ich wünsche,dass die Musterlösungen von solchen schweren Aufgaben im Netz gestellt werden.

ich wünsche Euch viel Erfolg und weiterso...:-)

Von „http://wiki.freitagsrunde.org/Javakurs2007/Variablen_und_Zuweisungen“

Javakurs2007/Cowsay

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Aufgabenstellung

Diese Aufgabe stellt eine etwas erweitere Version von HelloWorld dar. Als Vorbild dient das Unix-Programm Cowsay. Diesem Programm übergibt man einen Text und als Ergebnis erhält man eine kleines Bild (als ASCII) auf der Konsole, welches den übergeben Text enthält. Dies könnte so aussehen wie unter Beispiele zu sehen. Cowsay könnt ihr hier ausprobieren: http://www.linuxbox.co.uk/cowsay.php

1) Als ersten Schritt könnt ihr mit **System.out.println("");** erstmal ein wenig herumprobieren und versuchen eine ASCII Graphik, die ihr euch ausgedacht habt, auszugeben. Natürlich könnt ihr auch die aus dem Beispiel nehmen.

2) Als zweiten Schritt könnt ihr probieren, auch etwas Text mit in die Graphik einzubinden, wie in dem ersten Beispiel zu sehen.

Beispiel

```
 _____________
< Java rocks! >
 -------------
       \   ^__^
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
```

Hinweise

Wenn ihr auf der Konsole ein Backshlash \ augeben wollt, dann müsst ihr dieses Zeichen, da es selbst Escapezeichen ist, mit einem Backslash "escapen". Der Code um ein Backslash auszugeben, sieht also so aus.

```
System.out.println("\\");
```

weitere Beispiel

```
   oooo
   `888
    888  .oooo.   oooo    ooo  .oooo.
    888 `P  )88b   `88.  .8'  `P  )88b
    888  .oP"888    `88..8'    .oP"888
    888 d8(  888     `888'    d8(  888
.o. 88P `Y888""8o     `8'     `Y888""8o
`Y888P


                             oooo                .o.
                             `888                888
oooo d8b  .ooooo.   .ooooo.   888  oooo   .oooo.o 888
`888""8P d88' `88b d88' `"Y8  888 .8P'   d88(  "8 Y8P
 888     888   888 888        888888.    `"Y88b.  `8'
 888     888   888 888   .o8  888 `88b.  o.  )88b .o.
d888b    `Y8bod8P' `Y8bod8P' o888o o888o 8""888P' Y8P
```

```
    ____  ____  ____  ____
   /\   \/\   \/\   \/\   \
  /  \___\ \___\ \___\ \___\
  \  / __/_/   / /   / /   /
   \/_/\   \__/\/___/\/___/
     /  \___\     /  \___\
     \  / __/_   _\  /   /
      \/_/\   \/\ \/___/
        /  \__/  \___\
        \  / _\  /   /
         \/_/\ \/___/
           /  \___\
           \  /   /
            \/___/
```

Links

- http://www.network-science.de/ascii/
- http://de.wikipedia.org/wiki/ASCII-Art

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Kassenbon

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Kassenbon

Es soll ein Programm geschrieben werden, dass einen Kassenbon auf der Konsole ausgibt. Am Anfang des Programmes werden in Form von Variablen folgende Werte festgelegt:

- Anzahl von mindestens vier verschiedenen Waren, die gekauft werden sollen.
- Einzelpreise der Waren. (Ja, jede Warensorte hat einen anderen Preis)
- Inhalt der Brieftasche.

Das Programm soll den Gesamtpreis der eingangs festgelegten Waren ermitteln. Übersteigt der Gesamtpreis den Inhalt der Brieftasche, wird der Benutzer auf den fehlenden Betrag hingewiesen. Reicht der Inhalt der Brieftasche für den Einkauf aus, wird ein Kassenbon ausgegeben, der die gekauften Waren mit Anzahl und Preis sowie am Ende den Gesamtpreis auflistet.

Hinweis: Erstellt zuerst die Ausgabe des Kassenbons und füllt sie anschließend mit Berechnungen und der Fallunterscheidung.

Kassenbon + automatischer Einkauf

Erstellt eine Kopie eures soeben geschriebenden Programmes. Verändert das Programm so, dass nun der Kassenbon solange mit Waren gefüllt wird, bis der Inhalt der Brieftasche für keine weiteren Einkäufe ausreicht. Betrachtet der Einfachheit halber mehrere Produkte eines Typs als einen Einkauf.

Hinweis: Viele, viele Fallunterscheidungen!

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Unknown User

hallo alle freitagsrunde Team,

diese lehrveranstaltung finde ich persönlich super megageil, aber ich wünsche,dass die Musterlösungen von solchen schweren Aufgaben im Netz gestellt werden.

ich wünsche Euch viel Erfolg und weiterso...:-)

Robert

Der Eine oder Andere hat darauf bestanden auf seinem Kassenbon keine Hundertstelpfennige ausgeben zu wollen. Diese entstehen mitunter dadurch, dass Doublewerte mit einer gewissen Toleranz gespeichert werden. Hier ein Vorschlag für einen Workaround:

```
double kommazahl = 123.290402;

int ganzzahl = (int)kommazahl; // Nachkommastelle abschneiden
int kommastelle = (int)(kommazahl * 100)%10 ; // zwei Nachkommastellen extrahieren

System.out.println("Preis=" + ganzzahl + "." + kommastelle ); // Zahl in der Ausgabe zusammensetzen
```

Von „http://wiki.freitagsrunde.org/Javakurs2007/Kassenbon“

Javakurs2007/Javamenue

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Handwerkszeug

In vielen deiner Javaprogramme wirst du vom Benutzer eine Eingabe per Tastatur erfragen wollen. Da sich in Java das Lesen von Tastatureingaben normalerweise etwas umständlich gestaltet, haben wir dir an dieser Stelle etwas zur Erleichterung vorbereitet.

- kopiere die Datei Terminal.java (http://uebb.cs.tu-berlin.de/books/java/klassen/Terminal.java) in Dein Arbeitsverzeichnis.

Solange diese Datei in demselben Verzeichnis liegt, wie die java-Datei, an der Du arbeitest, stehen Dir einige Befehle zur Verfügung, die das Lesen von Tastatureingaben stark erleichtern. Wenn dich interessiert, warum das so ist, so erfährst du es hier.

Falls das nicht funktioniert, ist der sogenannte CLASSPATH falsch gesetzt. Gibt dazu in dem Terminal direkt nach dem Öffnen des Terminals folgendes Kommando ein:

```
CLASSPATH=".:$CLASSPATH"
```

Zurück zum Thema: Einer dieser Befehle lautet "Terminal.readString()". Er liest eine Tastatureingabe bis zum nächsten Zeilenumbruch von der Konsole und liefert ihn als String zurück.

Beispiel:

```
// Einlesen einer Tastatureingabe bis zum nächsten Zeilenumbruch in die Variable "eingegebeneTextzeil

String eingegebeneTextzeile;
eingegebeneTextzeile = Terminal.readString();
```

Erste Konsolenabfrage

Probieren wir das ganze einmal aus.

- Frage auf der Konsole nach dem Namen des Benutzers.
- speichere seine Eingabe mit Hilfe von "Terminal.readString()".
- Begrüße den Benutzer mit seinem Namen.

Menü mit Zahlen

Schonmal nicht schlecht. Nun zum interessanten Teil. Ein weiterer Befehl zur Vereinfachung der Konsolenbenutzung lautet Terminal.readInt(). Er liest eine Zahl von der Konsole und gibt sie als int zurück.

Beispiel:

```
// Einlesen einer Zahl per Tastatureingabe in die Variable "eingegebeneZahl"

int eingegebeneZahl;
eingegebeneZahl = Terminal.readInt();
```

- Frage auf der Konsole, ob der Benutzer:

  1) eine Frau ist.
  2) ein Mann ist.
  3) die Frage nicht beantworten möchte.

- Begrüße den Benutzer abhängig von seiner Eingabe mit "Herr", "Frau" oder geschlechtslos.
- Zeige eine Fehlermeldung an, wenn eine andere Zahl als 1,2 oder 3 eingegeben wird.

Tipps

Einen Text auf der Konsole ausgeben:

```
System.out.println("text");
```

Den Programmfluss abhängig vom Warheitsgehalt einer Bedingung lenken:

```
if (bedingung) {
 fall1
}
else {
 fall2
}
```

Zwei Bedingungen logisch miteinander verknüpfen:

```
if (bedingung1 && bedingung2) {...} // Logisches Und
```

```
if (bedingung1 || bedingung2) {...} // logisches Oder
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Von „http://wiki.freitagsrunde.org/Javakurs2007/Javamenue“

Javakurs2007/Glücksspiel

Aus FreitagsrundenWiki

Inhaltsverzeichnis

42%

Tipp: Mit dem Javabefehl *double randomValue = Math.random();* könnt ihr in der Variable *randomValue* einen zufälligen Fließkommawert zwischen 0 und 1 speichern.

- Schreibt ein Programm welches mit einer Wahrscheinlichkeit von 42% "Gewonnen!" und ansonsten "Verloren!" ausgibt.
- Kompiliert das Programm und führt es aus. Spielt ein wenig herum und ändert das Programm in dem ihr absichtlich Fehler einbaut, lest dann die Fehlermeldungen. Was genau für Informationen geben sie euch?

Begrüßung

- Schreibt ein Programm, welches drei Variablen kennt, die das Alter, den Namen und das Geschlecht einer Person speichern. Überlegt euch, von welchem Datentyp die drei Variablen sein müssen.
- Fügt eurem Programm dann eine Fallunterscheidung hinzu, die eine Begrüßung für die genannte Person ausgibt. Der Begrüßungstext unterscheidet sich für Personen, die männlich oder weiblich sind und unterscheidet zwischen Erwachsenen und Kindern. Folgende Begrüßungen sollen ausgegeben werden:
  - "Guten Tag Herr <name>!" für erwachsene Männer
  - "Guten Tag Frau <name>!" für erwachsene Frauen
  - "Hallo <name>!" für Mädchen und Jungen unter 18 Jahren

Glücksspiel

- Erweitert das Programm von Aufgabe "42%" zu einem kleinen Glücksspiel. Zwei Spieler wählen jeweils einen Wert zwischen 1 und 100. Dann wird eine zufällige weitere Zahl zwischen 1 und 100 ermittelt (mit Math.random()). Der Spieler, dessen Nummer am nächsten an der zufälligen Zahl liegt, hat gewonnen. Liegen beide Spieler gleich weit entfernt, dann gibt es ein Unentschieden.

Hinweis: Es reicht aus, wenn die Zahlen, die die Spieler wählen, direkt im Quellcode verankert sind, ihr müsst diese nicht von der Konsole einlesen.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine

Scheu, es geht nichts kaputt ;)

Javakurs2007/Arrays

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Eindimensionale Arrays

Zwei eindimensionale Arrays werden wie folgt deklariert und initialisiert.

- Was geben die anschließenden "System.out.println(...)" jeweils aus?
- Überprüfe deine Erwartungen, indem du die Codefragmente in einem eigenen Programm ausführst.

```
int[] meinArray1 = new int[5];
meinArray1[0] = 0;
meinArray1[1] = 1;
meinArray1[2] = 2;
meinArray1[3] = 3;
meinArray1[4] = 4;

int[] meinArray2 = new int[3];
meinArray2[0] = 0;
meinArray2[1] = 1;
meinArray2[2] = 2;
```

```
//1.
System.out.println(meinArray1.length);

//2.
System.out.println(meinArray2.length);

//3.
System.out.println("Ergebnis:"+meinArray1[1]);

//4.
System.out.println("Ergebnis:"+meinArray2[1]);

//5.
System.out.println("Ergebnis:"+meinArray1[4]);

//6.
System.out.println("Ergebnis:"+meinArray2[4]);

//7.
System.out.println(meinArray1.length-1);

//8.
System.out.println(meinArray2.length-1);

//9.
System.out.println("Ergebnis:"+meinArray1[0]);

//10.
System.out.println("Ergebnis:"+meinArray2[0]);
```

Zweidimensionale Arrays

Ein zweidimensionales Array wird wie folgt deklariert und initialisiert.

- Wie kann man sich ein zweidimensionales Array vorstellen?
- Was geben die anschließenden "System.out.println(...)" jeweils aus?
- Überprüfe deine Erwartungen, indem du die Codefragmente in einem eigenen Programm ausführst.

```
int[][] meinArray3 = new int[2][3];
meinArray3[0][0] = 1;
meinArray3[1][1] = 2;
meinArray3[0][2] = 1;
meinArray3[1][0] = 2;
meinArray3[0][1] = 1;
meinArray3[1][2] = 2;
```

```
//1.
System.out.println(meinArray3[0][0]+meinArray3[0][1]+meinArray3[0][2]);

//2.
System.out.println(meinArray3[1][0]+meinArray3[1][1]+meinArray3[1][2]);
```

Dreidimensionale Arrays

Ein dreidimensionales Array wird wie folgt deklariert und initialisiert.

- Wie kann man sich ein dreidimensionales Array vorstellen ?
- Was geben die anschließenden "System.out.println(...)" jeweils aus?
- Überprüfe deine Erwartungen, indem du die Codefragmente in einem eigenen Programm ausführst.

```
int[][][] meinArray3D = new int[2][2][2];
meinArray3D[0][0][0] = 1;
meinArray3D[0][1][1] = 2;
meinArray3D[0][1][0] = 2;
meinArray3D[0][0][1] = 1;

meinArray3D[1][0][0] = 3;
meinArray3D[1][1][1] = 4;
meinArray3D[1][1][0] = 4;
meinArray3D[1][0][1] = 3;
```

```
//1.
System.out.println(meinArray3D[0][0][0]+meinArray3D[0][0][1]);

//2.
System.out.println(meinArray3D[0][1][0]+meinArray3D[0][1][1]);

//3.
System.out.println(meinArray3D[1][0][0]+meinArray3D[1][0][1]);

//4.
System.out.println(meinArray3D[1][1][0]+meinArray3D[1][1][1]);
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Verschachtelte Schleifen

Aus FreitagsrundenWiki

Aufgabe

Schreibt ein Programm, welches das kleine Einmaleins ausgibt. Hierbei müssen zwei verschachtelte Schleifen verwendet werden (also eine Schleife innerhalb eines Blocks einer anderen Schleife). Die Ausgabe des Programms soll dann ca. so aussehen:

```
1 x 1 =   1
2 x 1 =   2
3 x 1 =   3
...
9 x 1 =   9
10 x 1 =  10

1 x 2 =   2
2 x 2 =   4
...
9 x 10 =  90
10 x 10 = 100
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Primzahlenaufgabe

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Primzahlen

Als Primzahlen bezeichnet man alle natürlichen Zahlen, die nur durch sich selbst und eins teilbar sind. Also zum Beispiel 3, 5, 7, 11, 13, usw. Die 1 ist per Definition keine Primzahl.

Einfacher Primzahlenfinder

Aufgabe:

Schreibt ein Programm, das die ersten n Primzahlen bestimmt und ausgibt.

Hinweise:

- Der einfachste Algorithmus um zu ermitteln, ob eine Zahl x eine Primzahl ist, besteht darin, x durch jede Zahl kleiner als x zu teilen und zu überprüfen, ob ein Rest übrig bleibt.

Für die 25 würde der Algorithmus also folgendermaßen ablaufen:

25 / 2 = 12 rest 1
25 / 3 = 8 rest 1
25 / 4 = 6 rest 1
25 / 5 = 5 rest 0 -> 25 ist keine Primzahl

Wenn der Algorithmus durchgelaufen ist, ohne einen Teiler für x gefunden zu haben, so muss x eine Primzahl sein.

- Der Modulo-Operator % gibt den Rest zurück, der beim Teilen von zwei Zahlen übrig bleibt.
- Ihr braucht zwei verschachtelte Schleifen. Eine, die die Zahlen 1 bis n hochzählt und eine, die jede dieser Zahlen durch jede kleinere Zahl teilt um auf Teilbarkeit zu überprüfen.

Optimierung des Algorithmus

Vorbemerkung:

Diese Aufgabe kann auf vielerlei Arten gelöst werden und ist weniger zum Verständnis von Schleifen, Arrays, Bedingungen und Co gedacht, sondern vielmehr, um die sinnvolle Anwendung dieser Konzepte zu üben.

Die Hinweise sind nur als Anregungen zum Lösen der Aufgabe gedacht, falls man selbst gar keine Idee hat. Da es viele Möglichkeiten gibt, die Aufgabe zu lösen, die Hinweise jedoch nur einen Weg aufzeigen, solltet ihr erstmal probieren, selbst eine Lösung zu finden.

Aufgabe:

Um zu ermitteln, ob eine Zahl `x` eine Primzahl ist, muss man diese nicht durch sämtliche kleineren Zahlen teilen. Tatsächlich reicht es, wenn man `x` nur auf Teilbarkeit durch alle Primzahlen, die kleiner als `x/2` sind, überprüft (warum dies so ist, wird weiter unten beim Punkt "Hintergrund" erklärt).

Optimiert nun den Algorithmus aus Aufgabe a) so, dass zum Überprüfen von Zahl `x` nur alle Primzahlen kleiner `x/2` durchlaufen werden.

Optionale Aufgabe:

Wie viele Primzahlen gibt es zwischen 1 und 10? Wieviele zwischen 1 und 100? 1 und 1000? ...

Schreibt euch ein Programm (oder eine Methode), das euch das Verhältniss von gefundenen Primzahlen zu der Anzahl der überprüften Zahlen ausgibt.

Hinweise:

- Speichert euch beim Durchlaufen der Zahlen jede gefundene Primzahl in einem Array.
- Wie groß muss das Array zum Speichern der gefundenen Primzahlen in Abhängigkeit von der Anzahl der zu überprüfenden Zahlen maximal sein? Die optionale Aufgabe hilft, dies herauszufinden.

Hintergrund:

Der Algorithmus aus Aufgabe a ist zwar einfach, aber nicht gerade sehr intelligent. Beispielsweise würde es auch reichen, beim Überprüfen einer Primzahl `x` nur die ersten `x/2` Zahlen zu durchlaufen, da beim Teilen durch alle größeren Zahlen nur noch ein Wert zwischen 1 und 2 rauskommen kann.

Veranschaulichung:

```
23 / 11 = 2 rest 1
23 / 12 = 1 rest 11
23 / 13 = 1 rest 10
23 / 14 = 1 rest 9
...
23 / 22 = 1 rest 1
```

Eine weitere Optimierungsmöglichkeit stellt die Tatsache dar, dass eine Zahl, die nicht durch zwei teilbar ist, auch nicht durch ein Vielfaches von zwei teilbar sein wird. Praktisch heißt das, dass wir die zu überprüfende Zahl nur auf Teilbarkeit durch die ungeraden Zahlen sowie der zwei überprüfen müssen.

Beispiel:

- 25 / 2 = 12 rest 1 -> nicht durch zwei teilbar, also auch nicht durch 4, 6, 8, 10, usw. -> nur ungerade Zahlen überprüfen

- 25 / 3 = 8 rest 1
- 25 / 5 = 5 rest 0 -> keine Primzahl

Wie man sieht, spart man hier schon ein paar Überprüfungen.

Die Regel, dass alle Zahlen, die nicht durch 2 teilbar, sind auch nicht durch ein Vielfaches von 2 teilbar sind, lässt sich noch verallgemeinern. Das heißt:

Eine Zahl die nicht durch `t` teilbar ist, ist auch nicht durch ein Vielfaches von `t` teilbar.

Dies bedeutet im Endeffekt, dass wir beim Überprüfen von `x` nur alle Primzahlen kleiner `x/2` überprüfen müssen. Da alle anderen Zahlen Vielfache einer kleineren Zahl sind, welche wir schon überprüft haben.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Wetterstation

Aus FreitagsrundenWiki

Aufgabe

Eine Wetterstation hat für 14 Tage folgende Temperaturwerte aufgenommen.

```
Tag:         01 02 03 04 05 06 07 08 09 10 11 12 13 14
Temperatur: 12 14 09 12 15 16 15 15 11 08 13 13 15 12
```

1. Speichere die folgende Tabelle mit einem geeigneten Datentyp.
2. Schreibe ein Programm, welches die Durchschnittstemperatur für die zwei Wochen bestimmt.
3. Schreibe ein Programm, welches die maximale und minimale Temperatur ausgibt.
4. Schreibe ein Programm, welches die beiden aufeinanderfolgenden Tage angeben kann, an denen es den größten Temperaturumschwung gab.
5. Schreibe ein Programm, welches die Tabelle schön tabellarisch auf der Konsole ausgibt.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Zweierpotenzen

Aus FreitagsrundenWiki

Aufgabe

Schreibt ein Programm, welches die ersten 20 Zweierpotenzen ausrechnet und in einem Array speichert.

Versucht in einem zweiten Schritt nur zwei Schleifen zu benutzen: Eine, welche das Array mit Werten füllt und eine Schleife, welche das Array testweise ausgibt.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Schleifen und Modulo

Aus FreitagsrundenWiki

Hinweise:

- Bevor du mit der Aufgabe beginnst, solltest du sie einmal komplett durchgelesen haben.
- Der Modulo Operator in Java ist "%". Der Modulo-Operator führt eine ganzzahlige Division durch und gibt den Rest zurück. *System.out.println("6 % 4 = " + 6 % 4)* ergibt also "6 % 4 = 2"

Aufgabe

Schreibe ein Programm, das die Zahlen 1 bis 100 (jeweils inklusive) auf einer eigenen Zeile ausgibt. Dabei soll jede durch 3 teilbare Zahl durch die Ausgabe "foo", jede durch 5 teilbare Zahl durch "bar" und jede durch 3 und 5 teilbare Zahl durch "foobar" ersetzt werden.

Die ersten Ausgaben des Programms sähen also so aus:

```
1
2
foo
4
bar
foo
7
8
foo
bar
```

etc.

Umsetzung

Erstelle eine neue Datei "LoopingFun.java" mit einer Klasse "LoopingFun" darin. Beispiel:

```
class LoopingFun {
        public static void main(String [] unused) {
                // Your code here ...
        }
}
```

1. Dein Code soll in die Main-Methode.
2. Ergänze die `main`-Methode um eine Schleife, die von 1 bis 100 (inklusive) zählt.
3. Gib in der Schleife die aktuelle Zahl aus.
4. Teste dein Programm.
5. Ergänze dein Programm, sodass es anstatt der durch 3 teilbaren Zahlen "foo" ausgibt.
6. Teste dein Programm.
7. Ergänze dein Programm, sodass es anstatt der durch 5 teilbaren Zahlen "bar" ausgibt. (Ignoriere dabei den Fall, wo eine Zahl durch 3 und durch 5 teilbar ist)
8. Teste dein Programm.
9. Ergänze dein Programm, sodass es eine durch 3 und 5 teilbare Zahl korrekt ausgibt.
10. Teste dein Programm.

Betrachte deine Lösung:

- Erklären die Namen der Variablen welchen Zweck sie haben?

- Enthält dein Programm Kommentare? Dokumentieren diese Kommentare was dein Programm tut (schlecht) oder warum du es tust (gut)?
- Wie stellt dein Programm fest, ob eine Zahl durch 3, 5 oder durch beide Zahlen teilbar ist?
- Wie hast du die Schleife implementiert? Mit einer For-Schleife oder mit einer while-Schleife? Verändere deine Implementierung, so dass die andere Schleifenart verwendet wird.
  - Welche der Schleifenkonstrukte ist für dieses Problem besser geeignet? Ändere (falls nötig) dein Programm so, dass die besser geeignete Syntax verwendet wird.
  - Einige dich mit deinen Nachbarn ob "for" oder "while" hier besser geeignet ist und teilt euer Ergebnis dem Tutor mit.
- Wie viele Fallunterscheidungen verwendet dein Programm? Drei, vier, fünf oder noch mehr? Wenn du mehr als drei Fallunterscheidungen hast, überlege wie Du mit drei Fallunterscheidungen auskommen könntest. Verändere dein Programm entsprechend. (Hinweis: *System.out.print**ln()*** fügt jeder Ausgabe einen Zeilenumbruch hinzu. *System.out.print()* tut das nicht.)
- Wie viele lokale Variablen verwendest du? Wenn es mehr als eine ist, überlege ob du dein Programm vereinfachen kannst, indem du nur noch eine Variable verwendest. (Hinweis: Auch ein Schleifenzähler ist eine lokale Variable)
- Vergleiche deine Lösung mit der eines anderen Kursteilnehmers und betrachtet die Unterschiede anhand der Fragen die hier aufgeführt sind.

Zum Abschluss: Führe dein Programm einem Tutor vor und programmiere seine Änderungsvorschläge.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Methoden

Aus FreitagsrundenWiki

Hinweis: Alle in diese Übung verwendeten Klassen findest du auch als Vorgabedateien unter http://docs.freitagsrunde.org/javakurs/2007/vorgaben/Methoden.tar

Inhaltsverzeichnis

Argumente

- Überlege dir, was der folgende Code macht. Was erwartest du als Ausgabe?
- Führe den Code aus und vergleiche das Ergebnis mit deinen Erwartungen.

```
class Beispiel {
	public static void main(String [] arguments) {
		ausgabe(1, "Martin");
		ausgabe("Arthur", 2);
		ausgabe(3, "Florian");
		ausgabe("Robert", 4);
	}

	static void ausgabe(int i, String string) {
		System.out.println(i + ": " + string);
	}
}
```

Rückgabewert

- Überlege dir, was der folgende Code in den folgenden Beispielen macht. Was erwartest du als Ausgabe?
- Führe den Code jeweils aus und vergleiche das Ergebnis mit deinen Erwartungen.

```
//file: Methoden1.java
class Methoden1 {
    public static void main (String[] arguments) {
        System.out.println( gibMirNeZahl() );
    }

    public static double gibMirNeZahl() {
        return 15.3;
    }
}
```

```
//file: Methoden2.java
class Methoden2 {
    public static void main (String[] arguments) {
        System.out.println( mathematik(1, 2) );
    }

    public static boolean mathematik(int argument1, int argument2) {
        return (argument1 + 5) < (argument2 * 2);
    }
}
```

```
//file: Methoden3.java
class Methoden3 {
    public static void main (String[] arguments) {
        System.out.println( mathematik(1, 2) );
        System.out.println( mathematik(1, 5) );
        System.out.println( mathematik(3, 4) );
    }

    public static boolean mathematik(int argument1, int argument2) {
        return (argument1 + 5) < (argument2 * 2);
    }
}
```

```
//file: Methoden4.java
class Methoden4 {
    public static void main (String[] arguments) {
        int eineZahl = 2;
        System.out.println( mathematik(4, eineZahl) );
        eineZahl = 5;
        System.out.println( mathematik(2, eineZahl) );
        System.out.println( mathematik(eineZahl, 4) );
    }

    public static boolean mathematik(int argument1, int argument2) {
        return (argument1 + 5) < (argument2 * 2);
    }
}
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Ran

das:

```
//file: Methoden2.java
class Methoden3
```

soll so:

```
//file: Methoden2.java
class Methoden2 {
```

sein...

Robert

Danke, ist geändert. -- Robert Buchholz 11:20, 11. Apr. 2007 (CEST)

Javakurs2007/MathematischeMethoden

Aus FreitagsrundenWiki

Im Folgenden wollen wir einige (mathematische)-Methoden programmieren.

- Lege eine neue Klasse "MatheAufgabe" an.
- Füge eine `public static void main(String[] arguments)` Methode hinzu.
- Deklariere innerhalb der `main`-Methode eine `double`-Variable.
- Belege diese Variable dem Wert `5.0/3.0` und gib den Inhalt der Variable auf dem Bildschirm aus.
- Schreibe nun eine Methode `public static double add(double x, double y)`, die die Summe der beiden übergebenen Zahlen `x` und `y` zurückgibt.
  - Teste deine Methode mit verschiedenen Eingaben wie z.B.

```
System.out.println(" 1.0 + 2.0 = " + add ( 1.0 , 2.0 ) );
double x = 5.0;
System.out.println( " " + x + " + 2.0 = " + add ( x , 2.0) );
System.out.println( " 2.0 + " + x + " = " + add ( 2.0 , x) );
x = add ( x , 2.0);
System.out.println( x);
```

- Schreibe nun eine Methode `sub`, welche die Differenz zweier übergebener Werte zurückgibt.
  - Teste deine Methode wie eben!
  - Überprüfe, ob x == (x - y) + y gilt! (natürlich mit deinen `sub` und `add`-Modthoden)
- Schreibe nun eine `mul`-Methode für `int`.
  - Benutze dazu nicht den operator *, sondern die Addition (+).
  - Teste deine Methode wie eben!
- Schreibe nun eine `public static double pow ( double basis, int exponent)`, die `basis` hoch `exponent` zurückgibt. `pow(2,3)` wurde z.B. 2^3=8 zurückgeben.
  - Benutze auch hier weder *, noch Math.pow(). Nutze stattdessen deine gerade geschriebene mul-Methode.
  - Teste deine Methode wie gehabt!
    Was liefert deine Methode wenn der Exponent 0 ist?
    Was liefert deine Methode wenn der Exponent negativ ist?

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

"Füge eine public static void main(Strings[] arguments) Methode hinzu." <-- Das muss "Füge eine public static void main(String[] arguments) Methode hinzu." heißen!

> Änder es ruhig. Nun hab ich's aber schon gemacht. -- Robert Buchholz

Von „http://wiki.freitagsrunde.org/Javakurs2007/MathematischeMethoden“

Javakurs2007/Schaltjahr

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Aufgabenstellung

Schreibt eine Methode, welche entscheidet, ob es sich bei der übergebenen Jahreszahl um ein Schaltjahr handelt, oder nicht. Die Jahreszahl könnte man als Integer (int) übergeben, der Rückgabewert des Methode sollte ein Wahrheitswert (boolean) sein.

Die Methode könnte so aussehen:

```
public static boolean isLeapYear(int year) {
[...]
}
```

Die Eigenschaften eines Schaltjahres sind

a) Ein Jahr ist kein Schaltjahr, wenn die Jahreszahl nicht durch 4 teilbar ist.

b) Ein Jahr ist ein Schaltjahr, wenn die Jahreszahl durch 4, aber nicht durch 100 teibar ist.

c) Ein Jahr ist ebenfalls ein Schaltjahr, wenn die Jahreszahl durch 4, durch 100 und durch 400 teilbar ist.

Zusatzaufgabe

Statte dein Programm mit einer nutzerfreundlichen Interaktion aus. Dazu kannst du die Klasse Terminal verwenden: Terminal.java (http://docs.freitagsrunde.org/javakurs/2007/vorgaben/Terminal.java) . Um die Klasse benutzen zu können, muss sie im gleichen Verzeichnis liegen wie dein Programm auch.

Beispiel zum Einlesen eines Integerwertes (int):

```
int number = Terminal.readInt();
```

oder mit einem Hinweis:

```
int number = Terminal.askInt("Please enter a number: ");
```

Probiert einfach mal aus, was die Klasse so alles kann. Übrigens ist sie sehr gut (auch Deutsch) kommentiert. Also einfach mal rein gucken.

Ziel der Aufgabe ist es, dass der Benutzer auf der Konsole aufgefordert wird, eine Jahreszahl einzugeben und draufhin eine Antwort bekommt, ob die eingegebene Jahreszahl ein Schaltjahr ist, oder nicht.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Crack the Password

Aus FreitagsrundenWiki

Crack the Password!

Nachdem du dich erfolgreich in Prof. Peppers' Computer gehackt hast, stehst du jetzt vor der letzten Barriere, bevor du dir die Musterlösung pünktlich vor der Klausur saugen kannst.

Der folgende Code überprüft das Passwort, mit dem du dich einloggen kannst. Alles was du aus internen Quellen weißt, ist, dass es nur 4 gültige Zahlen zwischen 1 und 1000 gibt um dich einzuloggen.

Kopiere folgende Methode, die überprüft ob die Passworteingabe richtig war, in dein Programm.

```
public static boolean checkPasscode(int H4X0R){
        // returns true if passcode is valid
        boolean result = false;
        for(int E1337=42; E1337<=(52^(0x6c)); E1337+=(3<<(14%6))){
                if(result=((++E1337|E1337+(2>>>1))^(1+H4X0R))==(123456789&0))
                        break;
        }
                return result;
}
```

1. Finde die 4 gültigen Zahlen mittels Brute-Force und gib sie auf der Konsole aus (Brute-Force = Rohe Gewalt, darunter versteht man das simple Ausprobieren aller möglichen Kombinationen).
2. Für die absoluten Cracks: Wie funktioniert der Algorithmus?

Hinweis: Eine Übersicht zu den verwendeten Java-Operatoren könnte nützlich sein:
http://www.java-forum.org/de/viewtopic.php?t=1545

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Rekursion vs. Iteration

Aus FreitagsrundenWiki

Das Ziel dieser Aufgabe ist es, dass Ihr möglichst viele Eurer bisher erworbenen Java-Kenntnisse Schritt für Schritt anwendet. Abschließend sollt Ihr noch einen kleinen Benchmark einer rekursiv und iterativ implementierten Funktion durchführen.

Inhaltsverzeichnis

Aufgabenstellung

Seht euch die Codebeispiele an und findet heraus, was der Code tut. Schreibt den Code anschließend neu, sodass der Code übersichtlicher und lesbarer ist.

Gegeben ist folgende kleine Funktion:

```
public static int doSomethingClever(int a){
   int b;
   if(a <= 1){
      if(a == 1){
         b = calcSomethingNice(a);
      }else{
         b = getSomeSmartInt(a - 1);
      }
   }else{
      return theSmartestMethodsAlwaysNeedRidiculouslyLongNames(a);
   }
   return b;
}
```

```
public static int getSomeSmartInt(int c){
   return ++c;
}
```

```
public static int calcSomethingNice(int d){
   int i = 1;
   for(i = d * 11; i > d; i--){
      i--;
   }
   return i;
}
```

```
public static int theSmartestMethodsAlwaysNeedRidiculouslyLongNames(int e){
   int container = doSomethingClever(e - 1);
   return container + doSomethingClever(e - 2);
}
```

Offensichtlich ist hier programmiertechnisch einiges schief gegangen. Denn obwohl der Code richtig

funktioniert, ist er nur schwer leserlich. Folgende Fehler wurden gemacht:

- nichtssagende Bennung der Variablen und Methoden,
- keine Kommentare,
- übertrieben Auslagerung in Extra-Methoden.

All das sollt Ihr jetzt schrittweise ändern:

1. Versucht, ohne den PC mit einigen Handsimulationen herauszufinden, was `doSomethingClever` eigentlich berechnet.
   - Hinweis: Beginnt mit kleinen Zahlen wie 0, 1, 2 oder 3 und macht mehrere Durchgänge.
   - Tipp: Es wird stets das n-te Glied einer bekannten Zahlenfolge berechnet.
2. Integriert zuerst die beiden Hilfsfunktionen `getSomeSmartInt` und `calcSomethingNice` in die Hauptfunktion, da beide ziemlich überflüssig sind.
3. Löst die verschachtelte Rekursion von `doSomethingClever` und `theSmartest...` auf.
4. Lasst Euch "sprechende" Namen sowohl für die Variablen, Methode(n) und als auch die Klasse einfallen.
5. Fügt ausreichend viele und prägnante Kommentare hinzu, bis das Programm für Euch gut lesbar ist.
6. **Wichtig:** Holt Euch nun einen **Tutor** ran, zeigt ihm Euer Programm und diskutiert die Lesbarkeit des Codes. Er kann euch sicherlich nützliche Tipps geben, was Ihr in Zukunft beim Programmieren unbedingt beachten solltet.

Eine Iterative Alternative

Schreibt nun ein Programm, das die gleiche Zahlenreihe nicht-rekursiv berechnet. Verwendet dazu ein Array von Integern und folgende Idee:

- Initialisiert ein Array, das alle Folgenglieder (bei 0 anfangend) bis zum Gesuchten aufnehmen kann.
- Berechnet die Folgenglieder in einer Schleife Eurer Wahl aufsteigend bei 0 beginnend und speichert sie in dem Array wie folgt: Folgenglied 0 steht an Position 0 im Array, Folgenglied 1 steht an Position 1 im Array, usw.

Hinweis: Wenn Ihr mit der Programmidee oder der Implementation Probleme habt, zögert nicht, einen Tutor zu fragen.

Benchmark und Auswertung

Schreibt nun ein letztes kurzes Programm, das zuerst die Folgenglieder 5, 10, 15, 20, 30 und 40 iterativ und dann noch einmal rekursiv berechnet und sofort auf der Console ausdruckt.

Diskutiert die Geschwindigkeitsunterschiede und die möglichen Vorteile der iterativen Implementation mit einem Tutor.

Tipp: Sollte Euer iteratives Programm nicht binnen weniger Sekunden sämtliche Zahlen berechnet haben, lasst unbedingt einen Tutor kurz drüberschauen.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Lineare Funktionen

Aus FreitagsrundenWiki

Aufgabenstellung

In dieser Aufgabe soll ein Programm geschrieben werden, welches verschiedene Rechenoperationen durchführt, die man mit linearen Funktionen so anstellen kann. Versucht einzelne Funktionalitäten eures Programms in sinnvolle Methoden auszulagern.

Eine lineare Funktion ist wie folgt definiert: y = m*x + n

1. Sei m=2 und n=-5 gegeben. Schreibe ein Programm, welches die Funktionswerte von x = 0, 1, 2, ..., 20 in einem Array speichert.
2. Erweitere dein Programm so, dass gezählt werden kann wie viele Funktionswerte größer 0 sind
3. Nun soll man den Graphen an der y-Achse beliebig verschieben können. Schreibe ein Programm, welches das Array und somit alle Funktionswerte von 0 bis 20 aktualisiert.
4. Desweiteren soll das Programm die Fläche zwischen dem Graphen und der x-Achse berechnen und ausgeben können.
5. Gegeben seien die vier Integer-Werte x1 und y1 sowie x2 und y2. Schreibe eine Methode, welche m und n bestimmt und in der Form y = m*x + n auf der Konsole ausgibt.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Fibonacci

Aus FreitagsrundenWiki

Fibonacci-Zahlen

Schreibt ein Programm, welches die ersten acht Folgeglieder der Fibonacci-Folge ausgibt. Zur Erinnerung:

- fibonacci(0) = 0
- fibonacci(1) = 1
- und für n > 1 gilt: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)

Aufgaben

1) Es soll eine Methode mit dem Namen fibonacci geschrieben werden. Diese Methode soll genau eine Variable vom Typ **int** und dem Namen *n* übergeben bekommen. Als Rückgabewert soll die Methode den zu *n* gehörigen Wert der Fibonacci-Folge liefern.

2) Nachdem nun die Methode mit der passenden Signatur geschrieben wurde, muss sie noch die nötige Berechnung tätigen. Im diesem Beipiel soll eine rekursive Lösung verwendet werden. Die obige Definition der Fibonacci muss lediglich in Java Code übersetzt werden.

3) Testet die fibonacci-Methode, ob sie korrekt funktioniert, indem ein paar Werte berechnet werden.

4) Der Datentyp int beinhaltet auch negative ganze Zahlen. Was passiert, wenn die fibonacci-Methode einen negativen Eingabewert erhält? Bitte erst darüber nachdenken und danach ausprobieren. Korrigiert die fibonacci-Methode, sodass sie bei Eingabe von negativen Zahlen eine Fehlermeldung ausgibt und einen Fehler-Code zurückgibt. Hier wäre etwa -1 als Ergebnis vorstellbar.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Brainfuck

Aus FreitagsrundenWiki

Aufgabenstellung

In dieser Aufgabe bauen wir uns eine simple Turingmaschine. Die Turingmaschine besitzt dabei nur ein Arbeitsband (workingTape) der Länge 10. Es existiert ein Lese/Schreibe-Kopf (head), der über das workingTape gleiten kann, dort Werte ändern und Werte ausgeben soll. Bewegt sich der Kopf über ein Ende des Arbeitsbandes hinaus, so taucht er auf der anderen Seite wieder auf (wenn also z.B der head auf das 11. Feld zeigt, dann soll er wieder auf das 1. zeigen).

Die Maschine selbst wird durch einen einfachen Programmcode gesteuert, der aus 5 Befehlen besteht:

- > Dieser Befehl bewegt den head ein Feld nach rechts
- < Dieser Befehl bewegt den head ein Feld nach links
- + Dieser Befehl erhöht den Wert des aktuellen Feldes um 1
- - Dieser Befehl verringert den Wert aktuelles Feldes um 1
- # Dieser Befehl gibt den aktuellen Wert auf der Konsole aus

Der Programmcode der die Turingmaschine steuert, soll in einem Array namens *sourceCode* stehen, der Code soll Schritt für Schritt durchgegangen und ausgeführt werden. Auf dem Arbeitsband werden Werte vom Typ **char** gespeichert, die anfangs alle den Wert 'a' haben. Der head zeigt am Anfang auf das erste Feld auf dem workingTape.

Hinweis 1: char-Werte werden intern als Zahlenwerte gespeichert, deshalb könnt ihr ohne Probleme sowas wie **char example = 'a'+1;** schreiben. Wenn ihr wissen wollt, welche Zeichen durch welche Zahlen repräsentiert werden, dann schaut in die ASCII-Tabelle.

Jetzt solltet ihr auch in der Lage sein, mit eurer eigenen Turingmaschine ein "Hello World"-Programm zu schreiben. Wenn ihr eure Turingmaschine eigentständig erweitern wollt, dann schaut lest mehr über die Sprache Brainfuck.

Hinweis 2: Zum Testen verwendet einfach diese Initialisierung der Variable Source-Code. Wenn ihr alles richtig gemacht habt, dann sollten die Buchstaben a, b und c ausgegeben werden. Die Schreibweise mit den geschweiften Klammern und den Kommawerten legt automatisch ein Array an und füllt es mit den beschriebenen Werten (Deklaration & Initialisierung in einem Schritt):

```
char[] sourceCode = {'#', '+', '#', '+', '#'};
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

FehlerKategorie

Um den Char um einen Wert zu erhöhen müsst ihr eure Variable zurück casten.

```
char Example = (char)('a'-1);
```

Das stimmt nicht man kann auch das schreiben:

```
char Example = 'a' - 1;
```

Von „http://wiki.freitagsrunde.org/Javakurs2007/Brainfuck“

Javakurs2007/Gauß-Algorithmus

Aus FreitagsrundenWiki

Aufgabenstellung

Ziel der Aufgabe ist es, eine (oder mehrere) Methoden zu schreiben, welche am Ende ein lineares Gleichungssystem nach Gauß lösen. Der Gauß-Algorithmus wird auch als Gaußsches Eliminationsverfahren bezeichnet.

Zunächst allerindings einige kleinere Aufgaben, die das ganze etwas Vorbereiten.

1.) Schreibt eine Methode printVektor(), die ein übergebenes, 1-dimensionales Array vom Typ double zeilenweise auf der Konsole ausgibt.

Mit welchen der während des Vortrages kennengelernten Schleifentypen lässt sich diese Methode realisieren?

2.) Schreibt nun eine Methode printMatrix(), die eine übergebenes, 2-dimensionales Array vom Typ double auf der Konsole ausgibt.

Testet eure beiden Methoden, indem ihr verschiedene Array deklariert und initialisiert, und anschließend den Methoden printVector() bzw. printMatrix() als Parameter übergebt.

3.) In dieser Aufgabe wollen wir eine Methode entwickeln, das mit Hilfe des gaußschen Eliminationsverfahrens ("Gauß-Algorithmus") ein lineares Gleichungssystem löst.

Ein lineares Gleichungssystem lässt sich in der Form Ax=b darstellen, wobei A die Koeffizientenmatrix, b der Zielvektor und x der gesuchte Lösungsvektor ist. Koeffizientenmatrix und Zielvektor lassen durch ein 2-dimensionales, bzw. durch ein 1-dimensionales Array darstellen. Eure Methode soll Koeffizientenmatrix und Zielvektor als Parameter übergeben bekommen, und den gesuchten Lösungsvektor als Rückgabewert haben. Bei eurer Implementierung braucht ihr die Pivotisierung vorerst nicht zu beachten.

Hinweise:

Der Algorithmus arbeitet in 2 Etappen:

a) Erzeugen der Zeilen-Stufen-Form

b) Bestimmen des Lösungsvektors durch rückwärtseinsetzen

Nutzt das bei Wikipedia angegebene Bespiel, um die korrekte Funktionsweise eures Programms zu testen und eure in 1.) und 2.) entwickelten Methoden, um Matrix und Vektoren an bestimmten Stellen des Algorithmus zu Debugging-Zwecken auszugeben.

4.) Bis jetzt kann euer Algorithmus nur eindeutig lösbare Gleichungssysteme lösen. Erweitert euren Algorithmus um einige Fallunterscheidungen, so dass dieser bei Gleichungssystemen mit keiner bzw. undendlich vielen Lösungen jeweils den Rückgabewert "null" hat.

Hinweise:

Aussagen über die Lösbarkeit eines linearen Gleichungssystems kann man nach Erzeugen der Zeilen-Stufen-Form treffen.

1.) Gibt es sogenannte Nullzeilen (alle Elemente einer Zeile der Koeffizientenmatrix und das Element der entsprechenden Zeile des Zielvektors sind 0), so existieren unendlich viele Lösungen.

2.) Sind alle Elemente einer Zeile der Koeffizientenmatrix gleich 0, das Element der entsprechenden Zeile des Zielvektors jedoch nicht, so hat das Gleichungssystem keine Lösung.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Zahlenumrechner

Aus FreitagsrundenWiki

In dieser Aufgabe soll ein Programm erstellt werden, das Dezimalzahlen in andere Darstellungen (Binär, Hexadezimal) umrechnet.

1. Schreibe eine Funktion 'dezimal_nach_binaer(int dezimal)', die eine Dezimalzahl als Eingabe bekommt und eine Binärzahl als String zurückgibt. Teste deine Funktion.
2. Schreibe eine Funktion 'dezimal_nach_hexadezimal(int dezimal)', die eine Dezimalzahl als Eingabe bekommt und eine Hexadezimalzahl als String zurückgibt.
3. Erweitere das entstandene Programm um ein Menü, so dass die Zahlen und die gewünschte Umrechnung vom Benutzer eingegeben werden können.

Hinweis: Du kannst die Klasse Terminal von http://docs.freitagsrunde.org/javakurs/2007/vorgaben/Terminal.java in das Verzeichnis kopieren, in welchem du dein Programm schreibst und die Methode `Terminal.readString()` benutzen, um eine Tastatureingabe von der Kommandozeile zu lesen. Wenn dich interessiert, warum das so funktioniert, so erfährst du es hier.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs 2007/Democracy

Aus FreitagsrundenWiki

Fehler im System

Führende PolitikwissenschaftlerInnen haben festgestellt, dass Demokratie einfach nicht funktioniert. Deshalb haben sie sich überlegt, dass es klüger wäre, wenn einfach der/die Älteste entscheidet. Sie haben ein Programm geschrieben, das unter den Parteivorsitzenden der großen Parteien, den/die älteste heraussucht. Leider sind sie keine gelernten Java-Programmierer und haben es nicht geschafft, lauffähigen Code zu produzieren. Findest du die Fehler im System?

```
public class DemocracyV2 {
	public static void main(String[] args) {
		String[] names = String[5];
		int[] ages = int[5];

		generateNamesAndAges(names, ages);
		System.out.println(whosTheOldest(names, ages));
	}

	private static void generateNamesAndAges(String[] names, int[] ages) {
		names[0] = Angela;
		ages[0] =  52;

		names[1] = Kurt;
		ages[1] = 58;

		names[2] = Lothar;
		ages[2] = 66;

		names[3] = Doppelspitze Claudia und Reinhard;
		ages[3] = 51 + 54;

		names[4] = Guido;
		ages[4] = 45;
	}

	private static void whosTheOldest(String[] names, int[] ages) {
		int oldest;
		for(int i; i <= 5; i++) {
			if ( oldest < ages[i] ) {
				int indexOfOldest = i;
			}
		}

		return "Oldest and wisest person is " + names[indexOfOldest] + " with an age of " + a
	}
}
```

Nachdem du die Fehler gefunden hast, wie könntest du sie beseitigen und so die Gesellschaft vor ihrem Untergang bewahren? Wie könnte mit dem Fall umgegangen werden, dass zwei PolitikerInnen gleich alt sind?

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Quersumme

Aus FreitagsrundenWiki

1. Aufgabe

Ein Programm zur Bestimmung der Quersumme einer natürlichen Zahl soll geschrieben werden. Schritt für Schritt soll dieses Programm um weitere Funktionen erweitert werden.

1. Spalte die einzelnen Ziffern mittels Modulo-Operation ab und addiere sie, um die Quersumme zu bestimmen.
2. Gib alle Zahlen von 0 - 1000 aus, welche die Quersumme 15 haben.
3. Gib alle Zahlen von 0 - 1000 aus, deren Quersumme ein Vielfaches von 7 ist.
4. Welche Quersumme der Zahlen von 0 - 1000 kommt am häufigsten vor? (Tipp: Überlegt Euch, wie viel verschiedene Quersummen vorkommen können, erstellt ein Array dieser Größe und speichert dort die Anzahl der Vorkomnisse)
5. Die iterierte Quersumme wird auch Ziffernwurzel genannt (Abk. zw) Beispiel:
   47 --> 4 + 7 = 11 --> 1 + 1 = 2, also zw(47)=2
   Schreibt ein Programm, welches die Ziffernwurzel für eine beliebige Zahl bestimmt.
6. Das Querprodukt ist wie folgt definiert: 68 = 6 * 8 = 48. Es gibt Zahlen, bei denen die Summe aus Quersumme und Querprodukt wieder die Zahl selbst ergibt Beispiel: 79 = 7 + 9 + 7*9 = 79. Gibt es weitere Zahlen zwischen 0 und 1000 mit dieser Eigenschaft? Wenn ja, welche ?

2. Aufgabe

Beim Gesellschaftsspiel "Die Böse Sieben" sitzen die Teilnehmer im Kreis und zählen reihum. Jede Zahl, welche durch 7 teilbar ist oder die 7 als Ziffer enthält, muss übersprungen werden. Schreibe ein Programm, welches für eine beliebige Zahl angibt, ob die nächste übersprungen werden muss oder nicht.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Appleman

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Vorüberlegungen

Lest euch zunächst etwas Wissen über Komplexe Zahlen an. Legt besonderes Augenmerk auf das Quadrieren von komplexen Zahlen in Normalform, andere Darstellungsformen von komplexen Zahlen werden hier nicht benötigt.

Appleman

Die Mandelbrot-Menge, oder auch Apfelmännchen genannt, ist eine ganz bestimme Menge von komplexen Zahlen. Sie gehört zu den sogenannten Fraktalen und man kennt sie aus unzähligen Darstellungen in Printmedien. Für eine genaue Definition und Berechnungsvorschrift sei auf die Wikipedia verwiesen: Mandelbrot-Menge, Definition über Rekursion. In dieser Aufgabe soll es darum gehen, ein Programm zur Darstellung eines solchen Apfelmännchens zu programmieren.

Theorie...

1. Lies dir die Definition der Mandelbrot-Menge durch.
2. Überlege dir, wie du die komplexen Zahlen c und z_n mit Java darstellen kannst. Welche Datentypen sind zur Berechnung notwendig?
3. Überlege dir, wie du die Folge berechnest (quadrieren komplexer Zahlen).
4. Überlege dir die Abbruchbedingungen für die Berechnung der Folge und was aus den beiden Fällen folgt.
5. Entwirf die grundsätzliche Struktur des Algorithmus. Plane dabei eine Methode *pointIteration(...)* ein, die die Iteration für jeden Punkt durchführt.

... und Praxis

Nachdem du nun hoffentlich den theoretischen Teil hinter dir hast, kannst du mit der Implementierung beginnen. Zur Grafikdarstellung gibt es die einfach zu bedienende Klasse "Pad", die ein Fenster erstellt, in dem du mit einfachen Methoden zeichnen kannst.

Falls du noch nicht die ÜBB-Klassen in deinem Javakurs-Verzeichnis hast, lade dir von der ÜBB-Seite (http://uebb.cs.tu-berlin.de/books/java/) die Klassen Pad, Point und Terminal herunter und lege sie in dem

Verzeichnis ab, in dem auch dein Programm entstehen soll. Wenn du dies getan hast, steht dir die Grafikfunktionalität zur Verfuegung.

```
//Mit
Pad drawPad = new Pad();
//wird ein neues Fenster erzeugt.

drawPad.setPadSize(int width, int height);
//setzt die Größe des Fensters.

drawPad.setVisible(true);
//zeigt das Fenster an.

drawPad.drawDot(int x, int y);
//zeichnet einen Punkt an der Stelle (x,y).
```

Du musst noch nicht verstehen, was dies sprachlich genau bedeutet. In Vorträgen 5 und 6 sollte dann klar werden was, drawPad ist.

1. Implementiere die *pointIteration(...)*-Methode
2. Teste die Methode. Teste dabei Werte die eindeutig innerhalb bzw. außerhalb der Mandelbrot-Menge liegen.
3. Lass diese Methode nun für jeden Punkt im Fenster ausführen. Dabei wird der Parameter *c* aus der aktuellen x- und y-Koordinate zusammengesetzt.
4. Zeichne die Punkte der Mandelbrot-Menge.
5. Fertig :)

Hinweise

Koordinaten im Computer

Am Computer werden Koordination, historisch bedingt, anders behandelt als im rechtwinkligen Koordinatensystem. Koordination im Computer werden von links nach rechts, als X-Achse und **von oben nach unten** als Y-Achse berechnet. Beachte dies bei der Verwendung von *drawPad.drawDot(x,y)*.

```
(0,0)-------------------- X
|                  (640,0)
|
|
|
|
| (0,480)          (640,480)
Y
```

Quadrieren komplexer Zahlen

.

.

.

.

ACHTUNG: Teaser - lies hier nur weiter, wenn du nicht weiter kommst oder ... dir den Spass verderben willst.

.

.

.

.

.

.

.

.

Komplexe Zahlen werden in Normalform als $z = a + b\,i$ dargestellt. Um z zu quadrieren, kann man sich entweder einen eigenen Datentyp schreiben, der die Multiplikation implementiert ODER man macht es sich einfach und berechnet das Quadrat Komponentenweise.

$z_{n+1} = z_n^2 + c = (a_{z_n}+b_{z_n}\mathrm{i}) + (a_c + b_c\mathrm{i})^2 = a_{z_n}^2 - b_{z_n}^2 + 2a_{z_n}b_{z_n}\mathrm{i} + a_c + b_c\mathrm{i} = (a_{z_n}^2 - b_{z_n}^2 + a_c) + (2a_{z_n}b_{z_n} + b_c)\mathrm{i}$

$z_{n+1} = a_{z_{n+1}} + b_{z_{n+1}}\mathrm{i}$ laesst sich also komponentenweise berechnen mit:
$a_{z_{n+1}} = (a_{z_n}^2 - b_{z_n}^2 + a_c)$
$b_{z_{n+1}} = (2a_{z_n}b_{z_n} + b_c)$

In Java sieht dies wie folgt aus:

```
double cReal = startX;
double cImg = startY;
double zReal = 0;
double zImg = 0;

zReal = zReal*zReal - zImg*zImg + cReal;
zImg = 2*zReal*zImg + cImg;
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/DiffusionLimitedAggregation

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Einleitung

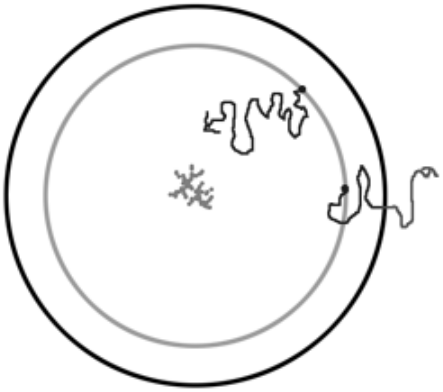
Als Diffusion Limited Aggregation (http://en.wikipedia.org/wiki/Diffusion-limited_aggregation) , kurz DLA, bezeichnet man physikalische Vorgaenge bei denen sich Teilchen durch Brownsche Bewegung zufällig bewegen und bei Kontakt mit anderen Teilchen anlagern. Diese Vorgänge können leicht am Computer simuliert werden und erzeugen sehr schöne Gebilde. In dieser Aufgabe soll es darum gehen, eine solche DLA-Simulation zu programmieren.

Eine solches Programm könnte z.B. so aussehen:

Die Theorie

Um ein möglichst gleichmäßiges Wachstum des DLA-Fraktals zu erreichen, setzt man in der Mitte des Bildschirms einen Keim und lässt auf einer Kreisbahn (grün) um das Zentrum einen neuen Partikel erscheinen.

Diesen Partikel bewegt man zufällig über die Ebene (blaue Bahnen). Entfernt sich der Partikel zu weit vom Zentrum (schwarzer Kreis), so wird er verworfen (rote Bahn) und ein neuer Partikel erzeugt.



Die Radien der Kreise sollten mit der Größe des DLA-Fraktals wachsen. Dazu überprüft man bei jeder Anlagerung eines Partikels, ob sich das Fraktal vergrößert hat und ändert gegebenenfalls den Radius. Ein Partikel hat außer seiner Existenz keine weiteren besonderen Eigenschaften.

Vorüberlegungen

Soweit so gut. **Nun bist du ein wenig gefragt!**
Lies dir die folgenden Fragestellungen durch, und denke darüber nach.

1. Die Partikel werden in einer 2-dimensionalen Ebene gespeichert. Welche Datenstruktur aus Java kann man besonders gut verwenden, um viele Partikel abzuspeichern?
2. Wie würde man überprüfen, ob ein Partikel einen anderen berührt?
3. Wie kann man einen Partikel über die Fläche bewegen? Wie erzeugt man eine Brownsche Bewegung?

.

.

.

.

.

.

.

.

Lösungen

1. Ein 2-dimensionales Array von genau der Größe der zu simulierenden Fläche bietet sich an. Als Datentyp würde *boolean* genügen, denn wir wollen nur wissen, ob sich an der Position (x,y) ein Partikel befindet oder nicht. Dem Autor ist aber *int* sympathischer ;)
   int[][] field = new int[256][256];
2. Jedes der 8 benachbarten Felder um die Position des gerade betrachteten Partikels muss überprüft werden. Ist dort ein Partikel vorhanden, so enthält das Feld einen Wert >0.
3. Math.random() (http://javadocs.org/Math) erzeugt zufällige Zahlen. Mit deren Hilfe kann man über eine

Fallunterscheidung die X- und Y-Koordinate des Partikels inkrementieren oder dekrementieren. Dadurch bewegt sich das Teilchen einen Schritt weiter.

Die Praxis

Da diese Aufgabe ziemlich komplex ist, werden schrittweise alle nötigen Bausteine entwickelt.

Grafik

Zur Grafikdarstellung gibt es die einfach zu bedienende Klasse "Pad", die ein Fenster erstellt, in dem du mit einfachen Routinen zeichnen kannst.

Falls du noch nicht die ÜBB-Klassen in deinem Javakurs-Verzeichnis hast, lade dir von der ÜBB-Seite (http://uebb.cs.tu-berlin.de/books/java/) die Klassen Pad, Point und Terminal herunter und lege sie in das Verzeichnis ab, in dem auch dein Programm entstehen soll. Wenn du dies getan hast, steht dir die Grafikfunktionalität zur Verfügung. In dieser Aufgabe werden die notwendigen Grafikbefehle vorgeben, du musst dir keine Sorgen machen, dass du dies nicht schaffst.

Schritt fuer Schritt

1. Erstelle eine Klasse *DLA*
2. Füge in der ersten Zeile folgenden import hinzu: *import java.awt.Color;*. Was genau dies bedeutet, wird in Vortrag 6 erklärt.
3. Schreibe eine Methode *public static int limit(int value, int lower, int upper)*, die überprüft, ob sich der Wert *value* innerhalb der Grenzen *lower* und *upper* befindet. Ist dies nicht der Fall, soll der Wert auf diesen Bereich beschränkt zurückgegeben werden.
4. Schreibe eine Methode *public static boolean testPixel(int[][]field, int x, int y)*, die überprüft, ob an einem bestimmten Punkt im 2-dimensionalen Array ein Wert >0 vorliegt. Fange dabei ungültige Werte für x und y ab. Benutze dazu die *limit* Methode.
5. Teste die *testPixel(...)*-Methode!
6. Schreibe eine Methode *public static boolean isTouching(int[][] field, int x, int y)*, die überprüft, ob sich in einem der 8 umliegenden Felder ein Partikel befindet. Benutzte dabei die *testPixel(...)* Methode.
7. Teste diese Methode!
8. Schreibe eine Methode *public static boolean inRange(int x, int y, int maxR, int maxX, int maxY)*, die überprüft, ob sich die Koordination *x* und *y* in einem Rechteck der Seitenlänge *maxX* und *maxY* innerhalb eines Abstandes *maxR* vom Mittelpunkt befindet.
9. Teste diese Methode!
10. Schreibe eine Methode *public static int moveRandom(int movePixel)*, die eine Zahl (Koordinatenkomponente) mit **gleichen** Wahrscheinlichkeiten inkrementiert (+1), unverändert lässt (+/-0) oder erniedrigt (-1).
11. Test kurz diese Methode.
12. Schreibe die Methode *public static void setPixel(Pad drawPad, int[][]field, int x, int y, Color c)*. Diese überprüft noch einmal die Gültigkeit der Koordinaten *x* und *y*, platziert dann im Array *field* an diesen Koordinaten einen Partikel und zeichnet den Partikel auf dem *drawPad* wie folgt:

```
drawPad.setColor(c);
drawPad.drawDot(x, y);
```

Main

Schreibe nun die main-Methode für das DLA-Programm. Wenn dir die Struktur des Programms schon klar ist, dann kannst du gern allein versuchen, das Problem zu lösen. Hier sei noch ein Code-Fragment zum Berechnen von Koordinaten auf einem Kreis gegeben:

```
//einen Punkt auf einem Kreis um den Mittelpunkt mit dem Radius
//maxR+20 erzeugen
double x = Math.toRadians(Math.random()*360);
int movingPixelX = limit((int)(Math.cos(x)*(maxR+20))+maxX/2, 0, maxX);
int movingPixelY = limit((int)(Math.sin(x)*(maxR+20))+maxY/2, 0, maxY);
```

zusätzlich noch Code um das Grafikfenster zu initialisieren:

```
//Fenster fuer Grafikausgabe vorbereiten und anzeigen
Pad drawPad = new Pad();
drawPad.setBackground(Pad.black);
drawPad.setPadSize(maxX, maxY);
drawPad.setVisible(true);
drawPad.setColor(Pad.white);
```

Die Klasse Pad bietet ausserdem einige Farbwerte vom Typ *Color* an, dir du einfach so benutzen kannst.

```
Pad.white
Pad.black
...
```

Falls dir noch nicht ganz klar ist, wie alles zusammen wirken soll, benutze folgenden Rumpf, der schon die Grafik-Funktionalität und andere Vorgaben enthält:

```
import java.awt.Color;

class DLA {
	public static void main(String[] args) {
		//Feld fuer Kollisionsueberpruefung erstellen
		int[][] field = ...


		//Grenzen fuer Koordinaten der Partikel aus field berechnen (.length verwenden)
		int maxX = ...
		int maxY = ...

		//Fenster fuer Grafikausgabe vorbereiten und anzeigen
		Pad drawPad = new Pad();
		drawPad.setBackground(Pad.black);
		drawPad.setPadSize(maxX, maxY);
		drawPad.setVisible(true);
		drawPad.setColor(Pad.white);


		//"Samen" erzeugen. Also den ersten Partikel in der Mitte des Feldes setzen (Color: P
		setPixel(...);

		//Radius des Partikels der am weitesten vom Zentrum entfernt ist
		int maxR=0;

		//Schleifenkopf, solange maxR nicht den Rand des Fensters
		//beruehrt, sollen neue Punkte gezeichnet werden
		while(...) {
			//einen Punkt auf einem Kreis um den Mittelpunkt mit dem Radius
			//maxR+20 erzeugen
			double x = Math.toRadians(Math.random()*360);
			int movingPixelX = limit((int)(Math.cos(x)*(maxR+20))+maxX/2, 0, maxX);
			int movingPixelY = limit((int)(Math.sin(x)*(maxR+20))+maxY/2, 0, maxY);

			//Schleife zum bewegen des Partikels
			//while(!isTouching(...)) && inRange(..., ..., maxR+30, ..., ...))
			{
				//Partikelkoordinaten zufaellig bewegen (moveRandom)
				movingPixelX = ...
				movingPixelY = ...
			}

			//Schleife zuende, ueberpruefe ob eine Beruehrung vorliegt:
			if(isTouching(...)) {
				//Farbwertberechnung fuer 1337 blau-Effekt
				int k = limit(255-(int)(255*((double)maxR*2)/(double)maxX), 0, 255);
				Color c = new Color(k, k, 255);

				//den Partikel setzen und zeichnen
				setPixel(...);

				//Radius zum Mittelpunkt berechnen,
				//wenn groesser als altes Maximum, ueberschreiben
				...
			}
		}

		return;
	}
...
}
```

Hinweis: Koordinaten im Computer

Am Computer werden Koordination, historisch bedingt, anders behandelt als im rechtwinkligen Koordinatensystem. Koordination im Computer werden von links nach rechts, als X-Achse und **von oben nach unten** als Y-Achse berechnet. Beachte dies bei der Verwendung von *drawPad.drawDot(x,y)*.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Palindrome

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Einleitung

Palindrome sind Wörter, die rückwärts und vorwärts gleich sind, z.B. Rentner.

Später brauchst du solche Palindrome als Eingabewerte für das Programm, das du schreiben sollst. In der Wikipedia gibt es dafür eine Liste mit Palindromwörtern.

Aufgabenstellung

- Schreibe ein Programm, das ein Wort umdreht und dann wieder ausgibt. War das eingegebene Wort ein Palindrom, so sollte das eingegebene Wort und das wieder ausgegebene Wort gleich sein (Hinweis: Der Vergleich von Strings mit == funktioniert nicht. Da müsst ihr euch selbst etwas überlegen, oder euren Tutor fragen).
- Überlege dir zuerst, wie in Vortrag 4 gelernt, wie du das Programm schreiben willst, z.B. welche Methoden du brauchst und was diese tun. Solltest du gar keine Ahnung haben, wie man das Problem angehen könnte, dann lass dir einfach von einen Tutor deines Vertrauens einen Tipp geben.

Fortgeschritten:

- Lies das umzudrehende Wort von der Konsole ein.

Tipps:

Speichern der Länge eines Strings

```
String palindrom = "rentner";
int length = palindrom.length();
```

Lesen des n-ten Buchstabens

```
int n = 5;
char c = palindrom.charAt(n);
```

Lesen von der Konsole

Wenn ihr beim Ausführen eures Programms in der Konsole einen Parameter mit übergebt, so findet ihr ihn an Position [0] im String-Array eurer main-Methode wieder.

Beispiel:

```
java Palindrome rentner
```

-> arguments[0] enthält rentner.

Du kannst auch die Klasse Terminal (http://docs.freitagsrunde.org/javakurs/2007/vorgaben/Terminal.java) von in das Verzeichnis kopieren, in welchem du dein Programm schreibst und die Methode `Terminal.readString()` benutzen, um eine Tastatureingabe von der Kommandozeile zu lesen. Wenn dich interessiert, warum das so funktioniert, so erfährst du es hier.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Hangman

Aus FreitagsrundenWiki

Hangman

Schreibe das in der Vorlesung besprochene Spiel Hangman für die Konsole.
Es soll von einem Spieler beliebig oft spielbar sein. In jedem Spielschritt soll die verbleibene Menge an Rateversuchen, sowie die bereits erratenen Buchstaben des Lösungswortes in der Form "HAN__AN" angezeigt werden. Die verbleibenden Rateversuche können einfach in Form einer Zahl angegeben werden.

1. Mache dir Notizen zum Spielablauf einer Hangman-Spielrunde.
2. Skizziere den Programmablauf.
3. Notiere den mittlerweile besser durchdachten Programmablauf möglichst übersichtlich und mit möglichst vielen Schritten.
4. Mache dir Gedanken darüber, welche Schritte sehr komplex sind und entwerfe auch für sie einen Programmablauf.

1. Kopiere und entpacke die Vorgabedateien unter http://docs.freitagsrunde.org/javakurs/2007/vorgaben/Hangman.tar in ein Verzeichnis deiner Wahl.
2. Benutze zum Schreiben deines Programmes die Vorgabedatei "Hangman.java" im Verzeichnis "Hangman".

Anmerkung: Das Einlesen von Zeichen über die Konsole gestaltet sich in Java üblicherweise etwas umständlich. Aus diesem Grund enthalten die Vorgaben zu dieser Übung eine Bibliotheksklasse "Terminal", welche dir einige leicht zu benutzende Funktionen für die Arbeit mit der Konsole zur Verfügung stellt. Solange sich die Datei "Terminal.java" im selben Verzeichnis wie deine Datei "Hangman.java" befindet, hast du Zugriff auf alle Funtionen der Bibliotheksklasse, indem du sie in der Form "Terminal.gewünschteFunktion()" aufrufst.

- public static String readString () // liest ein Wort von der Konsole
- public static char readChar () // liest ein enzelnes Zeichen von der Konsole

Fortgeschritte

Achtung, schwer!

- Gib die verbleibenden Rateversuche in Form eines ASCII-Hangmans aus.

Tipp: Ein Array zum Speichern der Graphik kann hier außerordentlich hilfreich sein.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine

Scheu, es geht nichts kaputt ;)

lol Rechtschreibfehler bei "Achtung, schwehr!", aber echt coole Aufgabe ;P

Von „http://wiki.freitagsrunde.org/Javakurs2007/Hangman“

Javakurs2007/SelectionSort

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Aufgabenstellung

Ziel dieser Aufgabe ist es, Werte in einem unsorierten Array zu sortieren. Das heißt, euer Programm bekommt ein Array zum Beipiel von ganzen Zahlen (int). Das könnte so aussehen:

```
int[] list = {5, 8, 3, 6, 8, 9, 4, 2};
```

Algorithmus

Bei Wikipedia findet ihr Erklärungen zu der Funktionsweise von Selection-Sort und dort gibt es auch ein Beispiel, welches das Vorgehen des Algorithmus' visualisiert. Falls ihr Fragen haben solltet, fragt euren Tutor, da es keinen Sinn macht, mit Programmieren anzufangen, wenn ihr den Algorithmus noch nicht verstanden habt.

Tipps

- Ihr werdet eine Methode brauchen, die zwei Werte in dem Array tauscht, diese könnte so aussehen:

```
public static void swap(int[] array, int position1, int position 2) {
[...]
}
```

Diese Methode soll das Array so ändern, dass der Wert, der an **position1** stand, dann an **position2** steht, und andersrum.

- Eine weitere Methode, die ihr braucht, ermittelt die Position des kleinesten Wertes in einem Teilarray.

```
public static int findMinimum(int[] array, int startIndex, int endIndex) {
[...]
}
```

startIndex und **stopIndex** sind die Positionen, an dem der zu untersuchende Teilbereich des Arrays anfängt, bzw aufhört. Diese Angaben sind nötig, da ihr später den kleinsten Wert nur in eurem unsortierten Teil des Arrays finden wollt.

- Um besser testen zu können, eignet sich eine Methode, die überprüft, ob ein Array sortiert ist oder nicht.

```
public static boolean isSorted(int[] array) {
[...]
}
```

- Auch eine Methode, die euer Array auf der Konsole ausgibt, sollte nicht fehlen, so könnt ihr am besten Schritt für Schritt überprüfen, ob euer Code richtig arbeitet oder nicht.
- Versucht dabei gleich passende Namen zu verwenden und auch Kommentare nicht zu vergessen. Solltet ihr zu zweit an einem Rechner sitzen, dann sollte die Person, die gerade nicht tippt, den Tippenden auf Fehler und Ungenauigkeiten hinweisen. So kontrolliert ihr euch gegenseitig und euer Code wird besser. Vergesst nicht, euch in regelmäßigen Abständen mit dem Tippen abzuwechseln.

Schließlich überlegt euch, wie ihr eure Methoden am besten testet. Führt z.B. gegenseitige Tests durch. Konntet ihr Fehler finden? --- beheben?

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Cäsar-Chiffre

Aus FreitagsrundenWiki

Im Folgenden soll ein Programm weitgehend selbsttändig entwickelt werden. Lies dir das nötige Wissen zur Cäsar-Chiffre im Web an und überlege dir selbst eine Struktur für dein Programm.

Mache dir auch Gedanken darüber, wie du die einzelnen Teile frühzeitig testen kannst.

1. Erstelle ein Programm, das einen Text als Code im Programmtext enthält, mit einem Passwort durch eine Cäsar-Chiffre verschlüsselt und auf der Konsole ausgibt.

Komfortabler, aber mit ein wenig Aufwand verbunden. Tipps siehe unten :

- der Text soll aus einer Datei eingelesen werden
- der chiffrierte Text soll in einer neuen Datei gespeichert werden.

1. Erweitere dein Programm so, dass es einen mit der selben Chiffre verschlüsselten Text bei gegebendem Passwort entschlüsseln kann und in einer seperaten Datei speichert.
2. Erweitere dein Programm so, dass es in der Lage ist, einen beliebigen mit einer Cäsar-Chiffre verschlüsselten Text mit Hilfe von Häufigkeitsanalyse, ohne das Passwort zu kennen, zu entschlüsseln. Gib auch das gefundene Passwort an.

Hinweis: Unsere Tutoren können dir bei allen auftretenden Javaproblemen helfen, wir können allerdings nicht garantieren, dass sie sich in ihrer Freizeit auch mit Kryptographie beschäftigen.

Verschlüsseln / Entschlüsseln

1. Trage einen von dir verschlüsselten Text mit min. 50 Zeichen hier im Wiki ein und verlinke ihn auf dieser Seite unter deinem Namen. Entschlüssle andere Texte, die hier verlinkt sind.

Wenn du willst, kannst du deinen Chiffrieralgorythmus auch verändern oder einen anderen implementieren, mache das im Link zu deinen damit verschlüsselten Texten aber auch kenntlich. Denke daran, dass du selbst in der Lage sein solltest, deine Texte bei gegebendem Passwort zu entschlüsseln ;)

Tipps:

- Zum Einlesen von Textdateien schaut in der Java API unter *BufferedReader*, *InputStreamReader* und *FileInputStream* nach. Hier ein Beispiel, wie man diese Klassen benutzen könnte (vergesst bei euch die import-Anweisungen nicht):

```
BufferedReader bufferedReader = new BufferedReader(
          new InputStreamReader(
          new FileInputStream("MeineDatei.txt")));
String line;
boolean endOfFile = false;
while ((line = ) != null) {
    line = bufferedReader.readLine();
    if(line == null){
        endOfFile = true;
    } else{
         // in dem String line steht jetzt die
         // aktuelle Zeile des Programms
    }
}
```

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Abzaehlspiel

Aus FreitagsrundenWiki

Problembeschreibung

10000 Leute stehen durchnummeriert im Kreis. Nacheinander muss jeder Dritte gehen. Welche 2 Personen bleiben übrig?

Arbeitsschritte

1. Analysiert das Problem, gibt es noch Fragen oder Ungenauigkeiten in der Aufgabenstellung, die zuvor geklärt werden müssen?
2. Entwerft den späteren Programmablauf auf Papier. Notiert euch dabei, welche Teilprobleme zu lösen sind.
3. Verfeinert euren Entwurf, wie im Vortrag 4 gezeigt wurde.
4. Schreibt das Programm, achtet darauf, häufig zu testen!

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2006/Rennschnecke

Aus FreitagsrundenWiki

Inhaltsverzeichnis

Rennschnecke

1. Erstelle eine Klasse "Rennschnecke"
2. Rennschnecken sollen folgende Eigenschaften(Objektvariablen) besitzen:
   - einen Namen
   - eine Rasse
   - eine Maximalgeschwindigkeit
   - die Schnecke soll wissen welchen Weg sie bereits zurück gelegt hat
3. Erstelle für die Klasse Rennschnecke einen Konstruktor, der den Instanzvariablen beim Erstellen einer neuen Instanz(Objekt zu dieser Klasse) Werte zuweist.
4. Lege in der Klasse "Rennschnecke" eine Methode "krieche()" an, welche die Schnecke abhängig von ihrer Maximalgeschwindigkeit eine zufällige Strecke weiter bewegt.
   - Tipp: Schaut euch die Methode Math.random() aus der Java API an.
5. Lege in der Klasse "Rennschnecke" eine Methode "public String toString()" an, welche die Daten der Schnecke mit return als String zurückgibt.
6. Teste deine Klasse, indem du probehalber ein Rennschneckenobjekt erzeugst und seine Daten auf der Konsole ausgibst.
   - Tipp:Verwende zum Ausgeben der Daten die toString() Methode der Rennschnecke.

Rennen

1. Erstelle eine Klasse "Rennen"
2. Ein Rennen hat folgende Eigenschaften:
   - einen Namen
   - die Anzahl der teilnehmenden Schnecken
   - die teilnemenden Schnecken selbst (z.B. in einer ArrayList)
   - die Länge der zu kriechenden Strecke
3. Überlege dir, welche dieser Werte bereits im Konstructor gesetzt werden sollten.
4. Lege in der Klasse "Rennen" eine Methode "void addRennschnecke(Rennschnecke neueSchnecke)" an, welche dem Rennen eine Schnecke hinzufügt.
5. Lege in der Klasse "Rennen" eine Methode "void removeRennschnecke(String name)" an, welche eine Schnecke aus dem Rennen entfernt.
6. Lege in der Klasse "Rennen" eine Methode "public String toString()" an, welche die Daten des Rennens mit return als String zurückgibt.
   - Tipp: Um die Daten der beteiligten Schnecken zurückzugeben, könnt ihr deren toString() Funktion

benutzen.

7. Teste deine Klasse vom Hauptprogramm aus!
8. Lege in der Klasse "Rennen" eine Methode "Rennschnecke ermittleGewinner()" an, welche **null** zurückliefert, wenn noch keine der teilnemenden Schnecken das Ziel erreicht hat und anderenfalls die Gewinnerschnecke zurückgibt.
9. Lege in der Klasse "Rennen" eine Methode "void lasseSchneckenKriechen()" an, welche alle teilnemenden Schnecken einmal kriechen lässt.
10. Lege in der Klasse "Rennen" eine Methode "void durchfuehren()" an, welche so lange lasseSchneckenKriechen() aufruft, bis eine der Schnecken das Ziel erreicht hat.
    - Tipp: Ob eine Schnecke im Ziel angekommen ist, kannst du mit deiner Methode ermittleGewinner() herrausfinden.

Wettbüro

1. Erstelle eine Klasse Wettbuero.
   Ein Wettbuero hat die folgenden Eigenschaften:
   - Es weiß, für welches Rennen es seine Wetten entgegennimmt.
   - Es verfügt über eine Liste (z.B. eine Arraylist) von angenommenden Wetten.
   - Es hat einen festen Faktor, mit welchem Wetteinsätze bei einem Gewinn multipliziert werden.
2. Lege in der Klasse "Wettbuero" eine Methode "wetteAnnehmen(String schneckenName, int wetteinsatz, String spieler)" an, welche eine Wette entgegennimmt. Die Wette ist bezogen auf eine Schnecke für das Rennen, das von dem Büro bearbeitet wird.
   - Um die einzelnen Wetten speichern zu können, sollten ihre Daten in eigenen Objekten der Klasse "Wette" gespeichert werden. Erstellt euch diese Klasse selbst.
   - Tipp: Denkt an toString().
3. Lege in der Klasse "Wettbuero" eine Methode "rennenDurchfueren()" an, welche das betreute Rennen durchfuert.
4. Lege in der Klasse "Wettbuero" eine Methode "toString()" an, welche die Daten des Wettbueros, die Daten des Rennens sowie sämtliche abgeschlossende Wetten als String zurückgibt.
5. Teste dein Programm!
6. Tausche die Klasse Rennen mit einem Kommilitonen, der ebenfalls bereits fertig ist.
   - Sollten eure Programme immer noch laufen?
   - Treten Fehler auf? Wenn ja: ändert eure Programme so ab, dass ihr problemlos Klassen austauschen könnt.

Zusatz

Es gibt noch ein paar Dinge, die in unserem Wettbüro nicht so laufen wie sie sollten. Macht sie besser.

- Dieselbe Schnecke kann in ein Rennen zweimal eingetragen werden.
- Es können negative Wetten abgeschlossen werden.
- Gehen zwei Schnecken gleichzeitig durchs Ziel, wird die Schnecke ausgegeben, auf die die Suche nach dem Gewinner zuerst stößt. Das ist ganz schön ungerecht.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle

dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Studentendatenbank LE5

Aus FreitagsrundenWiki

Im Rahmen dieser Aufgabe wollen wir eine kleine "Studentendatenbank" erstellen.

1.1. Schreibe eine Klasse *Student*, die Informationen wie Name, Matrikelnummer etc. über einen Studenten speichert. Erzeuge ein Objekt dieser Klasse und belege die Objektvariablen mit Werten. Lies diese Werte aus den Objektvariablen wieder aus und gib sie auf dem Bildschirm aus.

1.2. Schreibe einen Konstruktor für deine Klasse, der die Objektvariablen des erzeugten Objekts mit Werten füllt. Erzeuge mit Hilfe dieses Konsturktors ein Objekt, lies seine Werte aus und gib sie auf dem Bildschirm aus.

1.3. Erweitere deine Klasse um eine Methode *printMe()*, die die Daten des Studentenobjekts auf dem Bildschirm ausgibt. Teste deine Methode!

1.4. Lege ein Feld (Array) mit 100 Studentenobjekten an, die verschiedene Matrikelnummern haben und gib alle auf dem Bildschirm aus.

1.5. Das in der vorherigen Aufgabe angelegte Feld ist schon eine Art Studentendatenbank. Schreibe daher eine neue Klasse *StudentDatabase*, die ein Feld von Studentenobjekten enthält. Die Klasse soll einen Konstruktor haben, dem die Größe des zu erzeugenden Feldes übergeben wird. Implementiere auch eine Methode *printMe()*, die alle Datensätze ausgeben soll, sowie eine Methode *addStudent()*, die ein Studentenobjekt mit den zu übergebenen Daten anlegen soll und dieses in die Datenbank einfügen soll.

1.6. Zusatzaufgabe: Erweitere die Studentendatenbank um eine Methode *deleteStudent()*, die das Studentenobjekt mit der übergebenen Matrikelnummer aus der Datenbank löschen soll. Funktionert nach dem Löschen einiger Objekte deine Ausgabemethode printMe() noch?

1.7. Zusatzaufgabe: Erweitere die Studentendatenbank um eine Methode *numberOfStudents()*, die die Anzahl der in der Datenbank gespeicherten Studentenobjekte zurückliefert.

1.8. Zusatzaufgabe: Schreibe Methoden, die statistische Daten über die Studenten ermitteln. Beispiele sind: die Durchschnittsnote, das Durchschnittsalter . . .

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle

dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Multiarray

Aus FreitagsrundenWiki

Mehrdimensionale Arrays

Mehrdimensionale Arrays in Java werden gewöhnlich als Arrays von Arrays gespeichert. Je höher die Dimension und je größer die Arrays, desto höher ist der zusätzliche Speicherbedarf für diese Arrays. So kann es beispielsweise die Speicherung großer Bilder (dreidimensional: Pixelposition in x-Richtung, Pixelposition in y-Richtung, RGB-Farbwert) nötig machen, eine andere Strategie einzuschlagen: Die Daten werden in ein einzelnes Array in besonderer Reihenfolge abgespeichert, und eine Methode rechnet den dreidimensionalen Index in einen eindimensionalen um. Dies lässt uns auf dieses eine Array so zugreifen, als handle es sich um eine komplexere Struktur. Das ist praktisch, da die Daten dann einerseits in der Form vorliegen, in der wir sie direkt der Grafikkarte „füttern“ können, nämlich eindimensional, wir sie aber gleichzeitig benutzen können wie ein zweidimensionales Bild von RGB(Rot-, Grün-, Blau-Anteil)-Pixeln.

Diese Aufgabe kann bereits nach dem 2. Vortrag in Angriff genommen werden und nimmt mit jeder weiteren Unteraufgabe an Schwierigkeit und benötigtem Wissen zu. Die Aufgabe ist besonders für Leute geeignet, die schon Erfahrungen mit anderen (imperativen/objektorientierten) Programmiersprachen haben.

Aufgaben:

1. Wieviel Speicherbedarf benötigt die konventionelle Speicherung zusätzlich für die inneren Arrays im Vergeich zu unserem Ansatz für ein RGB-Bild der Größe 256x256? (Annahme: ein Array benötigt 1 Speichereinheit)

2. Lies dir folgende Website durch: http://webster.cs.ucr.edu/AoA/Windows/HTML/Arraysa2.html. Was bedeuten die Begriffe column-major und (fortgeschritten) row-major? Nach welcher Formel werden im zwei- bzw dreidimensionalen Fall die Offsets berechnet?

3. Erstelle eine Klasse Bild. Ein Bild soll seine Breite und seine Höhe kennen. Erstelle hierzu zwei private Variablen. Die dritte Dimension, die (Farb-)Tiefe ist mit 3 fest, da wir annehmen, dass es sich ausschliesslich um RGB-Bilder handelt.

4. Fortgeschritte: Erstelle für die 3 eine finale statische Konstante (Farb-)Tiefe, um die 3 nicht immer explizit in den Code schreiben zu müssen.

5. Zusätzlich soll eine Variable pixels die eigentlichen Pixeldaten enthalten. Erstelle hierzu ein Character-Array pixels der Größe Breite*Höhe*3 und initialisiere die RGB-Werte einheitlich mit fortlaufenden Nummern. Wie sind Breite und Höhe zu setzen, wenn es sich um ein quadratisches Bild handelt und der volle Datenumfang des characters (8 bit) ausgenutzt werden soll?

6. Es soll Getter-Methoden für die Höhe und die Breite sowie für das Datenarray geben.

7. Fortgeschritte: Schreibe einen Konstruktor, der die Höhe, die Breite, und eindimensionales Array als Parameter bekommt.

8. Erstelle eine allgemeine Getter- Methode die ein Character zurückgibt, und eine setter-Methode. Beide sollen auf unser eindimensionales Pixeldatenarray so zugreifen, als wäre es ein dreidimensionales.

9. Fortgeschritte: Wirf eine Exception, wenn einer der übergebenen Indizes über die im Objekt gespeicherte

Höhe, Breite oder (Farb-)Tiefe hinausgehen! Was passiert, wenn beim Zugriff der aus dem falschen dreidimensionalen Index berechnete eindimensionale Index innerhalb, was ausserhalb des Arrays pixels liegt? Überprüfe zudem im Konstruktor, dass Höhe*Breite*3 gleich pixels.length (by the way eine finale Instanzvariable. Was heißt das?) ist! Wirf ebenfalls eine Exception. Was wäre über Alternativen mit speziellen Rückgebewerten, die einen Fehler anzeigen zu sagen?

10. Erstelle eine Methode `float setPixel(int x, int y, char R, char G, char B)` und Methoden, die die Rot- Grün- und Blauwerte auslesen. Nutze hierbei die bereits implementierten, allgemeineren Setter- und Getter-Methoden.

11. Implementiere eine Methode, die das Pixelarray traversiert und die Rot-Grün-und Blauwerte auf den Bildschirm ausgibt. Nutze hierbei die bereits implementierten Getter-Methoden.

12. In der Bildbearbeitung werden auf zweidimensionale Bilder verschiedene Filter angewandt. Wir möchten hier einen einfachen Glättungsfilter implementieren, der harte Kanten weichzeichnet. Der funktioniert so: Nimm den oberen, den unteren, den linken, und den rechten Pixelwert plus den Wert des Pixels selbst, teile das durch 5 und setze das Pixel mit dem entsprechenden Wert. Hinweis: Pixel, die ausserhalb des Bildes liegen, werden als schwarz (0, 0, 0) angenommen. Fortgeschritten: Als Randbedingung kann man auch annehmen, dass sich das Bild an den Ränder so fortsetzt, wie es geendet hat.

13. Fortgeschritte: Verschiedene Grafik-APIs benutzen column-major oder row-major-Format. Ein Bild soll „wissen“, ob es column-major oder row-major ist. Implementiere Setter- und Getter-Methoden!

14. Fortgeschritte: Erweitere die Setter-Methode derart, dass sie das Pixelarray in die jeweils andere Speicherungsform konvertiert! Was zeigt dies über die Sinnhaftigkeit von privaten Attributen und den Gebrauch von Setter- und Getter-Methoden?

15. Fortgeschritte: Wie könnte man unsere Klasse Bild verfeinern, um auch Grauwertbilder und Bilder mit Alphakanal abspeichern zu können? Welche Attribute und Methoden kann man in eine gemeinsame Superklasse zentralisieren?

16. Fortgeschritten: Zelluläre Automaten werden in der Computergraphik eingesetzt, um komplexes Verhalten wie etwa Rauch, Feuer, aber auch Staubildung zu simulieren. Es gibt sogar ernstzunehmende Wissenschaftler, die behaupten, unsere gesamte Welt wäre ein zellulärer Automat. Zelluläre Automaten basieren auf dem Prinzip, dass Zellen lokal Informationen austauschen, d.h. dass der Zustand einer Zelle vom Zustand der direkten Nachbarzellen abhängt. Hatten wir nicht sowas ähnliches schon? Richtig, der Glättungsfilter aus Aufgabe 12 arbeitete auch so ähnlich.
Für Zelluläre Automaten, besonders dreidimensionale, ist Speichereffizienz besonders wichtig, da für realistische Animationen mehrere hundert mal hundert mal hundert Zellen angelegt werden müssen.
Lest diese Website über Conways Game of Life und habt Spaß: http://www.bitstorm.org/gameoflife/

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Von „http://wiki.freitagsrunde.org/Javakurs2007/Multiarray“

Javakurs2007/Canvas

Aus FreitagsrundenWiki

In dieser Aufgabe sollt Ihr eine Repräsentation einer dreidimensionalen Maloberfläche implementieren, die verschiedene geometrische Figuren wie Kugeln, Punkte, Geraden oder Würfel enthalten kann.

Beachtet bitte, dass diese Aufgabe ziemlich umfangreich ist und daher viele mögliche Fehlerquellen enthält. Fragt daher bei jedem auftauchenden Problem einen Tutor um Hilfe, da sich mögliche Fehler sonst bis zum Schluss durchziehen könnten.

Hinweis: Testet jede Eurer geschriebenen Methoden am besten, sobald Ihr sie geschrieben habt, um mögliche Fehler frühzeitig zu finden.

Inhaltsverzeichnis

3D-Punkt

Erstellt die Klasse Point3D, die die Repräsentation eines Punktes im dreidimensionalen Punktes darstellt.

Die Klasse soll folgende Daten, Methoden und Funktionen haben:

1. private Variablen für x-, y- und z-Koordinaten
2. einen Konstruktor, der Initialisierungswerte für x, y und z bekommt
3. eine Methode toString(), die den Punkt so darstellt: "(3, 2, 1)"
    - Tipp: Spätestens jetzt sollte ein erster Test des bisher Geschriebenen erfolgen.
4. eine Methode changeCoordinates, die eine nachträgliche Anpassung der Koordinaten ermöglicht.
5. eine Methode calcDistance, die den Abstand des Punktes zu einem zweiten berechnet
    - Tipp: Entscheidet, ob die Distanz exakt (float) und grob (int) berechnet werden soll. In beiden Fällen könnt Ihr unter java.lang.Math (http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Math.html) Nützliches finden.

Kugel

Erstellt die Klasse Sphere, die eine Kugel darstellen soll.

Sphere soll folgende Daten und Methoden kapseln:

1. private Variablen für Ursprung (Point3D) und Radius
2. Methoden zum Ändern und Initialisieren dieser Werte

3. eine Methode toString mit passender Visualisierung
    - Tipp: Heute schon getestet? ;-)
4. zwei Funktionen, die Volumen und Oberfläche der Kugel berechnen
5. eine Funktion, die diese Kugel und eine andere hinsichtlich ihrer Größe vergleicht (compareSpheres)

Weitere Figuren

Orientiert Euch an Sphere und Point3D und schreibt Klassen für weitere Figuren, wie z.B. Würfel, Zylinder oder Kegel.

Canvas

Schreibt eine Klasse Canvas, die

1. durch ein parameterlosen Konstruktor erzeugt wird.
2. ein Array für jede Sorte von geometrischer Figur besitzt.
    - Tipp: Macht Euch Gedanken wie Ihr ein über- oder unterlaufen des Arrays verhindern und abfangen könnt.
3. durch entsprechende Methoden Figuren jeder Sorte einzeln hinzufügen und löschen kann.
4. eine Methode zu Berechnung des Gesamtvolumens und der Gesamtoberfläche hat.
5. als toString() eine Liste aller Figuren und andere relevante Infos ausgibt.

Zusatz

Achtung nur für Cracks: Diese Zusatzaufgabe geht über den Inhalt und Sinn des Javakurses hinaus. Daher fangt sie nur an, wenn Ihr alles andere verstanden und keine weiteren Fragen mehr habt. Das meinen wir ernst.

Schreibt mithilfe der Terminalklasse eine Klasse, die eine Oberfläche zur Verfügung stellt, die interaktiv das Hinzufügen und Löschen von Figuren ermöglicht und alle Daten ausgeben kann.

- Tipp: Nutze folgendes Konstukt für die mehrstufige Fallunterscheidung:

```
public class MyClass{...
   char decision;
   ...
   //decision gets some value
   ...
   switch(decision){
     case 'a': doSth();break;
     case 'b': doSthDiff();break;
     default doAnotherThing();break;
   }
   ...
}
```

Das funktioniert natürlich auch mit mehr Fällen und ints.

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)

Javakurs2007/Doom

Aus FreitagsrundenWiki

ASCII-DOOM

In Ascii-Doom geht es darum, ein kleines Spiel zu implementieren, mit ASCII Grafiken. Es wird möglich sein, mit einer Spielfigur auf dem Spielfeld zu laufen, später könnt ihr auch Items, Waffen und Liv Tyler (http://venus.provocateuse.com/images/photos/liv_tyler_01.jpg) einbauen! Wir werden Klasse für Klasse das Spiel implementieren.

Inhaltsverzeichnis

class GameMain Teil 1

- Die Klasse GameMain ist die Klasse, die die main Methode enthalten wird und somit ausführbar ist.
- Sie wird die Schnittstelle zur Außenwelt sein und die Ausgabe auf die Konsole tätigen.
- Erstellt zunächst nur diese Klsse. Sobald ihr Klassen aus den späteren Aufgaben implementiert habt, könnt ihr in der main-Methode ihre Funktionalität testen.

---

class Position Teil 1

- Position repräsentiert eine Position auf einem 2D-Spielfeld.
- Enthält private Attribute x und y, gekapselt mit gettern/settern.

- Implementiert eine equals-Methode, die 2 Positions vergleicht.
- **TESTET euer equals!**

---

class GameTile Teil 1

- GameTile repräsentiert ein Tile (ein Feld) im späteren Spiel.
- Enthält eine private Variable, die für den Typ des Feldes steht (am besten ein int).
  - 1 entspricht wand
  - 2 entspricht leer
- Also auch hier braucht ihr Getter und Setter.
- Enthält außerdem eine private Variable vom Typ Position, welche die Position des GameTiles auf dem Feld speichert.
- Implementiert eine toString-Methode, die jedem GameTile einen String mit Länge 1 zuordnet.
  - wand = "x"
  - leer = " "
- **TESTET euer toString.**

---

class GameField Teil 1

- GameField ist eine zentrale Klasse in diesem Spiel, sie repräsentiert das Spielfeld als Ganzes und besteht aus GameTiles.
- Enthält eine private Variable für das Level (2D-Array aus GameTiles).
- Über den Konstruktor wird ein String dieser Form übergeben(ebenso seine Breite und Höhe):

```
String s = "########" +
           "#      #" +
           "#      #" +
           "###    #" +
           "#      #" +
           "########";
```

- - Es soll mit einer Schleife der String Zeichen für Zeichen ausgelesen werden und aus jedem Zeichen ein neues GameTile (fest oder leer) erzeugt werden. Diese GameTiles werden dann in dem Array gespeichert!
  - Wie man Strings Zeichen für Zeichen ausliest, findet ihr in der API (java.lang.String).
  - Implementiert ein toString, dieses soll folgenden String liefern:

```
  12345678
1 ########
2 #      #
3 #      #
4 ###    #
5 #      #
6 ########
```

- - Tipp: "\n" steht in einem String für Zeilenumbruch.
  - Es soll also aus dem GameField-Array wieder ein String generiert werden.
  - Die toString-Methode von GameField soll die toString Methode von GameTile verwenden, um diese Ausgabe zu erzeugen.
- **TESTET euer Programm mit verschiedenen Levels**

---

class World Teil 1

- World ist die Klasse, die alles verbinden wird, Spieler, Items, Spielfeld etc.
- Zunächst einmal gibt es nur das GameField, was wir einbinden können (später der Spieler).
- Die Main Klasse erzeugt ab jetzt nur noch eine Instanz von World nicht mehr von GameField. Denn World selbst soll sich das GameField erzeugen (im Konstruktor).
- Bei Java funktioniert es bei grafischen Anwendungen etwa so: Man hat ein Objekt, das eine Leinwand repräsentiert (Canvas), man darf auf diesem Objekt Sachen malen. In einem Spiel wird das Objekt in jedem Frame leer gemacht, danach werden alle beteiligten grafischen Objekte (Spieler, Spielfeld, Gegner) nacheinander auf diese Leinwand gemalt. Letzendlich wird die Leinwand auf dem Bildschirm ausgegeben. Das hat den Vorteil, dass jede Klasse die darauf Zugriff bekommt, irgendetwas malen kann ohne zu wissen, was bereits darauf ist oder noch darauf kommt. (Jedes Objekt kümmert sich selbst um seine grafische Entsprechung.)

Das werden wir hier auch machen, allerdings mit Strings, da uns die Grundlagen für Grafisches noch fehlen.

- Definiert eine solche Canvas Variable, die für die grafische Repräsentierung des Spielfelds stehen wird.
  - Das soll ein 2D-Array aus Strings der Länge 1 sein. Ein String pro Feld, dieses enthält dann "#" oder " ".

---

class GameField Teil 2

- Wir brauchen nun eine draw-Methode, die eine solche Leinwand (String[][]) aus World übergeben bekommt und sich selbst darauf malt. Diese Methode funktioniert nun ähnlich wie unsere toString-Methode, sie gibt nur nichts zurück, sondern modifiziert direkt unser Leinwand-Array und malt dort Zeichen für Zeichen die Felder rein. (Arrays sind keine primitiven Datentypen, daher wird hier nicht mit einer Kopie gearbeitet)

```
public void draw( String[][] canvas ){ }
```

- Was passiert eigentlich, wenn das GameField größer ist als unser String[][]?

---

class World Teil 2

- Nun kommt unsere wichtigste Methode `public void draw()`. Draw soll alle Inhalte (bisher nur GameField) auf unser Canvas-Objekt malen, indem es die draw-Methode von GameField aufruft und das Canvas-Array übergibt. Der Code beschränkt sich hier also auf eine Zeile.
  - Später fügen wir hier mehr ein.
- Implementiert auch eine toString-Methode, diese sollte das aktuelle Canvas-Array in der richtigen Form für die Konsolenausgabe liefern.
- **TESTET nun euere Grafik Ausgabe und euer toString().**

---

class GameMain Teil 2

- Die Klasse GameMain enthält als einzige eine Instanz der Klasse World und kann somit das Spielfeld zeichnen.
- Sofern ihr beim Testen versucht habt, das Spielfeld mehrmals auf die Konsole auszugeben, seht ihr, dass das alte Spielfeld noch immer sichtbar ist über dem neuen. Das sieht irgendwie blöd aus.
- Zählt nach, wieviele Zeilen euere Konsole hat.
- Bevor man das Level ausgibt, kann man soviele leere Zeilen ausgeben lassen, dass das alte Level nicht mehr sichtbar ist. `System.out.println("");` gibt eine leere Zeile aus.
- **TESTEN ob das klappt und gut aussieht**

---

class Player Teil 1

- Der Spieler wird die Möglichkeit haben, auf dem Spielfeld rumzulaufen und je nachdem, wie weit ihr kommt, noch viel mehr.
- Der Spieler besitzt eine Position (Variable vom Typ Position), wie immer private mit Getter/Setter
- Der Spieler wird durch ein "*" auf dem Spielfeld repräsentiert. Implementiert eine draw-Methode, die

genau so wie die draw-Methode aus GameField den Spieler auf unsere Leinwand malt, an die Position die in der position-Variable gepseichert ist.

- Player soll auch vier Methoden haben
  - public void moveLeft()
  - public void moveRight()
  - public void moveUp()
  - public void moveDown()
- Welche einfach nur die X- oder Y-Position um 1 erhöhen oder dekrementieren.

---

class World Teil 3

- Im Konstruktor soll nun auch eine Spielerinstanz erzeugt werden und zunächst an einer bel. Position gesetzt werden.
- Erweitert die draw-Methode so, dass nun auch der Spieler angezeigt wird.
- **TESTET ob das funktioniert. Was passiert eigentlich, wenn man den Spieler an der Position -1 | -1 setzt ?**

---

class World Teil 4

- Implementiert eine Methode `public boolean onKeyPressed( String key ){ ... }`. Sie wird dann aufgerufen werden, wenn eine Taste gedrückt wurde und soll `true` zurückgeben, wenn der Spieler sich bewegt hat, ansonsten `false`.
- Zunächst müssen wir in onKeyPressed prüfen, ob key einer der Strings "a" (links) , "s"(unten) , "d"(rechts) , "w"(oben) ist (equals benutzen!). Falls ja, wurde schonmal sicherlich eine gültige Taste gedrückt. Falls nein, können wir false zurückliefern.
- Wir müssen als nächstes prüfen, ob der Spieler durch die Bewegung nicht in eine Mauer reinläuft, dazu müssen wir die Spielerposition vergleichen und schauen, in welchem Tile er stehen würde, wenn wir rechts/links/hoch/runter liefen. Sofern dieses Tile leer ist, dürfen wir in diese Richtung laufen.
- Nun wo wir das alles geprüft haben, bewegen wir den Spieler mit einer der vier Bewegungsfunktionen und geben true zurück.
- **TESTET unbedingt manuell diese Methode, sie muss korrekt funktionieren!**

---

class GameMain Teil 2

- Nun kommt alles zusammen.
- Fuer die Eingabe von der Konsole bwnutzt wieder die Terminal Klasse.
- Es muss geprueft werden ob es sich um ein gueltiges Zeichen handelt und ob es eines der Zeichen a,s,w,d ist.
- Das eingegebene Zeichen soll dann an die World Klasse uebergeben werden und diese soll den Spieler bewegen, falls dies moeglich ist!

- Nun kann der Spieler im Level rumlaufen und es gibt eine Art Interaktion. Das ist das Basisgerüst für ein Spiel, ihr könnt nun weitere Sachen einfügen wie z.B. Items, die man einsammeln kann. Wie könnte man das machen? Was hat Liv Tyler in diesem Spiel für eine Aufgabe?

Kommentare

Wenn du Anmerkungen zur Aufgabe hast oder Lob und Kritik loswerden möchtest, ist hier die richtige Stelle dafür. Klicke einfach ganz rechts auf "bearbeiten" und schreibe deinen Kommentar direkt ins Wiki. Keine Scheu, es geht nichts kaputt ;)