
Java™

Wiederholung

- Deklaration einer Struktur
  - ```
    public class Strukturname {
        Attribute
    }
    ```
  - Strukturname (= Klassenname)
    - Möglichst ein Substantiv, groß geschrieben
  - Attribute (Variablen)
    - Wie Variablendeklaration,
      aber zusätzlich Schlüsselwort **`public`** vorangestellt
    - Initialwert ist der „Standard-Wert“ mit dem die leere Struktur startet
- Der Strukturname steht nach der Deklaration in anderen Java-Programmen als Datentyp zur Verfügung
    ```

Wiederholung

- Verwendung einer Struktur
  - Deklaration einer Struktur (muss initialisiert werden)
    ```
    Strukturname strukturName;
    ```
  - Deklaration und Initialisierung einer leeren Struktur
    ```
    Strukturname strukturName = new Strukturname();
    ```
  - Zuweisung eines Wertes zu einem Attribut der Struktur
    ```
    strukturName.attribut = Ausdruck;
    ```

Beispiel Punkt

- Deklaration einer Struktur „Punkt“
  - ```
    public class Point {
        public double x = 0;
        public double y = 0;
    }
    ```
  - Punkt besteht aus
    - einer x-Koordinate (double)
    - einer y-Koordinate (double)
- Verwendung der neuen Struktur

```
public class Programmname {
    public static void main (String[] args) {
        Point p = new Point();
        p.x = 15;
        p.y = 20;
    }
}
```
```

Klassen

Klasse

- Vorrichtung zur Speicherung und Organisation von Datensätzen
- Organisation (Operationen auf Klasse)
  - Daten hinzufügen, bearbeiten, löschen
  - Daten suchen, sortieren, filtern
  - Je nach Art des Datensatzes
    - Richtungswinkel aus Rechts- und Hochwerten bestimmen
    - Sammelmail an gewisse Personen schreiben
    - Bilder drehen, analysieren, komprimieren, …

Klassen in Java

- Deklaration einer Klasse
  - ```
    public class Klassenname {
        Attribute
        Konstruktoren
        Instanzmethoden
        statische Methoden
    }
    ```
  - Klassenname
    - Möglichst ein Substantiv, groß geschrieben
  - Attribute
    - Eigenschaften der Struktur (Variablen)
  - Konstruktoren
    - Methode, die eine Instanz dieser Klasse erstellt
  - Instanzmethoden
    - Operationen auf dieser Instanz
    ```

Erinnerung

- Statische Methoden
  - Deklaration in separater Klasse
  - ```
    public class Klassenname {
        public static RückgabeDatentyp
                              methodenname(Parameterdeklaration) {
            ...
        }
    }
    ```
  - Aufruf aus dem Hauptprogramm

    ```
    public class Programmname {
        public static void main(String[] args) {
            Klassenname.methodenname(Parameter);
        }
    }
    ```
    ```

Beispiel

- Statische Methoden
  - Deklaration in separater Klasse
  - ```
    public class OutputMethods {
        public static void printAll(int from, int to) {
            for (int nr = from; nr <= to; ++nr) {
                System.out.println(nr);
            }
        }
    }
    ```
  - Aufruf aus dem Hauptprogramm

    ```
    public class OutputTest {
        public static void main(String[] args) {
            OutputMethods.printAll(5,10);
        }
    }
    ```
    ```

Schlüsselwort `static`

- Statische Methoden
  - ```
    public class Klassenname {
        public static RückgabeDatentyp
                              methodenname(Parameterdeklaration) {
            ...
        }
    }
    ```
- Konstanten (Schlüsselwort `final`)
  - ```
    public class Klassenname {
        public static final Datentyp KONSTANTENNAME = Wert;
    }
    ```
- Sind nicht Teil der Instanzen, sondern nur der Klasse!

Instanzmethoden

- Methoden, die auf Objekten operieren
- Deklaration
  - Innerhalb der Klasse, auf dessen Instanzen die Methode operiert
  - Wie andere Methoden, ohne das Schlüsselwort **`static`**

```
public class Klassenname {

    public RückgabeDatentyp methodenname(Parameter) {
        ...
    }

}
```

Instanzmethoden

- Methoden, die auf Objekten operieren
- Aufrufbefehl
  - Variable
  - Punkt
  - Methodenaufruf (wie gehabt)

```
public class Programmname {

    public static void main(String[] args) {

        Klassenname instanz = new Klassenname();

        instanz.methodenname(…);

    }

}
```

Schlüsselwort `this`

- Innerhalb von Instanzmethoden greift man auf die Instanz zu
- Beispiel
- ```
  public class Point {
      public double x = 0;
      public double y = 0;
      public void print() {
          System.out.println("("+ this.x +","+ this.y +")");
      }
  }
  ```
- ```
  public class Programmname {
      public static void main(String[] args) {
          Point p = new Point();
          p.x = 5;
          p.y = 2;
          p.print();
      }
  }
  ```

Schlüsselwort `static`

- Statische Methoden sind unabhängig von Instanzen
- Alternative: Statische Methode
- ```
  public class Point {
      public double x = 0;
      public double y = 0;
      public static void print(Point p) {
          System.out.println("("+ p.x +","+ p.y +")");
      }
  }
  ```
- ```
  public class Programmname {
      public static void main(String[] args) {
          Point p = new Point();
          p.x = 5;
          p.y = 2;
          Point.print(p);
      }
  }
  ```

Konstruktor

- Wozu?
  - Erzeugen einer Instanz (Speicherplatz für das Objekt vorsehen)
  - Initialisierung der Attribute
- Methode ohne Rückgabe-Datentyp
  - Name der Methode wie Klassenname (auch groß)
  - Vorsicht: **`this`** gibt es eigentlich noch nicht, wird ja gerade erstellt!

```
public class Klassenname {

    public Datentyp attribut;

    public Klassenname(Parameter) {
        this.attribut = …;
    }
}
```

Konstruktor

- Beispiel ohne Konstruktor

```
public class Point {
    public double x = 0;
    public double y = 0;
    public void print() {
        System.out.println("("+ this.x +","+ this.y +")");
    }
}
```

```
public class Programmname {
    public static void main(String[] args) {
        Point p = new Point();
        p.x = 5;
        p.y = 2;
        p.print();
    }
}
```

Konstruktor

- Beispiel mit Konstruktor
- ```
  public class Point {
      public double x = 0;
      public double y = 0;
      public Point(double x, double y) {
          this.x = x;
          this.y = y;
      }
      public void print() {
          System.out.println("("+ this.x +","+ this.y +")");
      }
  }
  ```
- ```
  public class Programmname {
      public static void main(String[] args) {
          Point p = new Point(5,2);
          p.print();
      }
  }
  ```

Default Konstruktor

- Was ist ein Default Konstruktor?
  - Ein Konstruktor ohne Parameter, der nichts tut
- Wann ist er da?
  - Wird kein Konstruktor angegeben, existiert der Default Konstruktor
  - Wird dagegen ein Konstruktor deklariert, existiert er nicht
- Was macht er?
  - Erzeugt eine Instanz ohne den Attributen Werte zuzuweisen und macht auch sonst nichts, etwa so:

```
public class Klassenname {
   public Klassenname() {

   }
}
```

Default Konstruktor

- Das funktioniert nicht
- ```
  public class Point {
      public double x = 0;
      public double y = 0;
      public Point(double x, double y) {
          this.x = x;
          this.y = y;
      }
      public void print() {
          System.out.println("("+ this.x +","+ this.y +")");
      }
  }
  ```
- ```
  public class Programmname {
      public static void main(String[] args) {
          Point p = new Point();
          p.x = 5;
          p.y = 2;
          p.print();
      }
  ```

Sichtbarkeit

Sichtbarkeit

- Eigenschaften einer Klasse vor Nutzer der Klasse verbergen
- Wozu?
  - Voller Zugriff auf alle Attribute eines Objekts ist fehleranfällig
  - Stattdessen Zugriff über Methoden,
    die fehlerhafte Zuweisungen abfangen

Beispiel

- Datums-Klasse (schlecht)
  - ```
    public class Date {
        public int day = 1;
        public int month = 1;
        public int year = 1970;
    }
    ```
- Hauptprogramm
  - ```
    public class Programmname {
        public static void main(String[] args) {
            Date d = new Date();
            d.day = 50;
        }
    }
    ```
- Lösung
  - Attribute immer als **`private`** deklarieren
  - Getter und Setter deklarieren

Getter und Setter

- Getter
  - Methode, die den Wert eines Attributs zurück liefert
- Setter
  - Methode, um den Wert eines Attributs zu setzen (ohne Rückgabe)
- So sieht es aus
  - ```
    public class Klassenname {
        private Datentyp attribut;
        public Datentyp getAttribut() {
            return this.attribut;
        }
        public void setAttribut(Datentyp attribut) {
            this.attribut = attribut;
        }
    }
    ```
    ```

Beispiel

- Datums-Klasse (besser)
  - ```
    public class Date {
        private int day = 1;
        private int month = 1;
        private int year = 1970;
        public void setDay(int day) {
            if (day > 0 && day <= 31)
                this.day = day;
        }
    }
    ```
- Hauptprogramm
  - ```
    public class Programmname {
        public static void main(String[] args) {
            Date d = new Date();
            d.setDay(50);
        }
    }
    ```

Sichtbarkeit

- Methode/Attribut/Konstante nur innerhalb Klasse sichtbar
  - Schlüsselwort **`private`**
- Methode/Attribut/Konstante nur innerhalb Paket sichtbar
  - Ohne Schlüsselwort
  - Muss aus anderen Klassen importiert werden (**`import`**)
    oder durch Voranstellen des Klassennamen (mit Punkt)
- Methode/Attribut/Konstante überall sichtbar
  - Schlüsselwort **`public`**
  - Muss aus anderen Paketen und Klassen importiert werden (**`import`**)
    oder durch Voranstellen von Paket und Klasse (mit Punkt verbinden)

Privates

- Grundsätzlich gilt
  - Alles, was von außen nicht gebraucht wird, sollte privat sein
- Beispiel

```
public class Point {
    private double x = 0;
    private double y = 0;
    public void moveLeft (double amount) {
        x = x - amount;
    }
}
```

  - `Point p = new Point();`
  - Verbotener Zugriff von außen
    `p.x = p.x - 10;`
  - Erlaubter Zugriff von außen
    `p.moveLeft(10);`

Beispiele

Klasse Punkt

```
public class Point {

    // Attribute
    private double x;
    private double y;

    // Konstruktor
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }


}
```

Klasse Punkt

```
public class Point {

    // Attribute
    private double x;
    private double y;

    // Getter
    public double getX() {
        return this.x;
    }
    public double getY() {
        return this.y;
    }


}
```

Klasse Punkt

```
public class Point {

    // Attribute
    private double x;
    private double y;

    // Instanzmethoden (Beispiele)
    public void print() {
        System.out.println("("+ this.x +","+ this.y +")");
    }
    public double distanceTo(Point other) {
        double dx = this.x - other.x;
        double dy = this.y - other.y;
        return Math.sqrt(dx*dx + dy*dy);
    }


}
```

Klasse Kreis

```
public class Circle {

    // Attribute
    private Point center;
    private double radius;

    // Konstruktor
    public Circle(Point p, double r) {
        this.center = p;
        this.radius = r;
    }


}
```

Klasse Kreis

```
public class Circle {

    // Attribute
    private Point center;
    private double radius;

    // Getter
    public Point getCenter() {
        return this.center;
    }
    public double getRadius() {
        return this.radius;
    }


}
```

Klasse Kreis

```
public class Circle {

    // Attribute
    private Point center;
    private double radius;

    // Instanzmethoden (Beispiele)
    public double getArea() {
        return Math.PI * this.radius * this.radius;
    }
    public boolean contains(Point p) {
        return p.distanceTo(this.center) < this.radius;
    }
    public Point[] calcIntersectionPoints(Circle other) {
        …
    }


}
```

Bereits vorhandene Klassen

Klasse

- Vorrichtung zur Speicherung und Organisation von Datensätzen
- Organisation (Operationen auf Klasse)
  - Daten hinzufügen, bearbeiten, löschen
  - Daten suchen, sortieren, filtern
  - Je nach Art des Datensatzes
    - Richtungswinkel aus Rechts- und Hochwerten bestimmen
    - Sammelmail an gewisse Personen schreiben
    - Bilder drehen, analysieren, komprimieren, …

Klasse String

- Vorrichtung zur Speicherung und Organisation von Zeichenketten
- Deklaration einer String-Variablen (Auszug)
  - `String variable = new String();`
  - `String variable = "Initialtext";`
- Instanzmethoden für Strings (Auszug)
  - Anzahl der Zeichen in einem String
    `… = variable.length();`
  - Zeichen ersetzen (alle 'a' durch 'b')
    `… = variable.replace('a','b');`
  - Zeichen an einer gewissen Position ermitteln
    `… = variable.charAt(position);`

Klasse String

- Beispiel Java-Programm
  - Wie viele 'e' kommen in einer vom Benutzer eingegebenen Zeichenkette vor?

```
import java.util.Scanner;
public class StringTest {
   public static void main (String[] args) {
      Scanner scanner = new Scanner(System.in);
      String line = scanner.nextLine();
      int c = 0;
      for (int i=0; i<line.length(); ++i) {
         if (line.charAt(i) == 'e') {
            ++c;
         }
      }
      System.out.println("Eingabe enthält " + c + " e.");
      scanner.close();
   }
}
```

Klasse Scanner

- Vorrichtung zur Speicherung und Organisation von Scannern
- Deklaration einer Scanner-Variablen (Auszug)
  - `Scanner variable = new Scanner(System.in);`
  - `Scanner variable = new Scanner("2 3 4");`
- Instanzmethoden für Scanner (Auszug)
  - Nächsten Integer-Wert aus dem Datenstrom lesen
    `… = variable.nextInt();`
  - Scanner schließen
    `variable.close();`
  - Eine vollständige Zeile aus dem Datenstrom lesen
    `… = variable.nextLine();`

Klasse Scanner

- Beispiel Java-Programm
  - Summiere alle Werte in einer Datei auf

```
import java.io.File;
public class ScannerTest {
   public static void main (String[] args) {
      try {
         Scanner scanner = new Scanner(new File(NAME));
         double sum = 0;
         while (scanner.hasNextDouble()) {
            sum += scanner.nextDouble();
         }
         System.out.println("Summe: " + sum);
         scanner.close();
      }
      catch (FileNotFoundException e) {
         System.out.println("Datei existiert nicht");
      }
   }
}
```

Klasse File

- Vorrichtung zur Speicherung und Organisation von Dateien oder Verzeichnissen auf der Festplatte
- Deklaration einer File-Variablen (Auszug)
  - `File variable = new File("C:");`
- Instanzmethoden für Files (Auszug)
  - Existiert diese Datei bzw. dieses Verzeichnis auf der Festplatte?
    `… = variable.exists();`
  - Datei oder Verzeichnis von der Festplatte löschen
    `… = variable.delete();`
  - File-Array aller Dateien und Verzeichnisse in diesem Verzeichnis
    `… = variable.listFiles();`
  - Zeitpunkt der letzten Bearbeitung der Datei bzw. des Verzeichnisses
    `… = variable.lastModified();`

Klasse File

- Beispiel Java-Programm
  - Entferne alles, was ein 'x' enthält aus dem Ordner 'C:\Program Files'

```
import java.io.File;
public class FileTest {
   public static void main (String[] args) {
      String name = "C:"+File.separator+"Program Files";
      File folder = new File(name);
      File[] files = folder.listFiles();
      for (int i=0; i<files.length; ++i) {
         if (files[i].getName().contains("x")) {
            files[i].delete();
         }
      }
   }
}
```

Weitere Klassen

- Punkt
  - `Point`
- Datum
  - `Date`
- Java Application Program Interface (API)
  - http://docs.oracle.com/javase/8/docs/api/
  - > 1000 Klassen
- Frameworks, Libraries, Tools
  - Externe Bibliotheken
  - > 100 000 Klassen

Datenstrom

Datenstrom: wozu?

- Austausch von Daten
  - Eingabe und Ausgabe auf der Konsole
  - Dateien auf der Festplatte
  - Kommunikation mit einem anderen Gerät
    - An den PC angeschlossen (z.B. Drucker)
    - Im Netzwerk oder Internet (Server, andere PCs, Smartphones)

Datenstrom: wie geht das?

- Verbindung herstellen
  - Scanner oder Datei öffnen
  - Kommunikationskanal mit anderem Gerät sicherstellen
- Daten empfangen
  - Text aus einer Datei lesen
  - Kartenmaterial von einem Web Map Service (WMS) erhalten
  - Audio von einem Mikrofon aufnehmen
- Daten senden
  - Text in eine Datei schreiben
  - Bild zum Ausdrucken an den Drucker senden
- Verbindung beenden

Beispiel: Random Access File

- Verbindung herstellen
  - `RandomAccessFile file = new RandomAccessFile(NAME,TYP);`
  - `NAME` ist der Dateiname
  - `TYP` ist die Zugriffsart (`"r"` für lesend, `"rw"` für lesend+schreibend)
- Daten empfangen
  - `… = file.readXXX();`
  - `… = file.readLine();`
- Daten senden
  - `file.writeXXX(…);`
  - `file.writeBytes(…);`
- Verbindung beenden
  - `file.close();`

Fehlerbehandlung

- Verbindung kann nicht hergestellt werden
  - Datei kann nicht gefunden werden (`FileNotFoundException`)
- Verbindung wurde unterbrochen
  oder der Datenaustausch hat nicht korrekt funktioniert
  - Fehler bei der Ein- oder Ausgabe (`IOException`)
- Verbindung kann nicht beendet werden
  - Datei kann nicht geschlossen werden (`IOException`)
- Try-With-Resources (damit Verbindung beendet wird)
  - ```
    try (RandomAccessFile file = new RandomAccessFile(…)) {
        …
    }
    catch (…) {
        …
    }
    ```
    ```

Softwareentwurf

Softwareentwurf

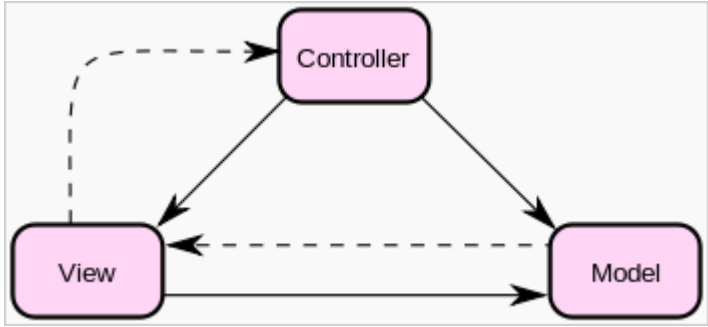
- Bevor man eine Software entwickelt sollte man …
  - abgeklärt haben, was die Software am Ende können soll
    - Lastenheft (Anforderungen des Auftraggebers)
    - Pflichtenheft (Konzeptentwurf des Auftragnehmers)
  - den Aufbau der Software entwerfen
    - Welche Komponenten hat die Software (Architektur)?
    - Welche Programmiersprachen werden verwendet?
    - Wie arbeiten die Entwickler zusammen?
    - …

Softwarearchitektur

- Welche Komponenten arbeiten wie zusammen?
  - Server (Programm im Internet)
  - Client (Programm, das der Benutzer sieht)
  - Datenbank (Programm, das Daten verwaltet)
  - …
- Welche Architekturmuster gibt es?
  - Mud-to-Structure (einfach drauf los)
  - Verteilte Systeme (zum Beispiel Client – Server)
  - Adaptive Systeme (Plug-Ins möglich)
  - Interaktive Systeme (z.B. mit Sensoren oder Benutzern)
    - Model View Controller (MVC)

Model View Controller

- Modell (model)
  - Zustand des Systems
  - Welche Variablen sind wie belegt?
  - Steuert nicht, verwaltet nur



- Präsentation (view)
  - Graphische Ausgabe des Zustands
  - Zeige in Abhängigkeit vom Modell etwas an
  - Greift den Zustand aus dem Modell ab
- Steuerung (controller)
  - Eingabe oder Interaktion
  - Modifiziert das Modell
  - Gibt bei Änderung Aktualisierungsbefehl an Präsentation

Einfache GUI mit Java

```
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Programmname extends JPanel {
   public static final long serialVersionUID = 1;



   public void paint (java.awt.Graphics g) {

   }

   public static void main (String[] args) {
      JFrame frame = new JFrame();
      frame.add(new Programmname());
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);

   }
}
```

Einfache GUI mit Java

```
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Programmname extends JPanel {
   public static final long serialVersionUID = 1;

   public void paint (java.awt.Graphics g) {
   }

   public static void main (String[] args) {
      JFrame frame = new JFrame();
      frame.add(new Programmname());
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

Model

View

Controller

Einfache GUI mit Java

```
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Programmname extends JPanel {
   public static final long serialVersionUID = 1;

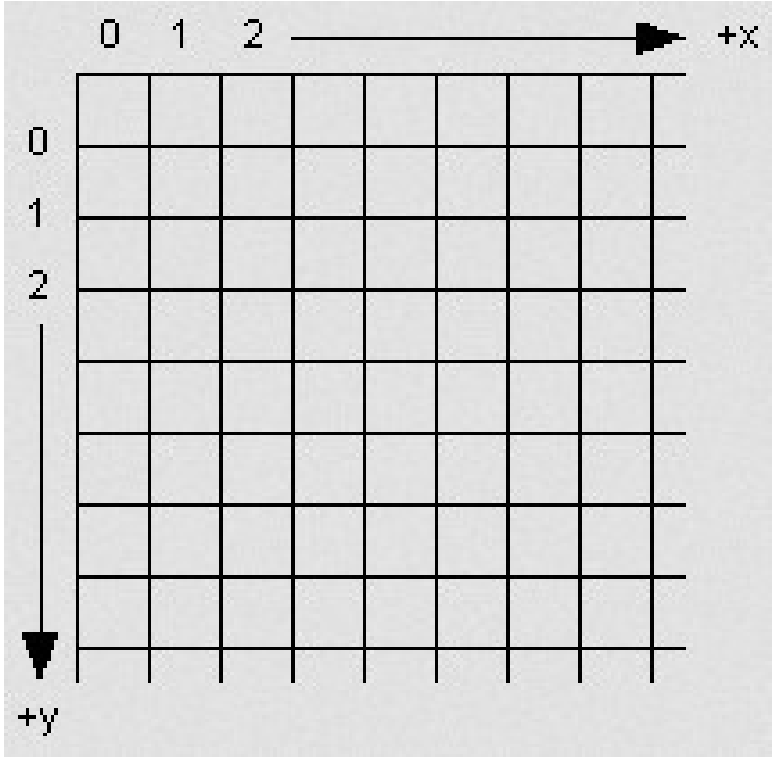
   static int posX = 50;
   static int posY = 50;
   static int radius = 10;

   public void paint (java.awt.Graphics g) {
      g.drawOval(posX,posY,radius,radius);
   }

   public static void main (String[] args) {
      JFrame frame = new JFrame();
      frame.add(new Programmname());
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
      frame.repaint();
   }
```

Ein paar Infos zum Einstieg

- Model
  - Beliebige Variablen mit `static` vorangesetzt
  - Sollten von der Präsentation nur aufgerufen werden
  - Sollten von der Steuerung verändert werden
- View
  - Das Koordinatensystem



Ein paar Methoden zum Einstieg

- View
  - Linie zeichnen

    ```
    g.drawLine(int xStart, int yStart, int xEnd, int yEnd)
    ```
  - Rechteck zeichnen

    ```
    g.drawRect(int x, int y, int width, int height)
    ```
  - Oval zeichnen

    ```
    g.drawOval(int x, int y, int width, int height)
    ```
  - Text zeichnen

    ```
    g.drawString(String text, int x, int y)
    ```
  - Gefülltes Rechteck zeichnen

    ```
    g.fillRect(int x, int y, int width, int height)
    ```
  - Farbe ändern auf rot

    ```
    g.setColor(Color.RED);
    ```

Ein paar Methoden zum Einstieg

- Controller
  - Fenster aktualisieren (View wird neu gezeichnet)
    `frame.repaint()`
  - Größe des Fensters setzen (Zeichenfläche + Rand)
    `frame.setSize(int width, int height)`
  - Position des Fensters auf dem Monitor setzen
    `frame.setLocation(int x, int y)`
  - Benutzereingabe mit `scanner` wie gehabt

Spezifikation

Unified Modeling Language (UML)

- Formalisierte Diagrammbeschreibung zur Darstellung von
  - Klassen
    - Attribute (Eigenschaften einer Klasse)
    - Instanzmethoden (Operationen auf Klasse)
  - Aktionen
    - Eingabe
    - Verarbeitung
    - Ausgabe
  - Aktivitäten
    - Kontrollfluss (siehe Flussdiagramm)
    - Objektfluss (siehe Datenströme)
    - Aktion (siehe Methoden)
  - …

UML-Klassendiagramm

- Pro Klasse ein großes Kästchen mit: (Auszug)
  - Klassenname
  - Attribute
    - Sichtbarkeit
    - Name : Datentyp
  - Operationen
    - Sichtbarkeit
    - Methodenname
    - (Parameter) : Rückgabe
- Sichtbarkeit
  - + public
  - # protected
  - - private
  - ~ package
- Parameter
  - Name : Datentyp

JavaDoc

- Software-Dokumentationswerkzeug
  - Damit kann aus Java-Quellcode mit Kommentaren eine html-Seite mit Java-API automatisch generiert werden
- JavaDoc steht als Kommentar (`/** */`) vor
  - Klassen (in der Zeile vor `class`)
  - Methoden (in der Zeile vor dem Methodenkopf)
  - Attribute und Konstanten außerhalb von Methoden (vor Deklaration)
  - Nur mit Schlüsselwort `public` sinnvoll
- Wie man JavaDoc schreibt
  - Mit Tags (beginnen mit `@`) beschreibt man spezielle Eigenschaften
  - Der restliche Text ohne Tags ist eine allgemeine Beschreibung

Die wichtigsten Tags und Parameter

- Klasse
  - Der Autor einer Klasse (eher unüblich)
  - `@author NAME`
  - Welche JDK-Version wird zum Kompilieren benötigt
  - `@since VERSION`
- Methoden
  - Parameter einer Methode inklusive Beschreibung der Parameter
  - `@param NAME DESCRIPTION`
  - Beschreibung des Rückgabewertes einer Methode
  - `@return DESCRIPTION`

Beispiel JavaDoc

```
/**
* This program prints the number 42, that's it.
* @author Martin Unold
*/
public class Example {
    /**
    * This method returns the number 42 as int in any case.
    * @return The number 42
    */
    public static int methode() {
        return 42;
    }
    /**
    * The program starts with this method.
    * @param args The command line arguments
    */
    public static void main(String[] args) {
        System.out.println(methode());
    }
}
```

Debugger

Debugger

- Werkzeug, das beim Diagnostizieren und Auffinden semantischer Fehler in Computersystemen unterstützt

Debugger

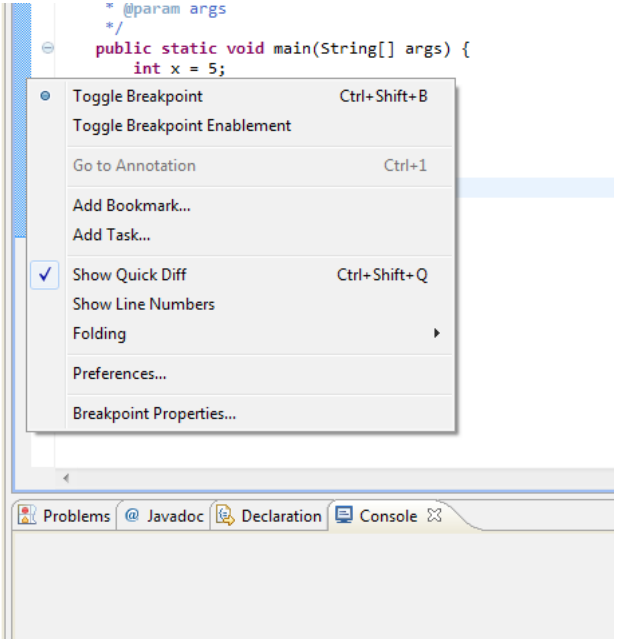
- Werkzeug, das beim Diagnostizieren und Auffinden semantischer Fehler in Computersystemen unterstützt
- Die wichtigsten Funktionen
  - Steuerung des Programmablaufs
  - Inspizieren und Modifizieren von Variablen

Debugger

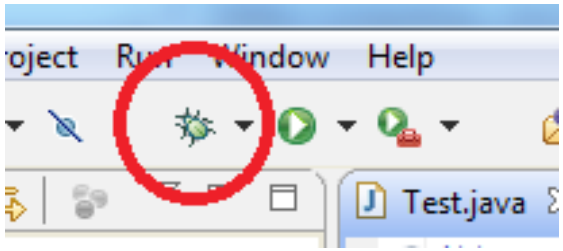
- Werkzeug, das beim Diagnostizieren und Auffinden semantischer Fehler in Computersystemen unterstützt
- Die wichtigsten Funktionen
  - Steuerung des Programmablaufs
  - Inspizieren und Modifizieren von Variablen
- In Eclipse
  - Haltepunkte (Breakpoints) setzen
  - Programm im „Debug-Modus“ starten
  - In die „Debug-Perspektive“ wechseln

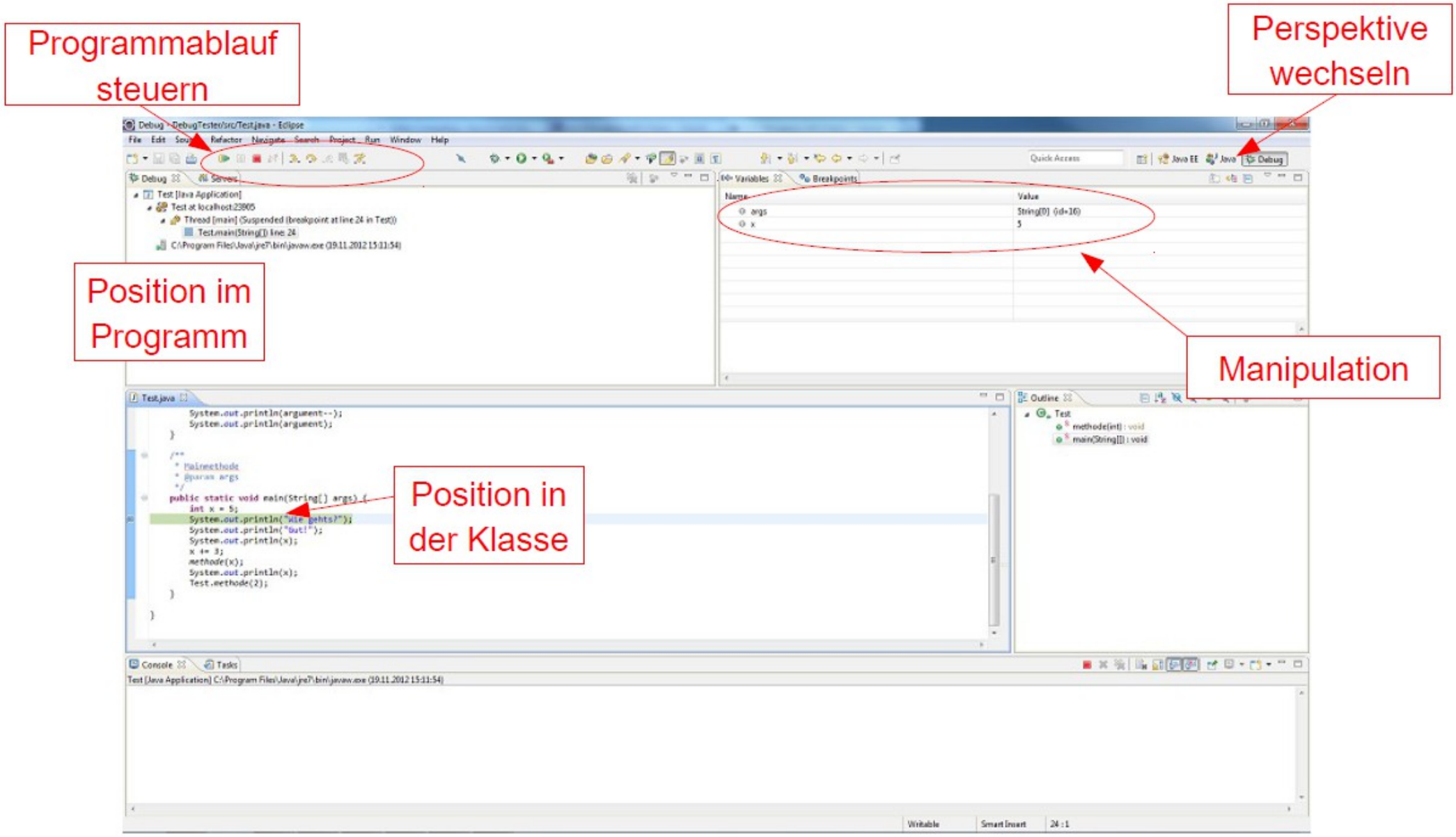
Debugger

- Haltepunkte setzen
  - Doppelklick an den linken Rand oder
  - Rechtsklick → Toggle Breakpoint
  - Genauso auch wieder entfernen



- Programm im Debug-Modus starten
  - F11 oder




Programmablauf steuern
Perspektive wechseln
Position im Programm
Manipulation
Position in der Klasse

Debugger

- Programmablauf steuern
  - Stop Beendet die Anwendung
  - F8 Gehe weiter bis zum nächsten Haltepunkt (normale Ausführung)
  - F7 Gehe bis zum Ende der Methode (zurück zum Aufruf)
  - F6 Führe eine Zeile des vorliegenden Codes komplett aus
  - F5 Führe eine Elementaranweisung aus (springt in Methoden)

Debugger

Ende

Zusammenfassung

Was ist eine Programmiersprache?

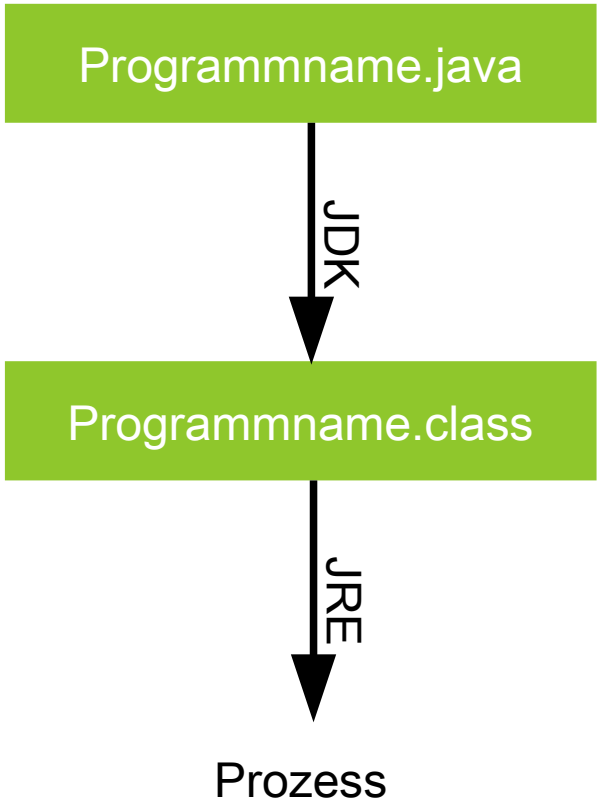
- Eine Sprache, die
  - Formal eindeutig in Maschinenbefehle übersetzbar ist
  - Für Menschen einfacher verständlich ist als Bytecode
- Zur Formulierung von
  - Datenstrukturen
    - Wie sollen Informationen gespeichert werden?
  - Algorithmen
    - Welche Anweisungen bzw. Rechenvorschriften sollen wann ausgeführt werden?

Was ist ein Compiler?

- Ein Programm, das
  - In einer Programmiersprache geschriebenen Text (Quellcode) in ein maschinenlesbares Format (Bytecode) umwandelt

Wie funktioniert das bei Java?

- Kompilieren: „javac Programmname.java“ (im JDK enthalten)
- Ausführen: „java Programmname“ (mit virtueller Maschine)
- Quellcode
  - In Programmiersprache geschrieben
  - Als .java-Datei auf der Festplatte
- Zwischencode
  - In Maschinennaher Sprache
  - Als .class-Datei auf der Festplatte
- Maschinencode
  - In Maschinensprache
  - Prozess wird ausgeführt



Wie sehen Java-Programme aus?

- Textdatei (Endung .java) auf der Festplatte
- Inhalt:

```
public class Programmname {

    public static void main (String[] args) {

        Anweisungen

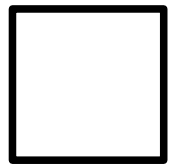
    }

}
```

Was ist eine Variable?

- Deklaration in Java

```
Datentyp name;
```

- Komponenten einer Variablen
  - Bezeichner (`name`)
  - Adresse (Zugewiesener Ort im Arbeitsspeicher: )
  - Wert
    - Was steht an der Speicherposition der Variablen?
    - Welche Bits sind 0 und welche sind 1?
  - Datentyp
    - Welche Art von Werten kann die Variable annehmen?
    - Wie sind die Bits an der Speicherposition zu interpretieren?

Wie greift man auf eine Variable zu?

- Schreibender Zugriff in Java
  - Bezeichner steht links vom Zuweisungs-Operator (=)

    ```
    name = wert;
    ```

- Lesender Zugriff in Java
  - Wert der Variablen wird als Parameter an eine Methode übergeben

    ```
    methode(…,name,…);
    ```
  - Bezeichner steht rechts vom Zuweisungs-Operator ()

    ```
    … = … name … ;
    ```
- Lesender und schreibender Zugriff in Java
  - Bezeichner steht sowohl links als auch rechts vom Zuweisungs-Operator oder bei Abkürzungen, z. B.

    ```
    ++name;
    ```

Welche Datentypen gibt es in Java?

- Primitive Datentypen
  - Boolscher Wahrheitswert
    - boolean
  - Unicode-Zeichen (UTF-16)
    - char
  - Ganzzahlen
    - byte, short, int, long
  - Gleitkommazahlen
    - float, double
- Klassen
- Arrays

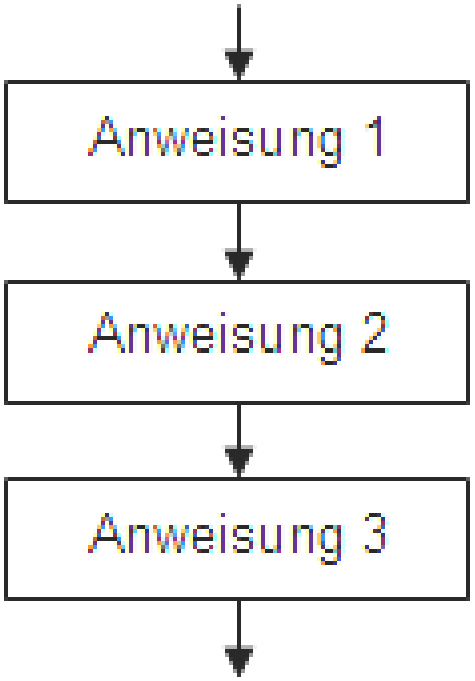
Wie erzeugt man Exemplare?

- Primitive Datentypen (durch Literale)
  - Boolscher Wahrheitswert durch die reservierten Wörter false bzw. true
    - `false true`
  - Unicode-Zeichen (UTF-16) durch einfache Anführungszeichen
    - `'…'`
  - Ganzzahlen (standardmäßig int) oder durch nachgestelltes l für long
    - `… …L`
  - Gleitkommazahlen mit Punkt (double), durch f (float) bzw. d (double)
    - `…F …D`
- Klassen durch Aufruf eines Konstruktors
  - `new Klassenname(…)`
- Arrays durch Angabe der Elemente in {} / als Default-Array
  - `new Datentyp[] { … , … , … } / new Datentyp[LENGTH]`

Was ist eine Sequenz?

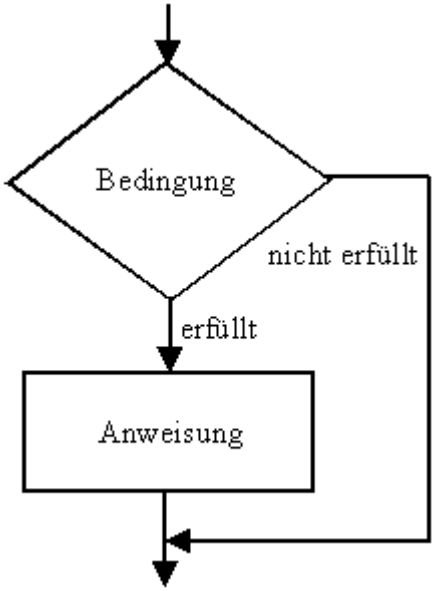
- Eine Anweisung wird nach der anderen ausgeführt
  - Anweisungen mit Semikolon beenden

```
…
Anweisung1;
Anweisung2;
Anweisung3;
…
```



Was ist eine Selektion?

```
if (...) {
   ...
}
```

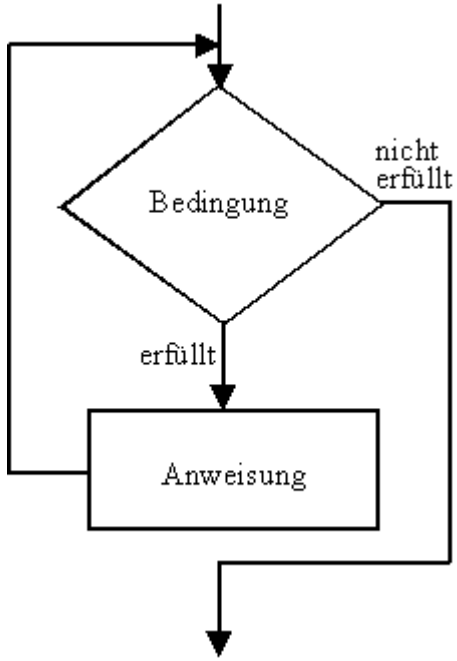


Welche Arten der Selektion gibt es in Java noch?

- If-Verzweigung
  - Anweisungen werden nur ausgeführt,
    wenn der Ausdruck in den runden Klammern `true` ergibt
- If-Else-Verzweigung
  - Wenn der Ausdruck in den runden Klammern `true` ergibt,
    werden die Anweisungen im if-Teil ausgeführt
  - Wenn der Ausdruck in den runden Klammern `false` ergibt,
    werden die Anweisungen im else-Teil ausgeführt
- Switch-Anweisung
  - Je nach Wert der Variablen in den runden Klammern,
    wird zur entsprechenden `case`-Stelle gesprungen

Was ist eine Iteration?

```
while (…) {
   …
}
```



Welche Arten der Iteration gibt es in Java noch?

- While-Schleife
  - Anweisungen werden ausgeführt, solange der Ausdruck in den runden Klammern (zu Beginn jedes Durchlaufes) `true` ergibt
- Do-While-Schleife
  - Anweisungen werden auf jeden Fall einmal ausgeführt
  - Danach wie While-Schleife
- For-Schleife
  - Wie While-Schleife
  - Zusätzlich kann eine Anweisung angegeben werden, die zu Beginn der Schleife einmal ausgeführt wird
  - Zusätzlich kann eine Anweisung angegeben werden, die am Ende jedes Durchlaufes ausgeführt wird

Wie ruft man in Java Methoden auf?

- Ohne Parameter, ohne Rückgabewert
  - `scanner.close();`
  - `System.out.println();`
- Mit Parameter(n), ohne Rückgabewert
  - `Thread.sleep(1000);`
  - `System.out.println(123);`
- Ohne Parameter, mit Rückgabewert
  - `int result = scanner.nextInt();`
  - `double result = Math.random();`
- Mit Parameter(n), mit Rückgabewert
  - `double result = Math.max(2.5,-2.5);`
  - `String result = "Hello World".replace('l','m');`

Wie schreibt man in Java statische Methoden?

- Deklaration einer statischen Methode
  - `public static RückgabeDatentyp name (Parameterliste) {`
    `Anweisungen`
    `}`
  - Rückgabe-Datentyp
    - Beliebiger Datentyp oder `void`, falls keine Rückgabe erfolgt
  - Name
    - Klein Schreiben (wie bei Variablen)
  - Parameterliste
    - Datentyp parameterName
    - Mit Komma getrennt (falls mehrere Parameter verwendet werden)
  - Anweisungen
    - Beliebige Anweisungen (siehe Kontrollstrukturen)
    - Letzte Anweisung muss `return`-Statement sein

Wie nutzt man in Java Arrays?

- Deklaration und Wertzuweisung

  - Deklaration eines Arrays (muss initialisiert werden)

```
Datentyp[] arrayName;
```

  - Deklaration und Initialisierung eines Arrays mit Werten

```
Datentyp[] arrayName = {wert0, wert1, … };
```

    Deklaration und Initialisierung eines „leeren“ Arrays der Länge N

```
Datentyp[] arrayName = new Datentyp[N];
```

  - Zuweisung eines Wertes zu dem Array-Eintrag an einer Position

```
arrayName[position] = Ausdruck;
```

Was sind die Bestandteile einer Klasse?

- Deklaration einer Klasse
  - ```
    public class Klassenname {
        Attribute
        Konstruktoren
        Instanzmethoden
        statische Methoden
    }
    ```
  - Klassenname
    - Möglichst ein Substantiv, groß geschrieben
  - Attribute
    - Eigenschaften der Struktur (Variablen)
  - Konstruktoren
    - Methode, die eine Instanz dieser Klasse erstellt
  - Instanzmethoden
    - Operationen auf dieser Instanz
    ```

Klausurvorbereitung

Beispielhafte Frage

- Warum wird bei der Ausgabe von String-Variablen nicht (wie bei anderen Objekten) die Speicherposition der Variablen ausgegeben?

```
String s = …;
System.out.println(s);
```

Klausurvorbereitung

- Termin
  - xx. Januar um xx:00 Uhr
- Erlaubtes Hilfsmittel
  - Ein Din-A4 Blatt beidseitig **handbeschrieben**
- Weiterhin mitbringen
  - Studentenausweis
  - Stift
  - Gelassenheit

Klausurvorbereitung

- Java ist auch eine Insel
  - insbesondere Kapitel 1 – 3

- Testat-Aufgaben (Programmieren, Nachdenken, Wissen)

- Übungsaufgaben
  - Ein zusätzliches Übungsblatt auf
    http://unold.net/informatik