Java Einführung
Abstrakte Klassen und Interfaces

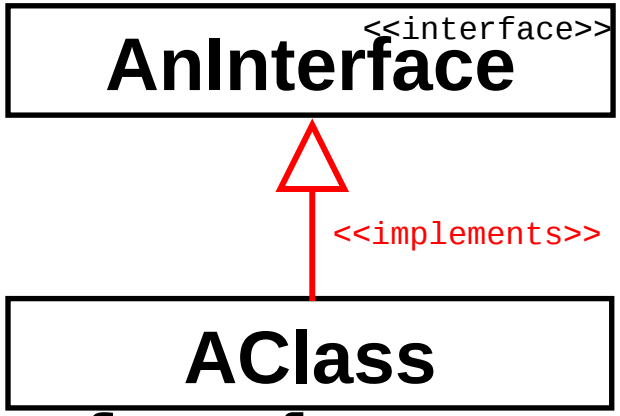
Interface

- **Interface** bieten in Java ist die Möglichkeit, einheitliche **Schnittstelle** für Klassen zu definieren, die
  - später oder/und
  - durch andere Programmierer
- implementiert werden.
- Interfaces können **definiert** und **implementiert** werden.

Definition und Implementierung eines Interfaces

Java-Syntax:

```
interface AnInterface {
...
}
```




```
class AClass implements AnInterface {
...
}
```

Bsp: Interface definieren

```
interface HasName {

  String pre = “My Name is “;
  //Variablen sind autom. static & final!

  String getName();
  //Methoden sind autom. public!
  //keine Implementierung!
}
```

Bsp: Interface implementieren

- Der Compiler prüft ob alle Methoden des Interfaces implementiert wurden.

```
class Person implements HasName {
   privateString myName; // Instanzvariablen

   public String getName() {
     return(pre + myName); // pre kommt vom Interface!
   }

  public void talk(String sentence) {
    System.out.println(myName+ " says: "+sentence);
  }
}
```

Bsp: Interface implementieren II

- Interfaces werden in der Java API gefunden. Z.B.: **`java.lang.Comparable`**

Method Summary int compareTo(Object o)

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

```
class fraction implements Comparable {
   private double n,d;
   public int compareTo(Object o) {
    Fraction oFrac = (Fraction) o; // upcast!
    double me = (double) n/ (double) d;
    double other = (double) oFrac.n / (double) oFrac.d;
    if (me < other) { return -1; }
    if (me > other) { return 1; }
    return 0; }
}
```

Abstrakte Klassen

- Soll genauso wie `Interface/implements` ein **einheitliches Interface** für alle abgeleiteten Klassen definiern. Hier ist es aber möglich einzelne Methoden bereits in der abstrakten Klasse zu implementieren und Instanzvariablen zu deklarieren.

- Wird in einer Klasse für mindestens eine Methode nur die Schnittstelle definiert (abstrakte Methode), muss die gesammte Klasse als abstrakt definiert werden.

Abstrakte Klassen II

- Die Kennzeichnung einer Klasse als abstrakt **verhindert**, dass Instanzen dieser Klasse erzeugt werden können.

- Alle abstrakten Methoden müssen in den nicht abstrakten Subklassen **implementiert** werden.

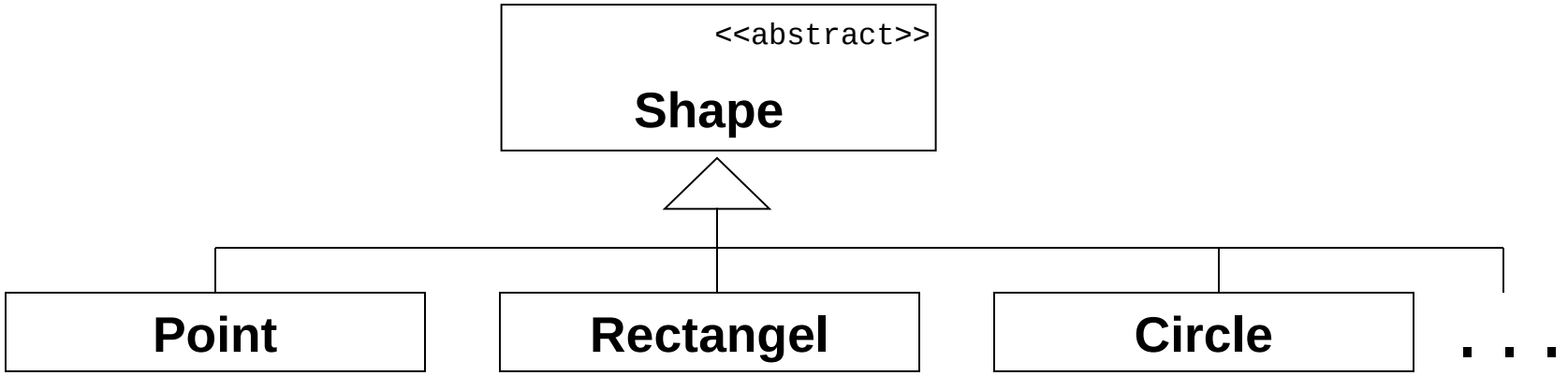
- Java-Syntax der Klassendeklaration:

```
abstract class KlassenName {
   void aNormalMethod(int a) {...}
   abstract void aAbstractMethod(int b);
}
```

Block der Methode fehlt

Abstrakte Klasse Beispiel

- Es sollen Verschiedene Formen implementiert werden, die das gleiche Interface verwenden. Das Interface und gemeinsame Teile werden in Shape definiert/implementiert



Abstrakte Klasse Beispiel II

```
public abstract class Shape {
  protected Point anchor;

  Shape() { this.anchor =new Point(); }

  public Point position() { return anchor; }

//abstract interface
  abstract public double area();
  abstract public double perimeter();
}
```

Abstrakte Klasse Beispiel II

```
public class Rectangle extends Shape {
  protected Point rightLowerCorner; // 2. Ecke

  public Rectangle() {
    super();
    this.rightLowerCorner = new Point();
  }
  // Implementierung der abstrakten Methoden
  public double area() { return (width()*heigth()); }
  public double perimeter() {
     return (2*(width()+heigth()));
  }
  // Zusätzliche Methoden
  private double width() {
    return (Math.abs(rightLowerCorner.x-anchor.x));
  } ...
```

Was sie nach dieser Einheit wissen sollten...

- Verwendung von Interfaces und abstrakten Klassen