Interface = 100 % abstrakte Klasse

↳ Eigenschaften

- Methoden haben keine Implementierung
- Es können keine Objekte eines Interface erzeugt werden
- Es können aber Variablen eines Interface deklariert werden

- Wenn Interface mit nur einer Methode ⇒ Lambas

```
interface Writer {
    void write(char c);
    void open();
    void close();
}
```

```
public class Writer implements Writer {
    public void open() { }
    public void close() { }
    public void write(char c) {
        System.out.println(c);
    }
}
```

Eine Klasse kann ein Interface "implementieren"

- Eine Klasse kann von einer Klasse erben, aber beliebig viele Interfaces implementieren
- Eine Klasse, die ein Interface implementiert, muss alle seine Methoden
  - implementieren (als public!) können auch durch abstrakte Meth implementiert werden
  - oder erben

```
public interface Iterable {
  Iterator getIterator();
}

public interface Iterator {
  boolean hasNext();
  int next();
}

public class List implements Iterable {
  :
  :
  public Iterator getIterator() {
    return new MyIterator();
  }
}

public class MyIterator implements Iterator {
  private Node curr = head;
  public boolean hasNext() {
    return curr != null;
  }
  public int next() {
    int v = curr.value;
    curr = curr.next;
    return v;
  }
}
```

z.B. soll ein Objekt

- vergleichbar sein →
```
interface comparable {
  int compareTo (Object do);
}
```
- ausgebbar sein →
```
interface printable {
  void print();
}
```
- verschlüsselbar sein →
```
interface Encodable {
  String getEncodedValue();
}
```

Diese Interfaces nicht man zur Klasse des Objekt dazu

```
class MyClass implements Comparable, Printable, Encodable {
...
}
```

Somit können MyClass-Objekt wie alle anderen Comparable, Printable und Encodable behandelt werden.

Abstrakte Klassen vs Interfaces

Abstracte Klassen können auch konkrete Methoden und Datenfelder enthalten. Interfaces nicht!

```
public abstract class Stream {

  public String name;
  public abstract void print(char c);
  public void printString(String s) {
    for (int i=0; i<s.length(); i++) {
      print(s.charAt(i));
    }
  }
}
```

Eine Klasse darf nicht von mehreren Klassen erben (auch nicht, wenn diese abstract sind).
Sie darf aber mehrere Interfaces implementieren.

```
public class File extends Stream implements Comparable, Encodable{
}....
```

Um Kompatibilität zu erreichen, sind Interfaces meist besser als abstrakte Klassen.

Vergleichbarkeit: Comparable

Für Klassen deren Objekte vergleichbar sein sollen

```
interface Comparable {
  int compareTo(Object ob);
}
```

a.compareTo(b)
< 0 wenn a < b.
0 wenn a == b
> 0 wenn a > b

```
public class Fraction implements Comparable {
  private final int z;
  private final int n;
  public Fraction(int z, int n) {
    this.z = z;
    this.n = n;
  }
  ...
  public int compareTo(Object o) {
    Fraction f = (Fraction) o;
    return z * f.n - f.z * n;
```

Bsp.:

List implements Iterable

void add (Comparable d);

↳ sortiert einfügen

⇒ sortierte Liste

- Iterable
- Iterator
- Comparale
- Comparater

wichtig!!!