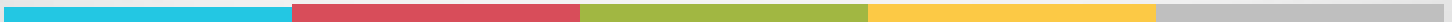
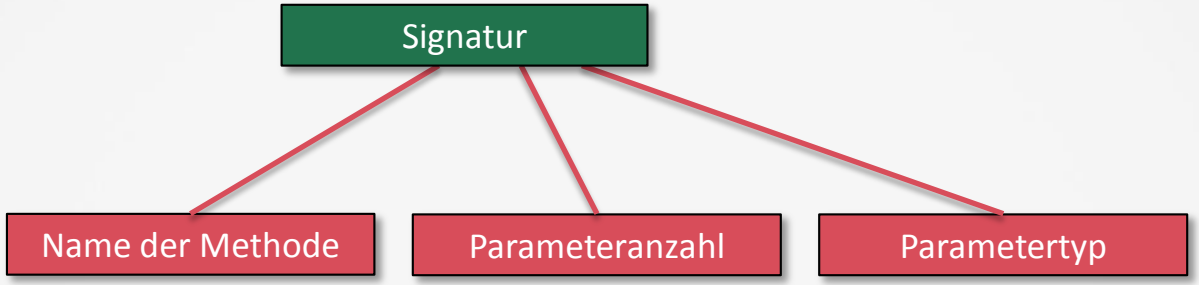
OOP - Polymorphismus



Signatur einer Methode



Überladen von Methoden – Beispiel - Parameteranzahl

```
class Animal
{
  public int noOfLegs(){.....}
  public int noOfLegs(int x){.....}
  public int noOfLegs(double x){.....}
  public double noOfLegs(String x){.....}
  public double noOfLegs(int x, int y){.....}
  public double noOfLegs(){.....}

  public double noOfLegs(int x){.....}
  //public double noOfLegs(int a, int b){.....}
}
```

Erlaubt: Unterschiedliche Parameteranzahl

Erlaubt: Unterschiedliche Parametertyp

Erlaubt: Unterschiedlicher Rückgabetyp

Erlaubt: Unterschiedl. Zugriffsmodifizierer

Verboten: Signatur vorhanden aber anderer Rückgabetyp

Verboten: Signatur bereits vorhanden aber mit unterschiedlichem Parameternamen

Überschreiben von Methoden II – Regeln – Unterschied Überladen

- ✓ Man kann in einer Subklasse eine Methode neu definieren, die in der Oberklasse bereits definiert ist.
- ✓ In diesem Fall wird die ererbte Methode überschrieben(“Overriding”)
- ✓ Eine Methode der Oberklasse wird nur dann an die Unterklasse vererbt, wenn sie dort nicht überschrieben wird
- ✓ Sie wird überschrieben, wenn in der Unterklasse eine Methode mit gleichem Namen und gleicher Anzahl sowie gleichen Typen der Argumente definiert wird
- ✓ Die Namen der Argumente spielen keine Rolle, aber die Argument-Typen müssen genau gleich sein, sonst handelt es sich um Überladen, nicht überschreiben, d.h. es gibt beide Methoden in der Unterklasse
- ✓ Der Zugriffsschutz darf in der Unterklasse schwächer sein(z.B. protected in der Oberklasse, public in der Unterklasse.)

Überschreiben von Methoden - Prinzip

Overriding:

Gleiche Methode, gleiche Parameter

```
class Dog{
   public void bark(){
      System.out.println("woof ");
   }
}
class Hound extends Dog{
   public void bark(){
      System.out.println("bowl");
   }
}
```

Overloading:

Gleiche Methode, unterschiedliche Parameter

```
class Dog{
   public void bark(){
      System.out.println("woof ");
   }
}
class Hound extends Dog{
public void bark(int num){
      for(int i=0; i<num; i++)
         System.out.println("woof ");
   }
}
```

Überschreiben von Methoden III - @Override

Wenn man eine Methode überschreiben will, ist empfohlen, die „Annotation“ @Override zu verwenden:

```
class Hound extends Dog{
...
@Override
public void bark(){ ... }
}
```

- ✓ Für den Leser des Programms ist klarer, was passiert.
- ✓ Der Compiler gibt eine Warnung bzw. Fehlermeldung aus, wenn doch kein Überschreiben vorliegt.

Überschreiben von Methoden VI - Polymorphie

- ✓ Allgemein heißt eine Methode/Funktion “polymorph”(“vielgestaltig”), wenn sie für unterschiedliche Klassen(Datentypen) unterschiedliche Implementierungen hat.
- ✓ Dies wurde eigentlich „überladene“ Methoden einschließen, aber in der objektorientierten Programmierung ist es üblich, nur Methoden, die in einer Subklasse überschrieben werden, als polymorph zu bezeichnen.

Überschreiben von Methoden VI - final

- ✓ Mit dem Schlüsselwort final vor einer Methode verbietet man das Überschreiben dieser Methode
- ✓ final vor einer Klasse verbietet, dass Subklassen dieser Klasse definiert werden