

Beispielprogramme aus Grundkurs Programmieren in Java, 6. Auflage

Hier finden Sie die Beispielprogramme aus dem Buch. Suchen Sie einfach anhand des Inhaltsverzeichnisses die Stelle, an der sich das Programm im Buch befindet, und navigieren Sie über den Link in den entsprechenden Teil des Dokuments.

Inhaltsverzeichnis

1. Einleitung

1. Java - mehr als nur kalter Kaffee?
2. Java für Anfänger - das Konzept dieses Buches
3. Weitere Infos und Kontakt zu den Autoren
4. Verwendete Schreibweisen

2. Einige Grundbegriffe aus der Welt des Programmierens

1. Computer, Software, Informatik und das Internet
2. Was heißt Programmieren?

3. Aller Anfang ist schwer

[1. Mein erstes Programm](#)

2. Formeln, Ausdrücke und Anweisungen
3. Zahlenbeispiele
4. Verwendung von Variablen
5. Auf den Schirm!
6. Das Programmgerüst
7. Eingeben, Übersetzen und ausführen
8. Übungsaufgaben

4. Grundlagen der Programmierung in Java

1. Grundelemente eines Java-Programms

1. Kommentare
2. Bezeichner und Namen
3. Literale
4. Reservierte Wörter, Schlüsselwörter
5. Trennzeichen
6. Interpunktionszeichen
7. Operatorsymbole
8. import-Anweisungen
9. Zusammenfassung
- [10. Übungsaufgaben](#)

2. Erste Schritte in Java

- [1. Grundstruktur eines Java-Programms](#)
2. Ausgaben auf der Konsole
3. Eingaben von der Konsole
- [4. Schöner programmieren in Java](#)
5. Zusammenfassung
- [6. Übungsaufgaben](#)

3. Einfache Datentypen

- [1. Ganzzahlige Datentypen](#)
- [2. Gleitkommatypen](#)
3. Der Datentyp char für Zeichen
4. Zeichenketten
5. Der Datentyp boolean für Wahrheitswerte
6. Implizite und explizite Typumwandlungen
7. Zusammenfassung
8. Übungsaufgaben

4. Der Umgang mit einfachen Datentypen

[1. Variablen](#)

2. Operatoren und Ausdrücke

3. Allgemeine Ausdrücke

[4. Ein- und Ausgabe](#)

[1. Statischer Import der IOTools-Methoden](#)

5. Zusammenfassung

[6. Übungsaufgaben](#)

5. Anweisungen und Ablaufsteuerung

1. Anweisungen

2. Blöcke und ihre Struktur

3. Entscheidungsanweisung

4. Wiederholungsanweisungen, Schleifen

5. Sprungbefehle und markierte Anweisungen

6. Zusammenfassung

[7. Übungsaufgaben](#)

5. Referenzdatentypen

[1. Felder](#)

1. Was sind Felder?

2. Deklaration, Erzeugung und Initialisierung von Feldern

[3. Felder unbekannter Länge](#)

[4. Referenzen](#)

[5. Ein besserer Terminkalender](#)

[6. Mehrdimensionale Felder](#)

7. Mehrdimensionale Felder unterschiedlicher Länge

8. Vorsicht, Falle: Kopieren von mehrdimensionalen Feldern

[9. Vereinfachte for-Schleifen-Notation](#)

10. Zusammenfassung

11. Übungsaufgaben

2. Klassen

1. Was sind Klassen?

2. Deklaration und Instantiierung von Klassen

3. Komponentenzugriff bei Objekten

[4. Ein erstes Adressbuch](#)

[5. Klassen als Referenzdatentyp](#)

[6. Felder von Klassen](#)

7. Vorsicht, Falle: Kopieren von geschachtelten Referenzdatentypen

[8. Auslagern von Klassen](#)

9. Zusammenfassung

[10. Übungsaufgaben](#)

6. Methoden, Unterprogramme

[1. Methoden](#)

1. Was sind Methoden?

2. Deklaration von Methoden

3. Parameter Übergabe und Ergebnissrückgabe

[4. Aufruf von Methoden](#)

5. Überladen von Methoden

[6. Variable Argument-Anzahl bei Methoden](#)

[7. Vorsicht, Falle: Referenzen als Parameter](#)

[8. Sichtbarkeit und Verdecken von Variablen](#)

9. Zusammenfassung

[10. Übungsaufgaben](#)

2. Rekursiv definierte Methoden

[1. Motivation](#)

[2. Gute und schlechte Beispiele für rekursive Methoden](#)

[3. Zusammenfassung](#)

3. Die Methode main

[1. Kommandozeilenparameter](#)

[2. Anwendung der vereinfachten for-Schleifen-Notation](#)

[3. Zusammenfassung](#)

[4. Übungsaufgaben](#)

4. Methoden aus anderen Klassen aufrufen

[1. Klassenmethoden](#)

[2. Die Methoden der Klasse java.lang.Math](#)

[3. Statischer Import](#)

5. Methoden von Objekten aufrufen

[1. Instanzmethoden](#)

[2. Die Methoden der Klasse java.lang.String](#)

[6. Übungsaufgaben](#)

7. Die objektorientierte Philosophie

1. Die Welt, in der wir leben
2. Programmierparadigmen ? Objektorientierung im Vergleich
3. Die vier Grundpfeiler objektorientierter Programmierung

1. Generalisierung

2. Vererbung
3. Kapselung
4. Polymorphismus
5. Weitere wichtige Grundbegriffe

4. Modellbildung ? von der realenWelt in den Computer
5. Zusammenfassung
6. Übungsaufgaben

8. Der grundlegende Umgang mit Klassen

1. Vom Referenzdatentyp zur Objektorientierung

2. Instanzmethoden

1. Zugriffsrechte

2. Was sind Instanzmethoden?

3. Instanzmethoden zur Validierung von Eingaben

4. Instanzmethoden als erweiterte Funktionalität

3. Statische Komponenten einer Klasse

1. Klassenvariablen und -methoden

2. Klassenkonstanten

4. Instantiierung und Initialisierung

- [1. Konstruktoren](#)
- [2. Überladen von Konstruktoren](#)
- [3. Der statische Initialisierer](#)
- [4. Der Mechanismus der Objekterzeugung](#)

5. Zusammenfassung

- [6. Übungsaufgaben](#)

9. Vererbung und Polymorphismus

1. Wozu braucht man Vererbung?

- 1. Aufgabenstellung
- 2. Analyse des Problems
- [3. Ein erster Ansatz](#)
- [4. Eine Klasse für sich](#)
- 5. Stärken der Vererbung
- [6. Vererbung verhindern durch final](#)
- [7. Übungsaufgaben](#)

[2. Die super-Referenz](#)

- 3. Überschreiben von Methoden und Variablen

[1. Dynamisches Binden](#)

- [2. Überschreiben von Methoden verhindern durch final](#)

- 4. Die Klasse java.lang.Object
- 5. Übungsaufgaben
- [6. Abstrakte Klassen und Interfaces](#)
- [7. Übungsaufgaben](#)
- 8. Weiteres zum Thema Objektorientierung

- 1. Erstellen von Paketen
- 2. Zugriffsrechte
- [3. Innere Klassen](#)
- [4. Anonyme Klassen](#)

- 9. Zusammenfassung
- [10. Übungsaufgaben](#)

[10. Exceptions und Errors](#)

- 1. Eine Einführung in Exceptions

- 1. Was ist eine Exception?
- 2. Übungsaufgaben
- [3. Abfangen von Exceptions](#)
- [4. Ein Anwendungsbeispiel](#)
- 5. Die RuntimeException
- [6. Übungsaufgaben](#)

- 2. Exceptions für Fortgeschrittene

- [1. Definieren eigener Exceptions](#)
- 2. Übungsaufgaben
- [3. Vererbung und Exceptions](#)
- [4. Vorsicht, Falle!](#)
- [5. Der finally-Block](#)

- 6. Die Klassen Throwable und Error
- 7. Zusammenfassung
- [8. Übungsaufgaben](#)

3. Assertions

- [1. Zusicherungen im Programmcode](#)
- 2. Compilieren des Programmcodes
- 3. Ausführen des Programmcodes
- 4. Zusammenfassung

11. Fortgeschrittene objektorientierte Programmierung

1. Aufzählungstypen

- [1. Deklaration eines Aufzählungstyps](#)
- [2. Instanzmethoden der enum-Objekte](#)
- [3. Selbstdefinierte Instanzmethoden für enum-Objekte](#)
- [4. Übungsaufgaben](#)

2. Generische Datentypen

- [1. Generizität in alten Java-Versionen](#)
- [2. Generizität ab Java 5.0](#)
- [3. Einschränkungen der Typ-Parameter](#)
- [4. Wildcards](#)
- [5. BoundedWildcards](#)
- [6. Generische Methoden](#)
- 7. Ausblick
- [8. Übungsaufgaben](#)

[3. Sortieren von Feldern und das Interface Comparable](#)

12. Einige wichtige Hilfsklassen

1. Die Klasse StringBuffer

[1. Arbeiten mit String-Objekten](#)

2. Arbeiten mit StringBuffer-Objekten
3. Übungsaufgaben

2. Die Wrapper-Klassen (Hüll-Klassen)

[1. Arbeiten mit eingepackten Daten](#)

2. Aufbau der Wrapper-Klassen
- [3. Ein Anwendungsbeispiel](#)
- [4. Automatische Typwandlung für die Wrapper-Klassen](#)
5. Übungsaufgaben

3. Die Klassen BigInteger und BigDecimal

[1. Arbeiten mit langen Ganzzahlen](#)

2. Aufbau der Klasse BigInteger
3. Übungsaufgaben
- [4. Arbeiten mit langen Gleitkommazahlen](#)
5. Aufbau der Klasse BigDecimal
- [6. Viele Stellen von Nullstellen gefälltig?](#)
7. Übungsaufgaben

4. Die Klasse DecimalFormat

- [1. Standard-Ausgaben in Java](#)
- [2. Arbeiten mit Format-Objekten](#)
- [3. Vereinfachte formatierte Ausgabe](#)
4. Übungsaufgaben

5. Die Klassen Date und Calendar

1. Arbeiten mit Zeitpunkten
- [2. Auf die Plätze, fertig, los!](#)
- [3. Spezielle Calendar-Klassen](#)
- [4. Noch einmal: Zeitmessung](#)
5. Übungsaufgaben

6. Die Klassen SimpleDateFormat und DateFormat

- [1. Arbeiten mit Format-Objekten für Datum/Zeit-Angaben](#)
2. Übungsaufgaben

7. Die Collection-Klassen

1. Sammlungen von Objekten ? Der Aufbau des Interface Collection
2. Sammlungen durchgehen ? Der Aufbau des Interface Iterator
3. Mengen

1. Das Interface Set
- [2. Die Klasse HashSet](#)
3. Das Interface SortedSet
- [4. Die Klasse TreeSet](#)

4. Listen

1. Das Interface List

[2. Die Klassen ArrayList und LinkedList](#)

[3. Suchen und Sortieren ? Die Klassen Collections und Arrays](#)

5. Übungsaufgaben

[8. Die Klasse StringTokenizer](#)

1. Übungsaufgaben

13. Aufbau grafischer Oberflächen in Frames - von AWT nach Swing

1. Grundsätzliches zum Aufbau grafischer Oberflächen

[2. Ein einfaches Beispiel mit dem AWT](#)

[3. Let's swing now!](#)

[4. Etwas Fill-in gefällig?](#)

5. Die AWT- und Swing-Klassenbibliothek im Überblick

6. Übungsaufgaben

14. Swing-Komponenten

1. Die abstrakte Klasse Component

2. Die Klasse Container

[3. Die abstrakte Klasse JComponent](#)

4. Layout-Manager, Farben und Schriften

- [1. Die Klasse Color](#)
- [2. Die Klasse Font](#)
3. Layout-Manager

- [1. Die Klasse FlowLayout](#)
- [2. Die Klasse BorderLayout](#)
- [3. Die Klasse GridLayout](#)

5. Einige Grundkomponenten

- [1. Die Klasse JLabel](#)
2. Die abstrakte Klasse AbstractButton
- [3. Die Klasse JButton](#)
- [4. Die Klasse JToggleButton](#)
- [5. Die Klasse JCheckBox](#)
- [6. Die Klassen JRadioButton und ButtonGroup](#)
- [7. Die Klasse JComboBox](#)
- [8. Die Klasse JList](#)
9. Die abstrakte Klasse JTextComponent
- [10. Die Klassen JTextField und JPasswordField](#)
- [11. Die Klasse JTextArea](#)
- [12. Die Klasse JScrollPane](#)
- [13. Die Klasse JPanel](#)

6. Spezielle Container, Menüs und Toolbars

1. Die Klasse JFrame
2. Die Klasse JWindow
- [3. Die Klasse JDialog](#)
4. Die Klasse JMenuBar
- [5. Die Klasse JToolBar](#)

[7. Übungsaufgaben](#)

15. Ereignisverarbeitung

1. Zwei einfache Beispiele

[1. Zufällige Grautöne als Hintergrund](#)

[2. Ein interaktiver Bilderrahmen](#)

2. Programmiervarianten für die Ereignisverarbeitung

1. Innere Klasse als Listener-Klasse

[2. Anonyme Klasse als Listener-Klasse](#)

[3. Container-Klasse als Listener-Klasse](#)

[4. Separate Klasse als Listener-Klasse](#)

3. Event-Klassen und -Quellen

[4. Listener-Interfaces und Adapter-Klassen](#)

[5. Listener-Registrierung bei den Event-Quellen](#)

[6. Auf die Plätze, fertig, los!](#)

[7. Übungsaufgaben](#)

16. Einige Ergänzungen zu Swing-Komponenten

1. Zeichnen in Swing-Komponenten

1. Grafische Darstellung von Komponenten

2. Das Grafik-Koordinatensystem

[3. Die abstrakte Klasse Graphics](#)

[4. Ein einfaches Zeichenprogramm](#)

5. Layoutveränderungen und der Einsatz von revalidate

2. Noch mehr Swing gefällig?
3. Übungsaufgaben

17. Applets

1. Erstellen und Ausführen von Applets

1. Vom Frame zum Applet am Beispiel
2. Applet in HTML-Datei einbetten
3. Applet Über HTML-Datei ausführen

2. Die Methoden der Klasse JApplet
3. Zwei Beispiele

1. Auf die Plätze, fertig, los!
2. Punkte verbinden im Applet

4. Details zur HTML-Einbettung

1. Der Applet-Tag
2. Die Methode showDocument

5. Sicherheitseinschränkungen bei Applets
6. Übungsaufgaben

18. Parallele Programmierung mit Threads

[1. Ein einfaches Beispiel](#)

2. Threads in Java

[1. Die Klasse Thread](#)

[2. Das Interface Runnable](#)

[3. Threads vorzeitig beenden](#)

3. Wissenswertes Über Threads

4. Thread-Synchronisation und -Kommunikation

[1. Das Leser/Schreiber-Problem](#)

[2. Das Erzeuger/Verbraucher-Problem](#)

5. Threads in Frames und Applets

[1. Auf die Plätze, fertig, los!](#)

[2. Spielereien](#)

3. Swing-Komponenten sind nicht Thread-sicher

[6. Übungsaufgaben](#)

19. Ein- und Ausgabe Über Streams

1. Grundsätzliches zu Streams in Java

[2. Dateien und Verzeichnisse ? Die Klasse File](#)

3. Ein- und Ausgabe Über Character-Streams

- [1. Einfache Reader- und Writer-Klassen](#)
- [2. Gepufferte Reader- und Writer-Klassen](#)
- [3. Die Klasse StreamTokenizer](#)
- [4. Die Klasse PrintWriter](#)
5. Die Klassen IOTools und Scanner

- [1. Was machen eigentlich die IOTools?](#)
- [2. Konsoleneingabe über ein Scanner-Objekt](#)

4. Ein- und Ausgabe Über Byte-Streams

- [1. Einige InputStream- und OutputStream-Klassen](#)
- [2. Die Serialisierung und Deserialisierung von Objekten](#)
3. Die Klasse PrintStream

5. Einige abschließende Bemerkungen
6. Übungsaufgaben

20. Client/Server-Programmierung in Netzwerken

1. Wissenswertes Über Netzwerk-Kommunikation

1. Protokolle
- [2. IP-Adressen](#)
3. Ports und Sockets

2. Client/Server-Programmierung

1. Die Klassen ServerSocket und Socket

- [2. Ein einfacher Server](#)
- [3. Ein einfacher Client](#)
- [4. Ein Server für mehrere Clients](#)
- [5. Ein Mehrzweck-Client](#)

3. Wissenswertes Über URLs

- [1. Client/Server-Kommunikation Über URLs](#)
- [2. Netzwerkverbindungen in Applets](#)

4. Übungsaufgaben

21. Neuerungen in Java 7

1. Spracherweiterungen

1. Elementare Datentypen und Anweisungen

- [1. Binäre ganzzahlige Literalkonstanten](#)
- [2. Unterstrich als Trennzeichen in Literalkonstanten](#)
- [3. Strings in der switch-Anweisung](#)

[2. Verkürzte Notation bei generischen Datentypen](#)

3. Ausnahmebehandlung

- [1. Mehrere Ausnahme-Typen in einem catch-Block](#)
- [2. try-Block mit Ressourcen](#)

2. Erweiterungen der Klassenbibliothek

- 1. Dateien und Verzeichnisse

- [1. Das Interface Path und die Klasse Paths](#)
- [2. Die Klasse Files](#)

[2. Grafische Oberflächen](#)

22. Blick über den Tellerrand

- 1. Der Vorhang fällt
- 2. A fool with a tool
- 3. Alles umsonst?
- 4. Und fachlich?
- 5. Zu guter Letzt

3.1: Mein erstes Programm

[Berechnung.java](#) | Gibt die Summe zweier Zahlen auf dem Bildschirm aus

4.1.10: Übungsaufgaben

[Berechnung.java](#) | Fehlerhafte Berechnungs-Klasse aus den Übungsaufgaben

4.2.1: Grundstruktur eines Java-Programms

[HalloWelt.java](#) | Das wohl berühmteste Einsteiger-Programm der Programmiergeschichte

4.2.4: Schöner programmieren in Java

[Unsorted.java](#) | Dieses Programm ist ein Beispiel dafür, wie das Einhalten weniger

[Version A](#)

[Version B](#)

4.2.6: Übungsaufgaben

[Strukturuebung.java](#) Übungsaufgabe zu den Coding-Richtlinien

4.3.1: Ganzzahlige Datentypen

[Longst.java](#)

4.3.2: Gleitkommatypen

[Floatdiv.java](#) Division bei Gleitkommazahlen

[Intdiv.java](#) Division bei ganzen Zahlen

[Version A](#)

[Version B](#)

4.4.1: Variablen

[StundenRechner1.java](#) Stundenlohn-Berechnung ohne Variablen

[StundenRechner2.java](#) Stundenlohn-Berechnung unter Verwendung von Variablen

4.4.4: Ein- und Ausgabe

[IOToolsTest.java](#) Beispielprogramm für die Eingabe über Tastatur

4.4.4.1: Statischer Import der IOTools-Methoden

[IOToolsTestMitStaticImport.java](#) Beispielprogramm für die Eingabe über Tastatur (mit statischem Import)

4.4.6: Übungsaufgaben

[Plus.java](#) Übungsaufgabe zur Verwendung von Operatoren

[Raviolita.java](#) Programm mit Lücken

4.5.7: Übungsaufgaben

[BreakAndContinue.java](#) Übungsaufgabe

[Falsch.java](#) Übungsaufgabe

[Quersumme.java](#) Übungsaufgabe

5.1: Felder

[UmstaendlicherKalender.java](#) Dieses Programm zeigt, wie ein Terminkalender ohne Verwendung von Feldern

5.1.3: Felder unbekannter Länge

[Swap.java](#) Vertauscht ein Feld von Zahlen

5.1.4: Referenzen

[Doppel.java](#) Das Programm verdoppelt eine einzelne Zahl

[DoppelFeld.java](#) Das Programm verdoppelt ein Feld von Zahlen

5.1.5: Ein besserer Terminkalender

[BessererKalender.java](#) Kalenderprogramm unter Verwendung von Arrays

5.1.6: Mehrdimensionale Felder

[MonatsKalender.java](#) Das Kalenderprogramm, gültig für einen ganzen Monat

5.1.9: Vereinfachte for-Schleifen-Notation

[MehrSchleifen.java](#) Traditionelle und neue, vereinfachte Schleifen-Notation mehrdimensional

[Schleifen.java](#) Traditionelle und neue, vereinfachte Schleifen-Notation eindimensional

5.2.4: Ein erstes Adressbuch

[AdressBuch_v1.java](#) Eine einfache Adressverwaltung Funktioniert nur mit einer Adresse

5.2.5: Klassen als Referenzdatentyp

[AdressBuch_v2.java](#) Eine einfache Adressverwaltung für zwei Adressen

5.2.6: Felder von Klassen

[AdressBuch_v3.java](#) Eine einfache Adressverwaltung für zwanzig Adressen

5.2.8: Auslagern von Klassen

[Adresse.java](#) Die Adresse, in eine eigene Java-Datei ausgelagert

5.2.10: Übungsaufgaben

[Komponente.java](#) Übungsaufgabe

[Referenzen.java](#) Übungsaufgabe

6.1: Methoden

[Eval1.java](#) Das Programm führt eine simple Berechnung aus (Potenz)

[Eval2.java](#) Das Programm führt dieselbe Berechnung wie Eval1 aus, jedoch mehr als einmal Der Co

6.1.4: Aufruf von Methoden

[AufrufTest.java](#) Was passiert, wenn wir den Inhalt von übergebenen Parametern in der Methode ab

6.1.6: Variable Argument-Anzahl bei Methoden

[Argumente.java](#) Summationsmethode mit variabler Argumentliste

6.1.7: Vorsicht, Falle: Referenzen als Parameter

[AufrufTest2.java](#) Vorsicht Falle: Übergabe von Arrays an Methoden

[AufrufTest3.java](#) Explizites Kopieren von Feldinhalten

6.1.8: Sichtbarkeit und Verdecken von Variablen

[VerdeckenTest.java](#) Verdecken von Variablen beim Methodenaufruf

6.1.10: Übungsaufgaben

[BooleanMethods.java](#) Übungsaufgabe

6.2.1: Motivation

[Unendlichkeit.java](#) Was passiert, wenn Methoden sich selbst aufrufen?

6.2.2: Gute und schlechte Beispiele für rekursive Methoden

[IndirekteRekursion.java](#) Beispiel für Endlosrekursion

6.3.1: Kommandozeilenparameter

[GrussWort.java](#) Das Programm liest Kommandozeilenparameter aus und verwertet sie in einem Gru

6.3.4: Übungsaufgaben

[Signatur.java](#) Übungsaufgabe

6.4.1: Klassenmethoden

[MeineMethoden.java](#) Klasse, die Methoden bereitstellt

[TesteMethoden.java](#) Klasse, die die Methoden der Klasse verwendet

6.4.3: Statischer Import

[StatischImports.java](#) Klasse, die statisch importierte Math-Methoden benutzt

6.5.1: Instanzmethoden

[Multiplizierer.java](#) Klasse mit Instanzvariable und Instanzmethode

[TesteMultiplizierer.java](#) Klasse, die Instanzmethoden von verschiedenen Objekten einsetzt

6.5.2: Die Methoden der Klasse java.lang.String

[StringTest.java](#) Einige Beispiele für den Umgang mit Strings

6.6: Übungsaufgaben

[Tausche.java](#) Übungsaufgabe

7.3.1: Generalisierung

[FaxAdresse.java](#) Erweiterung des Adress-Objektes

8.1: Vom Referenzdatentyp zur Objektorientierung

[Student.java](#) Studenten-Klasse, erster Versuch

8.2.1: Zugriffsrechte

[Student.java](#) Aus Gründen der Kapselung wurden die Instanzvariablen auf private gesetzt

8.2.2: Was sind Instanzmethoden?

[Student.java](#) Der Zugriff auf private Bestandteile der Klasse wird durch Methoden gekapselt

8.2.3: Instanzmethoden zur Validierung von Eingaben

[Student.java](#) Die Matrikelnummer kann auf Gültigkeit geprüft werden

8.2.4: Instanzmethoden als erweiterte Funktionalität

[Student.java](#) Die Klasse besitzt jetzt eine toString-Methode

8.3.1: Klassenvariablen und -methoden

[Student.java](#) Neuer Bestandteil: ein Instanzzähler

8.3.2: Klassenkonstanten

[Student.java](#) Konstanten definieren die Studienfächer

8.4.1: Konstruktoren

[Student.java](#) Neu: ein selbstdefinierter argumentloser Konstruktor

8.4.2: Überladen von Konstruktoren

[Student.java](#) Die Studenten-Klasse, diesmal mit zwei Konstruktoren

[Version A](#)

[Version B](#)

8.4.3: Der statische Initialisierer

[Student.java](#) Das Phantom der Uni - erzeugt durch einen statischen Initialisierer

8.4.4: Der Mechanismus der Objekterzeugung

[Student.java](#) Die Klasse Student

[Sub.java](#) Subklasse für den Test der Aufrufreihenfolge beim Erzeugen von Objekten

[Super.java](#) Superklasse für den Test der Aufrufreihenfolge beim Erzeugen von Objekten

8.6: Übungsaufgaben

[AchJa.java](#)

[Fehler1.java](#)

[Fehler2.java](#)

[Fehler3.java](#)

[Fehler4.java](#)

[Fehler5.java](#)

[Fehler6.java](#)

[Fuchs.java](#)

[Glasboden.java](#)

[Hund.java](#)

[IntKlasse.java](#)

[KarlsruherStudent.java](#)

[Katze.java](#)

[Klang.java](#)

[KlassenTest.java](#)

[Krach.java](#)

[Maus.java](#)

[Musik.java](#)

[Patient.java](#)

[Ratte.java](#)

[RefIntKlasse.java](#)

[Reifen.java](#)

[TennisSpieler.java](#)

[TestStrecke.java](#)

[TestZwei.java](#)

9.1.3: Ein erster Ansatz

[USDollar.java](#) Eine erste, konkrete Währung

[Waehrung.java](#) Diese Klasse stellt allgemein Geldwerte dar

9.1.4: Eine Klasse für sich

[Yen.java](#) Die japanische Währung

9.1.6: Vererbung verhindern durch final

[Kid.java](#) Test: Vererbung Verhindern durch final

[NoKidsPlease.java](#) Klasse zur Demonstration des Schutzes vor Vererbung

9.1.7: Übungsaufgaben

[Euro.java](#) Die europäische Währung

9.2: Die super-Referenz

[NonsenseDollar.java](#) Klasse mit einer völlig unsinnig überschriebenen Methode ;-)

9.3.1: Dynamisches Binden

[Sohn.java](#) Test: Dynamisches Binden

[Vater.java](#) Test: Dynamisches Binden

9.3.2: Überschreiben von Methoden verhindern durch final

[Kind.java](#) Test: Überschreiben von Methoden

[Papa.java](#) Test: Überschreiben von Methoden

9.6: Abstrakte Klassen und Interfaces

[Goldbarren.java](#) Ein simpler Wertgegenstand

[Kruegerrand.java](#) Eine Klasse, die sowohl Waehrung als auch Wertgegenstand ist

[Waehrung.java](#) Die Klasse Waehrung, erweitert um einige Methoden

[Wertgegenstand.java](#) Dieses Interface repräsentiert einen Wertgegenstand

9.7: Übungsaufgaben

[KreisPlot.java](#) Übungsaufgabe: Zeichnen von Kreisen

9.8.3: Innere Klassen

[Aufzaehlung.java](#) Mit Hilfe einer Folge wird durch eine Ansammlung von Objekten iteriert

[Version A](#)

[Version B](#)

[Folge.java](#) Interface fuer die Festlegung der Methoden zur Abarbeitung der Elemente

9.8.4: Anonyme Klassen

[Aufzaehlung.java](#) Hier wurde die Folge in einer anonymen Klasse erzeugt

9.10: Übungsaufgaben

[A.java](#)

[ABCD.java](#)

[B.java](#)

[Bildschirm.java](#)

[Brot.java](#)

[C.java](#)

[D.java](#)

[Mahlzeit.java](#)

[MetallPlatte.java](#)

[Mittagessen.java](#)

[Point.java](#)

[Salat.java](#)

[Sandwich.java](#)

[SpielFigur.java](#)

[Vesper.java](#)

[Wurst.java](#)

10: Exceptions und Errors

[Excep1.java](#) Ein sehr einfaches Programm, welches aber bereits eine Exception produzieren kann

10.1.3: Abfangen von Exceptions

[Excep2.java](#) Die bei Division durch 0 auftretende Exception wird abgefangen

10.1.4: Ein Anwendungsbeispiel

[LiesAusDatei.java](#) Das Programm liest einzelne Zeichen aus einer Datei

[LiesDatei_2.java](#) Das Programm liest einzelne Zeichen aus einer Datei und behandelt Exceptions g

[LiesDatei_2b.java](#) Das Programm liest einzelne Zeichen aus einer Datei und behandelt Exceptions

10.1.6: Übungsaufgaben

[Exueb1.java](#)

[Exueb2.java](#)

[Exueb3.java](#)

[Exueb4.java](#)

[Exueb5.java](#)

[Exueb6.java](#)

[Exueb7.java](#)

10.2.1: Definieren eigener Exceptions

[DigitException.java](#) Eigene Exception-Klasse zur Erkennung von Ziffern und Erzeugen entsprechen

[LiesAusDatei_3.java](#)

10.2.3: Vererbung und Exceptions

[PingException.java](#) Die Ping-Exception

[PingPong.java](#) Ein Programm, welches Exceptions auslöst

[Version A](#)

[Version B](#)

[PingPongException.java](#) Die PingPong-Exception

[PongException.java](#) Die Pong-Exception

10.2.4: Vorsicht, Falle!

[PingPong.java](#) Ein Programm, welches Exceptions auslöst

[Version A](#)

[Version B](#)

10.2.5: Der finally-Block

[PingPong.java](#)

Ein Programm, welches Exceptions auslöst

[Version A](#)

[Version B](#)

[Version C](#)

[Version D](#)

[Version E](#)

10.2.8: Übungsaufgaben

[Exueb8.java](#)

10.3.1: Zusicherungen im Programmcode

[AssertionTest.java](#) Methode kehrtwert mit einer Zusicherung

11.1.1: Deklaration eines Aufzählungstyps

[Jahreszeit.java](#) Die vier Jahreszeiten als enum-Typ

11.1.2: Instanzmethoden der enum-Objekte

[Aufzaehlungen.java](#) Verwendung des Jahreszeiten-enum-Typs

11.1.3: Selbstdefinierte Instanzmethoden für enum-Objekte

[Enumerations.java](#) Verwendung des Noten-Typs

[Noten.java](#) Die Noten auf der Klavier-Tastatur

11.1.4: Übungsaufgaben

[EsWarEinmal.java](#) Das Maerchen-Programm zu Aufgabe 143

[StudentenTest.java](#) Test-Klasse für Aufgabe 141

[StudentenTest2.java](#) Test-Klasse für Aufgabe 142

11.2.1: Generizität in alten Java-Versionen

[Ohrring.java](#) Simple Klasse fuer Ohrring-Objekte

[Paar.java](#) Klasse für Paare vom Typ Object

[PaarTest1.java](#) Testprogramm 1 fuer die Paar-Klasse

[PaarTest2.java](#) Testprogramm 2 fuer die Paar-Klasse

[Socke.java](#) Simple Klasse fuer Socken-Objekte

11.2.2: Generizität ab Java 5.0

[GenPaar.java](#) Generische Paar-Klasse

[GenPaarTest1.java](#) Tesprogramm 1 für die generische Paar-Klasse

[GenPaarTest2.java](#) Tesprogramm 2 für die generische Paar-Klasse

11.2.3: Einschränkungen der Typ-Parameter

[Hemd.java](#) Simple Klasse für Hemden-Objekte

[Hose.java](#) Simple Klasse für Hosen-Objekte

[Kleidung.java](#) Simple Klasse für Kleidungs-Objekte

[TollesPaar.java](#) Generische Paar-Klasse mit eingeschraenktem Typ-Parameter

[TollesPaarTest.java](#) Testprogramm für die Klasse TollesPaar

11.2.4: Wildcards

[TollesPaarTestWild1.java](#) Testprogramm ohne Wildcards fuer die Klasse TollesPaar

[TollesPaarTestWild2.java](#) Testprogramm mit Wildcards fuer die Klasse TollesPaar

11.2.5: Bounded Wildcards

[GenPaarTestWild1.java](#) Testprogramm 1 für die GenPaar-Klasse mit Bounded Wildcard

[GenPaarTestWild2.java](#) Testprogramm 2 für die GenPaar-Klasse mit Bounded Wildcard

[GenPaarTestWild3.java](#) Testprogramm 3 für die GenPaar-Klasse mit Bounded Wildcard

[Jeans.java](#) Simple Klasse fuer Jeans-Objekte

11.2.6: Generische Methoden

[GenPaarTest3.java](#) Testprogramm mit generischen Methoden

[GenPaarTest5.java](#) Testprogramm mit generischen Methoden und eingeschränkten Typ-Parametern

11.2.8: Übungsaufgaben

[Konzert.java](#) Programm zu Aufgabe 147

[RateMal.java](#) Programm zu Aufgabe 146

[TierKaefig.java](#) Programm zu Aufgabe 144

[Tierleben.java](#) Programm zu Aufgabe 145

11.3: Sortieren von Feldern und das Interface Comparable

[Ring.java](#) Klasse zur Darstellung von Ringobjekte, die nach ihrer Groesse sortiert werden koennen

[RingDemo.java](#) Testprogramm zur zufaelligen Erzeugung und Sortierung von zehn Ringen

12.1.1: Arbeiten mit String-Objekten

[StringRefs.java](#) Diese Klasse stellt String-Objekte als Referenzdatentyp vor

12.2.1: Arbeiten mit eingepackten Daten

[WrapperBeispiel.java](#) Diese Klasse zeigt die Funktionsweise von Wrapperklassen

12.2.3: Ein Anwendungsbeispiel

[Summiere.java](#) Diese Klasse stellt ein Summations-Programm dar

12.2.4: Automatische Typwandlung für die Wrapper-Klassen

[AutoBoxing.java](#) Verwendung von Wrapper-Objekten mit AutoBoxing

[AutoBoxingDangers.java](#) Gefahren bei der Verwendung von Wrapper-Objekten

[Boxing.java](#) Verwendung von Wrapper-Objekten mit explizitem Boxing

12.3.1: Arbeiten mit langen Ganzzahlen

[QuadrierungBigInt.java](#) Diese Klasse ist ein modifiziertes Quadrierungsprogramm

[QuadrierungLong.java](#) Diese Klasse quadriert die Zahl 16 mehrfach

12.3.4: Arbeiten mit langen Gleitkommazahlen

[ProduktSummeBigDec.java](#) Diese Klasse korrigiert Fehler der Klasse ProduktSummeDouble

[ProduktSummeDouble.java](#) Diese Klasse gibt bei der Rechnung mit Gleitkommazahlen einzelne Re

12.3.6: Viele Stellen von Nullstellen gefällig?

[BigNewton.java](#) Diese Klasse berechnet näherungsweise die Nullstellen der Funktion $\sqrt{2}$ mit Hil

12.4.1: Standard-Ausgaben in Java

[StandardFormat.java](#) Diese Klasse zeigt wie man das Ausgabeformat von double- Werten beeinflus

12.4.2: Arbeiten mit Format-Objekten

[MyFormats.java](#) Diese Klasse zeigt an weiteren Beispielen "selbst gemachte" Formate zur Ausgabe

12.4.3: Vereinfachte formatierte Ausgabe

[Ausgaben.java](#) Formatierte Ausgabe mit printf

12.5.2: Auf die Plätze, fertig, los!

[Stoppuhr.java](#) Diese Klasse realisiert eine einfache Stoppuhr

12.5.3: Spezielle Calendar-Klassen

[CalArith.java](#) Diese Klasse gibt "Zeitpunkte" aus

12.5.4: Noch einmal: Zeitmessung

[CalStoppuhr.java](#) Diese Klasse ist eine alternative Version der "einfachen Stoppuhr" und arbeitet mit

12.6.1: Arbeiten mit Format-Objekten für Datum/Zeit-Angaben

[MyDateFormats.java](#) Diese Klasse gibt noch einige weitere Beispiele für "selbst gestrickte" Formate

[MyStandardDateFormats.java](#) Diese Klasse gibt noch einige Beispiele für Standard-Formate zur Au

12.7.3.2: Die Klasse HashSet

[ZahlenMenge.java](#) Diese Klasse zeigt die Verwendung einer Collection vom Typ HashSet

12.7.3.4: Die Klasse TreeSet

[SortierteZahlenMenge.java](#) Diese Klasse zeigt die Verwendung einer Collection vom Typ TreeSet

12.7.4.2: Die Klassen ArrayList und LinkedList

[ZahlenListe.java](#) Diese Klasse zeigt die Anwendung der List- Klassen am Beispiel der Klasse Array

12.7.4.3: Suchen und Sortieren ? Die Klassen Collections und Arrays

[SortierteZahlenListe.java](#) Diese Klasse demonstriert die Anwendung der Klasse Collections

12.8: Die Klasse StringTokenizer

[StringTokens.java](#) Diese Klasse ist ein Beispielprogramm zur Verwendung von StringTokenizer- Ob

13.2: Ein einfaches Beispiel mit dem AWT

[FrameOhneInhaltAWT.java](#) Diese Klasse erzeugt einen Frame mit den Mitteln des AWT- Paketes

13.3: Let's swing now!

[FrameOhneInhalt.java](#) Diese Klasse erzeugt einen Swing- Frame mit den von uns benötigten Erwei

[FrameOhneInhaltSwing.java](#) Diese Klasse erzeugt einen Frame mit den Mitteln von Swing- Klassen

13.4: Etwas Fill-in gefällig?

[FrameMitText.java](#) Diese Klasse erzeugt einen Frame mit Text

14.3: Die abstrakte Klasse JComponent

[FrameMitTextUndToolTip.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.4.1: Die Klasse Color

[FarbigesLabel.java](#) Diese Klasse kann ein farbiges Label darstellen

[FrameMitSchwarzemLabel.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.4.2: Die Klasse Font

[FrameMitMonospacedText.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.4.3.1: Die Klasse FlowLayout

[FrameMitFlowLayout.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.4.3.2: Die Klasse BorderLayout

[FrameMitBorderLayout.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.4.3.3: Die Klasse GridLayout

[FrameMitGridLayout.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.1: Die Klasse JLabel

[FrameMitBild.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

[images.html](#) Grafiken die vom Beispielprogramm benötigt werden

14.5.3: Die Klasse JButton

[FrameMitButtons.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.4: Die Klasse JToggleButton

[FrameMitToggleButton.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.5: Die Klasse JCheckBox

[FrameMitCheckBoxes.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.6: Die Klassen `JRadioButton` und `ButtonGroup`

[FrameMitRadioButtons.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.7: Die Klasse `JComboBox`

[FrameMitComboBoxes.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.8: Die Klasse `JList`

[FrameMitListe.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.10: Die Klassen `JTextField` und `JPasswordField`

[FrameMitTextFeldern.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.11: Die Klasse `JTextArea`

[FrameMitTextArea.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.12: Die Klasse `JScrollPane`

[FrameMitScrollText.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

14.5.13: Die Klasse `JPanel`

[FrameMitPanels.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

[images.html](#) Grafiken die vom Beispielprogramm benötigt werden

14.6.3: Die Klasse `JDialog`

[TopLevelContainer.java](#) Diese Klasse demonstriert die Anwendung der drei Fenster-Arten `Frame`, `V`

14.6.5: Die Klasse `JToolBar`

[FrameMitMenuBar.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

[images.html](#) Grafiken die vom Beispielprogramm benötigt werden

14.7: Übungsaufgaben

[NotenEingabe.java](#) Übungsaufgabe

[VierButtonFrame.java](#) Übungsaufgabe

15.1.1: Zufällige Grautöne als Hintergrund

[Farbwechsel.java](#) Diese Klasse erzeugt ein Swing-Fenster mit einem Button, der in der Lage ist die

15.1.2: Ein interaktiver Bilderrahmen

[Bilderrahmen.java](#) Diese Klasse erzeugt ein Swing-Fenster mit einem Menue, einer Toolbar und ein

[images.html](#) Grafiken die vom Beispielprogramm benötigt werden

15.2.2: Anonyme Klasse als Listener-Klasse

[Farbwechsel2.java](#) Diese Klasse ist eine Version unserer Farbwechsel- Klasse, allerdings mit anony

15.2.3: Container-Klasse als Listener-Klasse

[Farbwechsel3.java](#) Diese Klasse ist eine Version der Farbwechsel- Klasse, die nun selbst eine Liste

15.2.4: Separate Klasse als Listener-Klasse

[ButtonListener.java](#) Diese Klasse ist ein Beispiel für eine eigenständige Listener- Klasse

[Farbwechsel4.java](#) Diese Klasse benutzt die Listener- Klasse `ButtonListener`

15.4: Listener-Interfaces und Adapter-Klassen

[CloseToggleButtons.java](#) Diese Klasse erzeugt ein Swing-Fenster mit zwei Toggle-Buttons, die zum

15.5: Listener-Registrierung bei den Event-Quellen

[LookAndFeel.java](#) Diese Klasse erzeugt ein Swing-Fenster, das mit Buttons und Combo-Box sein L

15.6: Auf die Plätze, fertig, los!

[StoppuhrFrame.java](#) Diese Klasse erzeugt ein Swing-Fenster mit Stoppuhrfunktion

15.7: Übungsaufgaben

[EuroConverter.java](#) Übungsaufgabe

16.1.3: Die abstrakte Klasse `Graphics`

[ZeichenPanel.java](#) Diese Klasse ist eine spezielle `JPanel`-Klasse mit verschiedenen `Graphics`- Meth

[Zeichnung.java](#) Diese Klasse erzeugt ein Swing-Fenster mit einer Zeichnung

16.1.4: Ein einfaches Zeichenprogramm

[PunkteVerbinden.java](#) Diese Klasse erzeugt ein Swing-Fenster mit einem Zeichenbrett

[Zeichenbrett.java](#) Diese Klasse ist eine spezielle JPanel-Klasse für unser Zeichenbrett

16.1.5: Layoutveränderungen und der Einsatz von revalidate

[NewButtonFrame1.java](#) Diese Klasse erzeugt ein Swing-Fenster, das sich wie erwartet verhält

[NewButtonFrame2.java](#) Diese Klasse erzeugt ein Swing-Fenster und demonstriert die Funktion von

17.1.1: Vom Frame zum Applet am Beispiel

[AppletMitText.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

[FrameMitText.java](#) Diese Klasse macht nichts anderes als ihr Name verspricht

17.1.2: Applet in HTML-Datei einbetten

[AppletMitText.html](#) HTML- Datei zur Einbettung eines Applets

17.2: Die Methoden der Klasse JApplet

[AppletMethoden.html](#) Dieses Applet implementiert die Methoden init, start, stop und destroy

[AppletMethoden.java](#) Dieses Applet implementiert die Methoden init, start, stop und destroy

17.3.1: Auf die Plätze, fertig, los!

[StoppuhrApplet.html](#) Dieses Applet realisiert eine Stoppuhr

[StoppuhrApplet.java](#) Dieses Applet realisiert eine Stoppuhr

17.3.2: Punkte verbinden im Applet

[PunkteVerbindenApplet.html](#) Dieses Applet erzeugt mit Hilfe unserer Zeichenbrett-Klasse ein Zeichenbrett

[PunkteVerbindenApplet.java](#) Dieses Applet erzeugt mit Hilfe unserer Zeichenbrett-Klasse ein Zeichenbrett

[Zeichenbrett.java](#) Die Zeichenbrett-Klasse

17.4.1: Der Applet-Tag

[ParameterApplet.html](#) Dieses Applet bekommt die Beschriftungen des Border-Layouts als Parameter

[ParameterApplet.java](#) Dieses Applet bekommt die Beschriftungen des Border-Layouts als Parameter

17.4.2: Die Methode showDocument

[GoogleButtonApplet.html](#) Dieses Applet hat einen Button, der in der Lage ist, auf die vom Browser angezeigte URL zu klicken

[GoogleButtonApplet.java](#) Dieses Applet hat einen Button, der in der Lage ist, auf die vom Browser angezeigte URL zu klicken

17.5: Sicherheitseinschränkungen bei Applets

[AppletMitBild.html](#) Dieses Applet zeigt ein Bild- Label, funktioniert aber nur im Appletviewer

[AppletMitBild.java](#) Dieses Applet zeigt ein Bild- Label, funktioniert aber nur im Appletviewer

[AppletMitBildBrowse.html](#) Dieses Applet zeigt ebenfalls ein Bild- Label und funktioniert auch im Webbrowser

[AppletMitBildBrowse.java](#) Dieses Applet zeigt ebenfalls ein Bild- Label und funktioniert auch im Webbrowser

[images.html](#) Grafiken die vom Beispielprogramm benötigt werden

17.6: Übungsaufgaben

[Utils.java](#) Übungsaufgabe

18.1: Ein einfaches Beispiel

[ABCPrinter.java](#) Diese Klasse gibt das Alphabet aus

[ABCThread.java](#) Diese Klasse gibt auch das Alphabet aus, ist jedoch ein Thread

[MehrmalsP.java](#) Diese Klasse führt zweimal hintereinander die run- Methode der ABCPrinter- Klasse aus

[MehrmalsT.java](#) Diese Klasse führt zweimal hintereinander die run- Methode der ABCThread- Klasse aus

18.2.1: Die Klasse Thread

[MachMal.java](#) Diese Klasse macht eine Sekunde lang gar nichts

[TVProgAuslosung.java](#) Diese Klasse lost zwischen versch TV- Programmen

[TVProgThread.java](#) Diese Klasse stellt die Funktionalität unserer Programmauslosung als Thread dar

18.2.2: Das Interface Runnable

[ABCRunnable.java](#) Diese Klasse gibt das Alphabet aus und implementiert das Runnable- Interface

[MehrmalsR.java](#) Diese Klasse führt zweimal hintereinander die run- Methode der ABCRunnable- Klasse aus
[TVProgAuslosungMitRunnable.java](#) Diese Klasse lost zwischen versch TV- Programmen
[TVProgRunnable.java](#) Diese Klasse stellt die Funktionalität unserer Programmauslosung dar und im

18.2.3: Threads vorzeitig beenden

[StoppuhrMitThread.java](#) Diese Klasse realisiert eine Stoppuhr mit sich aktualisierender Uhrzeit
[UhrzeitThread.java](#) Diese Klasse gibt Sekundenweise die aktuelle Uhrzeit zurück

18.4.1: Das Leser/Schreiber-Problem

[Figur.java](#) Diese abstrakte Klasse stellt eine Schachfigur dar
[FigurenThreads1.java](#) Diese Klasse demonstriert das Leser/ Schreiber- Problem
[Leser.java](#) Diese Klasse stellt unseren Leser dar
[SchlechteFigur.java](#) Diese Klasse führt bei Verwendung zu Problemen
[Schreiber.java](#) Diese Klasse stellt unseren Schreiber dar

18.4.2: Das Erzeuger/Verbraucher-Problem

[EVTest1.java](#) Diese Klasse ist ein Testprogramm zur Demonstration des Erzeuger/ Verbraucher- Problems
[EVTest2.java](#) Diese Klasse demonstriert die Lösung des Erzeuger/ Verbraucher- Problems
[EVTest3.java](#) Diese Klasse demonstriert die Deadlock- Situation
[Erzeuger.java](#) Diese Klasse erzeugt einen int- Wert
[GuterWert.java](#) In dieser Klasse ist das Problem gelöst
[KlemmWert.java](#) Diese Klasse führt bei Verwendung zum Deadlock
[SchlechterWert.java](#) Diese Klasse führt bei Verwendung zu Problemen
[Verbraucher.java](#) Diese Klasse "verbraucht" einen int- Wert
[Wert.java](#) Diese abstrakte Klasse repräsentiert einen int- Wert

18.5.1: Auf die Plätze, fertig, los!

[AnzeigeThread.java](#) Diese Klasse dient zur dynamischen Zeitanzeige
[StoppuhrFrameThread.java](#) Diese Klasse stellt eine inzwischen fast komplette Stoppuhr zur Verfügung

18.5.2: Spielereien

[AutomatApplet.html](#) Diese Klasse ist ein einfacher Spielautomat
[AutomatApplet.java](#) Diese Klasse ist ein einfacher Spielautomat
[ColorRunLabel.java](#) Diese Klasse brauchen wir für die Label
[KnopfListener.java](#) Diese Klasse ist ein Listener der beim Druck auf einen Button eine veränderliche Variable auslöst
[StartStopButton.java](#) Diese Klasse benötigen wir für die Knöpfe

18.6: Übungsaufgaben

[KartenTerminal.java](#) Übungsaufgabe
[KonzertDaten.java](#) Übungsaufgabe
[UseTerminals.java](#) Übungsaufgabe

19.2: Dateien und Verzeichnisse ? Die Klasse File

[Create.java](#) Diese Klasse arbeitet mit File- Objekten

19.3.1: Einfache Reader- und Writer-Klassen

[WriteToFile.java](#) Diese Klasse schreibt etwas in eine Datei

19.3.2: Gepufferte Reader- und Writer-Klassen

[BufferedWriteToFile.java](#) Diese Klasse kann gepuffert etwas in eine Datei schreiben

19.3.3: Die Klasse StreamTokenizer

[ZahlenSumme.java](#) Diese Klasse liest von der Tastatur eine Zeichenfolge ein um sie dann zu zerlegen

19.3.4: Die Klasse PrintWriter

[PrintWriting.java](#) Diese Klasse kann mit Hilfe der print- Methode verschiedene Werte ausgeben

19.3.5.1: Was machen eigentlich die IOTools?

[InTools.java](#) Diese Klasse zeigt wie die IOTools funktionieren

19.3.5.2: Konsoleneingabe über ein Scanner-Objekt

[Eingaben.java](#) Konsolen-Eingabe unter Verwendung der Scanner-Klasse

19.4.1: Einige InputStream- und OutputStream-Klassen

[DataWriteAndRead.java](#) Diese Klasse speichert elementare Werte in einer Datei und liest sie wieder

19.4.2: Die Serialisierung und Deserialisierung von Objekten

[Datensatz.java](#) Diese Klasse demonstriert das Prinzip der Serialisierung

[ObjectRead.java](#) Diese Klasse schreibt Objekte

[ObjectWrite.java](#) Diese Klasse liest Objekte

20.1.2: IP-Adressen

[DNSAnfrage.java](#) Diese Klasse startet eine DNS- Anfrage

20.2.2: Ein einfacher Server

[DateTimeProtokoll.java](#) Diese Klasse stellt das Protokoll für unseren Zeitserver zur Verfügung

[DateTimeServer.java](#) Diese Klasse realisiert einen einfachen Zeit- Server

20.2.3: Ein einfacher Client

[DateTimeClient.java](#) Diese Klasse ist ein Client für unseren Zeitserver

20.2.4: Ein Server für mehrere Clients

[DateTimeDienst.java](#) Diese Klasse stellt das Protokoll für unseren DateTimeMultiServer zur Verfügung

[DateTimeMultiServer.java](#) Diese Klasse kann mehrere Client gleichzeitig bedienen

20.2.5: Ein Mehrzweck-Client

[MyClient.java](#) Diese Klasse ist ein Multifunktions- Client

20.3.1: Client/Server-Kommunikation über URLs

[LiesURL.java](#) Diese Klasse liest den Text eines Webdokuments

20.3.2: Netzwerkverbindungen in Applets

[DateTimeApplet.java](#) Diese Klasse ist ein Applet das Anfragen an unseren DateTimeMultiServer stellt

21.1.1.1: Binäre ganzzahlige Literalkonstanten

[BinaereLiterale.java](#) Diese Klasse arbeitet mit Literalkonstanten zu verschiedenen Basen

21.1.1.2: Unterstrich als Trennzeichen in Literalkonstanten

[Unterstrich.java](#) Diese Klasse demonstriert die Verwendung von Unterstrichen in numerischen Literalen

[UnterstrichFalsch.java](#) Diese NICHT COMPILIERBARE Klasse demonstriert UNZULÄSSIGE Verwendung von Unterstrichen

21.1.1.3: Strings in der switch-Anweisung

[StringSwitchDemo.java](#) Diese Klasse demonstriert den Einsatz von Strings in der switch-Anweisung

21.1.2: Verkürzte Notation bei generischen Datentypen

[GPaar.java](#) Generischer Datentyp für die Beispielprogramme

[GPaarDemo.java](#) Bisherige Notation beim Einsatz generischer Datentypen

[GPaarDiamondDemo.java](#) Verkürzte Notation beim Einsatz generischer Datentypen

[GPaarDiamondFalsch.java](#) Unzulässige Verwendung der verkürzten Notation beim Einsatz generischer Datentypen

[ListDiamondDemo.java](#) Verkürzte Notation beim Einsatz generischer Collections

[ListDiamondFalsch.java](#) Unzulässige Verwendung der verkürzten Notation beim Einsatz generischer Collections

21.1.3.1: Mehrere Ausnahme-Typen in einem catch-Block

[Wurzel.java](#) Programm mit herkömmlichem Exception-Handling

[WurzelMultiCatch.java](#) Programm mit Exception-Handling für zwei Ausnahme-Typen in einem catch-Block

[WurzelNochmalMultiCatch.java](#) Programm mit Exception-Handling für drei Ausnahme-Typen in einem catch-Block

[WurzelNochmalMultiCatchFalsch.java](#) NICHT COMPILIERBARES Programm mit unzulässigem Exception-Handling

21.1.3.2: try-Block mit Ressourcen

[Kopiere.java](#) Umständlicher Umgang mit mehreren Strömen und Ausnahmebehandlung in herkömmlichem try-Block

[KopiereTWR.java](#) Vereinfachter Umgang mit mehreren Strömen und Ausnahmebehandlung durch t

[ZeigeDateiInhalt.java](#) Umständlicher Umgang mit einem Strom und Ausnahmebehandlung in herkö

[ZeigeDateiInhaltTWR.java](#) Vereinfachter Umgang mit einem Strom und Ausnahmebehandlung durc

21.2.1.1: Das Interface Path und die Klasse Paths

[PathBeispiele.java](#) Zugriffe auf Dateisysteme-Pfade mittels Path und Paths

21.2.1.2: Die Klasse Files

[DateiOperationen.java](#) Dateioperationen mit der Klasse Files aus dem Paket java.nio.file

21.2.2: Grafische Oberflächen

[LookAndFeelWithNimbus.java](#) Demonstration der Look-and-Feel-Varianten Nimbus und Metal