
Java™

Wiederholung

- Anweisungen durch Methodenaufrufe
  - Ausgabe auf der Konsole
    ```
    System.out.println(...);
    ```
  - Benutzereingabe einlesen
    ```
    … = scanner.nextDatentyp();
    ```
  - Mathematische Funktionen
    ```
    Math.sqrt(…), Math.sin(…), Math.random(), usw.
    ```

Weitere Methodenaufrufe

- 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');`

Eigene Methoden schreiben

- Deklaration einer 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

Wiederholung

- Anweisungen mit Variablen

  - Deklaration einer Variablen (muss initialisiert werden)
    ```
    Datentyp variablenName;
    ```

  - Deklaration und Initialisierung einer Variablen
    ```
    Datentyp variablenName = Wert;
    ```

  - Zuweisung eines Wertes zu einer Variablen
    ```
    variablenName = Ausdruck;
    ```

Wiederholung

- Anweisungen bei Arrays

  - 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

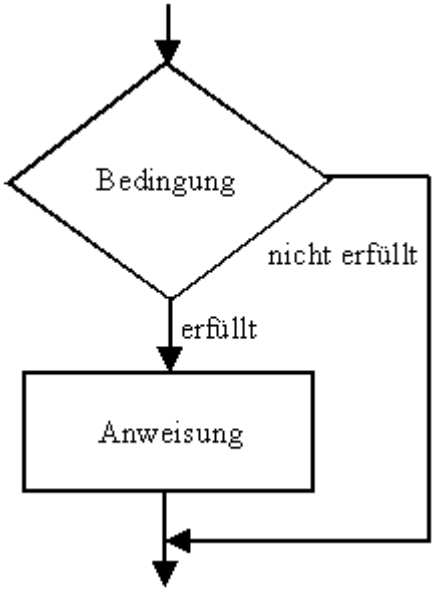
    ```
    arrayName[position] = Ausdruck;
    ```

Kontrollstrukturen

- Sequenz
  - Eine Anweisung wird nach der anderen ausgeführt
- Selektion
  - Anweisung wird in Abhängigkeit einer Bedingung ausgeführt
- Iteration
  - Anweisung wird mehrfach wiederholt ausgeführt

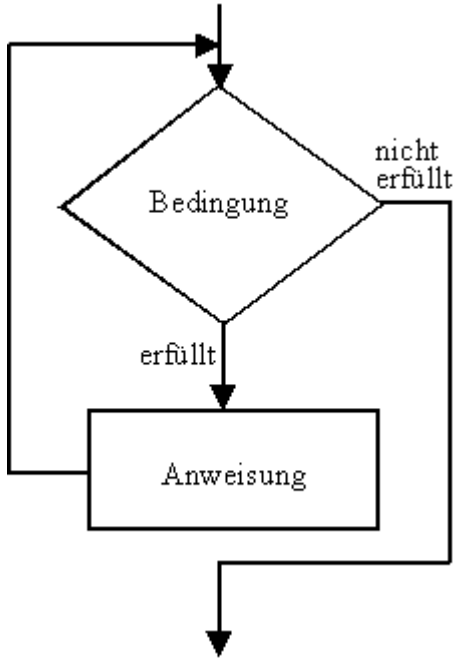
Selektion

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



Iteration

```
while (…) {
   …
}
```



Strukturen

(Daten-)struktur

- Vorrichtung zur Speicherung und Organisation von Datensätzen

(Daten-)struktur

- Vorrichtung zur Speicherung und Organisation von Datensätzen
- Arten von Datensätzen
  - Zahlen, Buchstaben, Worte, Texte
  - Messwerte, Koordinaten, Geometrien
  - Personendaten, Technische Daten
  - Bilder, Videos, Musik
  - …

(Daten-)struktur

- Vorrichtung zur Speicherung und Organisation von Datensätzen
- Organisation (Operationen auf Datensätzen)
  - 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, …

(Daten-)struktur

- Vorrichtung zur Speicherung und Organisation von Datensätzen
- Speicherung
  - Array
    - Beliebige, aber feste Anzahl von Variablen eines Datentyps
  - Struktur
    - Vorher festgelegte und von der Anzahl beschränkte Zusammenstellung von Variablen verschiedener Datentypen
    - Rekursive Strukturen nennt man Datenstrukturen (später)

Strukturen in Java

- 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
    ```

Erinnerung

- Elementare Anweisungen bei primitiven Datentypen

  - Deklaration einer Variablen (muss initialisiert werden)
    ```
    Datentyp variablenName;
    ```

  - Deklaration und Initialisierung einer Variablen
    ```
    Datentyp variablenName = Wert;
    ```

  - Zuweisung eines Wertes zu einer Variablen
    ```
    variablenName = Ausdruck;
    ```

Erinnerung

- Elementare Anweisungen bei Arrays

  - Deklaration eines Arrays (muss initialisiert werden)

    ```
    Datentyp[] variablenName;
    ```

  - Deklaration und Initialisierung eines Arrays mit Werten

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

    Deklaration und Initialisierung eines „leeren“ Arrays der Länge N

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

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

    ```
    variablenName[position] = Ausdruck;
    ```

Neu

- Elementare Anweisungen bei Strukturen

  - Deklaration einer Struktur (muss initialisiert werden)

```
Strukturname variablenName;
```

  - Deklaration und Initialisierung einer leeren Struktur

```
Strukturname variablenName = new Strukturname();
```

  - Zuweisung eines Wertes zu einem Attribut der Struktur

```
variablenName.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;
    }
}
```
```

Beispiel Person

- Deklaration einer Struktur „Person“
  - ```
    public class Person {
        public String name;
        public int age;
    }
    ```
  - Person besteht aus
    - einem Namen (String)
    - einem Alter (int)
- Verwendung der neuen Struktur

```
public class Programmname {
    public static void main (String[] args) {
        Person p = new Person();
        p.name = "Max Mustermann";
        p.age = 66;
    }
}
```
```

Beispiel Zeitangaben

```
public class Time {
    public int hour;
    public int minute;
    public int second;
}

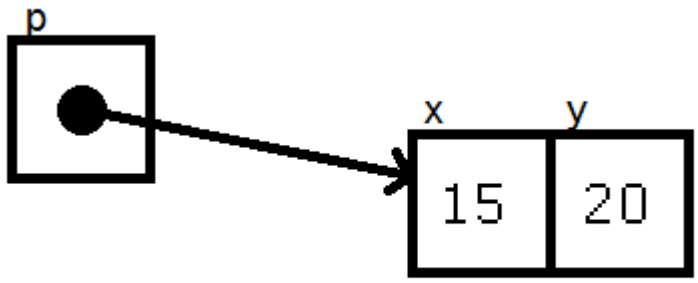
public class Timespan {
    public Time begin;
    public Time end;
}

public class Programmname {
    public static void main (String[] args) {
        Timespan informatik = new Timespan();
        informatik.begin = new Time();
        informatik.begin.hour = 8;
        informatik.begin.minute = 15;
        informatik.begin.second = 0;
        informatik.end = new Time();
    }
}
```

Situation im Speicher

```
public class Point {
    public double x = 0;
    public double y = 0;
}

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



Übung

- ```
  public class Point {
      public double x = 0;
      public double y = 0;
  }
  ```
- ```
  public class Test {
      public static void main (String[] args) {
          Point p = new Point();
          p.x = 1;
          p.y = 2;
          Point q = new Point();
          q.x = p.y;
          q.y = p.x;

          p.x *= 2;

          double dx = p.x-q.x;
          double dy = p.y-q.y;
          System.out.println(Math.sqrt(dx*dx+dy*dy));
      }
  }
  ```

Struktur / Klasse

- ```
  public class Point {
      public double x = 0;
      public double y = 0;
  }
  ```
- ```
  public class Test {
      public static void main (String[] args) {
          Point p = new Point();
          p.x = 1;
          p.y = 2;
          Point q = new Point();
          q.x = p.y;
          q.y = p.x;

          p.x *= 2;

          double dx = p.x-q.x;
          double dy = p.y-q.y;
          System.out.println(Math.sqrt(dx*dx+dy*dy));
      }
  }
  ```

Attribute der Klasse Point

- ```
  public class Point {
      public double x = 0;
      public double y = 0;
  }
  ```
- ```
  public class Test {
      public static void main (String[] args) {
          Point p = new Point();
          p.x = 1;
          p.y = 2;
          Point q = new Point();
          q.x = p.y;
          q.y = p.x;

          p.x *= 2;

          double dx = p.x-q.x;
          double dy = p.y-q.y;
          System.out.println(Math.sqrt(dx*dx+dy*dy));
      }
  }
  ```

Instanz / Objekt

```
public class Point {
    public double x = 0;
    public double y = 0;
}
```

```
public class Test {
    public static void main (String[] args) {
        Point p = new Point();
        p.x = 1;
        p.y = 2;
        Point q = new Point();
        q.x = p.y;
        q.y = p.x;

        p.x *= 2;

        double dx = p.x-q.x;
        double dy = p.y-q.y;
        System.out.println(Math.sqrt(dx*dx+dy*dy));
    }
}
```

Zuweisung von Objekten

- Flache Kopie

```
Point p = new Point();
p.x = 1;
p.y = 2;

Point q = p;
q.x = 10;

System.out.println(p.x); // 10
```

Referenzen vergleichen

- Das ergibt false

```
Point p = new Point();
p.x = 1;
p.y = 2;
Point q = new Point();
q.x = 1;
q.y = 2;
System.out.println(p == q);
```

- Das ergibt true

```
Point p = new Point();
p.x = 1;
p.y = 2;
Point q = p;
System.out.println(p == q);
```

Null-Referenz

- Schlüsselwort **`null`**

- Bezeichnet eine unbesetzte Referenz
- Es wird auf kein Objekt verwiesen

- NullpointerException
  - Tritt auf, wenn man auf Attribute einer Null-Referenz zugreifen will

```
Point p = null;
p.x = …;
```

Methoden mit Klassen

- Call by Reference
-

```
public class MethodenMitKlassen {
    public static double distance (Point a, Point b) {
        double dx = a.x-b.x;
        double dy = a.y-b.y;
        return Math.sqrt(dx*dx+dy*dy);
    }
    public static void main (String[] args) {
        Point p = new Point();
        p.x = 1;
        p.y = 2;
        Point q = new Point();
        q.x = p.y;
        q.y = p.x;

        System.out.println(distance(p,q));
    }
}
```

Variable ausgeben

```
Point p = new Point();

System.out.println(p); // Speicherposition (hexadezimal)
```

Punkt ausgeben

```
Point p = new Point();

System.out.println( "(" +p.x+ "," +p.y+ ")" );
```

Klasse oder Array ?

- Klasse
  - Variablen haben Namen
  - Verschiedene Datentypen
- Point
  - Deklaration und Initialisierung

```
Point p
      = new Point();
```

  - Zugriff auf Attribute

```
p.x
p.y
```

- Array
  - Beliebige Anzahl
  - Variablen eines Datentyps
- Double-Array der Länge 2
  - Deklaration und Initialisierung

```
double[] a
      = new double[2];
```

  - Zugriff auf Elemente

```
a[0]
a[1]
```

Organisatorisches

- Vorlesung
  - Am 16.12. findet die nächste Vorlesung statt
  - Das ist auch die letzte Vorlesung für dieses Semester
- Übungen
  - Finden durchgehend statt, außer in der Woche zwischen den Jahren (Keine Übungen vom 26.12. bis 29.12.)
- Prüfungsleistung
  - Klausurtermin wird noch bekannt gegeben
- Studienleistung
  - Testat Level 15 vor der Klausur erreichen
  - Python-Aufgabe gelöst
  - Mitarbeit und Anwesenheit in den Übungen

Rekursive Strukturen

Erinnerung

- 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
    ```

Rekursive Strukturen

- Eine Struktur heißt rekursiv, wenn mindestens eines der Attribute als Datentyp die Struktur selbst hat
- Deklaration einer rekursiven Struktur
  - ```
    public class Strukturname {
        public Strukturname attributname;
        … weitere Attribute
    }
    ```
- Dann ist so etwas möglich

```
public class Programmname {
    public static void main (String[] args) {
        Strukturname test = new Strukturname();
        test.attributname.attributname.attributname...
    }
}
```
```

Wozu Rekursive Strukturen?

- Arrays haben den Nachteil, dass die Anzahl der Elemente nicht dynamisch verändert werden kann
- Wie kann man mit Hilfe von Strukturen Listen erstellen?

Rekursive Strukturen können

- Dynamisch Elemente hinzufügen oder entfernen
- Flexibel sortieren oder Daten geeigneter vorbereiten

- Elemente einer Rekursiven Struktur heißen Knoten (Node)

Rekursive Strukturen in Java

- Erstes Beispiel:
  Einfach verkettete Liste mit ganzzahligen Werten

  - ```
    public class Node {

        public int value;
        public Node next;

    }
    ```

  - Attribute
    - Der eigentliche Wert dieses Knotens
    - Verweis auf einen weiteren Knoten
    ```

Rekursive Strukturen in Java

- Erstes Beispiel:
  Einfach verkettete Liste mit ganzzahligen Werten

  - ```
    public class Node {

        public int value = 0;
        public Node next = null;

    }
    ```

  - Attribute
    - Der eigentliche Wert dieses Knotens
    - Verweis auf einen weiteren Knoten
    ```

Verwendung der einfach verketteten Liste

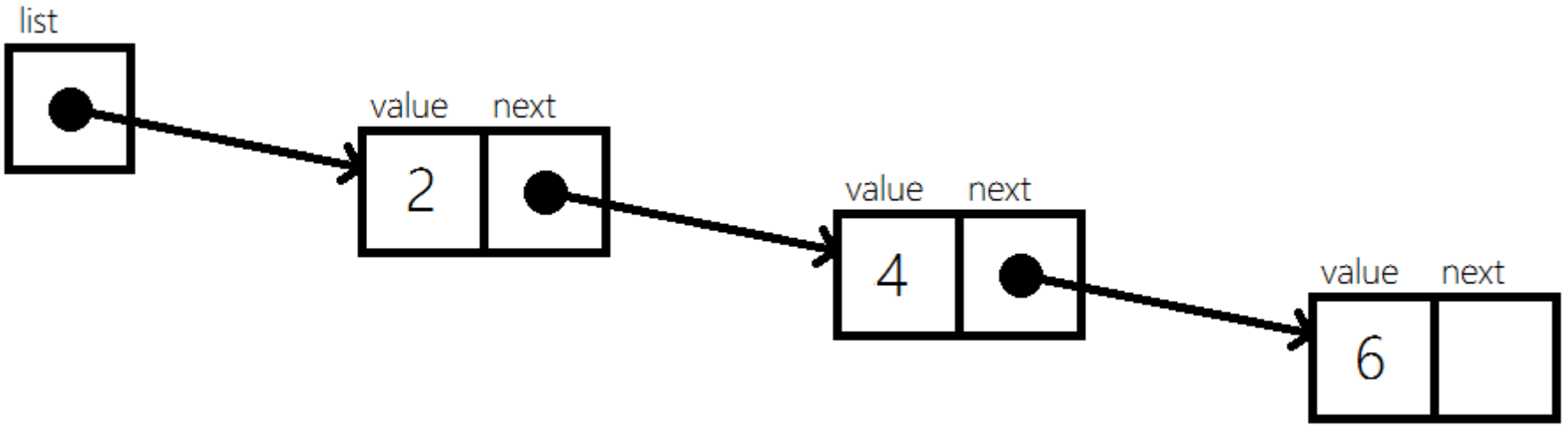
- Array

```
public class ListCompare {
   public static void main (String[] args) {
      int[] list = {2,4,6};
   }
}
```

- Liste

```
public class ListCompare {
   public static void main (String[] args) {
      Node list = new Node();
      list.value = 2;
      list.next = new Node();
      list.next.value = 4;
      list.next.next = new Node();
      list.next.next.value = 6;
   }
}
```

Verwendung der einfach verketteten Liste




```
public class ListCompare {
   public static void main (String[] args) {
      Node list = new Node();
      list.value = 2;
      list.next = new Node();
      list.next.value = 4;
      list.next.next = new Node();
      list.next.next.value = 6;
   }
}
```

Verwendung der einfach verketteten Liste

- Knoten an Liste anhängen (hinten)

```
public class ListCompare {
   public static void main (String[] args) {
      Node node;
      Node head = new Node();
      head.value = 2;

      node = head;
      while (node.next != null)
         node = node.next;
      node.next = new Node();
      node.next.value = 4;

      node = head;
      while (node.next != null)
         node = node.next;
      node.next = new Node();
      node.next.value = 6;
   }
}
```

Verwendung der einfach verketteten Liste

- Knoten an Liste anhängen (vorne)

```
public class ListCompare {
    public static void main (String[] args) {
        Node node;

        Node head = new Node();
        head.value = 6;

        node = new Node();
        node.next = head;
        head = node;
        head.value = 4;

        node = new Node();
        node.next = head;
        head = node;
        head.value = 2;

    }
}
```

Verwendung der einfach verketteten Liste

- Knoten entfernen (nach Position)

```
public class ListCompare {
   public static void main (String[] args) {

      Node head = …;
      int position = …;

      Node node = head;
      for (int i=0; i<position-1; ++i) {
         node = node.next;
      }
      node.next = node.next.next;

   }
}
```

Verwendung der einfach verketteten Liste

- Knoten entfernen (nach Wert)

```
public class ListCompare {
   public static void main (String[] args) {

      Node head = …;
      int value = …;

      Node node = head;
      while (node.next.value != value) {
         node = node.next;
      }
      node.next = node.next.next;

   }
}
```

Verwendung der einfach verketteten Liste

- Methoden mit Listen (Ausgabe auf Konsole)

```
public class ListCompare {

    public static void print (Node node) {
        while (node != null) {
            System.out.println(node.value);
            node = node.next;
        }
    }

    public static void main (String[] args) {

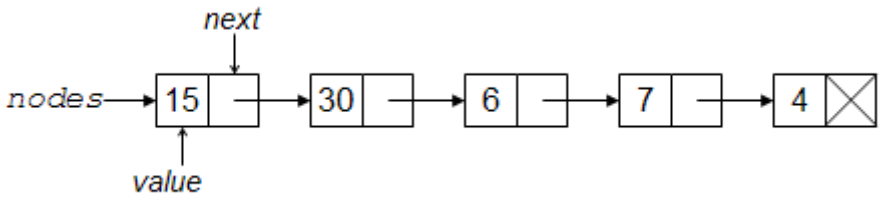
        Node head = …;

        print(head);

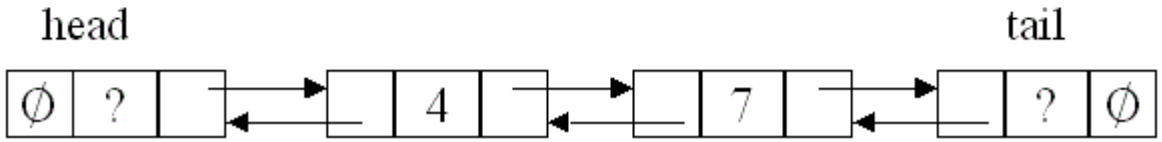
    }
}
```

Datenstrukturen

- Es gibt noch weitere
  - Einfach verkettete Liste



  - Doppelt verkettete Liste



  - Und noch viele mehr...

Ausnahmebehandlung

Ausnahmebehandlung

- Exception (Ausnahme)
  - Tritt zur Laufzeit eines Programms ein
  - Unterbricht den normalen Kontrollfluss
- Handling (Behandlung)
  - Problem führt nicht zum Absturz,
    sondern wird vom Programmierer abgefangen

Beispiele

- Benutzer gibt keine Zahl ein
  oder eine Zahl im falschen Format

  ```
  scanner.nextInt()
  ```

  - InputMismatchException
- Division durch 0

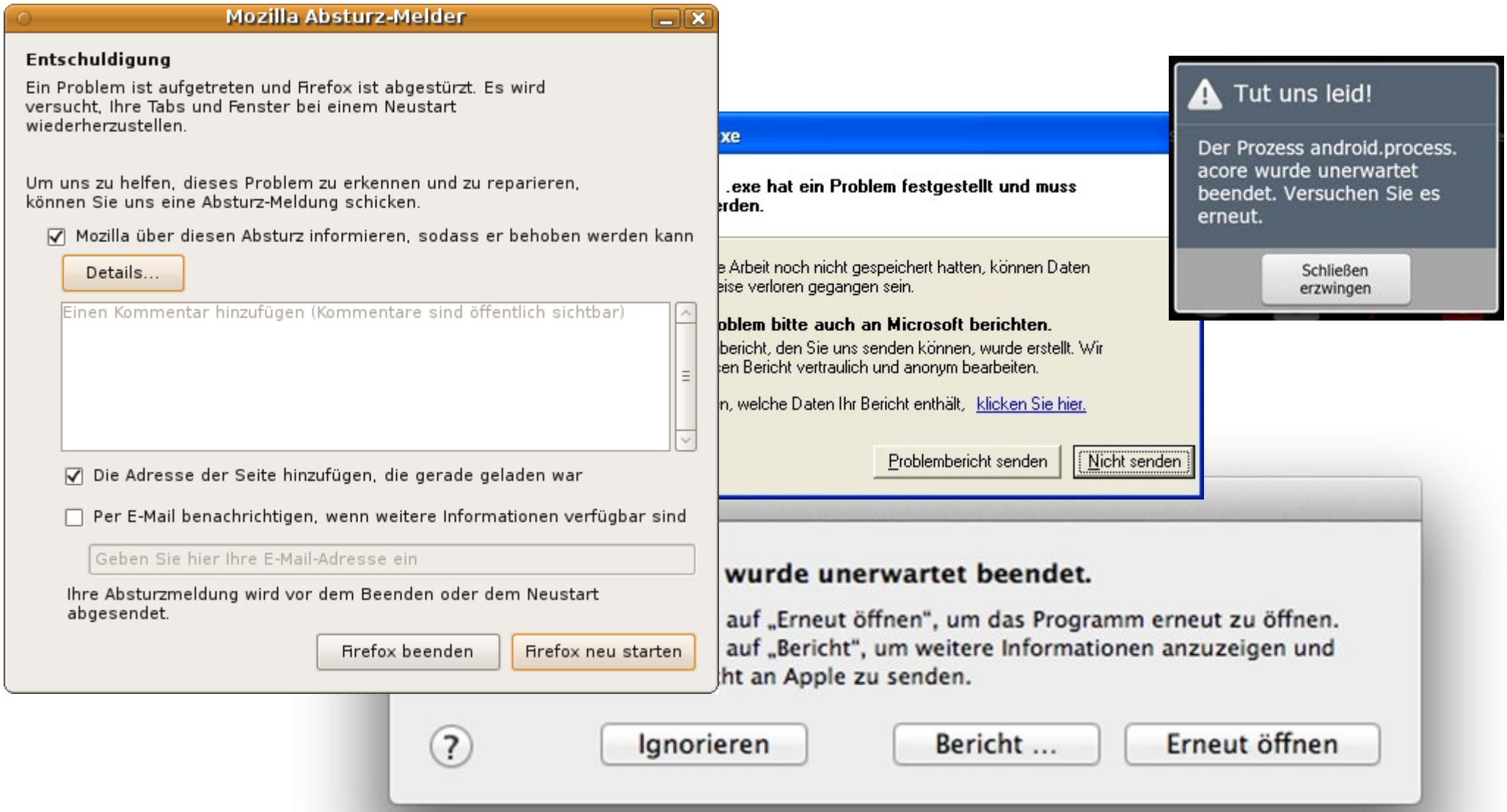
  ```
  1/0
  ```

  - ArithmeticException
- Zugriff auf unzulässigen Index eines Arrays

  ```
  array[-1]
  ```

  - ArrayIndexOutOfBoundsException

Wie machen es die Profis?



Ausnahmen behandeln

- In Java

```
try {

   Anweisungen, bei denen Exceptions auftreten könnten

}
catch (ExceptionType e) {

   Anweisungen, die ausgeführt werden,
   wenn ein Fehler der angegebenen Fehlerart oben auftritt

}
finally {

   Anweisungen, die auf jeden Fall ausgeführt werden

}
```

Ausnahmen behandeln

- Beispiel

```
int x = 42;

try {

   System.out.print("Geben Sie eine ganze Zahl ein: ");
   x = scanner.nextInt();

}
catch (InputMismatchException e) {

   System.out.println("Das war aber keine ganze Zahl");
   System.out.println("Jetzt wird weiter gemacht mit 42");

}
```

Sprachbeschreibung

Token in natürlicher Sprache

- Die Vorlesung am 2. Dezember 2015 findet gerade statt, wobei der Dozent (Martin Unold) etwas zu Token erklärt.
  - Die
  - Vorlesung
  - am
  - 2
  - Dezember
  - 2015
  - findet
  - gerade
  - statt
  - wobei
  - ...

Token in Java

- ```
  public class Programmname {
      public static void main (String[] args) {

      }
  }
  ```
- Token
  - public
  - class
  - Programmname
  - public
  - static
  - void
  - main
  - String
  - args
  ```

Token

- Quellcode ist ein Text
- Text besteht aus Token
- Token werden getrennt durch
  - Leerzeichen
  - Semikolon, Komma, Punkt (nicht bei Kommazahlen)
  - Klammern
- Token sind
  - Literale
  - Bezeichner
  - Reservierte Schlüsselwörter

Literale

- Konstante Ausdrücke,
  um Werte in verschiedenen Datentypen anzugeben
  - boolean (die reservierten Schlüsselworte true und false)

    ```
    true
    false
    ```
  - char (in einfachen Anführungszeichen angegebenes Zeichen)

    ```
    'x'
    '2'
    '\n'
    ```
  - String (in normalen Anführungszeichen angegebener Text)

    ```
    "Das ist ein String"
    "22"
    "\n"
    ```

Literale

- Konstante Ausdrücke,
  um Werte in verschiedenen Datentypen anzugeben
  - (byte)
    `123`
  - (short)
    `123`
  - int (ganze Zahl)
    `123`
  - long (ganze Zahl mit L)
    `123L`
  - float (Kommazahl mit F)
    `123f`
  - long (Kommazahl mit D)
    `123d`

Bezeichner

- Namen von
  - Variablen
  - Konstanten
  - Methoden
  - Paketen
  - Klassen
- Zum Beispiel
  - `Programmname`
  - `main`
  - `java`
  - `lang`
  - `Math`
  - `PI`

Reservierte Schlüsselwörter

- Sind nicht als Bezeichner zulässig
  - Man darf Variablen, Klassen, … nicht wie ein Schlüsselwort nennen
- Werden in Eclipse lila dargestellt
- Sind immer in einem Wort komplett klein geschrieben
- Beispiele
  - `public`
  - `static`
  - `void`
  - `float`
  - `boolean`
  - `new`
  - `if`
  - `while`

Hausaufgaben

3.1 Objektorientierte Programmierung (OOP)

3.2 Eigenschaften einer Klasse

3.4 Neue Objekte erzeugen