

## Script generated by TTT

Title: groh: profile1 (10.06.2016)

Date: Fri Jun 10 13:29:20 CEST 2016

Duration: 94:29 min

Pages: 76

The screenshot shows a PowerPoint slide titled "Access Modifiers, Packages". The slide content includes:

- Access Modifier für Methoden und Attribute:

	Class	Package	Subclasses	World
public	✓	✓	✓	✓
protected	✓	✓	✓	
no modifier	✓	✓		
private	✓			

- Packages:

  - Kapseln (hierarchisch organisiert) Mengen von Klassen und Interfaces
  - Deklaration: `package nameOfPackage;`
  - Beispiele: `java.math`, `java.lang`, `java.net`, `de.tum.in`

The slide is part of a presentation with 201 slides, currently on slide 135. The Windows taskbar at the bottom shows the time as 13:29 on 10.06.2016.

## Access Modifiers, Packages

- Access Modifier für Methoden und Attribute:

	Class	Package	Subclasses	World
public	✓	✓	✓	✓
protected	✓	✓	✓	
no modifier	✓	✓		
private	✓			

- Packages:

- Kapseln (hierarchisch organisiert) Mengen von Klassen und Interfaces
- Deklaration: `package nameOfPackage;`
- Beispiele: `java.math`, `java.lang`, `java.net`, `de.tum.in`

## Access Modifiers, Packages

- Access Modifier für Methoden und Attribute:

	Class	Package	Subclasses	World
public	✓	✓	✓	✓
protected	✓	✓	✓	
no modifier	✓	✓		
private	✓			

- Packages:

- Kapseln (hierarchisch organisiert) Mengen von Klassen und Interfaces
- Deklaration: `package nameOfPackage;`
- Beispiele: `java.math`, `java.lang`, `java.net`, `de.tum.in`

## Access Modifiers: static und final

- **static:**
  - Methode oder Attribut **gehört zur Klasse** und nicht zum Objekt (Attribut: existiert nur einmal, ist für alle Objekte **dasselbe**)  
„Klassenmethode, Klassenattribut“ <---> „Instanzenmethode, Instanzenattribut“
- **final:**
  - für *Attribute*: können nicht mehr geändert werden (**Konstanten**)
  - für *Methoden*: können **nicht overridden** oder hidden werden (kommt gleich)
  - für *Klassen*: Es können **keine Unterklassen** davon abgeleitet werden.

```
final class MyClass {  
    static int sameForAllInstances = 3;  
    final int constantMayBeDifferentForEachInstance;  
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;  
  
    static void methodOne() { /* ... */ }  
    final void methodTwo() { /* ... */ }  
    static final void methodThree() { /* ... */ }  
    void methodFour() { /* ... */ }  
    ...  
}
```

136

## Access Modifiers: static und final

- **static:**
  - Methode oder Attribut **gehört zur Klasse** und nicht zum Objekt (Attribut: existiert nur einmal, ist für alle Objekte **dasselbe**)  
„Klassenmethode, Klassenattribut“ <---> „Instanzenmethode, Instanzenattribut“
- **final:**
  - für *Attribute*: können nicht mehr geändert werden (**Konstanten**)
  - für *Methoden*: können **nicht overridden** oder hidden werden (kommt gleich)
  - für *Klassen*: Es können **keine Unterklassen** davon abgeleitet werden.

```
final class MyClass {  
    static int sameForAllInstances = 3;  
    final int constantMayBeDifferentForEachInstance;  
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;  
  
    static void methodOne() { /* ... */ }  
    final void methodTwo() { /* ... */ }  
    static final void methodThree() { /* ... */ }  
    void methodFour() { /* ... */ }  
    ...  
}
```

136

## Access Modifiers: static und final

- **static:**
  - Methode oder Attribut **gehört zur Klasse** und nicht zum Objekt (Attribut: existiert nur einmal, ist für alle Objekte **dasselbe**)  
„Klassenmethode, Klassenattribut“ <---> „Instanzenmethode, Instanzenattribut“
- **final:**
  - für *Attribute*: können nicht mehr geändert werden (**Konstanten**)
  - für *Methoden*: können **nicht overridden** oder hidden werden (kommt gleich)
  - für *Klassen*: Es können **keine Unterklassen** davon abgeleitet werden.

```
final class MyClass {  
    static int sameForAllInstances = 3;  
    final int constantMayBeDifferentForEachInstance;  
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;  
  
    static void methodOne() { /* ... */ }  
    final void methodTwo() { /* ... */ }  
    static final void methodThree() { /* ... */ }  
    void methodFour() { /* ... */ }  
    ...  
}
```

136

## Access Modifiers: static und final

- **static:**
  - Methode oder Attribut **gehört zur Klasse** und nicht zum Objekt (Attribut: existiert nur einmal, ist für alle Objekte **dasselbe**)  
„Klassenmethode, Klassenattribut“ <---> „Instanzenmethode, Instanzenattribut“
- **final:**
  - für *Attribute*: können nicht mehr geändert werden (**Konstanten**)
  - für *Methoden*: können **nicht overridden** oder hidden werden (kommt gleich)
  - für *Klassen*: Es können **keine Unterklassen** davon abgeleitet werden.

```
final class MyClass {  
    static int sameForAllInstances = 3;  
    final int constantMayBeDifferentForEachInstance;  
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;  
  
    static void methodOne() { /* ... */ }  
    final void methodTwo() { /* ... */ }  
    static final void methodThree() { /* ... */ }  
    void methodFour() { /* ... */ }  
    ...  
}
```

136

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMayBeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMayBeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMayBeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMayBeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMayBeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMayBeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMayBeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMayBeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMaybeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMaybeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMaybeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMaybeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMaybeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMaybeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

## Access Modifiers: static und final

```
public class MyClass {
    static int sameForAllInstances = 3;
    final int constantMaybeDifferentForEachInstance;
    static final int CONSTANT_SAME_FOR_ALL_INSTANCES = 7;
    public int instVar;

    public MyClass(int iv, int const) {
        instVar = iv;
        constantMaybeDifferentForEachInstance = const;
    } // constructor

    public static void methodOne() { /* ... */ }
    public void methodTwo() { /* ... */ }
}
```

Muss im Constructor gesetzt werden (oder mit Initializer direkt bei Deklaration (bspw. final int c = 0;))

```
MyClass m1 = new MyClass(20, 11);
MyClass m2 = new MyClass(30, 9);
MyClass.CONSTANT_SAME_FOR_ALL_INSTANCES = 3333; //ERROR
MyClass.sameForAllInstances = 99;
System.out.println(m1.sameForAllInstances); // 99
System.out.println(m2.sameForAllInstances); // 99
System.out.println(MyClass.sameForAllInstances); // 99
System.out.println(m1.instVar); // 20
System.out.println(m2.instVar); // 30
m1.instVar = 77;
MyClass.methodOne();
m1.methodTwo();
MyClass.methodTwo(); //ERROR
```

137

```
3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
package zue3Wzw;

public class MeadowDemo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LittleBee maja;
        maja = new LittleBee();
        LittleBee willi = new LittleBee();
        maja.fly();
        maja.sting();
        willi.snooze();
        AngryHornet evil;
        evil = new AngryHornet();
        evil.sting();

        ICanSting someStinger;
        someStinger = evil;
        someStinger.sting();

        Flower flower1 = new Flower();
        Flower flower2 = new Flower();

        maja.collectPollen(flower1, 20.0d);
    }
}
```

```
4Wzw/src/zue4Wzw/Demo.java - Eclipse
package zue4Wzw;

public class Demo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int result = 0;
        Demo demo = new Demo();
        result = demo.minimum(5, 10*2);
        System.out.println("das ergebnis ist: " + String.valueOf(result));
        long result22 = demo.faculty(-3);
        System.out.println("das ergebnis ist: " + String.valueOf(result22));
        double result222 = demo.powerMitWhile(2.0d, 3);
        System.out.println("das ergebnis ist: " + String.valueOf(result222));
        double result2 = demo.exp(2.0d);
        System.out.println("das ergebnis ist: " + String.valueOf(result2));
        double[] theArray = new double[3];
        theArray[0] = 1.0d;
        theArray[1] = 2.0d;
        theArray[2] = 3.0d;
        double result3 = demo.expectation(theArray);
        System.out.println("das ergebnis ist: " + String.valueOf(result3));
        double result4 = demo.variance(theArray);
        System.out.println("das ergebnis ist: " + String.valueOf(result4));
        double[][] matrix = new double[3][3];
        System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
    }

    public double variance(double a[]){
        double result = 0.0d;
        double oneOverNMinusOne = 0.0d;
        if(a.length >=2)
            oneOverNMinusOne = 1.0d / ((double)(a.length) - 1);
        else
            System.out.println("array is not long enough");
        double expectation = expectation(a);
        for(int i=0; i<a.length; i++){
            result = result + (a[i] - expectation) * (a[i] - expectation);
        }
    }
}
```

```
4Wzw/src/zue4Wzw/Demo.java - Eclipse
package zue4Wzw;

public class Demo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int result = 0;
        Demo demo = new Demo();
        result = demo.minimum(5, 10*2);
        System.out.println("das ergebnis ist: " + String.valueOf(result));
        long result22 = demo.faculty(-3);
        System.out.println("das ergebnis ist: " + String.valueOf(result22));
        double result222 = demo.powerMitWhile(2.0d, 3);
        System.out.println("das ergebnis ist: " + String.valueOf(result222));
        double result2 = demo.exp(2.0d);
        System.out.println("das ergebnis ist: " + String.valueOf(result2));
        double[] theArray = new double[3];
        theArray[0] = 1.0d;
        theArray[1] = 2.0d;
        theArray[2] = 3.0d;
        double result3 = demo.expectation(theArray);
        System.out.println("das ergebnis ist: " + String.valueOf(result3));
        double result4 = demo.variance(theArray);
        System.out.println("das ergebnis ist: " + String.valueOf(result4));
        double[][] matrix = new double[3][3];
        System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
    }

    public double variance(double a[]){
        double result = 0.0d;
        double oneOverNMinusOne = 0.0d;
        if(a.length >=2)
            oneOverNMinusOne = 1.0d / ((double)(a.length) - 1);
        else
            System.out.println("array is not long enough");
        double expectation = expectation(a);
        for(int i=0; i<a.length; i++){
            result = result + (a[i] - expectation) * (a[i] - expectation);
        }
        result = oneOverNMinusOne * result;
        return result;
    }

    public double expectation(double[] a){
        double result = 0.0d;
        double oneOverN = 0.0d;
        if(a.length >= 1)
            oneOverN = 1.0d / a.length;
        else
            System.out.println(" das array ist leer");
        for(int i=0; i<a.length; i++){
            result = result + a[i];
        }
        return result * oneOverN;
    }

    public int minimum(int a, int b){
        int result;
    }
}
```

```
4Wzw/src/zue4Wzw/Demo.java - Eclipse
package zue4Wzw;

public class Demo {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int result = 0;
        Demo demo = new Demo();
        result = demo.minimum(5, 10*2);
        System.out.println("das ergebnis ist: " + String.valueOf(result));
        long result22 = demo.faculty(-3);
        System.out.println("das ergebnis ist: " + String.valueOf(result22));
        double result222 = demo.powerMitWhile(2.0d, 3);
        System.out.println("das ergebnis ist: " + String.valueOf(result222));
        double result2 = demo.exp(2.0d);
        System.out.println("das ergebnis ist: " + String.valueOf(result2));
        double[] theArray = new double[3];
        theArray[0] = 1.0d;
        theArray[1] = 2.0d;
        theArray[2] = 3.0d;
        double result3 = demo.expectation(theArray);
        System.out.println("das ergebnis ist: " + String.valueOf(result3));
        double result4 = demo.variance(theArray);
        System.out.println("das ergebnis ist: " + String.valueOf(result4));
        double[][] matrix = new double[3][3];
        System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
    }

    public double variance(double a[]){
        double result = 0.0d;
        double oneOverNMinusOne = 0.0d;
        if(a.length >=2)
            oneOverNMinusOne = 1.0d / ((double)(a.length) - 1);
        else
            System.out.println("array is not long enough");
        double expectation = expectation(a);
        for(int i=0; i<a.length; i++){
            result = result + (a[i] - expectation) * (a[i] - expectation);
        }
        result = oneOverNMinusOne * result;
        return result;
    }

    public double expectation(double[] a){
        double result = 0.0d;
        double oneOverN = 0.0d;
        if(a.length >= 1)
            oneOverN = 1.0d / a.length;
        else
            System.out.println(" das array ist leer");
        for(int i=0; i<a.length; i++){
            result = result + a[i];
        }
        return result * oneOverN;
    }

    public int minimum(int a, int b){
        int result;
    }
}
```

```

12 double result = 0.0d;
13 double oneOverNMinusOne = 0.0d;
14 if(a.length > 2)
15     oneOverNMinusOne = 1.0d / ((double)(a.length) - 1);
16 else
17     System.out.println("array is not long enough");
18 double expectation = expectation(a);
19 for(int i=0; i<a.length; i++){
20     result = result + (a[i] - expectation) * (a[i] - expectation);
21 }
22 result = oneOverNMinusOne * result;
23 return result;
24 }
25
26
27 public static double expectation(double[] a){
28     double result = 0.0d;
29     double oneOverN = 0.0d;
30     if(a.length >= 1)
31         oneOverN = 1.0d / a.length;
32     else
33         System.out.println("das array ist leer");
34     for(int i=0; i < a.length; i++){
35         result = result + a[i];
36     }
37     return result * oneOverN;
38 }
39
40
41 public static int minimum(int a, int b){
42     int result;
43     if(a < b){
44         result = a;
45         //System.out.println("kjhjhhi");
46         //System.out.println("hhhhh");
47     }
48     else
49         result = b;
50     return result;

```

```

83 double result = 1;
84 for(int i=0; i<m; i++){
85     result = result * argument;
86 }
87 return result;
88 }
89
90
91 public static double powerMitWhile(double argument, int m){
92     double result = 1.0d;
93     int i = m;
94     while(i > 0){
95         result = result * argument;
96         i = i - 1;
97     }
98     return result;
99 }
100
101
102 public static int minimum2(int a, int b){
103     if(a < b)
104         return a;
105     else
106         return b;
107 }
108
109
110 public static long faculty(int a){
111     long result = 1;
112     while(a > 1){
113         result = result * a;
114         a = a - 1;
115     }
116     return result;
117 }
118
119 }
120

```

```

1 package zue4Wzw;
2
3 public class Demo {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int result = 0;
8         result = Demo.minimum(5, 10*2);
9         System.out.println("das ergebnis ist: " + String.valueOf(result));
10        long result22 = Demo.faculty(-3);
11        System.out.println("das ergebnis ist: " + String.valueOf(result22));
12        double result222 = Demo.powerMitWhile(2.0d, 3);
13        System.out.println("das ergebnis ist: " + String.valueOf(result222));
14        double result2 = Demo.exp(2.0d);
15        System.out.println("das ergebnis ist: " + String.valueOf(result2));
16        double[] theArray = new double[3];
17        theArray[0] = 1.0d;
18        theArray[1] = 2.0d;
19        theArray[2] = 3.0d;
20        double result3 = Demo.expectation(theArray);
21        System.out.println("das ergebnis ist: " + String.valueOf(result3));
22        double result4 = Demo.variance(theArray);
23        System.out.println("das ergebnis ist: " + String.valueOf(result4));
24        double[][] matrix = new double[3][3];
25        System.out.println("das ergebnis ist: " + String.valueOf(matrix[2][2]));
26    }
27 }

```

```

<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 13:56:56)
das ergebnis ist: 8.0
das ergebnis ist: 7.3887125220458545
das ergebnis ist: 2.0
das ergebnis ist: 1.0
das ergebnis ist: 0.0

```

## Overloading

Overloading: In einer Klasse mehrere Methoden mit gleichem Namen, aber verschiedener Parameterliste:

```

class OverloadingDemoClass {
    public int doSomething () {
        return 1 + 1;
    }

    public int doSomething(int param) {
        return param + 2;
    }
}

```

```

OverloadingDemoClass odc = new OverloadingDemoClass ();
int result1 = odc.doSomething();
int result2 = odc.doSomething(33);

```

Sinn: Flexibilität (speziellere und weniger spezielle Varianten der Methode anbieten, Abstraktion (--> APIs):

**Overloading:** In **einer** Klasse mehrere Methoden mit **gleichem Namen**, aber **verschiedener Parameterliste**:

```
class OverloadingDemoClass {
    public int doSomething() {
        return 1 + 1;
    }

    public int doSomething(int param) {
        return param + 2;
    }
}

OverloadingDemoClass odc = new OverloadingDemoClass();
int result1 = odc.doSomething();
int result2 = odc.doSomething(33);
```

**Sinn:** Flexibilität (speziellere und weniger spezielle Varianten der Methode anbieten, Abstraktion (--> APIs):

**Overriding:** In einer **Unterklasse** Methode mit **gleichem Namen**, und **gleicher Parameterliste** wie in Oberklasse:

```
class Bicycle {
    int speed;
    public void speedUp(int increment) {
        speed = speed + increment;
        System.out.println("superclass instance-method");
    }
}

class MountainBike extends Bicycle {
    public void speedUp(int increment) {
        speed = speed + increment;
        System.out.println("subclass instance-method");
    }
}

MountainBike mb = new MountainBike();
mb.speedUp(10); // mb.speed == 20
```

➔ Ausgabe: **subclass instance-method**

**Sinn:** Unterklasse bietet speziellere Version der Methode an (Aspekt von Polymorphie)

**Overriding:** In einer **Unterklasse** Methode mit **gleichem Namen**, und **gleicher Parameterliste** wie in Oberklasse:

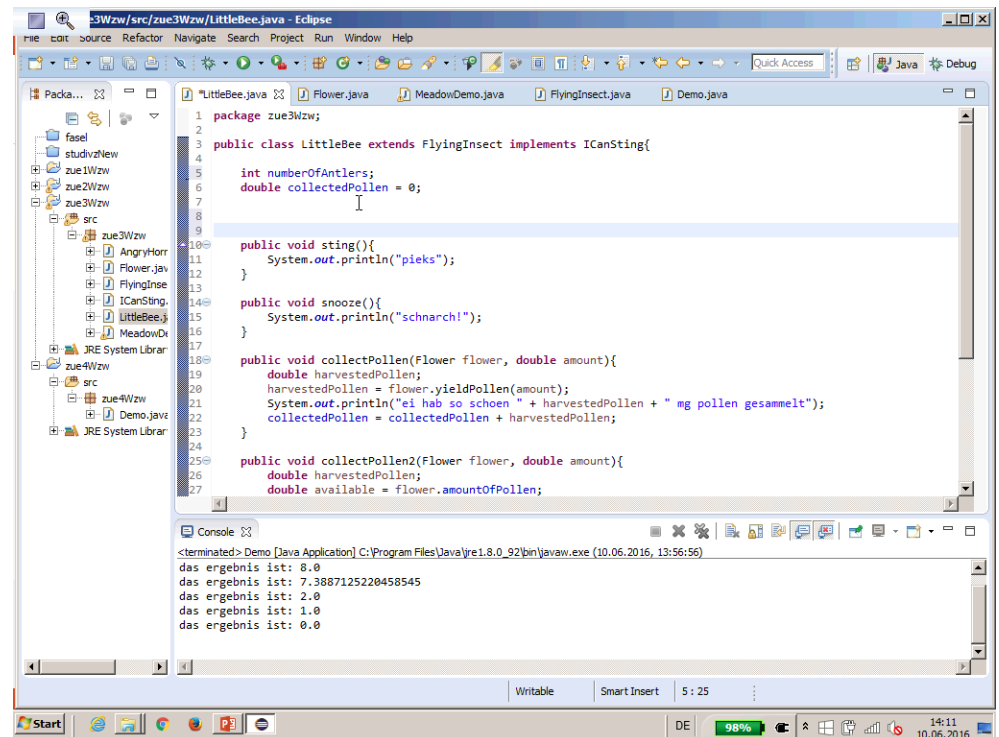
```
class Bicycle {
    int speed;
    public void speedUp(int increment) {
        speed = speed + increment;
        System.out.println("superclass instance-method");
    }
}

class MountainBike extends Bicycle {
    public void speedUp(int increment) {
        super(2 * increment); // call overridden method of superclass
        System.out.println("subclass instance-method");
    }
}

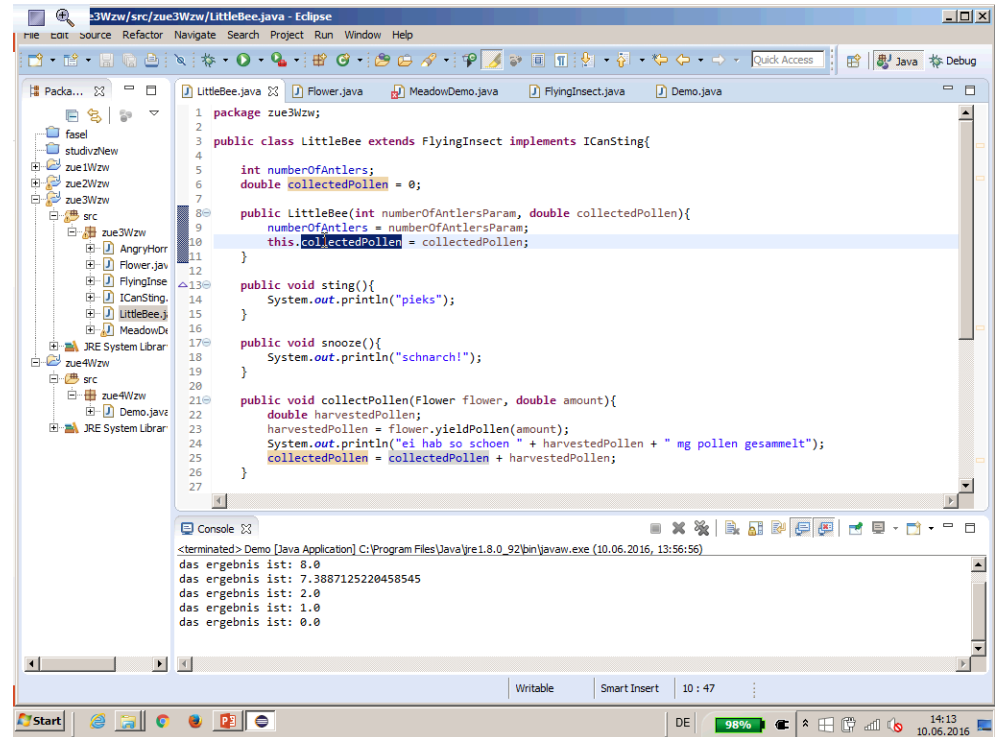
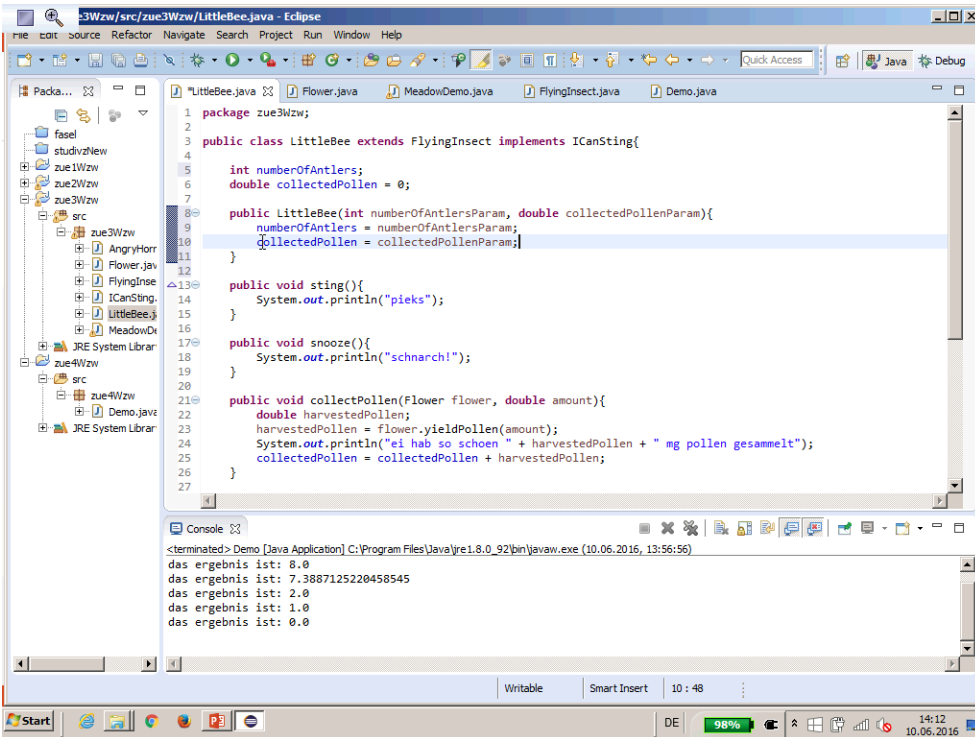
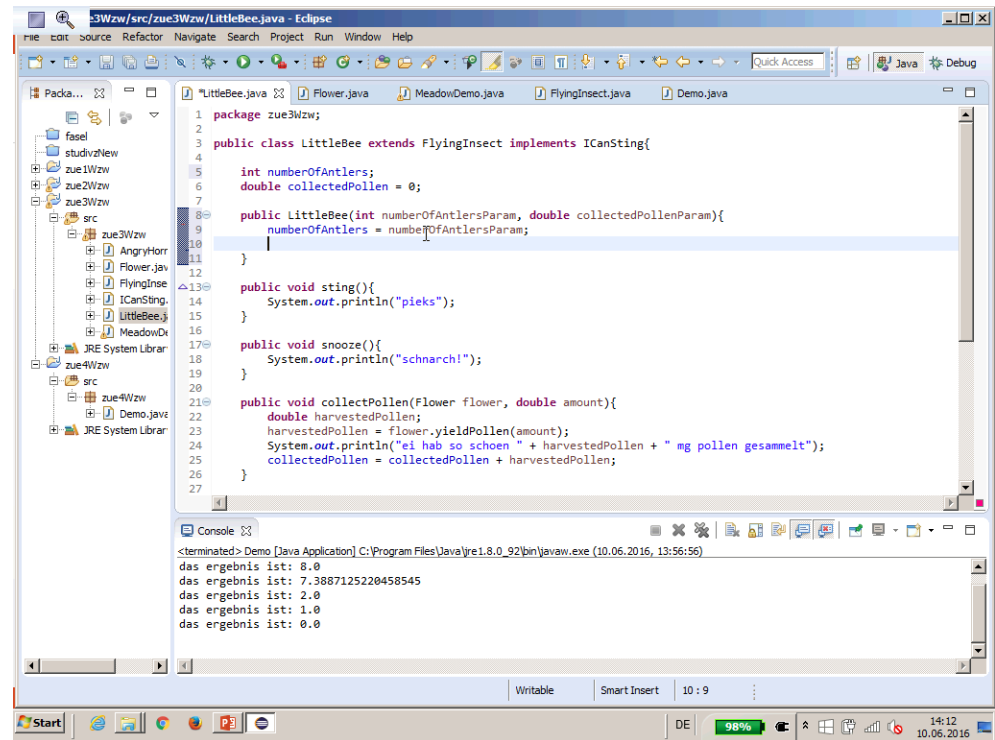
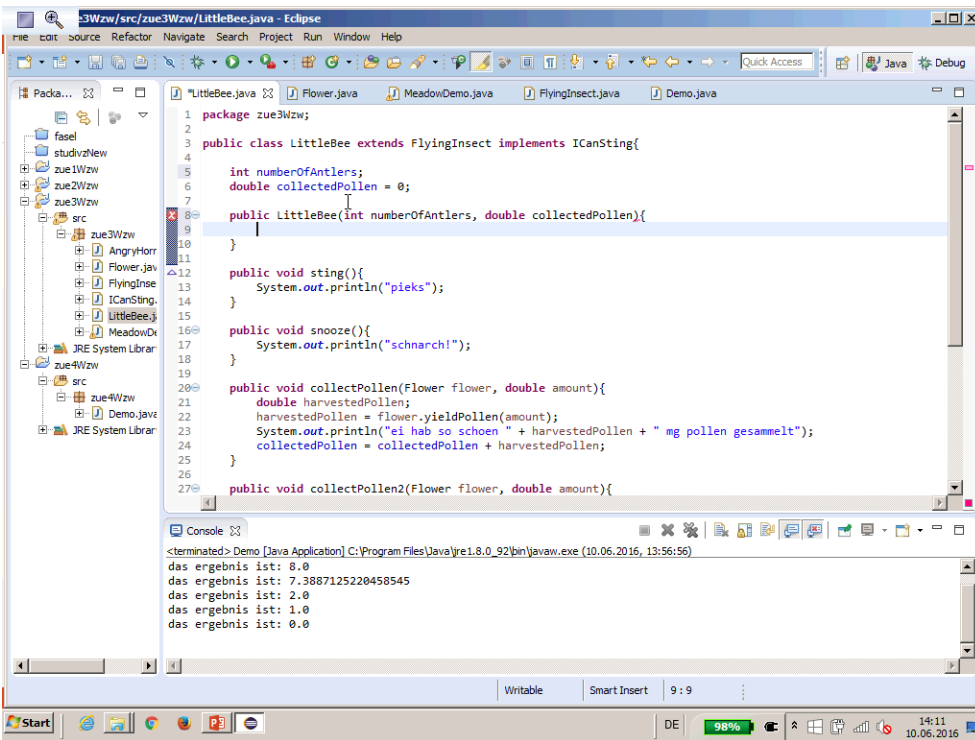
MountainBike mb = new MountainBike();
mb.speedUp(10); // mb.speed == 20
```

➔ Ausgabe: **superclass instance-method**  
**subclass instance-method**

**Sinn:** Unterklasse bietet speziellere Version der Methode an (Aspekt von Polymorphie)









```
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     int numberOfAntlers;
6     double collectedPollen = 0;
7
8     public LittleBee(int numberOfAntlersParam, double collectedPollen){
9         numberOfAntlers = numberOfAntlersParam;
10        this.collectedPollen = collectedPollen;
11    }
12
13    public void sting(){
14        System.out.println("pieks");
15    }
16
17    public void snooze(){
18        System.out.println("schnarch!");
19    }
20
21    public void collectPollen(Flower flower, double amount){
22        double harvestedPollen;
23        harvestedPollen = flower.yieldPollen(amount);
24        System.out.println("ei hab so schoen " + harvestedPollen + " mg pollen gesammelt");
25        collectedPollen = collectedPollen + harvestedPollen;
26    }
27 }
```

Console:  
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (10.06.2016, 13:56:56)  
das ergebnis ist: 8.0  
das ergebnis ist: 7.3887125220458545  
das ergebnis ist: 2.0  
das ergebnis ist: 1.0  
das ergebnis ist: 0.0

```
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public void sting(){
15        System.out.println("pieks");
16    }
17
18    public void snooze(){
19        System.out.println("schnarch!");
20    }
21
22    public void collectPollen(Flower flower, double amount){
23        double harvestedPollen;
24        harvestedPollen = flower.yieldPollen(amount);
25        System.out.println("ei hab so schoen " + harvestedPollen + " mg pollen gesammelt");
26    }
27 }
```

Console:  
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (10.06.2016, 13:56:56)  
das ergebnis ist: 8.0  
das ergebnis ist: 7.3887125220458545  
das ergebnis ist: 2.0  
das ergebnis ist: 1.0  
das ergebnis ist: 0.0

```
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = numberOfAntlersParam;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void sting(){
20        System.out.println("pieks");
21    }
22
23    public void snooze(){
24        System.out.println("schnarch!");
25    }
26
27 }
```

Console:  
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (10.06.2016, 13:56:56)  
das ergebnis ist: 8.0  
das ergebnis ist: 7.3887125220458545  
das ergebnis ist: 2.0  
das ergebnis ist: 1.0  
das ergebnis ist: 0.0

```
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void sting(){
20        System.out.println("pieks");
21    }
22
23    public void snooze(){
24        System.out.println("schnarch!");
25    }
26
27 }
```

Console:  
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0\_92\bin\javaw.exe (10.06.2016, 13:56:56)  
das ergebnis ist: 8.0  
das ergebnis ist: 7.3887125220458545  
das ergebnis ist: 2.0  
das ergebnis ist: 1.0  
das ergebnis ist: 0.0

```

3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java
1 package zue3Wzw;
2
3 public class MeadowDemo {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         LittleBee maja;
8         maja = new LittleBee(1, 20.0d);
9         LittleBee willi = new LittleBee(2);
10        maja.fly();
11        maja.sting();
12        willi.snooze();
13        AngryHornet evil;
14        evil = new AngryHornet();
15        evil.sting();
16
17        ICanSting someStinger;
18        someStinger = evil;
19        someStinger.sting();
20
21        Flower flower1 = new Flower();
22        Flower flower2 = new Flower();
23
24        maja.collectPollen(flower1, 20.0d);
25
26    }
27 }
28 }
29
Writeable Smart Insert 8: 35
Start DE 98% 14:16 10.06.2016

```

```

3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java Console
1 package zue3Wzw;
2
3 public class MeadowDemo {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         LittleBee maja;
8         maja = new LittleBee(2, 20.0d);
9         LittleBee willi = new LittleBee(3);
10        maja.fly();
11        maja.sting();
12        willi.snooze();
13        AngryHornet evil;
14        evil = new AngryHornet();
15        evil.sting();
16
17        ICanSting someStinger;
18        someStinger = evil;
19        someStinger.sting();
20
21        Flower flower1 = new Flower();
22        Flower flower2 = new Flower();
23
24        maja.collectPollen(flower1, 20.0d);
25
26    }
27 }
28 }
29
Console
<terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:17:28)
brumm
pieks!
schnarch!
MEGA-pieks
MEGA-pieks
uebrige pollen: 80.0
ei hab so schoen 20.0 mg pollen gesammelt
Writeable Smart Insert
Start DE 98% 14:16 10.06.2016

```

```

Java - zue3Wzw/src/zue3Wzw/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void fly(){
20        System.out.println("arg0");
21    }
22
23    public void sting(){
24        System.out.println("pieks");
25    }
26
27    public void snooze(){
28        System.out.println("schnarch!");
29    }
30
31    public void collectPollen(Flower flower, double amount){
32        double harvestedPollen;
33        harvestedPollen = flower.yieldPollen(amount);
34        System.out.println("ei hab so schoen " + harvestedPollen + " mg pollen gesammelt");
35        collectedPollen = collectedPollen + harvestedPollen;
36    }
37
38    public void collectPollen2(Flower flower, double amount){
39
40    }
41
Writeable Smart Insert 20: 26
Start DE 98% 14:18 10.06.2016

```

```

Java - zue3Wzw/src/zue3Wzw/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java Console
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void fly(){
20        System.out.println("bienenbrumm");
21    }
22
23    public void sting(){
24        System.out.println("pieks");
25    }
26
27
28
Console
<terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:18:57)
bienenbrumm
pieks
schnarch!
MEGA-pieks
MEGA-pieks
uebrige pollen: 80.0
ei hab so schoen 20.0 mg pollen gesammelt
Writeable Smart Insert 19: 23
Start DE 98% 14:18 10.06.2016

```

```
Java - zue3Wzw/src/zue3Wzw/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void fly(){
20        super();
21        System.out.println("bienenbrumm");
22    }
23
24    public void sting(){
25        System.out.println("pieks");
26    }
27
28
29 Console
30 <terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:19:34)
31 Exception in thread "main" java.lang.Error: Unresolved compilation problem:
32 Constructor call must be the first statement in a constructor
33
34 at zue3Wzw.LittleBee.fly(LittleBee.java:20)
35 at zue3Wzw.MeadowDemo.main(MeadowDemo.java:10)
```

```
Java - zue3Wzw/src/zue3Wzw/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void fly(){
20        this.super();
21        System.out.println("bienenbrumm");
22    }
23
24    public void sting(){
25        System.out.println("pieks");
26    }
27
28
29 Console
30 <terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:19:59)
31 Exception in thread "main" java.lang.Error: Unresolved compilation problem:
32 Constructor call must be the first statement in a constructor
33
34 at zue3Wzw.LittleBee.fly(LittleBee.java:20)
35 at zue3Wzw.MeadowDemo.main(MeadowDemo.java:10)
```

```
Java - zue3Wzw/src/zue3Wzw/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void fly(){
20        System.out.println("bienenbrumm");
21    }
22
23    public void sting(){
24        System.out.println("pieks");
25    }
26
27
28 Console
29 <terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:20:30)
30 bienenbrumm
31 pieks
32 schmarch!
33 MEGA-pieks
34 MEGA-pieks
35 uebrige pollen: 80.0
36 ei hab so schoen 20.0 mg pollen gesammelt
```

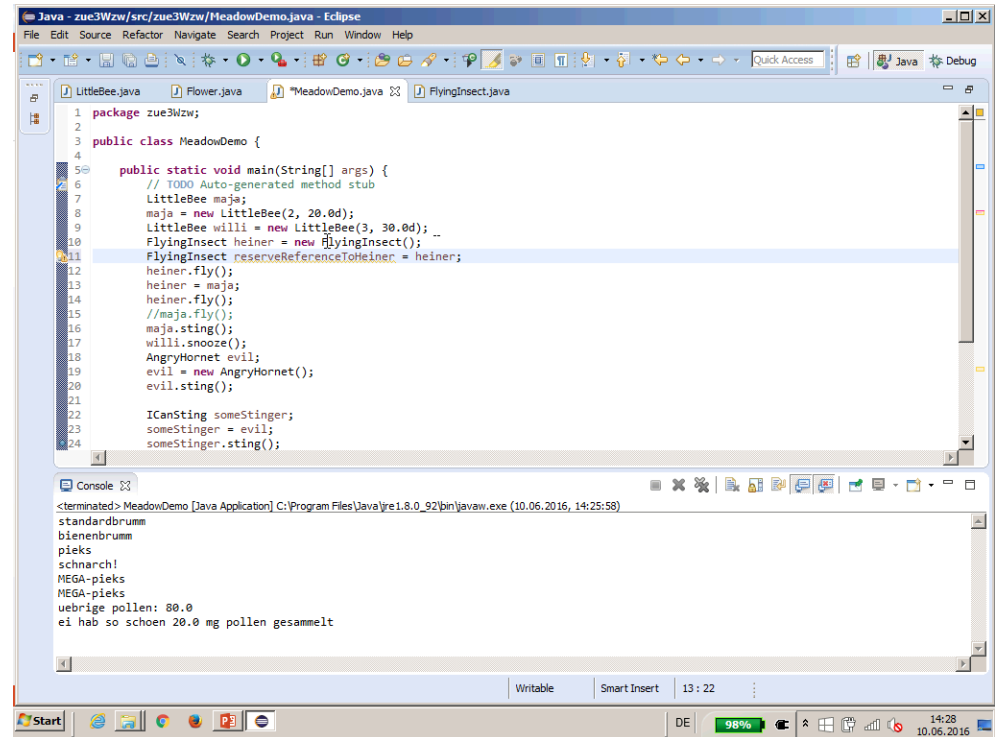
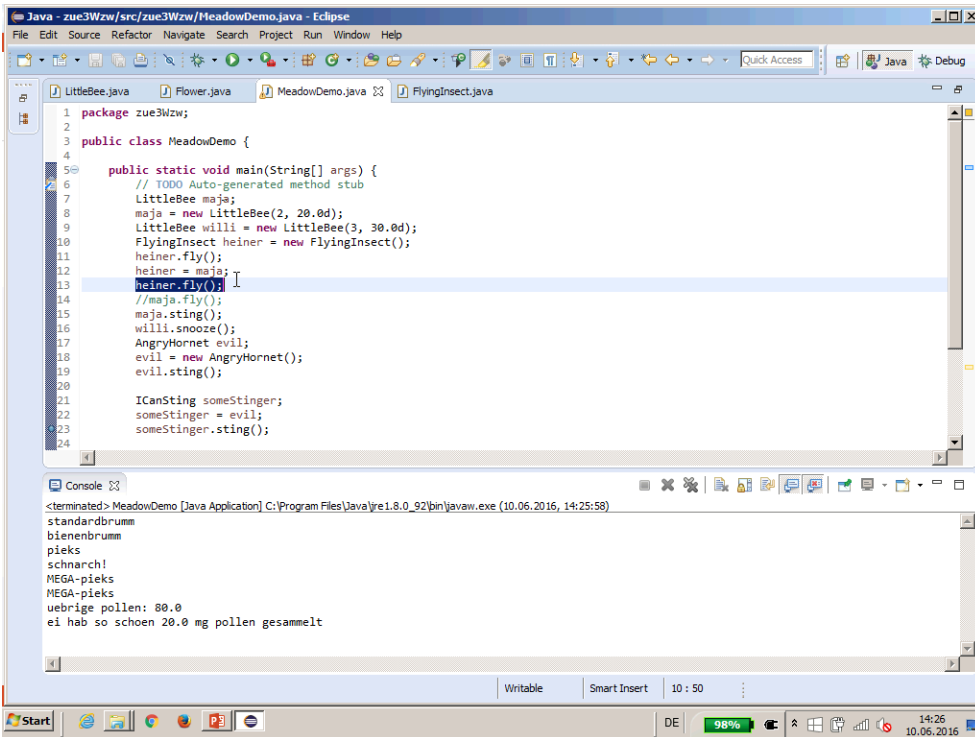
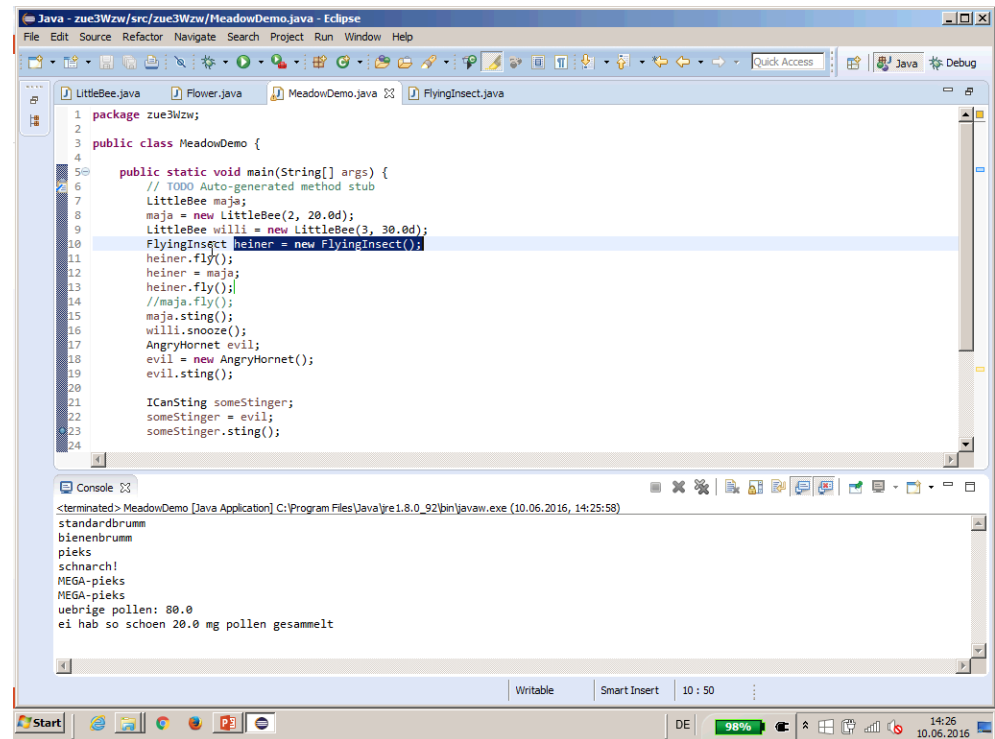
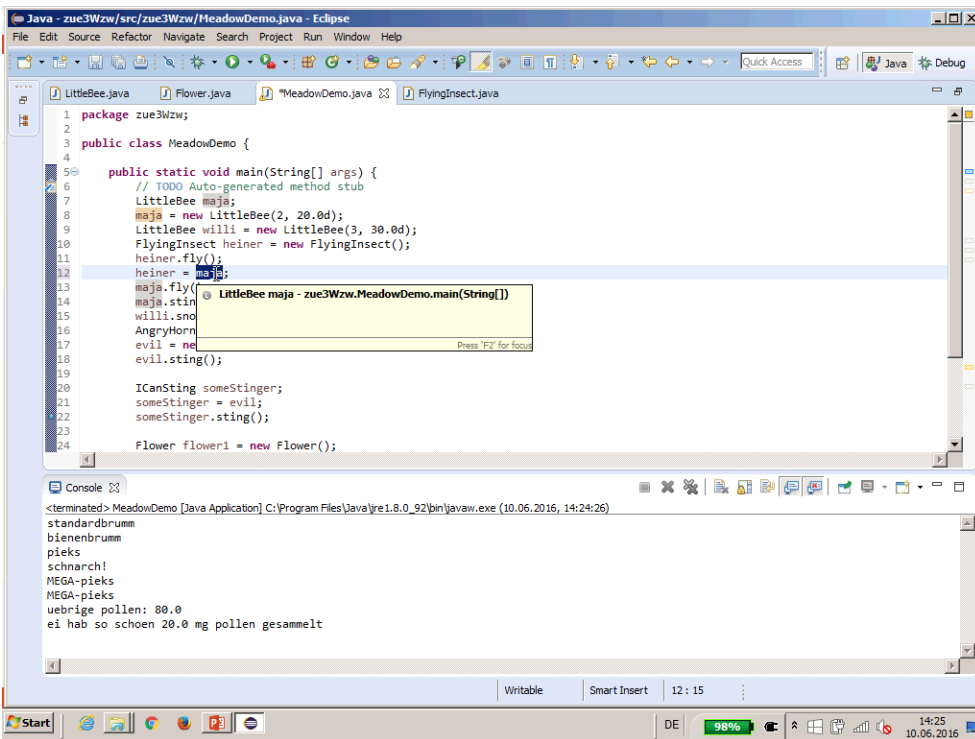
```
Java - zue3Wzw/src/zue3Wzw/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
LittleBee.java Flower.java MeadowDemo.java FlyingInsect.java
1 package zue3Wzw;
2
3 public class LittleBee extends FlyingInsect implements ICanSting{
4
5     static final int TYPICAL_NO_OF_ANTLERS = 2;
6     int numberOfAntlers;
7     double collectedPollen = 0;
8
9     public LittleBee(int numberOfAntlersParam, double collectedPollen){
10        numberOfAntlers = numberOfAntlersParam;
11        this.collectedPollen = collectedPollen;
12    }
13
14    public LittleBee(double collectedPollen){
15        numberOfAntlers = TYPICAL_NO_OF_ANTLERS;
16        this.collectedPollen = collectedPollen;
17    }
18
19    public void fly(){
20        System.out.println("bienenbrumm");
21    }
22
23    public void sting(){
24        System.out.println("pieks");
25    }
26
27
28 Console
29 <terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:20:30)
30 bienenbrumm
31 pieks
32 schmarch!
33 MEGA-pieks
34 MEGA-pieks
35 uebrige pollen: 80.0
36 ei hab so schoen 20.0 mg pollen gesammelt
```

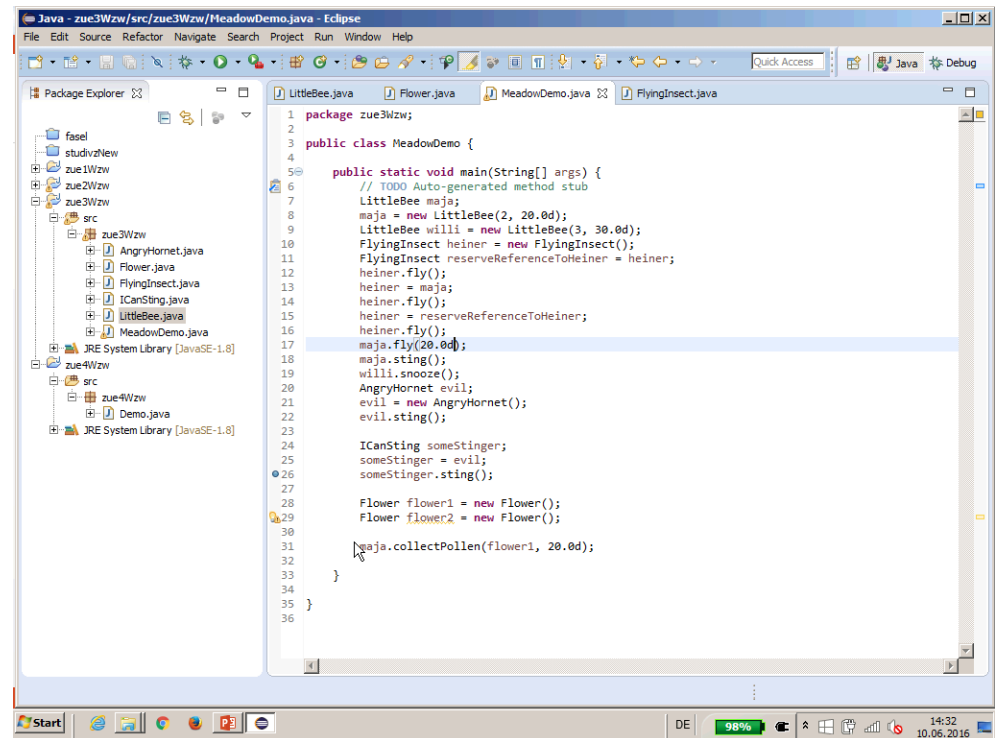
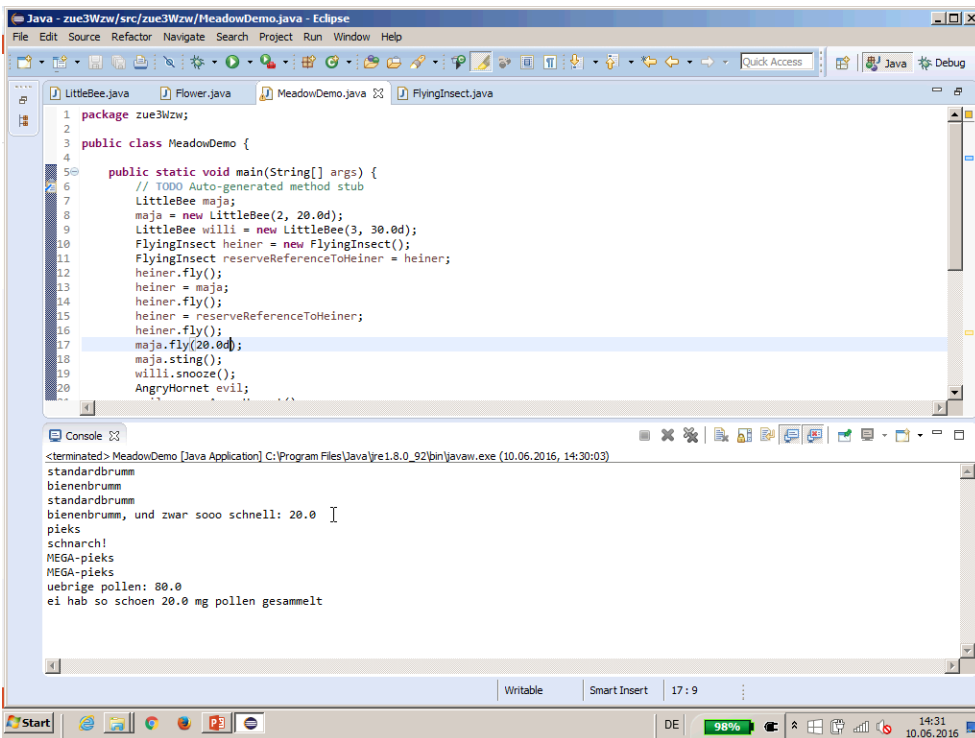
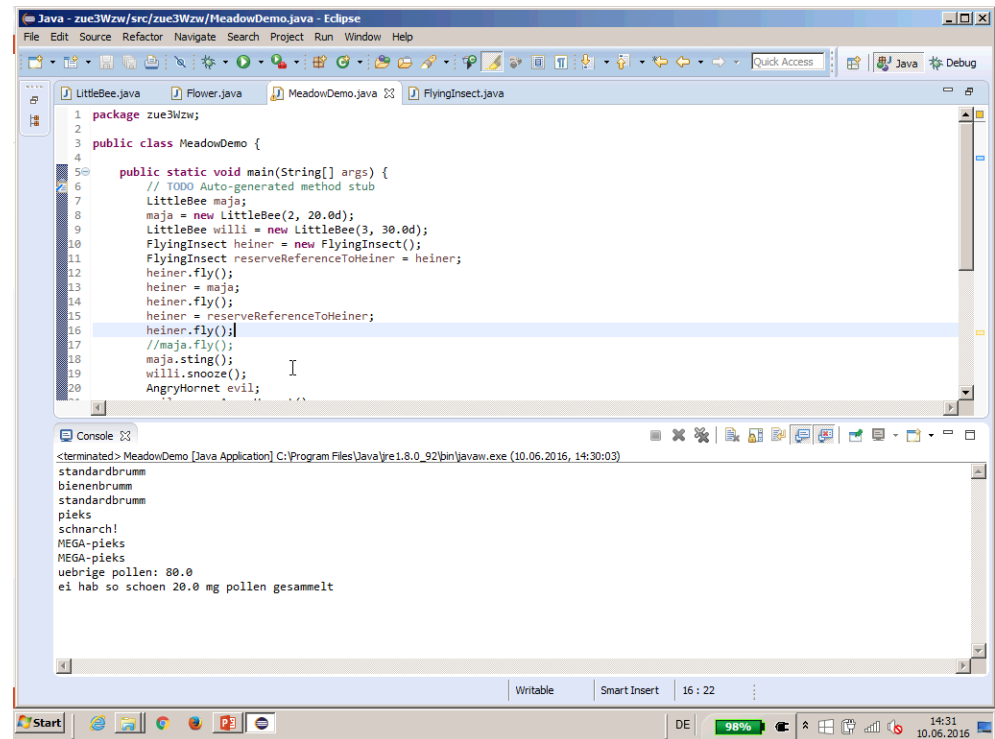
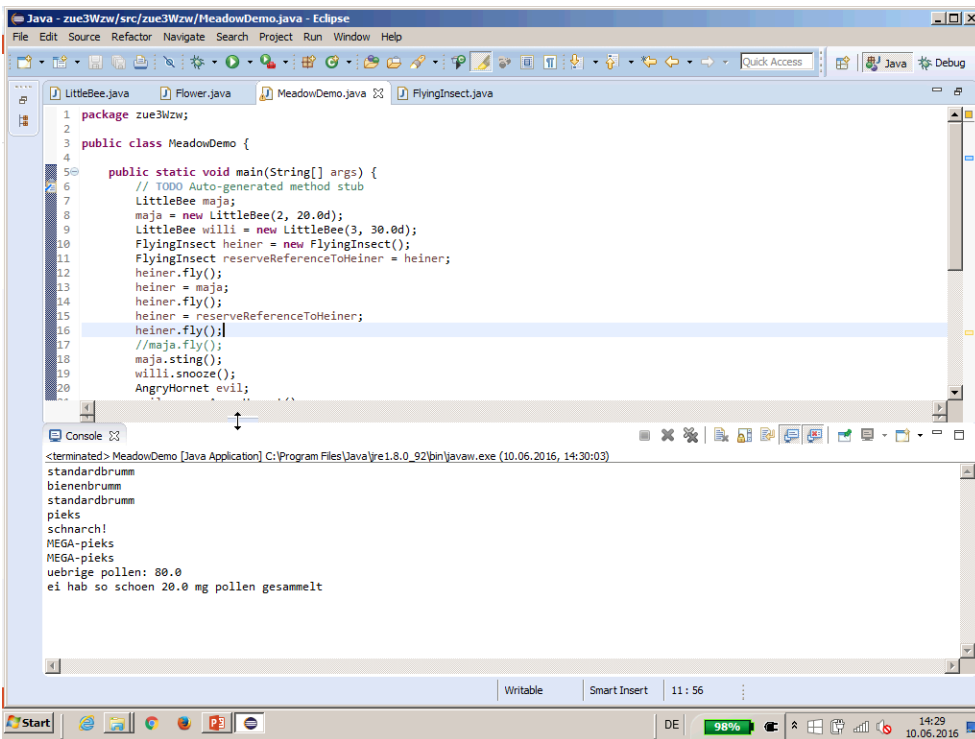
```
Java - zue3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
package zue3Wzw;
public class MeadowDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LittleBee maja;
        maja = new LittleBee(2, 20.0d);
        LittleBee willi = new LittleBee(3, 30.0d);
        maja.fly();
        maja.sting();
        willi.snooze();
        AngryHornet evil;
        evil = new AngryHornet();
        evil.sting();
        ICanSting someStinger;
        someStinger = evil;
        someStinger.sting();
        Flower flower1 = new Flower();
        Flower flower2 = new Flower();
    }
}
Console
<terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:20:30)
bienenbrumm
pieks
schnarch!
MEGA-pieks
MEGA-pieks
uebrige pollen: 80.0
ei hab so schoen 20.0 mg pollen gesammelt
9: 51
```

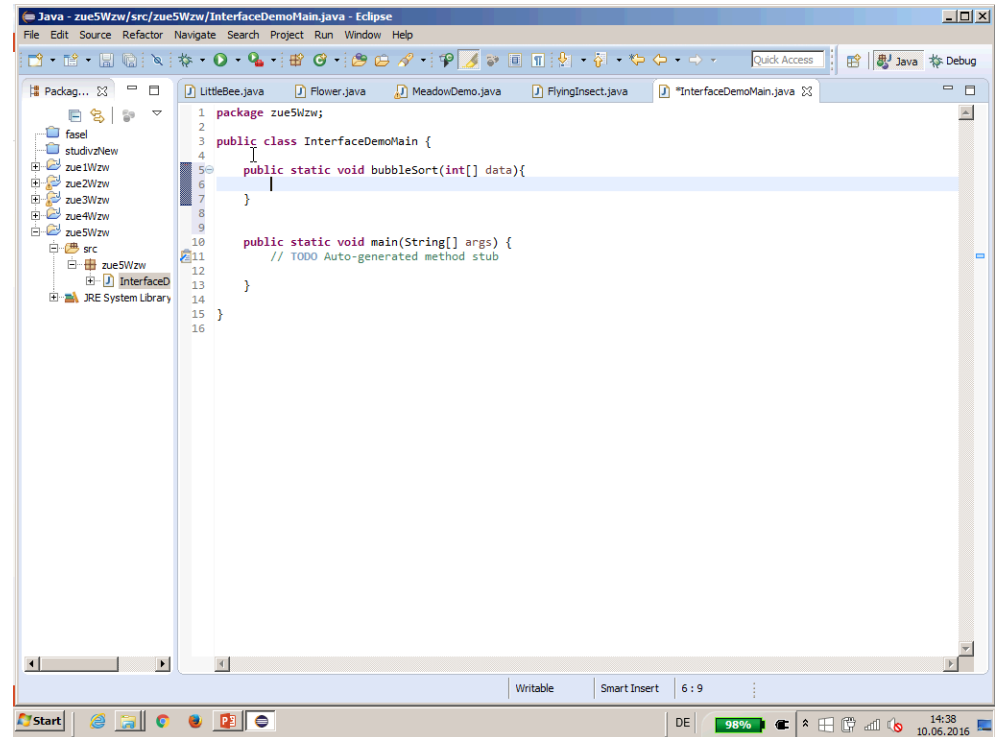
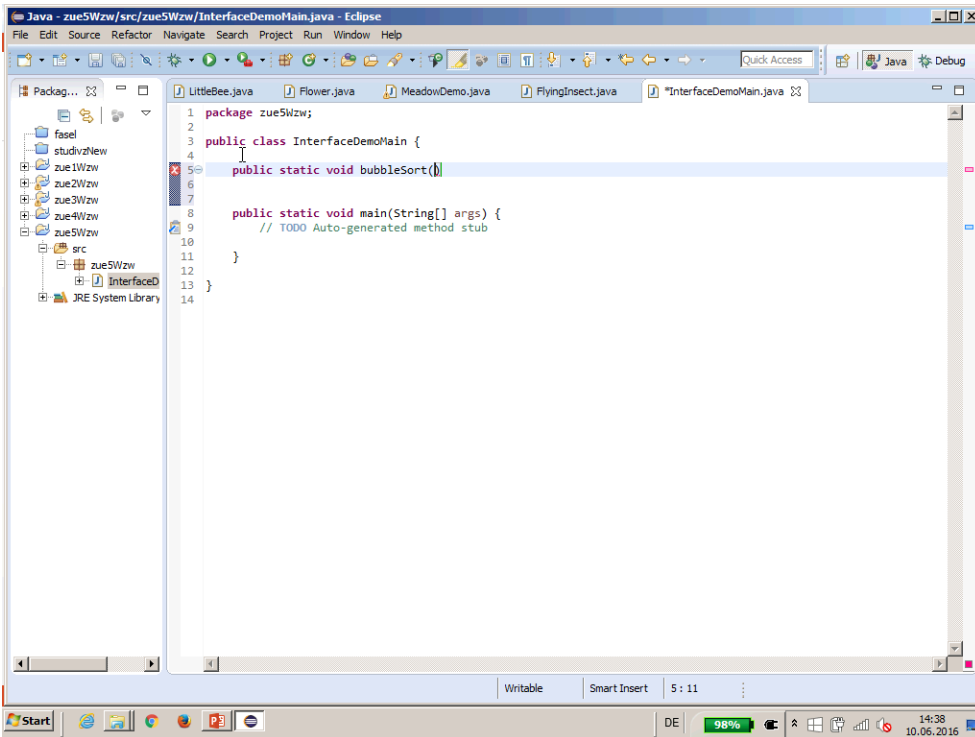
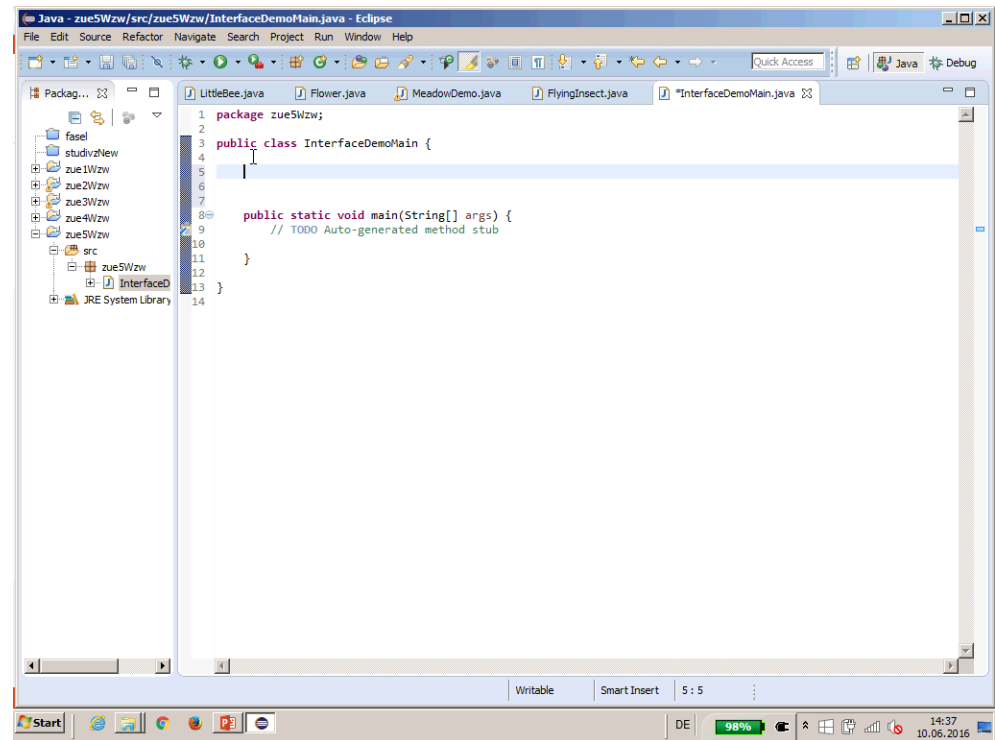
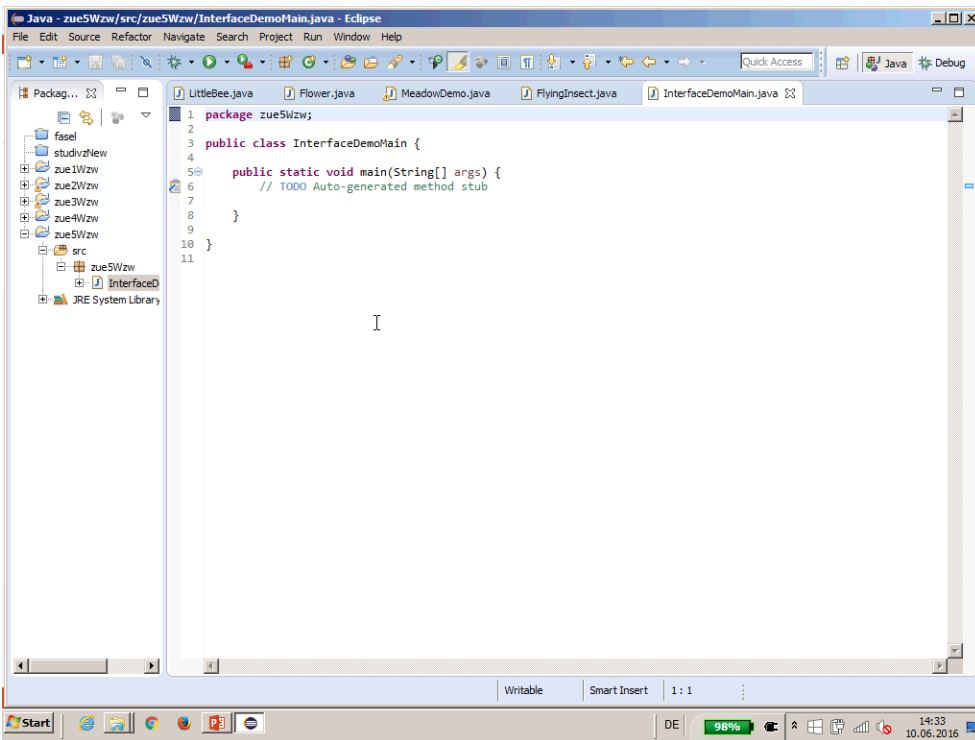
```
Java - zue3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
package zue3Wzw;
public class MeadowDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LittleBee maja;
        maja = new LittleBee(2, 20.0d);
        LittleBee willi = new LittleBee(3, 30.0d);
        FlyingInsect heiner = new FlyingInsect();
        maja.fly();
        maja.sting();
        willi.snooze();
        AngryHornet evil;
        evil = new AngryHornet();
        evil.sting();
        ICanSting someStinger;
        someStinger = evil;
        someStinger.sting();
        Flower flower1 = new Flower();
        Flower flower2 = new Flower();
    }
}
Console
<terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:20:30)
bienenbrumm
pieks
schnarch!
MEGA-pieks
MEGA-pieks
uebrige pollen: 80.0
ei hab so schoen 20.0 mg pollen gesammelt
9: 51
```

```
Java - zue3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
package zue3Wzw;
public class MeadowDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LittleBee maja;
        maja = new LittleBee(2, 20.0d);
        LittleBee willi = new LittleBee(3, 30.0d);
        FlyingInsect heiner = new FlyingInsect();
        heiner.fly();
        maja.fly();
        maja.sting();
        willi.snooze();
        AngryHornet evil;
        evil = new AngryHornet();
        evil.sting();
        ICanSting someStinger;
        someStinger = evil;
        someStinger.sting();
        Flower flower1 = new Flower();
    }
}
Console
<terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:24:26)
standardbrumm
bienenbrumm
pieks
schnarch!
MEGA-pieks
MEGA-pieks
uebrige pollen: 80.0
ei hab so schoen 20.0 mg pollen gesammelt
11: 22
```

```
Java - zue3Wzw/src/zue3Wzw/MeadowDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
package zue3Wzw;
public class MeadowDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        LittleBee maja;
        maja = new LittleBee(2, 20.0d);
        LittleBee willi = new LittleBee(3, 30.0d);
        FlyingInsect heiner = new FlyingInsect();
        heiner.fly();
        heiner = maja;
        maja.fly();
        maja.sting();
        willi.snooze();
        AngryHornet evil;
        evil = new AngryHornet();
        evil.sting();
        ICanSting someStinger;
        someStinger = evil;
        someStinger.sting();
        Flower flower1 = new Flower();
    }
}
Console
<terminated> MeadowDemo [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:24:26)
standardbrumm
bienenbrumm
pieks
schnarch!
MEGA-pieks
MEGA-pieks
uebrige pollen: 80.0
ei hab so schoen 20.0 mg pollen gesammelt
13: 20
```









```
Java - zue5Wzw/src/zue5Wzw/InterfaceDemoMain.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer: fasel, studivzNew, zue1WzW, zue2WzW, zue3WzW, zue4WzW, zue5WzW, src, zue5Wzw, interfaceD, JRE System Library
Editor:
1 package zue5Wzw;
2
3 public class InterfaceDemoMain {
4
5     public static void bubbleSort(int[] data){
6         for(int i = data.length-1; i>0; i--){
7             for(int j=0; j<i; j++){
8                 |
9             }
10        }
11    }
12
13
14    public static void main(String[] args) {
15        // TODO Auto-generated method stub
16    }
17
18
19 }
20
Status: Writable Smart Insert 7: 36 14:41 10.06.2016
```

```
Java - zue5Wzw/src/zue5Wzw/InterfaceDemoMain.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer: fasel, studivzNew, zue1WzW, zue2WzW, zue3WzW, zue4WzW, zue5WzW, src, zue5Wzw, interfaceD, JRE System Library
Editor:
1 package zue5Wzw;
2
3 public class InterfaceDemoMain {
4
5     public static void bubbleSort(int[] data){
6         for(int i = data.length-1; i>0; i--){
7             for(int j=0; j<i; j++){
8                 if(data[j]> data[j+1]){
9                     //swap elements
10                    data[j] = data[j+1];
11                    data[j+1] = data[j];
12                }
13            }
14        }
15    }
16
17
18
19    public static void main(String[] args) {
20        // TODO Auto-generated method stub
21    }
22
23
24 }
25
Status: Writable Smart Insert 11: 39 14:42 10.06.2016
```

```
Java - zue5Wzw/src/zue5Wzw/InterfaceDemoMain.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer: fasel, studivzNew, zue1WzW, zue2WzW, zue3WzW, zue4WzW, zue5WzW, src, zue5Wzw, interfaceD, JRE System Library
Editor:
1 package zue5Wzw;
2
3 public class InterfaceDemoMain {
4
5     public static void bubbleSort(int[] data){
6         for(int i = data.length-1; i>0; i--){
7             for(int j=0; j<i; j++){
8                 if(data[j]> data[j+1]){
9                     //swap elements
10                    data[j] = data[j+1];
11                    data[j+1] = data[j];
12                }
13            }
14        }
15    }
16
17
18
19    public static void main(String[] args) {
20        // TODO Auto-generated method stub
21    }
22
23
24 }
25
Status: Writable Smart Insert 11: 39 14:42 10.06.2016
```

```
Java - zue5Wzw/src/zue5Wzw/InterfaceDemoMain.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer: fasel, studivzNew, zue1WzW, zue2WzW, zue3WzW, zue4WzW, zue5WzW, src, zue5Wzw, interfaceD, JRE System Library
Editor:
6     int backup;
7     for(int i = data.length-1; i>0; i--){
8         for(int j=0; j<i; j++){
9             if(data[j]> data[j+1]){
10                //swap elements
11                backup = data[j];
12                data[j] = data[j+1];
13                data[j+1] = backup;
14            }
15        }
16    }
17
18
19
20
21
22    public static void main(String[] args) {
23        // TODO Auto-generated method stub
24        int[] foo = {5,3,7,1,2,10};
25        showArray(foo);
26
27
28    public static void showArray(int[] theArray){
29        for(int i=0; i<theArray.length; i++){
30            System.out.print(" " +theArray[i]);
31        }
32    }
33
Console:
<terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (10.06.2016, 14:46:29)
, 5
, 3
, 7
, 1
, 2
, 10
Status: Writable Smart Insert 24: 9 14:46 10.06.2016
```

