DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Programmieren in Java

Übungen

Edition 0.1, 28.01.2013
Autor: Stephan Rupp

Inhaltsverzeichnis

1. Entwicklungsumgebung erstellen 3
2. Ein eigenes Paket 6
3. Eingabe und Ausgabe von Werten 7
4. Quellcode und Objektcode 9
5. Reflexionsfaktor 9
6. Verzweigungen und Schleifen 9
7. Methoden 10
8. Reflexionsfaktor modular programmieren 11
9. Arrays 11
10. Faltung 12
11. Spektralanalyse 13
12. Grafik einbinden 14
13. Übertragungsfunktion 16
14. Zweidimensionale Felder in der Bildverarbeitung 17
15. Java Methoden für Mikrocontroller 19
16. Zeichen und Zeichenketten 20
17. Klassen 20
18. Objektorientierung 20
19. Erzeugung und Initialisierung mit dem Konstruktor 21
20. Methodenkeller (Stack) 21
21. Warteschlange 22
22. Vererbung 22
23. Klassifikation 24
24. Dynamische Bindung 24
25. Ausnahmebehandlung 25
26. Threads 26

1. Entwicklungsumgebung erstellen

Für die praktischen Übungen installieren Sie bitte die Entwicklungsumgebung Eclipse (siehe Literaturverzeichnis [2]) und die Ein- und Ausgabebibliotheken [3] zur Vorlesung. Die Vorlesung basiert auf dem Buch [1] und der diesbezüglichen Mustervorlesung siehe [3]. Gehen Sie hierbei wie folgt vor.

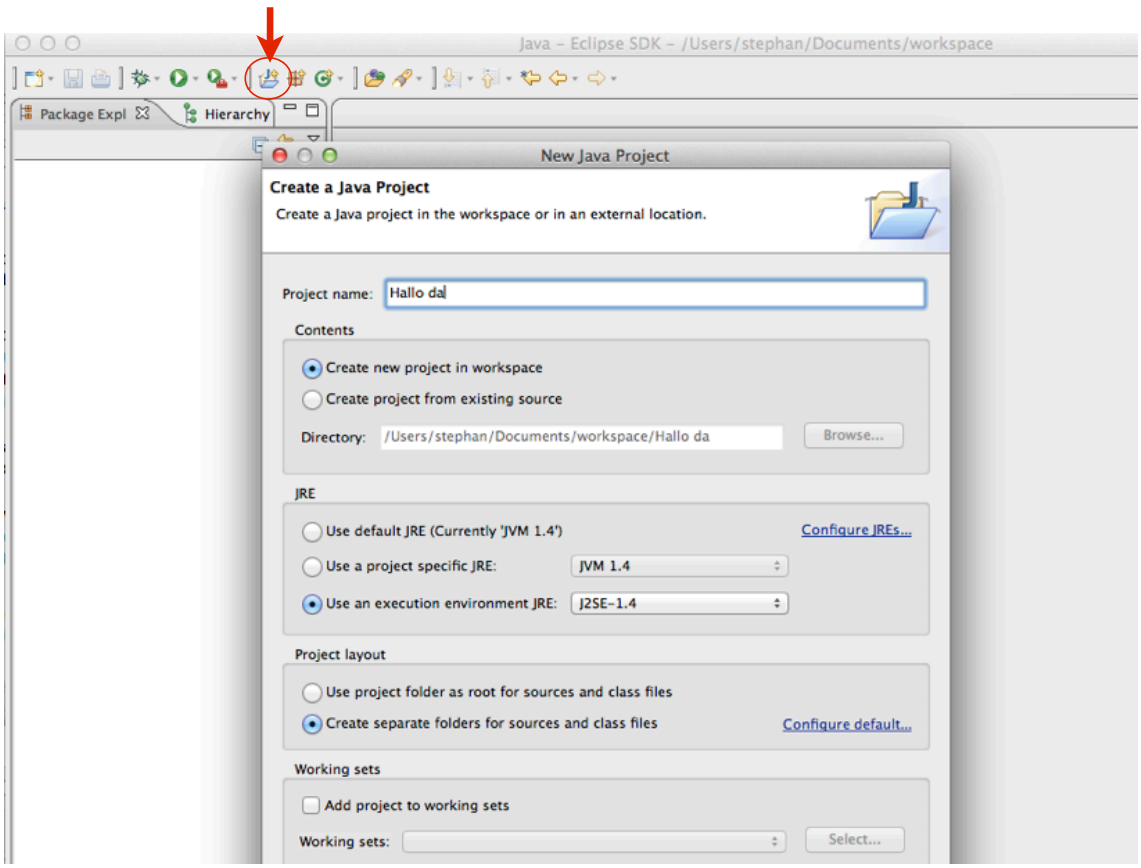
Schritt 1: Laden Sie das „Eclipse IDE for Java Developers“ von der in [2] angegebenen Web-Seite für das von Ihnen verwendete Betriebssystem (Mac OS, Windows, Linux). Installieren Sie das Paket gemäß Installationsanweisungen.

Schritt 2: Erzeugen Sie auf Ihrem Rechner ein Verzeichnis „Java“ und laden Sie dorthin von der Web-Seite [3] den Quellcode der Ein- und Ausgangsbibliotheken „In.java“, und „Out.Java“.

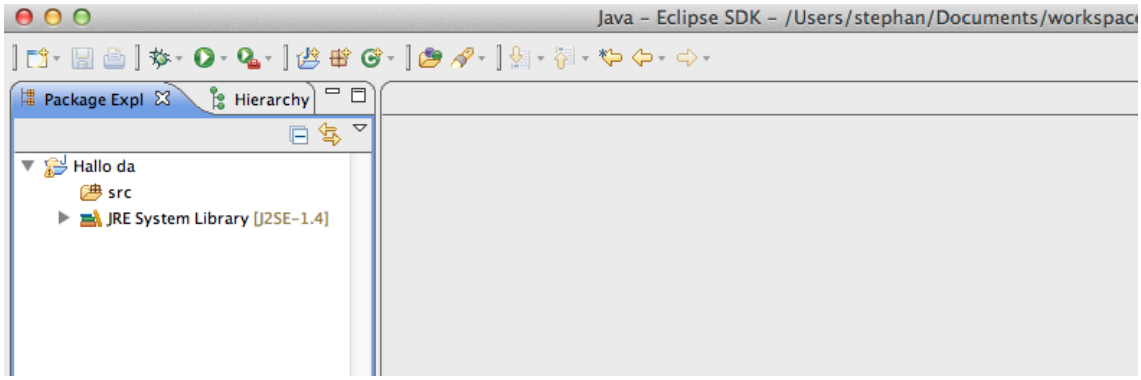
Schritt 3: Öffnen Sie Eclipse. Folgen Sie der Aufforderung, einen Arbeitsbereich („Workspace“) anzulegen, indem Ihr Quellcode und andere Dateien abgelegt werden. Hinweis: in den Übungen wird die englische Version von Eclipse verwendet, da die Menüs und der Sprachgebrauch auf diese Weise einheitlicher ist. Machen Sie sich mit der Oberfläche vertraut. Die wichtigsten Ansichten (Views) sind

- Package Explorer: zeigt die erstellten Projekte zusammen mit anderen Java-Packages und Dateien. Diese Projekte sind im Arbeitsbereich abgelegt.
- Outline: zeigt die Struktur der aktuell geöffneten Datei an
- Console: Ein- und Ausgabefenster des aktuell laufenden Programms (Terminalfenster)

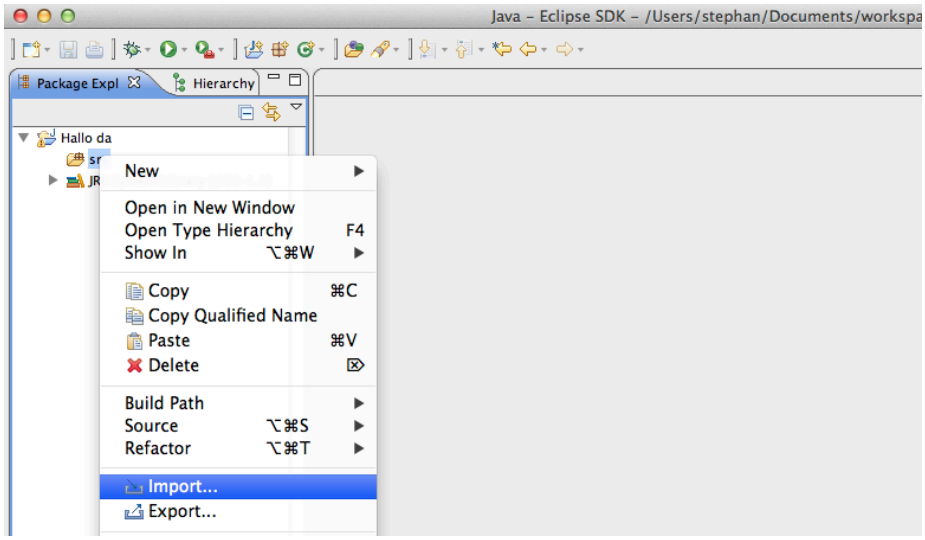
Schritt 4: Erstellen Sie ein neues Projekt, indem Sie das entsprechende Symbol in der oberen Menüleiste klicken. Geben Sie einen Projektnamen und die in folgender Abbildung gezeigten Einstellungen ein.



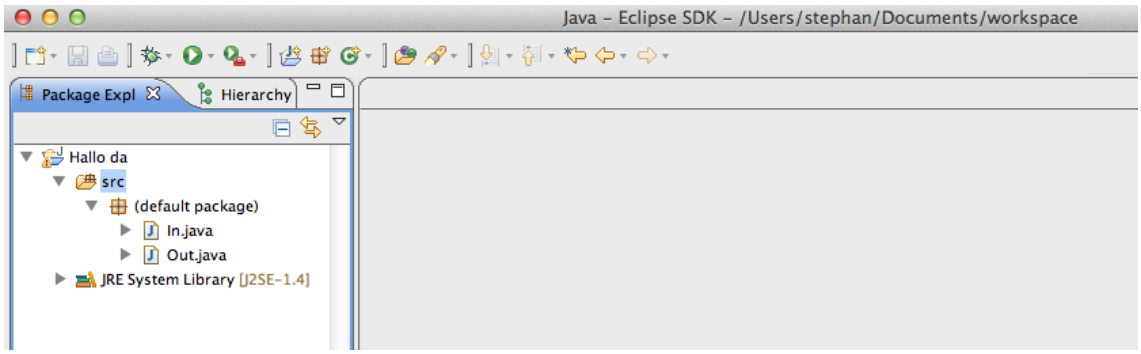
Schritt 5: Nach klicken der „Finish“-Taste sollten Sie im Package Explorer das neue Projekt sehen.



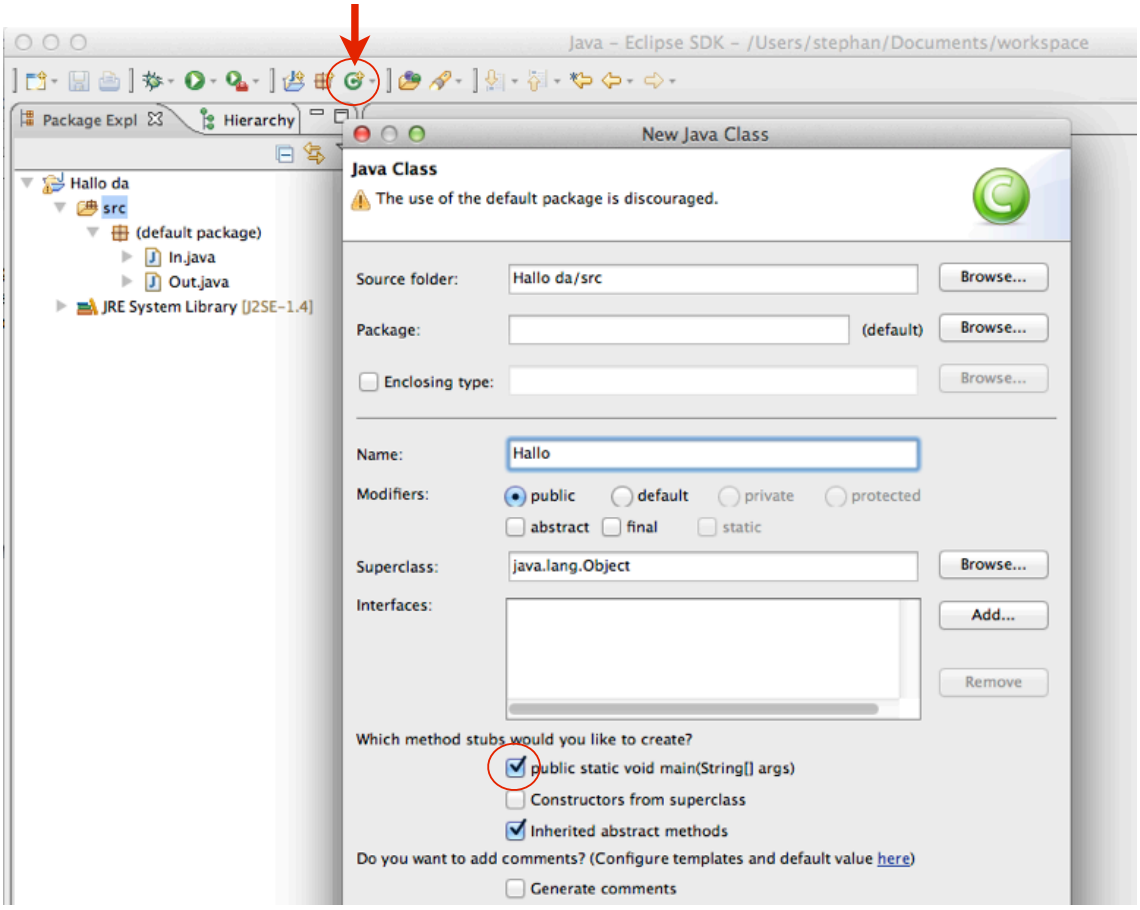
Schritt 6: Importieren Sie die Ein- und Ausgabeklassen. Wählen Sie hierzu das Verzeichnis aus („Browse“), auf das Sie die Dateien geladen haben.



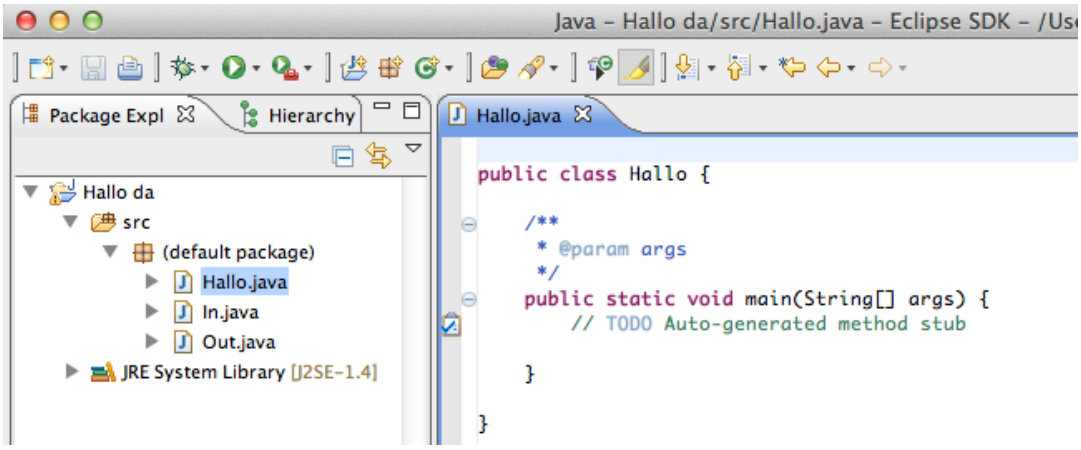
Ergebnis:



Schritt 7: Erstellen Sie eine eigene Java Klasse. Geben Sie z.B. den folgenden Namen, das oben generierte Paket ein, wählen Sie „public static void main“ aus und klicken Sie „Finish“. Bemerkung: Die gewählte Einstellung erzeugt einen Methodenrumpf für ein Hauptprogramm.



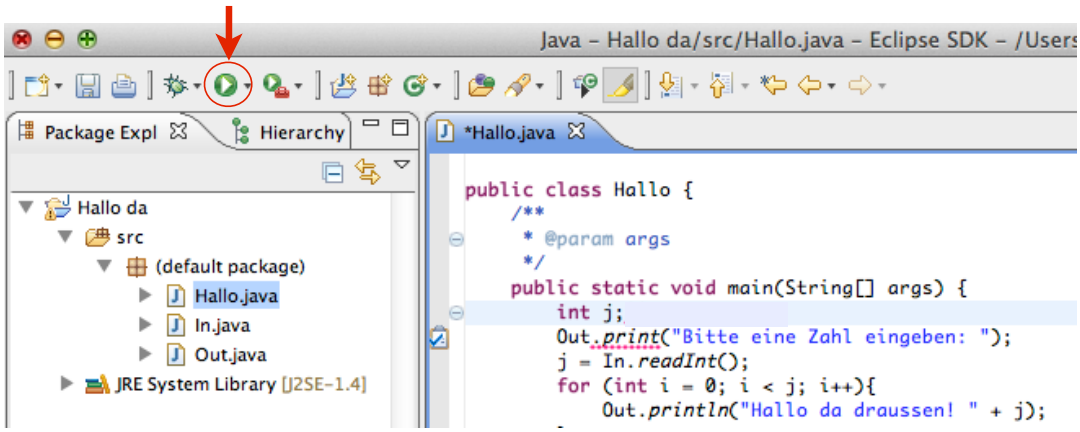
Ergebnis:



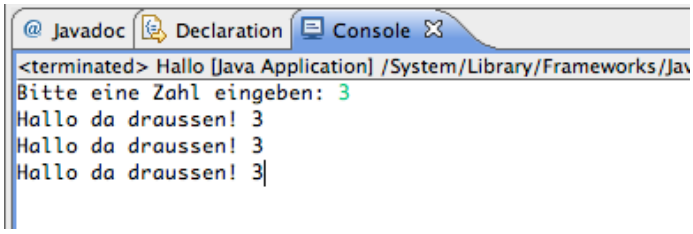
Schritt 8: Erstellen Sie das Allerweltsprogramm. Ersetzen Sie hierzu die Kommentarzeile „// TODO ...“ durch folgenden Programmtext:

```
int j;
Out.print("Bitte eine Zahl eingeben: ");
j = In.readInt();
for (int i = 0; i < j; i++){
      Out.println("Hallo da draussen! " + j);
}
```

Schritt 9: Führen Sie das Allerweltsprogramm aus. Klicken Sie hierzu die Schaltfläche „Run“, wie in folgender Abbildung gezeigt. Auf dem Konsolenfenster erscheint die Aufforderung des Programms zu Eingabe einer Zahl. Geben Sie eine Zahl ein.



Ergebnis:

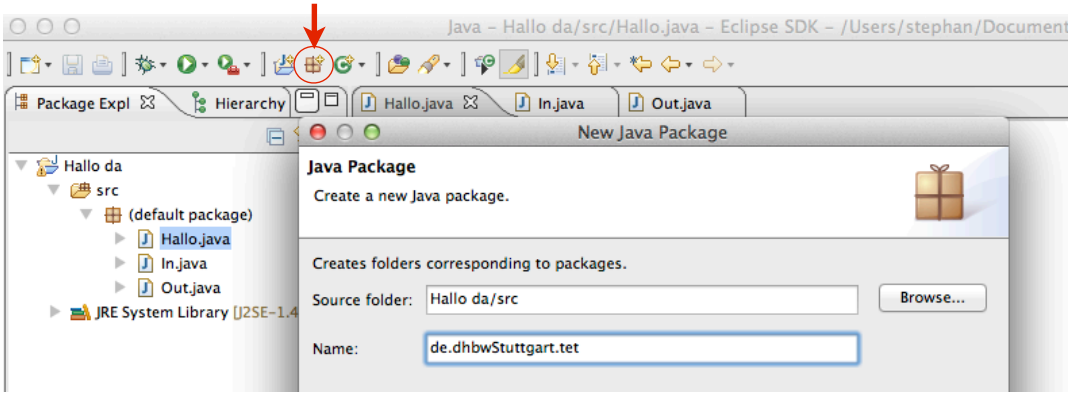


Schritt 10: Erkunden Sie die Entwicklungsumgebung. Klicken auf die importierten Dateien „In.java“ und „Out.java“ zeigt den Quelltext im mittleren Fenster an. Dieses Fenster ist der Editor, wie Sie bereits in Schritt 8 bemerkt haben. Das Fenster „Outline“ rechts daneben zeigt den Aufbau der jeweils gewählten Klasse.

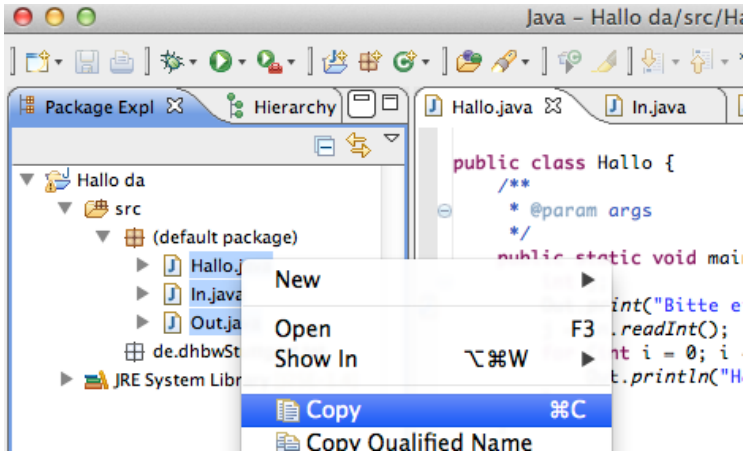
2. Ein eigenes Paket

Die Verwendung eines eigenen Pakets ermöglicht die Unterbringung Ihrer Klassen in einem eigenen Namensraum. Diese Vorgehensweise ist die bessere Programmierpraxis und Grund für die Warnung „The use of the default package is discouraged“ in Schritt 7, Übung 1.

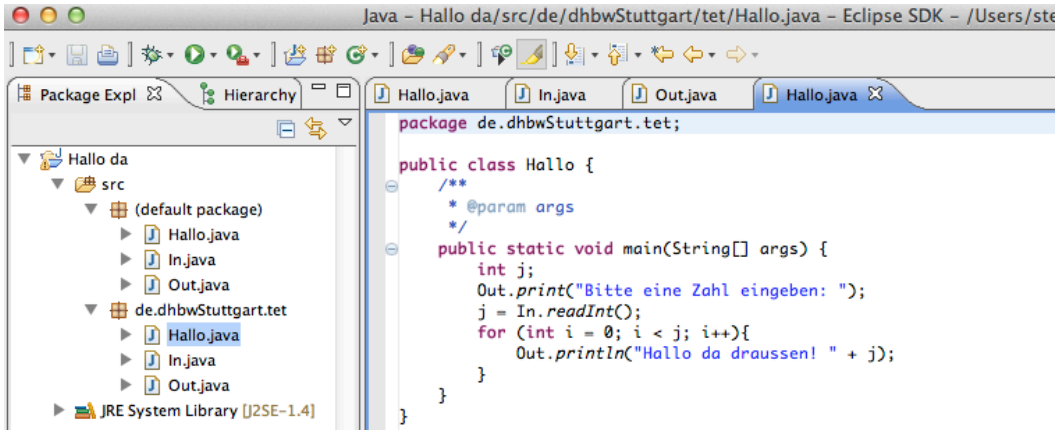
Schritt 1: Erstellen Sie ein eigenes Paket (Package). Geben Sie z.B. als Namen „de.dhbwStuttgart.tet“ ein und klicken Sie „Finish“.



Schritt 2: Kopieren Sie die 3 Java-Klassen aus dem Default Package in das eigene Paket.



Ergebnis:



Schritt 3: Öffnen der Klassen aus dem eigenen Paket zeigt, dass eine Deklaration des Paketnamens in die Klasse übernommen wurde. Hierdurch wird der eigene Namensraum gekennzeichnet. Starten Sie das Programm im eigenen Paket.

3. Eingabe und Ausgabe von Werten

Für die Eingabe und Ausgabe von Werten von Tastatur und Bildschirm, bzw. von einer Datei, stellen die Eingabeklassen und Ausgabeklassen In.java und Out.java geeignete Methoden bereit (siehe Vorlesungsfolien Teil 2, Folien 14 und 15).

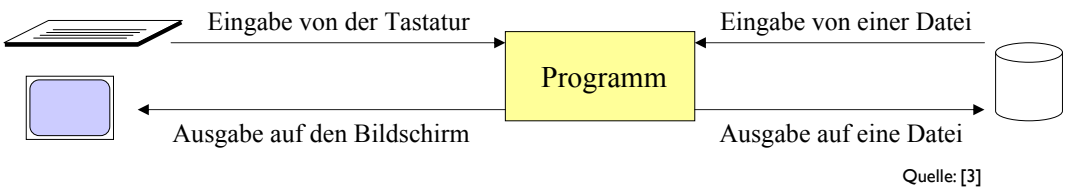


Bild 3.1 Eingabe und Ausgabe von Werten

Frage 1: Erstellen Sie ein Programm, das zwei ganze Zahlen a und b von der Tastatur einliest und Ausgaben auf dem Bildschirm für folgende Berechnungen ausgibt: Summe a+b, Produkt a*b, Quadratwurzel aus $a^2 + b^2$. Hinweis: die Quadratwurzel findet der Editor durch Eingabe des Methodennamens `Math.sqrt()` in der Standardbibliothek, die in der Laufzeitumgebung eingebunden ist.

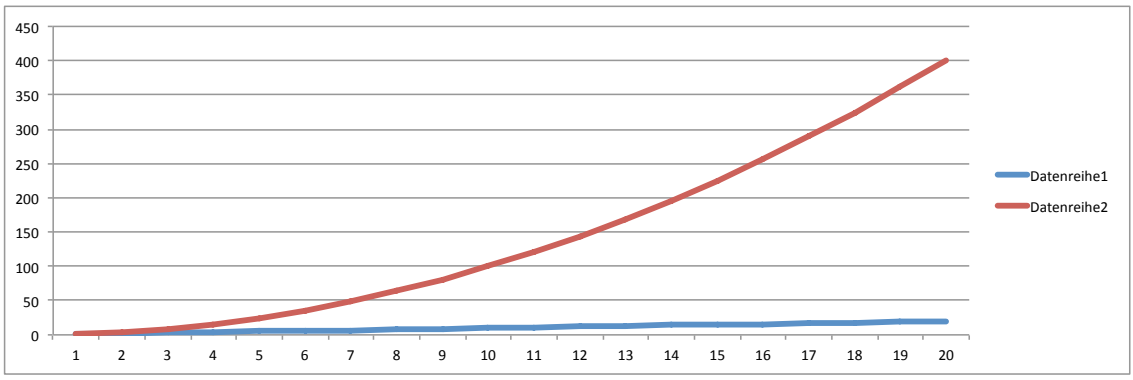
Frage 2: Modifizieren Sie das Programm so, dass die Eingabe aus einer Datei erfolgt. Kommentieren Sie das Programm sinnvoll. Hinweis: Speichern Sie die Tastatureingaben von Frage 1 in einer Datei, die dann später ausgelesen wird. Bemerkung: Bei Auswahl der Klasse In.Java sind im Fenster „Outline“ rechts neben dem Editor alle Methoden der Klasse zu finden, wie z.B. `In.Open(String)`, `In.close()`, ... Bei anklicken der Methoden im Fenster „Outline“ wird im Editorfenster der zugehörige Quelltext dargestellt.

Frage 3: Erstellen Sie ein Programm, das die Ergebnisse aus der in Frage 3.2 erzeugten Datei liest und anschliessend die Ergebnisse in eine andere Datei schreibt.

Frage 4: Suchen Sie die Datei im Arbeitsbereich (Workspace) von Eclipse. Offnen Sie die Datei mit einem geeigneten Programm (z.B. ein Texteditor). Öffnen Sie die Datei mit einem Programm zur Tabellenkalkulation (z.B. MS Excel). Hinweis: Der Pfad zum Arbeitsbereich wird in der Titelzeile des Eclipse Fensters angezeigt.

Frage 5: Schreiben Sie ein Programm zur Erzeugung einer kommaseparierten Datei. Nehmen Sie bitte für die Datensätze: (1) die Zahlen von 1 bis 20, (2) die Quadrate der Zahlen. Kommentieren Sie das Programm sinnvoll. Hinweis: Bei einer kommaseparierten Datei (engl. csv - comma separated values) werden Daten in einer Zeile durch ein Komma oder einen Strichpunkt als Trennzeichen getrennt. Jede Zeile als Datensatz wird durch einen Zeilenumbruch (Carriage Return) begrenzt. Der Zeilenumbruch wird durch die Funktion `Out.println()` (für „print line“) erzeugt.

Frage 6: Stellen Sie den Datensatz aus Frage 5 in einem Tabellenkalkulationsprogramm (z.B. MS Excel) als Diagramm dar. Hinweis: Die Trennzeichen innerhalb der Datensätze (Komma bzw. Strichpunkt) und der Dezimalpunkt (Punkt oder Komma je nach Betriebssystem) müssen beim Öffnen der Datei mit dem Programm zur Tabellenkalkulation richtig eingestellt werden. Ergebnis:



Frage 7: Erzeugen Sie mit dem Tabellenkalkulationsprogramm (z.B. Excel) einen Datensatz (mit N Werten in einer Zeile) und speichern Sie die Daten als kommaseparierte Datei (z.B. im Microsoft .csv Format) in den Arbeitsbereich von Eclipse ab. Schreiben Sie ein Programm, das

die Daten unter Verwendung der Methode In.done() einliest und mit Trennzeichen auf dem Bildschirm ausgibt. Das Programm soll ausserdem die Menge der Datensätze mitzählen und ausgeben. Hinweis: Verwenden Sie eine Schleife mit dem Schlüsselwort „while“.

Frage 8: Wenn die Daten aus Frage 7 keine ganzen Zahlen sind, sondern als Floating-Point Werte, so werden sie mit Dezimalpunkt dargestellt und abgespeichert. Hierbei spielt es eine Rolle, ob das Betriebssystem und Tabellenkalkulationsprogramm einen Punkt oder ein Komma als Dezimalpunkt verwenden. Erzeugen Sie mit dem Tabellenkalkulationsprogramm einen Datensatz mit zwei Nachkommastellen (durch umformatieren der Zellen aus Frage 3.7) und versuchen Sie diesen Datensatz einzulesen.

4. Quellcode und Objektcode

Lokalisieren Sie im Arbeitsbereich den Quellcode (.java) und Objektcode (.class). In welchen Verzeichnissen finden sich die in den bisherigen Übungen erzeugten Dateien? Woher kommt der Objektcode? Welche Rolle spielt der Paketname? Geben Sie die Verzeichnisstruktur als Diagramm (Graph, Baumstruktur) wieder.

5. Reflexionsfaktor

Eine Leitung mit Wellenwiderstand 50 Ohm ist mit einer Lastwiderstand R_L abgeschlossen. Schreiben Sie ein Programm, das folgende Berechnungen durchführt.

Frage 1: Berechnen Sie den Reflexionsfaktor am Ende der Leitung. Der Lastwiderstand R_L soll per Tastatur eingeben werden.

Frage 2: Die Leitung hat die Länge l, die ebenfalls per Tastatur eingegeben wird. Berechnen Sie den Reflexionsfaktor am Anfang der Leitung. Hinweis: Um die Berechnung zu vereinfachen, können Sie die Leitungslänge in Vielfachen der Wellenlänge bzw. in Vielfachen von $\lambda/4$ eingeben. Überprüfen Sie die Plausibilität der Ergebnisse für ausgewählte Fälle (offene Leitung, kurzgeschlossene Leitung, mit Wellenwiderstand abgeschlossene Leitung, Länge = $\lambda/4$, Länge = $\lambda/2$, ...).

Frage 3: Erweitern Sie das Programm um die Berechnung der Eingangsimpedanz.

Frage 4: Verbessern Sie das Programm um sinnvolle numerische Operationen (Abfangen von Divisionen durch Null, Ausgabe von Betrag und Phase der Eingangsimpedanz, Rundung von Betrag und Phase der Eingangsimpedanz auf ganze Zahlen, ...). Muster:

```
Bitte die Lastimpedanz (in [Ohm]) eingeben: 0
Bitte die Leitungslänge (in [lambda/4]) eingeben: 1
Reflexionsfaktor am Leitungsende: -1.0
Reflexionsfaktor am Leitungsanfang: 1.0
Eingangsimpedanz unendlich
```

6. Verzweigungen und Schleifen

Verzweigungen werden mit der „if“ Anweisung realisiert, wobei das Schlüsselwort „else“ den alternativen Pfad angibt. Für Verzweigungen mit mehreren Wegen kommt eine Konstruktion mit Hilfe von „switch“ zu den unterschiedlichen mit „case“ gekennzeichneten Pfaden.

Für Schleifen bekannter Länge eignet sich die Anweisung „for“ in Kombination mit einem Zähler. Schleifen mit einer allgemeinen boolschen Bedingung lassen sich mit der Anweisung „while“ in Kombination mit der Bedingung (Abweisschleife, da die Bedingung erst überprüft wird), bzw. die Anweisung „do“ gefolgt von der Anweisung „while“ in Kombination mit der Bedingung (Durchlaufschleife, da die Bedingung erst im Anschluss an den ersten Durchlauf geprüft wird).

Frage 1: Monatskalender. Erstellen Sie ein Programm, das zu einem eingegebenen Monat für das Jahr 2013 einen nach Wochen gegliederten Monatskalender ausgibt, z.B. in der unten gezeigten Form. Hinweis: Für Schaltjahre finden sich im Web folgende Regeln: (1) Ein Jahr ist ein Schaltjahr, wenn die Jahreszahl (ohne Rest) durch 4 teilbar ist. (2) Ein Jahr ist kein Schaltjahr, wenn es durch 4 und 100 (ohne Rest) teilbar ist. (3) Ein Jahr ist ein Schaltjahr, wenn es sowohl durch 4, durch 100 und durch 400 teilbar ist.

Muster:

```
Enter month: 4
Monatskalender 4 2013
Mo  Di  Mi  Do  Fr  Sa  So
 1   2   3   4   5   6   7
 8   9  10  11  12  13  14
15  16  17  18  19  20  21
22  23  24  25  26  27  28
29  30
```

Frage 2: Erweitern Sie den Monatskalender von Aufgabe 6.1 um die Eingabe beliebiger Jahreszahlen.

Frage 3: Schnee zu jeder Jahreszeit: Schreiben Sie ein Programm, das auf dem Terminal per Zufallsgenerator Schnee einer einstellbaren Stärke schreibt. Hinweis: Verwenden Sie hierzu die Funktion Math.random(). Muster:

```
Geben Sie die Schneestärke ein [1 - 10]: 3

 * *  *   ***         *  ****    *       ***  *  *   *         * *    *     **
   *  *    **   *     *    * *  ** ****   **    *     * ** **       *      *  *
*        *   * * *  **   ** **   *          *         ** *  ** * * *
* ****    *  *     *     *   * * * **   *            **   * * * *    * *       *
  * **   *   ** *      * *    **    ** *   ** *  *     **** *  *     *  ** *   *
    ** * **     ** * **    *    *    *      *     **  ** *    *   *   *    ** *
     **   *  *   * ***    *  *** **  *       *  *   * ** ***  *        *    * *
 *  *    *    *    *   ** *     *        * *   *** **         *
   * *  *  **    **   *     * * * **   ** *  *  *  *      **         * *  * *
   ****    ** *  * *    * *     *         * *  * *    * *  ***** *        *   *
***  * * **  **  *  *  *  ***  *   **    *  ****    *            *        * **
      **    *  ** * *  * * *     *      *    ** **  ** *    * *   **        **
  **     *   *     *  ** * *        ** ****  *   * ** *   **       *  **     * *
  *  *            *     **** * * * **  *    *      * **     *  ** *** * ***
*    ***  ***  ***              *    **   * ***  * *   * *           *   *
*       *    *            *      *          *    *      * * * **   * *     ** *
       *   * ** ** * *       * *** *      ** *      *   *     **   *       ** *
*   *       * *       *    *    * * * * *        *         **      *  *    *
```

Frage 4: Stellen Sie sich ein eigenes zweckfreies oder nützliches Problem zum Programmieren.

7. Methoden

Häufig benötigte Berechnungen lassen sich mit Hilfe von Methoden als Module behandeln. Die Methode wird so im Sinne einer mathematischen Funktion behandelt, wie z.B. y = f (x, y). Aus den Übergabeparametern x und y berechnet das Modul das Ergebnis und gibt es als Rückgabewert y

zurück. Methoden ohne Rückgabewert werden im Unterschied zu den genannten Funktionen als Prozeduren bezeichnet.

Frage 1: Schreiben Sie eine Methode, die zu einer in kartesischen Koordinaten gegebenen komplexen Zahl (d.h. Realteil und Imaginärteil) den Betrag ausrechnet. Benutzen Sie die Methode in einem Programm, das Real- und Imaginärteil vom Benutzer abfragt und den Betrag ausgibt.

Frage 2: Erweitern Sie das Programm um eine weitere Methode, die zu einer in kartesischen Koordinaten gegebenen komplexen Zahl (d.h. Realteil und Imaginärteil) die Phase ausrechnet. Erweitern Sie das Programm um die Ausgabe der Phase.

Frage 3: Schreiben Sie eine Klasse für komplexe Zahlen in Polarkoordinaten (d.h. mit Betrag und Phase).

Frage 4: Verwenden Sie die Klasse aus Frage 3 in einem Programm, das eine komplexe Zahl in kartesischen Koordinaten (d.h. Realteil und Imaginärteil) in eine komplexe Zahl in Polarkoordinaten (d.h. Betrag und Phase) umwandelt. Hinweis: Schreiben Sie eine Methode den Übergabeparametern Realteil und Imaginärteil (siehe Frage 2), jedoch mit Rückgabewert vom Typ Polarkoordinaten (Klasse aus Frage 3).

8. Reflexionsfaktor modular programmieren

Verwenden Sie Methoden und eine eigene Klasse für komplexe Zahlen, um Aufgabe 5 in einem modular aufgebauten Programm zu lösen.

9. Arrays

Für eindimensionale oder mehrdimensionale Datenreihen lassen sich Tabellen verwendet, die aus gleichartigen Elementen bestehen: sogenannte Arrays. Praktische Verwendung finden Arrays als Vektoren und Matrizen für numerische Anwendungen. So lassen sich beispielsweise Vektoroperationen, Matrizenrechnung und Berechnungen mit zeitdiskreten Signalen und Systemen mit Hilfe von Arrays ausführen (Bemerkung: die Software Matlab basiert auf Arrays).

Frage 1: Schreiben Sie ein Programm, das einen Vektor der Länge N durch Benutzereingabe erzeugt und in einer Datei speichert.

Frage 2: Schreiben Sie eine Methode zur Berechnung des Skalarproduktes zweier Vektoren und testen Sie die Methode in einem Programm.

Frage 3: Schreiben Sie eine Methode zur Berechnung des Kreuzproduktes zweier Vektoren und testen Sie die Methode in einem Programm.

Frage 4: Schreiben Sie ein Programm zur Multiplikation zweier Matrizen und testen Sie die Methode in einem Programm. Hinweis: Es genügt, wenn Sie die Zeilenzahl und Spaltenzahl einer Matrix durch Benutzereingabe abfragen und beide Matrizen im Programm mit Werten belegen, die es ermöglichen, den Algorithmus durch das Ergebnis auf Plausibilität zu prüfen. Beispiel: Matrix a(n,m) mit beliebigen Werten belegen, Matrix b(m,n) als Einheitsmatrix belegen, dann multiplizieren und Vorgaben a(n,m), b(m,n) und Ergebnis c(n,n) anzeigen. Muster:

```
Matrix a:
Geben Sie bitte die Anzahl der Zeilen n ein: 2
Geben Sie bitte die Anzahl der Spalten m ein: 3
```

```
a(0, 0) = 0.0  a(0, 1) = 1.0  a(0, 2) = 2.0
a(1, 0) = 3.0  a(1, 1) = 4.0  a(1, 2) = 5.0

b(0, 0) = 1.0  b(0, 1) = 0.0
b(1, 0) = 0.0  b(1, 1) = 1.0
b(2, 0) = 0.0  b(2, 1) = 0.0

c(0, 0) = 0.0  c(0, 1) = 1.0
c(1, 0) = 3.0  c(1, 1) = 4.0
```

10. Faltung

Die Faltung zweier zeitdiskreter Funktionen x(i) und h(i) wird durch die Faltungssumme

$$y[n] = \sum h[k] \cdot x[n-k]$$

beschrieben, wobei der Index k über den gesamten Wertebereich verläuft (von - ∞ bis + ∞ bzw. über alle vorhandenen Stützstellen von h(k)).

Frage 1: Schreiben Sie ein Programm, mit dem sich für h(n) und x(n) für eine ausreichende Anzahl Stützstellen sinnvolle Werte erzeugen lassen. Beispiele: (1) die Impulsantwort h(n) als Delta-Impuls, Rechteck-Impuls (Halteglied), Tiefpass (abklingender Verlauf), Verzögerung, etc. und (2) die Eingangsfunktion x(n) als periodische Funktion, Impuls, Rechteck etc.

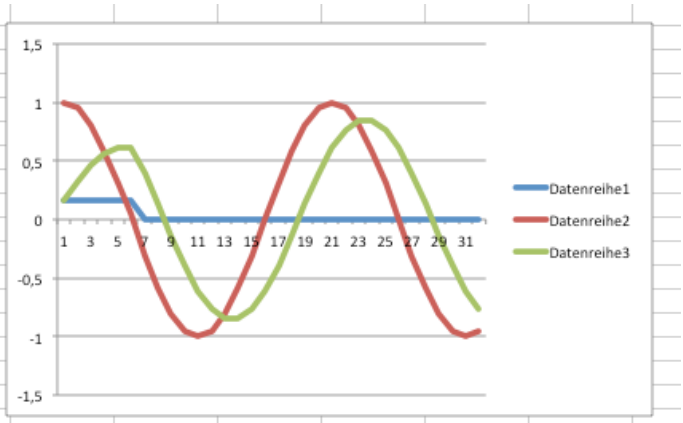
Frage 2: Ergänzen Sie das Programm um eine Methode, die die Faltung beider Signale durchführt.

Frage 3: Führen Sie das Programm für einige Signale aus und testen Sie das Programm durch Bewertung der Ergebnisse auf Plausibilität.

Frage 4: Schreiben Sie die Vorgaben und Ergebnisse in eine Datei, die Sie nach Excel importieren können. Visualisieren Sie die Ergebnisse in Excel. Muster:

```
Faltung - bitte Anzahl der Stützstellen eingeben: 32

(1) Delta bei i=0
(2) Delta bei i=10
(3) Rechteck von i=0 bis i=5 mit Höhe 1/6
(4) Abklingen h(i)=e(-i/4.0)
Bitte Impulsantwort h(i) auswählen: 3

(1) Delta bei i=10
(2) Rechteck von i=0 bis i=10
(3) periodisch x(i) = cos (i * 2 pi/20)
Bitte Eingangssignal x(i) auswählen: 3
```

0	0,16666667	1	0,16666667
1	0,16666667	0,95105652	0,32517609
2	0,16666667	0,80901699	0,46001225
3	0,16666667	0,58778525	0,55797646
4	0,16666667	0,30901699	0,60947929
5	0,16666667	6,12E-02	0,60947929
6	0	-0,309017	0,39130979
7	0	-0,5877853	0,13483617
8	0	-0,809017	-0,1348362
9	0	-0,9510565	-0,3913098
10	0	-1	-0,6094793
11	0	-0,9510565	-0,7679887
12	0	-0,809017	-0,851322
13	0	-0,5877853	-0,851322
14	0	-0,309017	-0,7679887
15	0	0,00E+00	-0,6094793
16	0	0,30901699	-0,3913098
17	0	0,58778525	-0,1348362
18	0	0,80901699	0,13483617



11. Spektralanalyse

Reelle Funktionswerte f(t), die im Intervall T an N Stützstellen abgetastet werden lassen, seien als Datenreihe f(k) beschreiben. Die Fourier-Transformation der Reihe wird beschrieben durch:

$$F[u] = \sum f[k]\, e^{-j2\pi k u/N} \qquad \text{mit dem Summenindex } k = 0 \text{ bis } k = N-1 \qquad (11.1)$$

In dieser Schreibweise wurde auf das Abtastintervall normiert, d.h. $\omega = 2\pi u /N$. die Datenreihe F[u] beschreibt das Spektrum des reellen Signals f[k]. Das Spektrum F[u] ist hierbei komplex in dem Sinne, dass F[u] einen Realteil und einen Imaginärteil besitzt. Das Spektrum F[u] besteht also aus zwei Reihen: Re{F[u]} und Im{F[u]}.

Die Rücktransformation lässt sich beschreiben durch:

$$f[k] = 1/N \sum F[u]\, e^{+j2\pi u k/N} \qquad \text{mit dem Summenindex } u = 0 \text{ bis } u = N-1 \qquad (11.2)$$

Frage 1: Schreiben Sie eine Methode zur Fourier-Transformation eines reellen Signals f[k] mit einer Anzahl Stützstellen N.

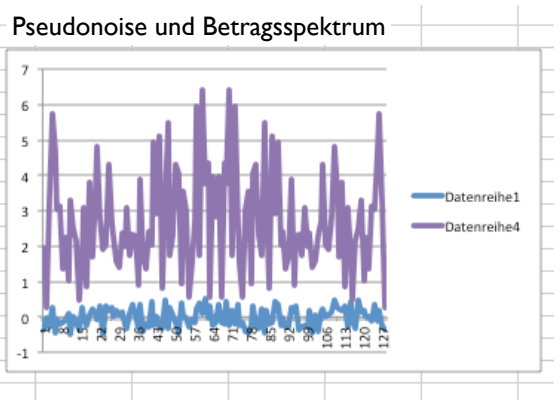
Frage 2: Schreiben Sie eine Methode, die das Spektrum F[u] in Betrag und Phase wandelt.

Frage 3: Schreiben Sie eine Methode zur Transformation des Rücktransformation des Spektrums F[u] in den Zeitbereich.

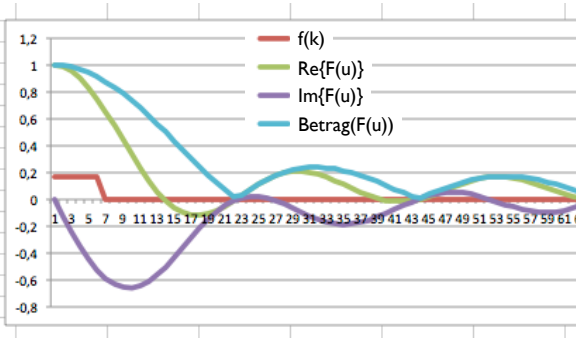
Frage 4: Testen Sie Ihre Methoden in einem Programm an geeigneten Signalen auf Plausibilität, z.B. Impuls, Rechteck (Spaltfunktion), periodische Funktion, Rauschen. Muster:

```
Spektralanalyse - bitte Anzahl der Stützstellen eingeben: 128

Bitte Signal f(k) auswählen: 6
(1) Impuls bei i=0
(2) Impuls bei i=10
(3) Rechteck von i=0 bis i=5
(4) Abklingen f(i)=e(-i/3.0)
(5) periodisch f(i) = cos (i * 2pi/20)
(6) zufällige Zahlenfolge (Pseudonoise)
```

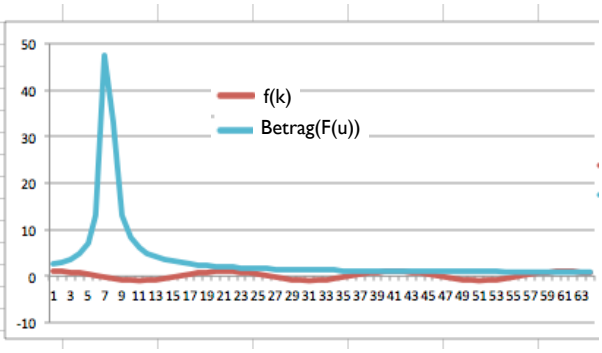
0	-0,3578635	-1,9641288	0	1,96412884	0
1	-0,0416955	0,22968378	-0,1152782	0,2569897	-26,652058
2	-0,2768497	-0,4272871	-3,2999772	3,32752516	82,6222855
3	0,24546772	-5,3173389	-2,0989416	5,71661164	21,5409013
4	-0,4714741	-1,6988196	-4,3649925	4,68392435	68,7344165
5	-0,0751493	-2,4185191	1,82214323	3,02810845	-36,994837
6	-0,2142149	2,93854932	1,10758896	3,14035435	20,6522751
7	-0,1608437	0,07583782	-1,3355448	1,33769626	-86,749996
8	-0,0663344	-1,4059329	-1,7237071	2,22436814	50,7977708
9	0,1079794	-0,9384516	0,39756594	1,01919093	-22,959473
10	-0,4934705	-3,2906151	0,13474114	3,29337256	-2,3447861
11	0,02624451	1,53944705	1,96423914	2,49562269	51,9128847
12	-0,1577918	-2,1274532	0,40114604	2,16494236	-10,678145
13	-0,3585516	0,47811713	-0,0493707	0,4806594	-5,8955045
14	0,2763282	-1,5396632	-0,7059463	1,69378951	24,6317859
15	0,01999846	3,0952255	-0,1147451	3,09735167	-2,1230756
16	-0,2555507	-0,561006	-0,6241373	0,83921098	48,0492004
17	0,05401948	-3,5300793	1,40922057	3,8009686	-21,762065



0	0,16666667	1	0	1	0
1	0,16666667	0,98899554	-0,121981	0,9964896	-7,03125
2	0,16666667	0,95645186	-0,2395787	0,98600107	-14,0625
3	0,16666667	0,90375448	-0,3486166	0,9686618	-21,09375
4	0,16666667	0,83313524	-0,4453201	0,94468211	-28,125
5	0,16666667	0,74756062	-0,5264916	0,91435238	-35,15625
6	0	0,65058397	-0,5896549	0,87803898	-42,1875
7	0	0,54616952	-0,6331623	0,83617916	-49,21875
8	0	0,43849772	-0,6562582	0,78927504	-56,25
9	0	0,33176219	-0,6590983	0,73788663	-63,28125
10	0	0,22996913	-0,6427206	0,68262408	-70,3125
11	0	0,13674969	-0,608974	0,62413928	-77,34375
12	0	0,05519511	-0,5604054	0,56311692	-84,375
13	0	-0,0122771	-0,5001144	0,50026505	88,59375
14	0	-0,0640193	-0,4315831	0,43630544	81,5625
15	0	-0,0992075	-0,3584898	0,37196379	74,53125



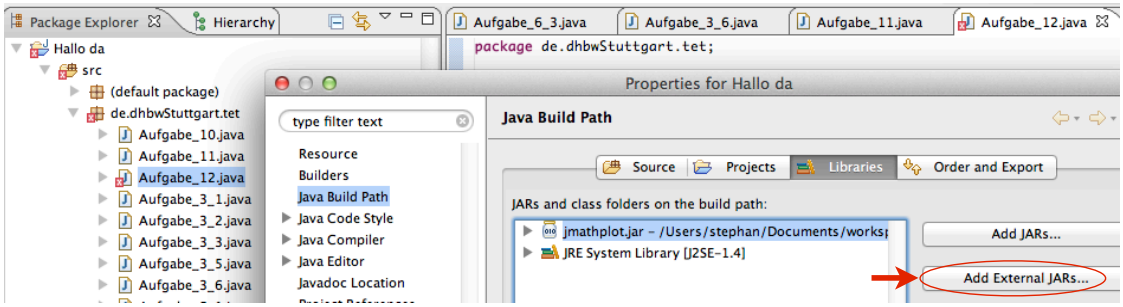
0	1	2,76007351	0	2,76007351	0
1	0,95105652	2,80689293	0,92968399	2,95684971	18,3256258
2	0,80901699	2,96255289	2,00908524	3,5795451	34,1435555
3	0,58778525	3,28692799	3,48161026	4,78805865	46,6475341
4	0,30901699	3,9593888	5,93569424	7,13507014	56,2948369
5	6,12E-17	5,69075752	11,5826448	12,90513	63,8343001
6	-0,309017	16,3396892	44,6249175	47,5222969	69,8894835
7	-0,5877853	-8,6427486	-32,033777	33,1792097	74,9010466
8	-0,809017	-2,4372095	-12,736528	12,9676188	79,1670723
9	-0,9510565	-1,0250257	-8,2164867	8,28017705	82,8889574
10	-1	-0,4091168	-6,1673653	6,18091994	86,2048018
11	-0,9510565	-0,0685961	-4,9825429	4,98301511	89,2112422
12	-0,809017	0,14507905	-4,2021156	4,20461931	-88,022635
13	-0,5877853	0,29024783	-3,6443611	3,65590089	-85,446404
14	-0,309017	0,39442805	-3,2228408	3,24688717	-83,022542
15	-1,84E-16	0,47225964	-2,8910766	2,92939467	-80,722628



12. Grafik einbinden

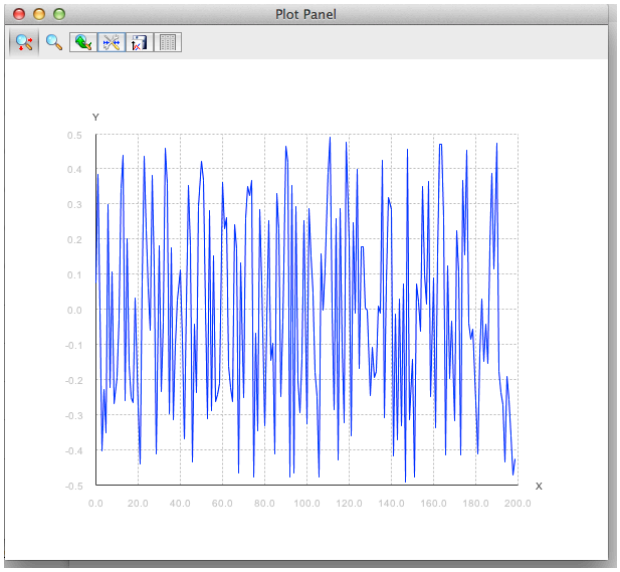
Die Darstellung der Ergebnisse durch Export in eine Datei und Import in Excel erscheint dem Programmierer etwas umständlich. Er möchte daher lieber eine Grafik-Bibliothek einbinden. Im Web findet sich hierzu beispielsweise „jmathplot“ mit der Beschreibung „easy Java scientific plot for math engineering“.

Frage 1: Laden Sie die Bibliothek aus dem Web und binden Sie die Bibliothek in Eclipse ein. Hinweis: Eclipse verwendet einen eigenen Pfad für die Quelltexte und den Objektcode. Um das Archiv „jmathplot.jar“ einzubinden, wählen Sie bitte unter Eclipse im Menü „Project“ den Punkt „Properties“, dann weiter den Punkt „Build Path“. Von dort aus können Sie im Menü „Libraries“ mit der Anweisung „Add external JARs“ (für Java ARchives) die Bibliothek importieren (von dem Verzeichnis, auf das Sie die Datei gespeichert haben). Die Datei sollte auf diesem Verzeichnis verbleiben. Eclipse verwaltet nur eine Pfadangabe auf das Verzeichnis.



Frage 2: Testen Sie die Bibliothek mit einem einfachen Beispiel. Hinweis: Verwenden Sie die Instruktionen „Example Java code“ auf der Web-Seite. Als Werte lassen sich beispielsweise ein Vektor für die x-Achse und Zufallszahlen für die y-Achse verwenden, siehe Beispiel unten.

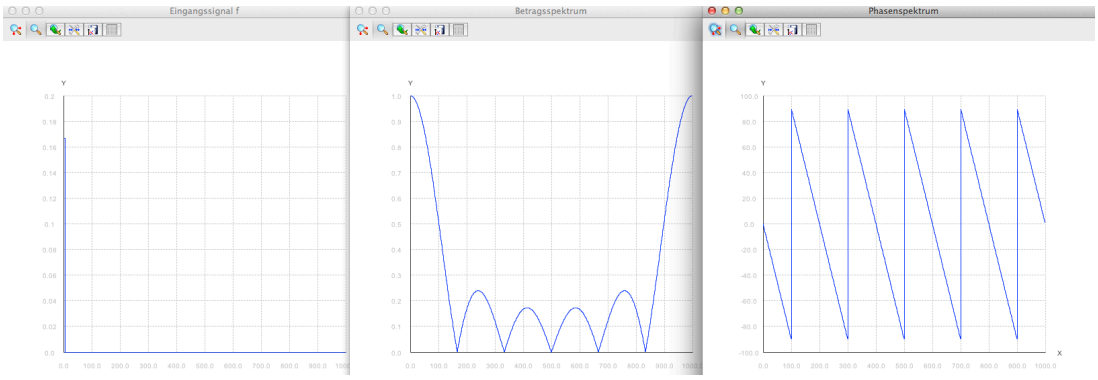
```
package de.dhbwStuttgart.tet;

import javax.swing.*;
import org.math.plot.*;

public class Aufgabe_12 {

      /**
       * @param args
       */
      public static void main(String[] args) {

            // define your data
            double[] x = new double[200];
            double[] y = new double[200];
            for (int i = 0; i<= 199; i++){
                  x[i] = i;
                  y[i] = Math.random() - 0.5;
            };

            // create your PlotPanel
            Plot2DPanel plot = new Plot2DPanel();

            // add line plot to the PlotPanel
            plot.addLinePlot("New Plot", x, y);

            // put the PlotPanel in a JFrame
            JFrame frame = new JFrame("Plot Panel");
            frame.setSize(600, 600);
            frame.setContentPane(plot);
            frame.setVisible(true);

      } //end main
}
```



Frage 3: Verwenden Sie die Grafik-Bibliothek, um grafische Ausgaben zu Aufgabe 10 zu erstellen.

Frage 4: Verwenden Sie die Grafik-Bibliothek, um grafische Ausgaben zu Aufgabe 11 zu erstellen.

Muster (Betragsspektrum und Phasenspektrum für Zeitsignal Rechteck über 6 Stützstellen):



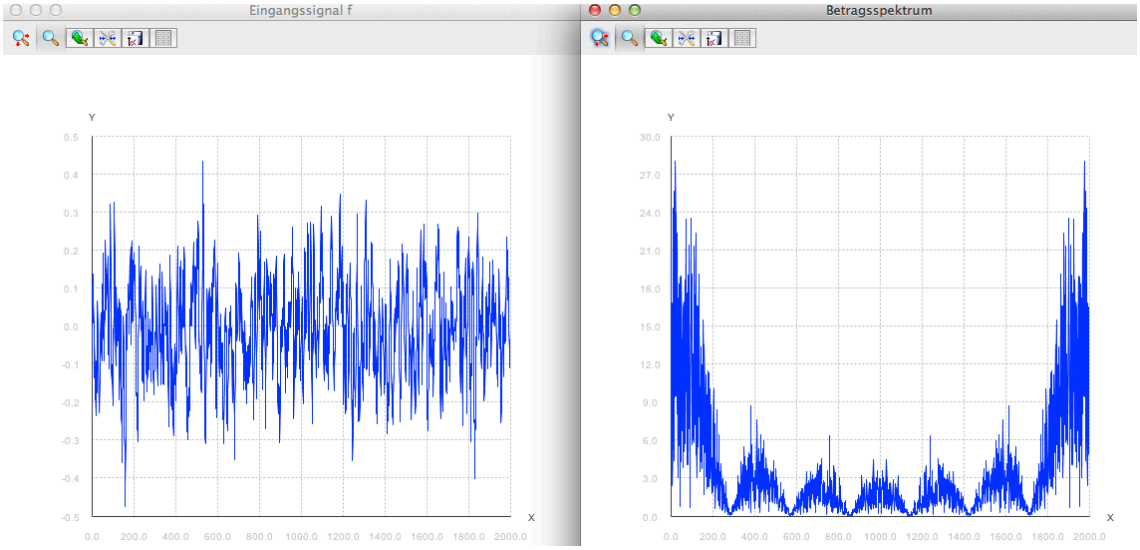
13. Übertragungsfunktion

Das Spektrum der Impulsantwort h[k] eines Systems wird als Übertragungsfunktion bezeichnet. Während im Zeitbereich die Auswirkung des Systems auf ein Eingangssignal x[k] durch Faltung mit der Impulsantwort beschrieben werden kann, ergibt sich im Frequenzbereich der Einfluss des Systems durch Multiplikation der Spektren des Eingangssignals X[u] mit dem Spektrum der Impulsantwort H[u], der sogenannten Übertragungsfunktion.

Frage 1: Ermitteln Sie die Impulsantwort eines Systems durch einen Delta-Impuls als Eingangssignal. Berechnen Sie die Übertragungsfunktion. Hinweis: Geben Sie sich eine Übertragungsfunktion vor, z.B. ein Rechteck, testen Sie diese mit einem Delta-Impuls und transformieren Sie das Ergebnis in den Frequenzbereich. Dieser Fall wäre mit Aufgabe 12.4 bereits abgedeckt.

Frage 2: Ermitteln Sie den Einfluss einer Verzögerung auf die Übertragungsfunktion, beispielsweise durch ein Rechteck, das um einige Stützstellen zeitverschoben reagiert. Hinweis: Hierfür wäre Aufgabe 12.4 geeignet zu erweitern.

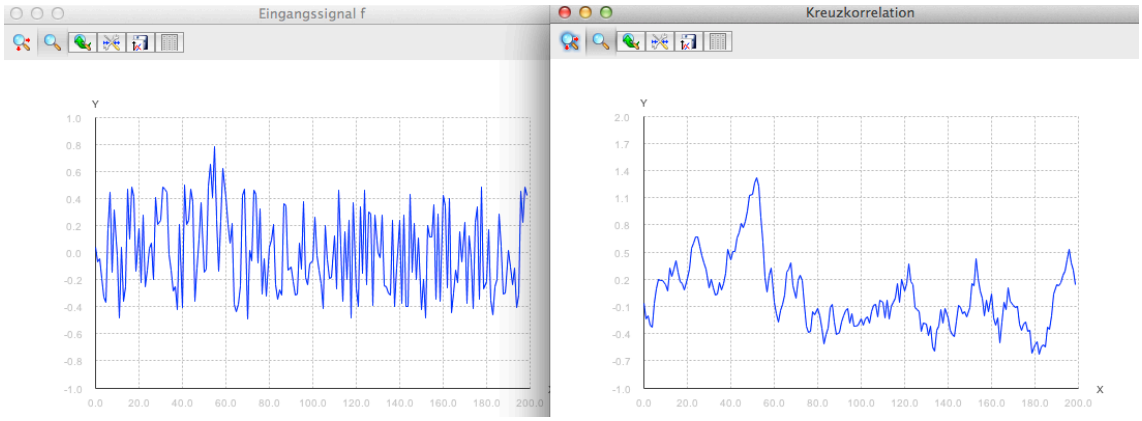
Frage 3: Ermitteln Sie das Betragsspektrum der Übertragungsfunktion eines Systems durch Rauschen als Eingangssignal. Muster: Pseudonoise gefaltet mit einer Rechteckfunktion (siehe Aufgabe 12.4) und zugehöriges Betragsspektrum.



Frage 4: In einem verrauschten Signal lässt sich ein vermutetes Signal durch Kreuzkorrelation ermitteln. Überlagern Sie ein Signal, z.B. ein Rechteck, mit Rauschen und korrelieren Sie das Ergebnis mit dem gesuchten Muster (z.B. ein Rechteck).

Hinweis: die Kreuzkorrelation lässt sich beschreiben durch $z[n] = \sum x[k] \cdot y[k+n]$, wobei der Index k über den gesamten Wertebereich verläuft (d.h. über alle vorhandenen Stützstellen des gesuchten Musters x(k)).

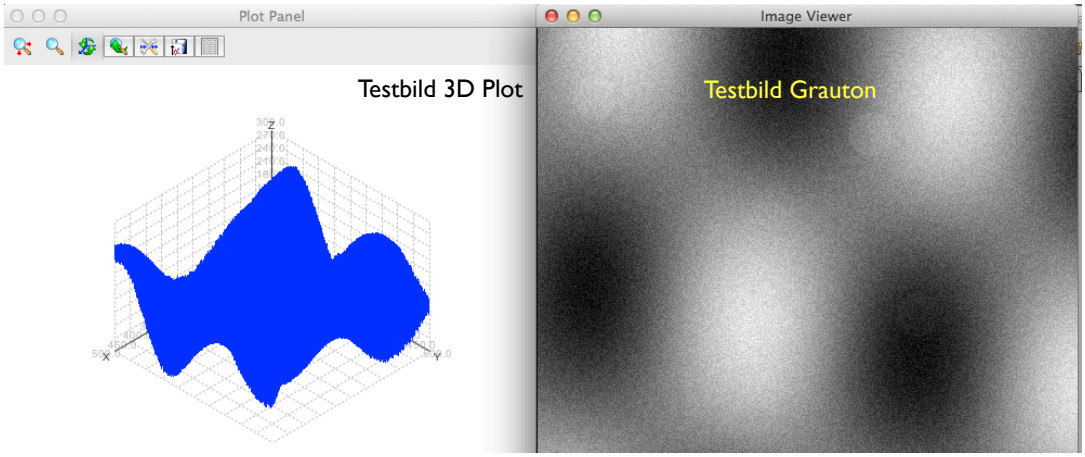
Beispiel: Zwischen den Stützstellen 50 bis 60 ist ein Signal der Amplitude 1/3 im Rauschen enthalten. Die Kreuzkorrelation zeigt die Wahrscheinlichkeit, das Signal irgendwo zu finden. Die Kreuzkorrelation liesse sich weiter normieren auf den Wert 1, indem man durch das Produkt der Effektivwerte beider Signale dividiert (durch $\text{sqrt}\left(\sum x[k]^2 \cdot \sum y[k+n]^2\right)$).



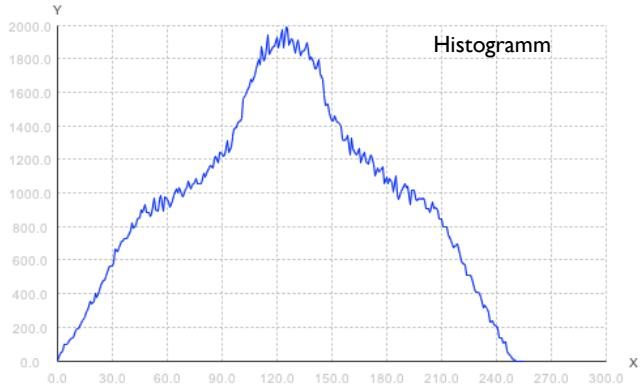
14. Zweidimensionale Felder in der Bildverarbeitung

Bilder lassen sich als zweidimensionale Felder x[k, l] darstellen, wobei jedes Feld einen Wert enthält, der der Helligkeit des Bildes an dieser Stelle enthält. Farbaufnahmen benötigen für jede der Grundfarben (z.B. RGB) ein solches Feld.

Frage 1: Erzeugen Sie ein Testbild, beispielsweise mit einem Grautonverlauf nach einem periodischen Muster. Addieren Sie etwas Rauschen. Addieren Sie einige geometrischen Mustern (z.B. Scheiben, Quadrate, ...). Hinweis: Zur Visualisierung stehen auf der Web-Seite der Übungen einige Klassen mit Hilfsfunktionen zur Verfügung (als 3D-Plot mit „jmathplot“ bzw. Schreiben und Datei und ein „Image Viewer“ zur Anzeige als als Grautonbild). Beispiel:



Frage 2: Berechnen Sie ein Histogramm des Bildes. Zeigen Sie das Histogramm als 2D-Plot an.



Frage 3: Zeichnen Sie das Bild mit einem Tiefpassfilter weich (z.B. ein Rechteck über 5x5 Pixel).

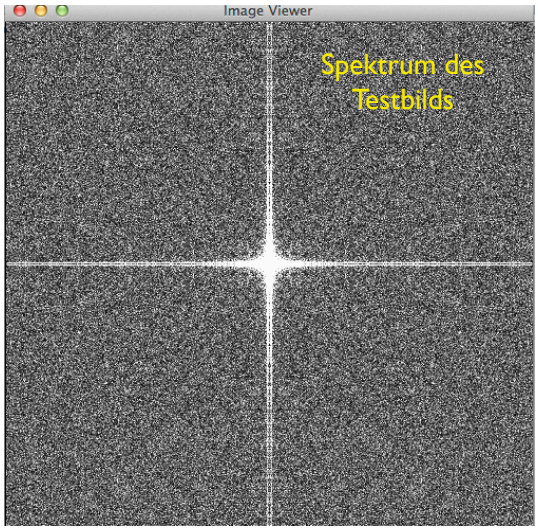
Frage 4: Schärfen Sie das Bild, indem Sie das unscharfe Bild subtrahieren.

Frage 5: Berechnen Sie das Spektrum des Bildes. Hinweis: Die zweidimensionale Fourier-Transformation berechnet sich aus:

$$F[u,v] = \sum \sum f[k,l]\, e^{-j2\pi (k\,u/N + l\,v/M)} \qquad \text{mit } k = 0 \text{ bis } k = N-1,\ l = 0 \text{ bis } M-1 \qquad (14.1)$$

wobei die Indizes k und l über den gesamten Wertebereich verlaufen (alle vorhandenen Stützstellen). Da die Transformation nicht über der Zeit geschieht, sondern über den Ort der Bildpunkte (Pixel), bezeichnet man den Wertebereich [u, v] der Transformierten auch als Ortsfrequenzen. Für Bilder ist speziell das Betragsspektrum (der Amplitudengang) von Interesse. Die Rücktransformation berechnet sich zu:

$$f[k,l] = 1/NM \sum \sum F[u,v]\, e^{+j2\pi (u\,k/N + v\,l/M)} \qquad \text{mit } u = 0 \text{ bis } u = N-1,\ v = 0 \text{ bis } M-1 \quad (14.2)$$



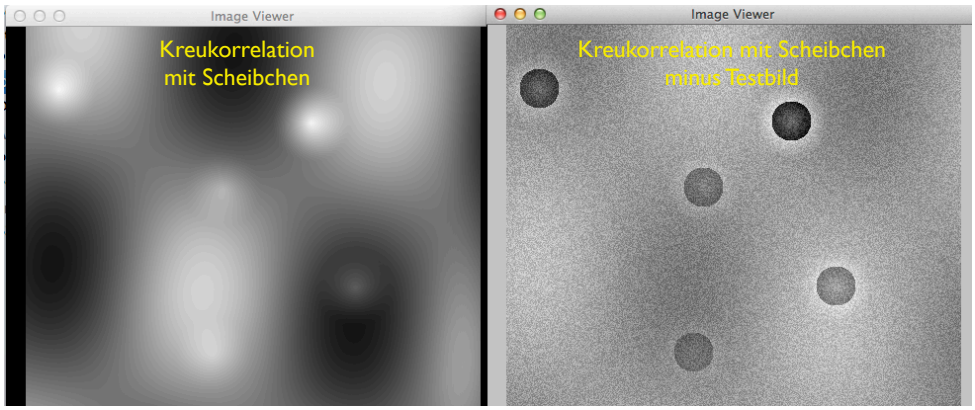
Bemerkungen zum Beispiel: (1) Die Bildgröße beträgt 500x500 Bildpunkte. Die niederfrequenten Anteile liegen also deutlich über der Grösse der Filtermaske (5x5 Bildpunkte). Durch Beschneidung des Spektrums und Rücktransformation liessen sich die niederfrequenten Anteile möglicherweise zugunsten der überlagerten Objekte (kreisförmige Scheiben) beseitigen. (2) Die Transformation benötigt reichlich Rechenzeit. Durch Verwendung der Fast-Fourier-Transformation liesse sich das Verfahren deutlich beschleunigen.

Frage 6: Auch in Bildern lassen sich durch Kreuzkorrelation Objekte suchen. Suchen Sie nach einem der vorhandenen geometrischen Mustern (Scheiben, Quadrate etc.). Hinweis: Den eindimensionalen Fall der Kreuzkorrelation aus Aufgabe 13.4 erweitert man hierzu auf

$$z[u,v] = \sum \sum x[k,l] \cdot y[k+u, l+v] \tag{14.3}$$

Hierbei durchlaufen die Indizes k und l alle vorhandenen Stützstellen des gesuchten Musters x(k)).

Für die Scheibchen im Testbild erhält man folgendes Ergebnis:



15. Java Methoden für Mikrocontroller

Java war ursprünglich als portable Plattform auch für kleine Geräte konzipiert (nach dem Konzept: „write once, run anywhere“). Die quelloffene Hardware-Plattform Arduino verfügt über eine integrierte Entwicklungsplattform (IDE) und ist per Java programmierbar. Hierzu genügen die Implementierung beiden Methoden setup() und loop().

Setup() erlaubt die Definition der Peripherie (PINs als Eingänge oder Ausgänge) und wird einmalig beim Start des Programms aufgerufen (entweder nach dem Übertragen des Programms auf das Board oder nach Drücken des Reset-Tasters). Die zweite Methode loop() stellt die Programmschleife dar, sie wird zyklisch durchlaufen, solange das Arduino-Board eingeschaltet ist.

Beispiel (Quelle: Wikipedia):

```
int ledPin = 13; // die LED ist an Pin 13 angeschlossen, was in der Variablen
                 // ledPin gespeichert ist

void setup() {
    pinMode(ledPin, OUTPUT); // legt den LED-Pin als Ausgang fest
}

void loop() {
    digitalWrite(ledPin, HIGH); // LED anschalten
    delay(1000); // 1 Sekunde warten(=1000ms , delay() braucht den Parameter in ms)
    digitalWrite(ledPin, LOW); // LED ausschalten
    delay(1000); // 1 Sekunde warten
}
```

Wenn Sie Interesse an hardwarenaher Programmierung in Java haben, wäre die Arduino-Plattform ggf. eine geeignete Basis für eigene Experimente.

16. Zeichen und Zeichenketten

Zeichen sind entweder als Bytes kodiert bzw. im Unicode Format mit 2 Bytes. Eine Zeichenkette wird im Format „String“ deklariert und grundsätzlich ähnlich wie eine Datenreihe indiziert. Wie beim Array ist die Stringvariable ein Zeiger auf die Stringobjekte.

Frage 1: Erstellen Sie eine Textdatei mit einem Mustertext. Schreiben Sie ein Programm, das eine Prüfsumme zur Textdatei berechnet. Hinweis: Verwenden Sie für die Verknüpfung von Zeichen zu einer Prüfsumme die Funktion XOR.

Frage 2: Ermitteln Sie die Häufigkeit der Buchstaben im Text.

Frage 3: Kodieren Sie den Text durch Verschieben der Textzeichen um einen vorgegeben Betrag.

Frage 4: Suchen Sie den Text nach einem Schlüsselwort oder einer Buchstabenfolge ab.

17. Klassen

Klassen wurden bereits in Aufgabe 7 verwendet, um Datentypen zu definieren, die über die einfachen Datentypen double, int, boolean etc. hinausgehen, beispielsweise komplexe Zahlen. Dieses Vorgehen ist nicht neu und wird z.B. in der Programmiersprache C durch Strukturen mit dem Schlüsselwort „struct“ verwendet (z.B. struct name { komponente_1, ..., komponente_n }).

Wie bei Arrays und Strings ist die Bezeichnung der Klasse nur der Zeiger auf das Objekt. Wie Arrays müssen Klassen nach ihrer Deklaration vor ihrer ersten Benutzung erzeugt werden. Zuweisungen von Klassen weisen dem Zeiger das gewünschte Objekt zu. Die als Klasse definierten Datentypen lassen sich auch als Rückgabewerte von Methoden verwenden.

Frage 1: Definieren Sie einen Datentyp für komplexe Zahlen.

Frage 2: Schreiben Sie eine Methode für die Umwandlung einer komplexen Zahl in kartesischen Koordinaten in eine komplexe Zahl in Polarkoordinaten. Schreiben Sie eine Methode für die Umkehrfunktion hierzu, bzw. bringen Sie beide Transformationen in einer Methode unter.

Frage 3: Schreiben Sie Methoden für die Addition zweier komplexer Zahlen und für die Multiplikation zweier komplexer Zahlen.

Frage 4: Testen Sie Ihre Methoden in einem Programm, das mit komplexen Zahlen rechnet, beispielsweise zur Berechnung der Summe bzw. des Produktes.

18. Objektorientierung

Klassen als Datenstrukturen und Methoden als modulare Funktionen oder Prozeduren sind nicht neu. Von Objektorientierung ist dann die Rede, wenn die Datenstrukturen innerhalb der Klassen nach dem Prinzip der Kapselung durch klasseneigene Methoden zugänglich sind. Objekte als Instanzen der Klasse tragen ihren individuellen Zustand in den Daten. Für die Kommunikation des Zustands bzw. für Zustandsänderungen stellen die Objekte eigene Methoden zur Verfügung.

Frage 1: Erweitern Sie Ihre Klassen für komplexe Variable um getter() und setter() Methoden.

Frage 2: Erweitern Sie die Klasse so, dass die Klasse sowohl kartesische Koordinaten (Realteil und Imaginärteil) als auch Polarkoordinaten (Betrag und Phase) enthält. Erweitern Sie die getter() und setter() entsprechend.

Frage 3: Ergänzen Sie eine Methode, die bei gegebenen kartesischen Koordinaten die Polarkoordinaten erzeugt. Ergänzen Sie eine Methode, die bei gegebenen Polarkoordinaten die kartesischen Koordinaten erzeugt.

Frage 4: Testen Sie Ihre Klasse mit einem Programm, das mit komplexen Zahlen rechnet, beispielsweise zur Berechnung der Summe bzw. des Produktes. Vergleichen Sie das Konzept mit Aufgabe 17.

Frage 5: Unterbinden Sie den direkten Zugriff auf die Daten der Klassen von außen durch Setzen der Attribute „private“ für Realteil, Imaginärteil, Betrag und Phase. Der Zugriff sollte jetzt nur noch durch die getter() und setter() funktionieren.

Frage 6: Testen Sie das Programm erneut und vergleichen Sie mit Aufgabe 17.

19. Erzeugung und Initialisierung mit dem Konstruktor

Nach der Deklaration eines Objekts einer Klasse wird der Konstruktor mit dem Schlüsselwort „new“ aufgerufen. Der Konstruktor findet sich in der Klassendefinition. Dort ist hinterlegt, was beim Aufruf des Konstruktors zur Initialisierung eines Objektes als Instanz der Klasse geschieht. Ist kein Konstruktor definiert, werden die Daten des Objekts per Default-Konstruktor mit 0 initialisiert.

Frage 1: Definieren Sie eine Klasse für 3-dimensionale Vektoren v(x,y,z).

Frage 2: Definieren Sie einen Konstruktor, mit deren Hilfe sich ein Vektor durch Übergabe seiner Werte als Parameter direkt initialisieren lässt.

Frage 3: Definieren Sie Vektoroperationen für Addition, Multiplikation mit einem Skalar, das Skalarprodukt zweier Vektoren und das Kreuzprodukt zweier Vektoren.

Frage 4: Schreiben Sie ein Programm zum Testen der Vektoroperationen.

20. Methodenkeller (Stack)

Unter einem Methodenkeller versteht man eine Ablage, die nach dem Prinzip „last in - last out“ organisiert ist. Man stapelt also neue Daten auf den alten und nimmt mit dem Arbeitsfortschritt die zuletzt abgelegten Daten wieder vom Stapel. Bei verschachtelten Methodenaufrufen werden so die lokalen Variablen und Rückgabewerte abgelegt und abgearbeitet.

Frage 1: Schreiben Sie eine Klasse vom Typ „Stack“ mit folgenden Attributen: ein Array für die Daten und einen Stapelzeiger auf die Spitze des Stapels.

Frage 2: Ergänzen Sie den Konstruktor.

Frage 3: Ergänzen Sie folgende Methoden: ein Datum ablegen (push()), ein Datum vom Stapel nehmen (pop());

Frage 4: Schreiben Sie ein Programm zum Testen der Klasse. Hinweis: Testen Sie zunächst einfache Operationen, wie Addition mit Hilfe zweier Variabler. Testen Sie dann eine rekursive Funktion, wie zum Beispiel n! (Fakultät von n). Muster (für einen Stack der Tiefe 10):

```
stackpointer at: 0 data = 3
stackpointer at: 1 data = 6
3 + 6 = 9

stackpointer at: 0 data = 5
```

```
stackpointer at: 1 data = 4
stackpointer at: 2 data = 3
stackpointer at: 3 data = 2
stackpointer at: 4 data = 1
faculty (5): 120

stackpointer at: 0 data = 11
stackpointer at: 1 data = 10
stackpointer at: 2 data = 9
stackpointer at: 3 data = 8
stackpointer at: 4 data = 7
stackpointer at: 5 data = 6
stackpointer at: 6 data = 5
stackpointer at: 7 data = 4
stackpointer at: 8 data = 3
stackpointer at: 9 data = 2
stack overflow
```

21. Warteschlange

Eine andere Art der Organisation für die Bedienung von Anfragen folgt dem Prinzip „wer zuerst kommt, wird zuerst bedient“ (engl. FIFO: first in - first out). Man erhält eine Warteschlange. Auch die Warteschlange hat in der Realisierung eine endliche Anzahl von Plätzen. Die Warteschlange lässt sich hierbei als Ringpuffer organisieren.

Frage 1: Schreiben Sie eine Klasse „Queue“ mit einer gegebenen Anzahl von Speicherplätzen, einem Zeiger für den Kopf der Schlange (der als nächstes bedient wird), und einem Zeiger für das Ende der Schlange (wo die nächste Anfrage angestellt wird).

Frage 2: Ergänzen Sie den Konstruktor der Klasse.

Frage 3: Ergänzen Sie die Methoden „put()“ zum Anstellen ans Ende der Warteschlange und „get() für die Entnahme des nächsten Datums vom Kopf der Warteschlange.

Frage 4: Schreiben Sie ein Programm zum Testen der Warteschlange.

22. Vererbung

Klassen lassen sich weiter spezialisieren. Programmiertechnisch erfolgt dies durch Definition einer neuen Klasse, die man als Erweiterung der vorhandenen Klasse deklariert: class Special extends General{}. Hierbei vererbt die allgemeine Klasse ihre Methoden an die spezialisierte Klasse. Die spezialisierte Klasse kann die Methoden verfeinern, überschreiben, bzw. weitere Methoden und Attribute einführen.

Frage 1: Erweitern Sie den Methodenkeller (Stack) aus Aufgabe 20 so, dass der Stack sich automatisch erweitert, falls Seine Größe nicht reicht. Hinweis: Erweitern Sie hierzu die Methode push().

Frage 2: Testen Sie den erweiterten Stack aus Frage 1 mit einem Programm (z.B. indem Sie eine Stackgröße vorgeben, und diese dann durch sukzessives Aufrufen der push()-Methode überschreiten, bzw. mit dem in Aufgabe 20 bereits verwendetem Testprogramm). Muster:

```
Testprogramm:
```

```
stackpointer at: 0 data = 1
stackpointer at: 1 data = 2
stackpointer at: 2 data = 3
size = 2
stackpointer at: 3 data = 4
stackpointer at: 4 data = 5
stackpointer at: 5 data = 6
size = 5
stackpointer at: 6 data = 7
size = 6
7 0 5 4 0 2

Testprogramm aus Aufgabe 20 für GrowableStack extends Stack:
stackpointer at: 0 data = 3
stackpointer at: 1 data = 6
3 + 6 = 9

stackpointer at: 0 data = 5
stackpointer at: 1 data = 4
stackpointer at: 2 data = 3
stackpointer at: 3 data = 2
stackpointer at: 4 data = 1
faculty (5): 120

stackpointer at: 0 data = 11
stackpointer at: 1 data = 10
stackpointer at: 2 data = 9
stackpointer at: 3 data = 8
stackpointer at: 4 data = 7
stackpointer at: 5 data = 6
stackpointer at: 6 data = 5
stackpointer at: 7 data = 4
stackpointer at: 8 data = 3
stackpointer at: 9 data = 2
stackpointer at: 10 data = 1
faculty (11): 39916800
```

Frage 3: Erweitern Sie die der Warteschlange aus Aufgabe 21 zu einer Warteschlange mit Prioritäten.

Frage 4: Schreiben Sie ein Programm zum Testen der der Warteschlange. Muster:

```
public class Aufgabe_22_4 {

	public static void main(String args[]) {
		PriorityQueue q = new PriorityQueue(20);

		q.put(1, 4);		//check put(): add values x at prio p to queue
		q.put(2, 9);
		q.put(3, 3);
		q.put(4, 9);
		q.put(5, 1);
		q.put(6, 2);
		q.put(7, 5);
		q.put(8, 6);

		Out.println();
		q.dump();		//dump queue in order of priorities
```

```
            while (q.length() > 0){
                  Out.print(q.get() + " ");  //check get(): take values from queue
            }
      }
}

      Testlauf:
      put 1 at prio 4 to index 0              //q.put();
      put 2 at prio 9 to index 0
      put 3 at prio 3 to index 2
      put 4 at prio 9 to index 1
      put 5 at prio 1 to index 4
      put 6 at prio 2 to index 4
      put 7 at prio 5 to index 2
      put 8 at prio 6 to index 2

      prio: 9 - data: 2                       //q.dump();
      prio: 9 - data: 4
      prio: 6 - data: 8
      prio: 5 - data: 7
      prio: 4 - data: 1
      prio: 3 - data: 3
      prio: 2 - data: 6
      prio: 1 - data: 5

      2 4 8 7 1 3 6 5                         //q.get();
```

23. Klassifikation

Durch Klassen lassen sich Dinge nach einem einheitlichen Verfahren sortieren, nämlich durch Anwendung eines Klassifikationsschemas. Solche Schemata sind beispielsweise in der Biologie gebräuchlich durch Einteilung von Lebewesen in Klasse (z.B. Säugetiere), Familie (z.B. Katzen), Gattung (z.B. Altwelt-Wildkatzen) und Art (z.B. Wildkatzen). Bei objektorientierten Programmiersprachen verwendet man diese Art der Orientierung zur Klassifikation der Anwendungsumgebung (Domain Model).

Frage 1: Welche Beziehungen zwischen Klassen in der objektorientierten Programmierung kennen Sie? Nennen Sie Beispiele.

Frage 2: Wie werden diese Beziehungen programmiertechnisch in Java abgebildet? Geben Sie einige Beispiele.

Frage 3: Welcher Unterschied besteht zwischen Klassen und Objekten? Was geschieht beim Laden eines Programmcodes? Was geschieht zur Laufzeit eines Programms?

Frage 4: Was versteht man unter einer abstrakten Klasse? Was versteht man unter einem Interface?

24. Dynamische Bindung

Gegeben sei folgende Hierarchie von Tierklassen: (1) „Animal“ als abstrakte Klasse mit dem Attribut „eats“ und den Methoden „saySomething()“ und „getAngry()“. Das Attribut „eats“ soll eine Präferenz für eine bestimmte Sorte Futter wieder geben. Die Klassen sollen konkretisiert werden

durch einen Hund, eine Katze und einen Vogel, die die Methoden spezifisch implementieren und unterschiedliche Präferenzen für Futter haben.

Frage 1: Implementieren Sie die abstrakte Klasse und zugehörigen konkreten Klassen in Java. Je nach Präferenz für Futter erfolgt ein zustimmender Laut der Methode „saySomething()“, bzw. eine Unmutsäusserung durch die Methode „getAngry()“.

Frage 2: Schreiben Sie ein Testprogramm, das die Klassen instanziert, und die Präferenz für Futter sowie die Implementierung der Methoden testet.

Frage 3: Erweitern Sie das Programm so, dass einer Instanz der Klasse Hund, Katze oder Vogel einer Instanz der übergeordneten Klasse „Animal“ zugeordnet wird. Testen Sie das Verhalten, indem Sie die Methoden der Klasse Animal aufrufen.

Frage 4: Erweitern Sie das Programm so, dass die Zuordnung zur übergeordneten Klasse dynamisch erfolgt, z.B. zufällig. Führen Sie eine Fütterung der Tiere durch. Muster:

```
food: hotdog - de.dhbwStuttgart.tet.JavaDog@3343c8b3
woof

food: dogfood - de.dhbwStuttgart.tet.JavaBird@272d7a10
peck

food: grain - de.dhbwStuttgart.tet.JavaCat@1aa8c488
claw

food: grain - de.dhbwStuttgart.tet.JavaBird@272d7a10
tweet
```

25. Ausnahmebehandlung

Laufzeitfehler, wie beispielsweise eine Division durch 0, Zugriff über den Null-Zeiger oder die Überschreitung von Grenzen in Arrays werden von der Laufzeitumgebung behandelt. Das Programm stürzt in einem solchen Fall mit einer Fehlermeldung ab. Darüber hinaus hat der Anwendungsprogrammierer die Möglichkeit, Fehler zu behandeln und es dem Programm zu ermöglichen, im Fehlerfall andere sinnvolle Entscheidungen zu treffen. Zu solchen Fehlern oder Ausnahmesituationen gehören beispielsweise fehlerhaften Daten, fehlenden Daten, bzw. Betriebsmittel, die nicht zur Verfügung stehen.

Die Fehlerbehandlung in Java erfolgt durch Trennung des fehlerfreien Falls vom Fehlerfall: Der fehlerfreie Fall wird als geschützter Block innerhalb einer „try“-Anweisung versuchsweise ausgeführt. Im Ablauf eventuell auftretende Fehlerfälle werden als Ausnahmen gemeldet und in einer Ausnahmeroutine behandelt. Die Ausnahmeroutine ist mit dem Schlüsselwort „catch“ gekennzeichnet. Die Fehlermeldungen sind hierbei in einer Fehlerklasse spezifiziert, die sich für anwendungsspezifische Fehler weiter spezialisieren lässt.

Frage 1: Schauen Sie sich die Methode In.ReadInt() zum Lesen einer Zahl näher an. Wie ist die Methode gegen fehlerhafte Datenformate abgesichert? Wie wird eine Fehlfunktion an den Nutzer der Methode signalisiert?

Frage 2: Zweck der Methode In.readInt() ist die Rückgabe von Integer-Werten. Schreiben Sie in einer eigenen Klasse eine alternative Implementierung der Methode unter der Bezeichnung readInt()

unter Verwendung der Bibliotheksmethode char c = In.read(). Die eigene Methode readInt() soll eine Ausnahme melden, wenn ein gelesenes Zeichen keine Ziffer war.

Frage 3: Schreiben Sie ein Programm zum Testen der alternativen Methode, beispielsweise durch Lesen der Werte aus einer Datei. Testen Sie die Methode.

Frage 4: Welche Art von Ausnahmen braucht man in Java als Anwendungsprogrammierer nicht abzufangen? Was muss man tun, wenn man in einer Methode Benutzerausnahmen melden möchte? Welche Konsequenz hat die erforderliche Spezifikation für den Java-Compiler? Wie wird die Ausnahmemeldung ausgelöst? Wer empfängt die Ausnahmemeldung?

26. Threads

Java Threads sind Kontrollflüsse, die innerhalb der Laufzeitumgebung quasiparallel ausgeführt werden, also nebenläufig zueinander. Die Klasse Thread stellt für quasiparallele Abläufe die Methoden start(), run() und sleep() zur Verfügung und wird vom Anwender weiter spezialisiert. Nebenläufige Aktivitäten lassen sich so durch Instanzierung der Klasse Thread erzeugen. Die Aktivitäten des Threads werden in der Methode run() untergebracht. Der Thread wird durch Aufruf der Methode start() gestartet. Mit Hilfe der Methode sleep(int ms) kann man den Thread für eine Zeit (in Millisekunden) in den Wartezustand versetzen.

Da Threads im Adressraum des Programms laufen, das sie erzeigt und startet, haben Threads auch Zugriff auf die diesbezüglichen Daten und Methoden. Der quasiparallele Betrieb ist naturgemäß nicht frei von Konflikten. Zur Synchronisation nebenläufiger Threads stehen daher weitere Methoden zur Verfügung. Hierzu gehören (1) die Anweisung synchronized(), die es erlaubt, kritische Bereiche zu definieren, die nur jeweils von einem Thread durchlaufen werden dürfen. Während ein Thread diesen Bereich beansprucht, müssen andere Threads warten. Die Methoden (2) wait() ermöglicht es Threads, die auf das belegte Betriebsmittel zugreifen möchten, zu warten. Die Methode (3) notify() ermöglicht es, wartende Threads zu aktivieren.

Frage 1: Schreiben Sie eine Methode, die als Thread ein vorgegebenes Zeichen auf der Konsole ausgibt. Stellen Sie die Geschwindigkeit der Ausgabe mit Hilfe der Sleep-Methode so ein, dass man die Ausgabe bequem beobachten kann.

Frage 2: Schreiben Sie einen weiteren Thread, der von der Konsole eingegebene Zeichen (Benutzereingabe) liest. Das Zeichen soll dem schreibenden Thread mitgegeben werden. Definieren Sie ein Zeichen, das beide Tasks beendet.

Frage 3: Schreiben Sie ein Programm zum Test beider Threads.

Frage 4: Schreiben Sie ein Programm, das einen Puffer einer vorgegebenen Größe enthält, den ein Thread beschreibt und ein anderer Thread liest. Um Zugriffskonflikte bzw. nicht konsistente Daten zu vermeiden, soll die Zugriffsmethoden des Puffers zum Schreiben und Lesen mit Hilfe der Methode synchronized() geschützt werden.

Deutsch - Englisch

Allerweltsprogramm	Hello world program
Ansichten	views
Arbeitsbereich	workspace
Ausnahmebehandlung	exception handling
Ausnahme melden	to throw an exception
baumelnde Else	dangling else
Betrag	absolute value
Bildpunkt	pixel (picture element)
Datenreihe (Feld, Matrix)	array
Gleitkommazahl	floating point number
Klassenkampf	class struggle
Luftwurzel	aerial root
Methodenkeller	stack
Warteschlange	queue
Wortversteinerung	fossile expression
Zeichenkette	string
Zusicherung	assertion

Abkürzungen

IDE	Integrated Development Environment
SDK	Software Development Kit

Literatur

(1) Hanspeter Mössenböck, Sprechen Sie Java?: Eine Einführung in das systematische Programmieren, dpunkt Verlag, 2011 (4. Auflage), ISBN-13: 978-3898645959

(2) Eclipse Entwicklungsumgebung für Java: http://www.eclipse.org/downloads/

(3) Ein- und Ausgabebibliotheken für diese Übungen: http://ssw.jku.at/JavaBuch/, sowie Übungsaufgaben und Vorlesungsfolien zu (1)

(4) Ed Burnette, Jörg Staudenmeyer, Eclipse IDE - kurz & gut, Verlag: O'Reilly, 2009 (2. Auflage), ISBN-13: 978-3897215528

(5) Kent Beck, JUnit Pocket Guide, Verlag: O'Reilly Media, 2004, ISBN-13: 978-0596007430

(6) Kathy Sierra, Bert Bates, Lars Schulten, Elke Buchholz, Java von Kopf bis Fuß, Verlag: O'Reilly, 2006, ISBN-13: 978-3897214484

(7) Übersicht über die Java-Klassenbibliothek (Javadoc): http://docs.oracle.com/javase/7/docs/api/index.html

(8) Stefan Middendorf, Reiner Singer, Jörn Heid, Java Programmierhandbuch und Referenz für die Java 2-Plattform, Standard Edition, dpunkt Verlag, 2002 (3. Auflage), ISBN-13: 978-3898641579, Online-Referenz:http://www.dpunkt.de/java/index.html