UNIVERSITÄT
LEIPZIG

Einführung in die Objekt-Orientierte Modellierung und Programmierung

Wintersemester 2025/2026

Dirk Zeckzer

Institut für Informatik

Teil XV

Modellierung: Interface

Interfaces

- Beschreiben
  - welche Methoden (Name)
  - mit welchen Parametern (Typen, Reihenfolge)
  - und welcher Rückgabe (Typ)

  zur Verfügung gestellt werden
- Abstrakte Methoden ohne Implementierung
- Keine nicht-statischen Attribute

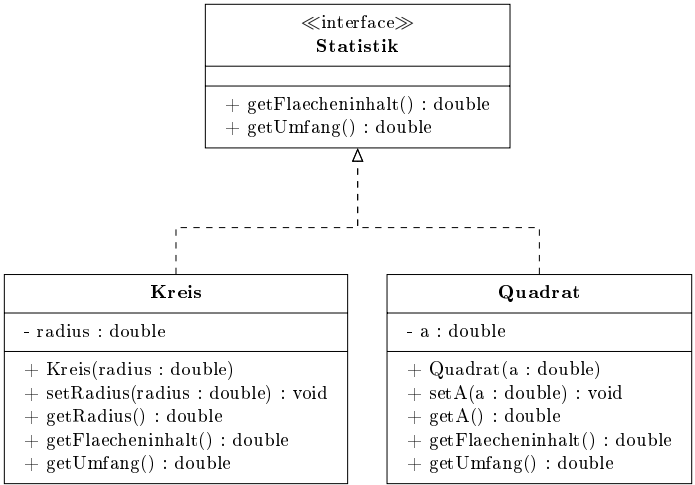
- In neueren Java-Versionen ist das Konzept aufgeweicht worden
- Hauptgrund:
  Kompatibilität zu älteren Java-Versionen innerhalb der Java-Bibliotheken
- Guideline:
  Neuere Funktionalitäten nur dann verwenden, wenn es unabdingbar ist

Interfaces

Kreis
- radius : double
+ Kreis(radius : double) + setRadius(radius : double) : void + getRadius() : double + getFlaecheninhalt() : double + getUmfang() : double

Quadrat
- a : double
+ Quadrat(a : double) + setA(a : double) : void + getA() : double + getFlaecheninhalt() : double + getUmfang() : double

Interfaces


≪interface≫
Statistik
+ getFlaecheninhalt() : double
+ getUmfang() : double
Kreis
- radius : double
+ Kreis(radius : double)
+ setRadius(radius : double) : void
+ getRadius() : double
+ getFlaecheninhalt() : double
+ getUmfang() : double
Quadrat
- a : double
+ Quadrat(a : double)
+ setA(a : double) : void
+ getA() : double
+ getFlaecheninhalt() : double
+ getUmfang() : double

Interfaces

Grammatik Interface:

```
1   public abstract interface Name {
2     ...
3   }
```

- Zulässige Sichtbarkeit:
  - public
  - package
- Implizit immer `abstract`

Interfaces

Grammatik Konstante in einem Interface:

- Implizit immer `public static final`

```
1   public static final int KONSTANTE = 2;
```

Grammatik Methode in einem Interface:

- Implizit immer `public abstract`

```
1   public abstract double getRadius();
```

Interfaces

```
1  package eoomp;
2
3  public interface Statistik {
4
5    public double getFlaecheninhalt();
6
7    public double getUmfang();
8  }
```

Interfaces: Implementierung

```java
package eoomp;

public class Kreis
  implements Statistik {

  private double radius;

  public Kreis(
    final double radius
  ) {
    setRadius(radius);
  }

  ...

  @Override
  public double getFlaecheninhalt() {
    return ...;
  }

  @Override
  public double getUmfang() {
    return ...;
  }
}
```

```java
package eoomp;

public class Quadrat
  implements Statistik {

  private double a;

  public Quadrat (
    final double a
  ) {
    setA(a);
  }

  ...

  @Override
  public double getFlaecheninhalt() {
    return ...;
  }

  @Override
  public double getUmfang() {
    return ...;
  }
}
```

Interfaces: Implementierung

Annotation `@Override`

- Besagt, dass diese Methode bereits in einem Interface definiert wurde
- Optional
- Wichtig als Schutz gegen Fehler
  - Stimmt die Deklaration der Methode nicht mit der Deklaration in einem Interface überein
    $\rightarrow$ Fehlermeldung des Compilers

Interface $\leftrightarrow$ Klasse

Interface wird als **Datentyp** verwendet

- Typ eines Attributes
- Typ einer Variablen
- Typ eines Parameters
- Typ des Rückgabewertes

Klasse (Implementierung des Interfaces) wird zur **Instantiierung** verwendet

Interfaces: instanceof

instanceof

```
1  Statistik statistik = new Kreis(1.0);
2
3  boolean isKreis = statistik instanceof Kreis;
4  // Ergebnis: true
5
6  boolean isQuadrat = statistik instanceof Quadrat;
7  // Ergebnis: false
```

Interfaces: instanceof

instanceof

```
1   if (statistik instanceof Kreis) {
2     Kreis kreis = (Kreis) statistik;
3     System.out.println("Radius: " + kreis.getRadius());
4   } else if (statistik instanceof Quadrat) {
5     Quadrat quadrat = (Quadrat) position;
6     System.out.println("Seitenlaenge: " + quadrat.getA());
7   } else {
8     ...
9   }
```

```
1   if (statistik instanceof Kreis kreis) {
2     System.out.println("Radius: " + kreis.getRadius());
3   } else if (statistik instanceof Quadrat quadrat) {
4     System.out.println("Seitenlaenge: " + quadrat.getA());
5   } else {
6     ...
7   }
```

Interfaces: Implementierung

```java
package eoomp;

public class BeispielKreise {

  public static void main (
    final String[] args
  ) {
    Kreis[] elemente = new Kreis[2];
    elemente[0] = new Kreis(7.0);
    elemente[1] = new Kreis(5.0);
    System.out.println(
      getSummeDerUmfaenge(elemente)
    );
  }

  public static double getSummeDerUmfaenge(
    final Kreis[] elemente
  ) {
    double result = 0.0;

    for (Kreis element : elemente) {
      result += element.getUmfang();
    }

    return result;
  }
}
```

```java
package eoomp;

public class BeispielQuadrate {

  public static void main (
    final String[] args
  ) {
    Quadrat[] elemente = new Quadrat[2];
    elemente[0] = new Quadrat(7.0);
    elemente[1] = new Quadrat(5.0);
    System.out.println(
      getSummeDerUmfaenge(elemente)
    );
  }

  public static double getSummeDerUmfaenge(
    final Quadrat[] elemente
  ) {
    double result = 0.0;

    for (Quadrat element : elemente) {
      result += element.getUmfang();
    }

    return result;
  }
}
```

Interfaces: Implementierung

```java
package eoomp;

public class BeispielInterface {

  public static void main(
    final String[] args
  ) {
    Statistik[] elemente = new Statistik[2];
    elemente[0] = new Kreis(7.0);
    elemente[1] = new Quadrat(7.0);
    System.out.println(
      getSummeDerUmfaenge(elemente)
    );
  }

  public static double getSummeDerUmfaenge(
    final Statistik[] elemente
  ) {
    double result = 0.0;

    for (Statistik element : elemente) {
      result += element.getUmfang();
    }

    return result;
  }
}
```

Interfaces und Erweiterungen

	Erweiterung	Interface
Darstellung in UML	durchgezogene Linie	gestrichelte Linie
Anzahl	1	$n \in \mathbb{N}$

	Basisklasse	Interface
Modifizierer Attribute	beliebig	nur `static final`

	Basisklasse	Interface
Sichtbarkeit Klasse	alle	nur `public` und package
Sichtbarkeit Methoden	alle	nur `public`
Sichtbarkeit Attribute	alle	nur `public`

- Interfaces können andere Interfaces erweitern

Interfaces und Erweiterungen

	Basisklasse	Interface
Konstanten	Ja	Ja
Attribute (ohne Konstanten)	Ja	Nein
Methoden-Deklaration	Ja	Ja
Methoden-Definition	Ja	Nein
Erweiterung/-Implementierung	Eine	Mehrere