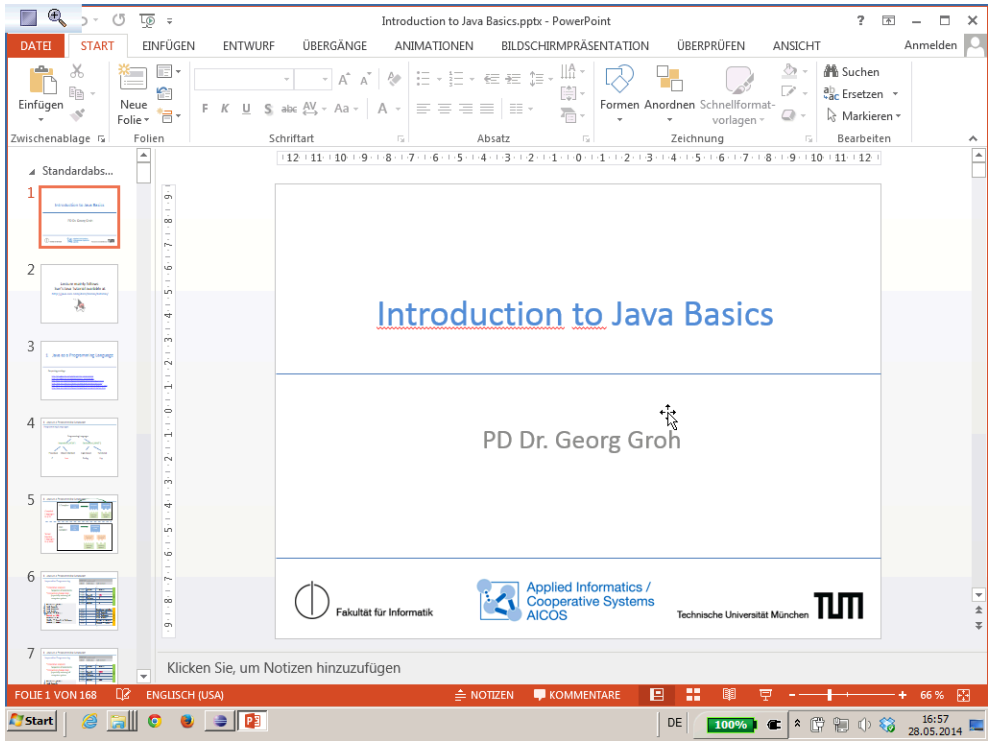
Script generated by TTT

Title: groh: profile1 (28.05.2014)

Date: Wed May 28 16:57:52 CEST 2014

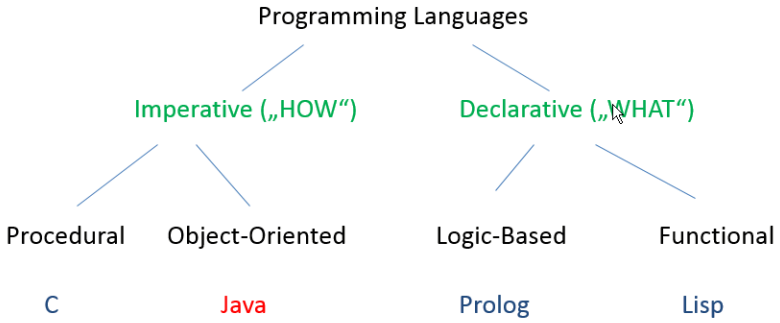
Duration: 90:22 min

Pages: 121

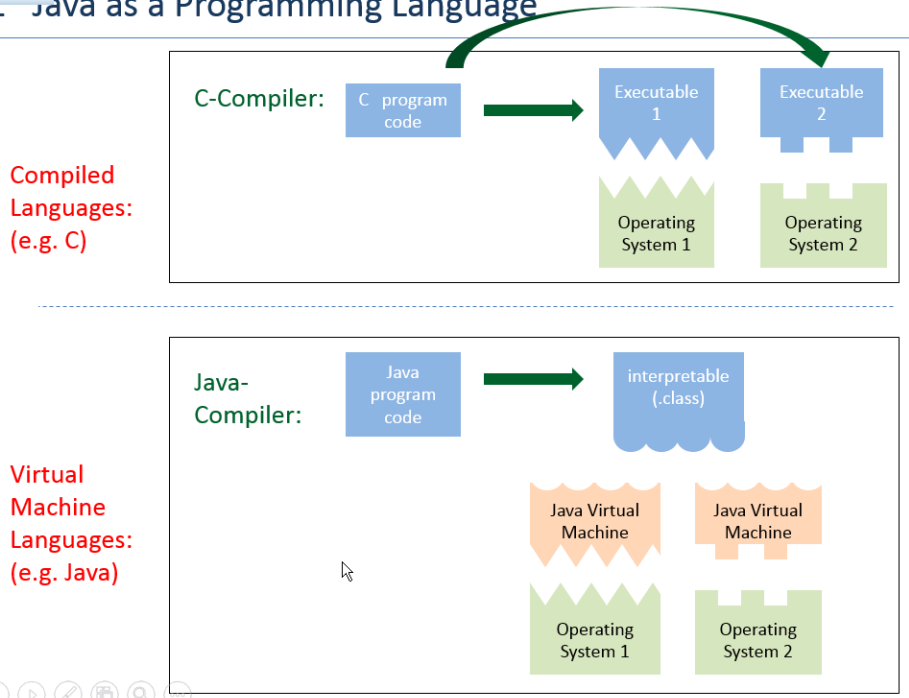


Java as a Programming Language

Programming Languages



Java as a Programming Language



Imperative Programming

- Imperative program:
  Sequence of statements
- Instructions change state (especially memory) of computer system

control flow

```
boolean plato;
int horst;
int heiner;
int fritz;
plato = false;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
```

memory (simplified model)

cell nr	cell name	cell content	
1123	plato	false	data
1124			
1125	horst	101	
1126	heiner	0	
1127			
1128	fritz	0	
4027		boolean plato;	instructions
4028		int horst;	
4029		int heiner;	
4030		int fritz;	
4029		plato = false;	
4030		horst = 101;	

1 Java as a Programming Language

Imperative Programming

- Imperative program:
  Sequence of statements
- Instructions change state (especially memory) of computer system

control flow

```
boolean plato;
int horst;
int heiner;
int fritz;
plato = false;
horst = 101;
heiner = 2;
fritz = horst + heiner;
horst = 2000;
```

memory (simplified model)

cell nr	cell name	cell content	
1123	plato	false	data
1124			
1125	horst	101	
1126	heiner	0	
1127			
1128	fritz	0	
4027		boolean plato;	instructions
4028		int horst;	
4029		int heiner;	
4030		int fritz;	
4029		plato = false;	
4030		horst = 101;	

1 Java as a Programming Language

Procedural Programming

- Group sequences of instructions into named „procedures“ („functions“, „methods“, „sub-routines“ etc.)

$f(x) = (x + x)^2$

```
int doSelfSumSquare(int someNumber){
        int a;
        a = someNumber + someNumber;
        a = a * a;
        return a;
}
```

- Advantages
  - no copying of instruction sequences
  - better testing
  - modularity (e.g. code change inside function doesn‘t affect caller)
  - code re-use
  - etc.

1 Java as a Programming Language

Procedural Programming

- Group sequences of instructions into named „procedures“ („functions“, „methods“, „sub-routines“ etc.)

$f(x) = (x + x)^2$

```
int doSelfSumSquare(int someNumber){
        int a;
        a = someNumber + someNumber;
        a = a * a;
        return a;
}
```

- Advantages
  - no copying of instruction sequences
  - better testing
  - modularity (e.g. code change inside function doesn‘t affect caller)
  - code re-use
  - etc.

Java as a Programming Language

```
int horst;
int heiner;
horst = 101;
heiner = 2;
heiner = doSelfSumSquare(horst);
heiner = doSelfSumSquare(117);
horst = horst + 2;
```

⋮

```
int doSelfSumSquare(int someNumber){
        int a;
        a = someNumber + someNumber;
        a = a * a;
        return a;
}
```

- In the example: Control flow is transferred to function, back to main program, back to function and back to main program

Java as a Programming Language

Object-oriented Programming

- Object-oriented programming:

  Group data and functions into objects ←→
  Models of state and behaviour of **real world** objects
  state: „fields“ ; behaviour: „methods“

- Methods should mainly act on an object's fields
- Classes: Blueprints for objects → Objects: Instances of classes
- Advantages
  - Intuitive models
  - Information hiding
  - Increased modularity, locality etc.
  - Increased code re-use
  - etc.

Java as a Programming Language

Object-oriented Programming

- Object-oriented programming:

  Group data and functions into objects ←→
  Models of state and behaviour of **real world** objects
  state: „fields“ ; behaviour: „methods“

- Methods should mainly act on an object's fields
- Classes: Blueprints for objects → Objects: Instances of classes
- Advantages
  - Intuitive models
  - Information hiding
  - Increased modularity, locality etc.
  - Increased code re-use
  - etc.

Java as a Programming Language

Object-oriented Programming

- Object-oriented programming:

  Group data and functions into objects ←→
  Models of state and behaviour of **real world** objects
  state: „fields“ ; behaviour: „methods“

- Methods should mainly act on an object's fields
- Classes: Blueprints for objects → Objects: Instances of classes
- Advantages
  - Intuitive models
  - Information hiding
  - Increased modularity, locality etc.
  - Increased code re-use
  - etc.

Object-oriented Programming

- Object-oriented programming:

  Group data and functions into objects ←→
  Models of state and behaviour of **real world** objects
  state: „fields“ ; behaviour: „methods“

- Methods should mainly act on an object's fields
- Classes: Blueprints for objects → Objects: Instances of classes
- Advantages
  - Intuitive models
  - Information hiding
  - Increased modularity, locality etc.
  - Increased code re-use
  - etc.

1 Java as a Programming Language

Object-oriented Programming

- Object-oriented programming:

  Group data and functions into objects ←→
  Models of state and behaviour of **real world** objects
  state: „fields“ ; behaviour: „methods“

- Methods should mainly act on an object's fields
- Classes: Blueprints for objects → Objects: Instances of classes
- Advantages
  - Intuitive models
  - Information hiding
  - Increased modularity, locality etc.
  - Increased code re-use
  - etc.

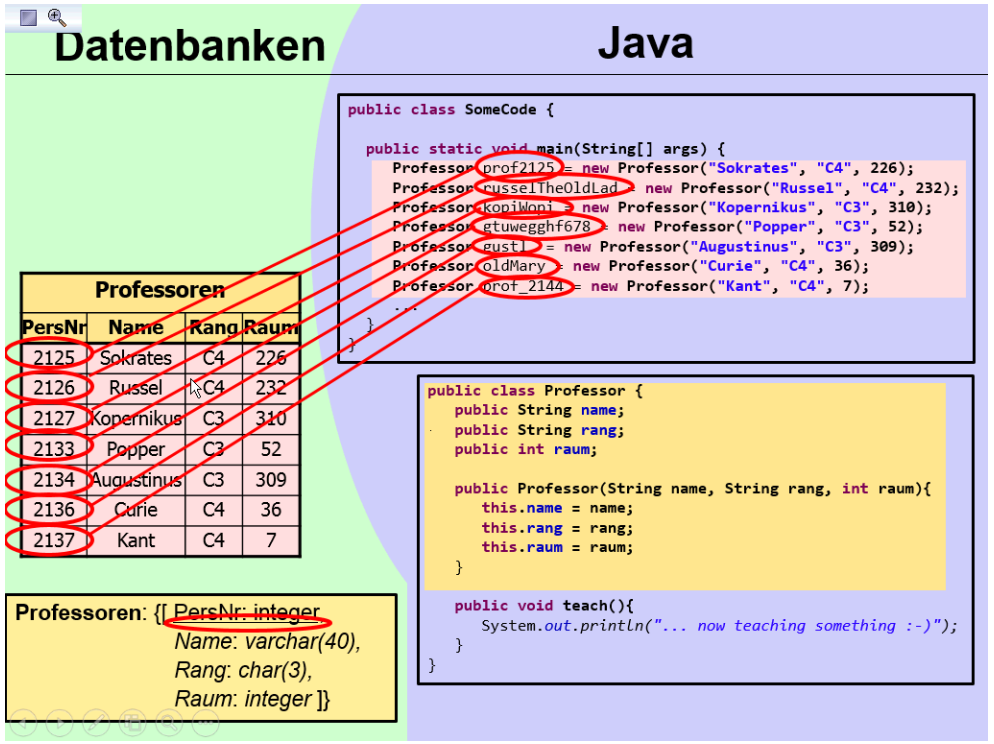
1 Java as a Programming Language

Object-oriented Programming

- Object-oriented programming:

  Group data and functions into objects ←→
  Models of state and behaviour of **real world** objects
  state: „fields“ ; behaviour: „methods“

- Methods should mainly act on an object's fields
- Classes: Blueprints for objects → Objects: Instances of classes
- Advantages
  - Intuitive models
  - Information hiding
  - Increased modularity, locality etc.
  - Increased code re-use
  - etc.



Datenbanken

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Professoren: {[PersNr: integer,
Name: varchar(40),
Rang: char(3),
Raum: integer]}

Java

```
public class SomeCode {

  public static void main(String[] args) {
     Professor prof2125 = new Professor("Sokrates", "C4", 226);
     Professor russelTheOldLad = new Professor("Russel", "C4", 232);
     Professor kopiWopi = new Professor("Kopernikus", "C3", 310);
     Professor gtuwegghf678 = new Professor("Popper", "C3", 52);
     Professor gustl  = new Professor("Augustinus", "C3", 309);
     Professor oldMary = new Professor("Curie", "C4", 36);
     Professor prof_2144 = new Professor("Kant", "C4", 7);
     ...
  }
}
```

```
public class Professor {
   public String name;
   public String rang;
   public int raum;

   public Professor(String name, String rang, int raum){
      this.name = name;
      this.rang = rang;
      this.raum = raum;
   }

   public void teach(){
      System.out.println("... now teaching something :-)");
   }
}
```

Datenbanken

???

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Professoren: {[PersNr: integer,
Name: varchar(40),
Rang: char(3),
Raum: integer]}

Java

```
public class SomeCode {

  public static void main(String[] args) {
     Professor prof2125 = new Professor("Sokrates", "C4", 226);
     Professor russelTheOldLad = new Professor("Russel", "C4", 232);
     Professor kopiWopi = new Professor("Kopernikus", "C3", 310);
     Professor gtuwegghf678 = new Professor("Popper", "C3", 52);
     Professor gustl  = new Professor("Augustinus", "C3", 309);
     Professor oldMary = new Professor("Curie", "C4", 36);
     Professor prof_2144 = new Professor("Kant", "C4", 7);
     ...
  }
}
```

```
public class Professor {
   public String name;
   public String rang;
   public int raum;

   public Professor(String name, String rang, int raum){
      this.name = name;
      this.rang = rang;
      this.raum = raum;
   }

   public void teach(){
      System.out.println("... now teaching something :-)");
   }
}
```

1 Java as a Programming Language

```
class Bicycle {
       int cadence = 0;
       int speed = 0;
       int gear = 1;

       void changeCadence(int newValue) {
            cadence = newValue;
       }

       void changeGear(int newValue) {
            gear = newValue;
       }

       void speedUp(int increment) {
            speed = speed + increment;
       }

       void applyBrakes(int decrement) {
            speed = speed - decrement;
       }
}
```

fields (state)

methods (behaviour)

class

Source: [JTutorial]

1 Java as a Programming Language

```
class Bicycle {
       int cadence = 0;
       int speed = 0;
       int gear = 1;

       void changeCadence(int newValue) {
            cadence = newValue;
       }

       void changeGear(int newValue) {
            gear = newValue;
       }

       void speedUp(int increment) {
            speed = speed + increment;
       }

       void applyBrakes(int decrement) {
            speed = speed - decrement;
       }
}
```

fields (state)

methods (behaviour)

class

Source: [JTutorial]

Java as a Programming Language

```
class BicycleDemo {
     public static void main(String[] args) {
          // Create two different Bicycle objects
          Bicycle bike1 = new Bicycle();
          Bicycle bike2 = new Bicycle();

          // Invoke methods on these objects
          bike1.changeCadence(50);
          bike1.speedUp(10);
          bike1.changeGear(2);

          bike2.changeCadence(50);
          bike2.speedUp(10);
          bike2.changeGear(2);
          bike2.changeCadence(40);
          bike2.speedUp(10);
          bike2.changeGear(3);
       }
}
```

```
class Bicycle {
       int cadence = 0;
       int speed = 0;
       int gear = 1;

       void changeCadence(int newValue) {
            cadence = newValue;
       }

       void changeGear(int newValue) {
            gear = newValue;
       }

       void speedUp(int increment) {
            speed = speed + increment;
       }

       void applyBrakes(int decrement) {
            speed = speed - decrement;
       }
}
```

Source: [JTutorial]

Datenbanken

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Professoren: {[PersNr: integer,
Name: *varchar(40)*,
Rang: *char(3)*,
Raum: *integer*]}

Java

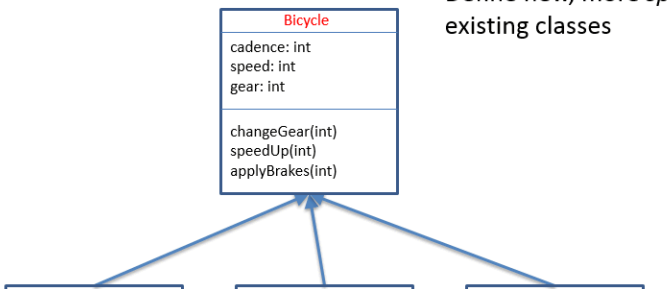
```
public class SomeCode {

  public static void main(String[] args) {
     Professor prof2125 = new Professor("Sokrates", "C4", 226);
     Professor russelTheOldLad = new Professor("Russel", "C4", 232);
     Professor kopiWopi = new Professor("Kopernikus", "C3", 310);
     Professor gtuwegghf678 = new Professor("Popper", "C3", 52);
     Professor gustl  = new Professor("Augustinus", "C3", 309);
     Professor oldMary = new Professor("Curie", "C4", 36);
     Professor prof_2144 = new Professor("Kant", "C4", 7);
     ...
  }
}
```

```
public class Professor {
   public String name;
   public String rang;
   public int raum;

   public Professor(String name, String rang, int raum){
      this.name = name;
      this.rang = rang;
      this.raum = raum;
   }

   public void teach(){
      System.out.println("... now teaching something :-)");
   }
}
```

Java as a Programming Language

Inheritance

Inheritance:
Define new, more *specialized* classes from existing classes

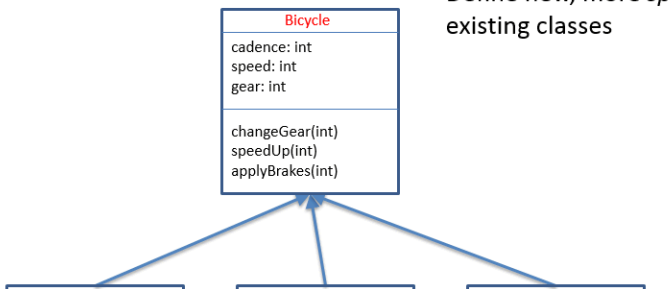



```
class RoadBike extends Bicycle {
    // additional fields and
methods
    // that define a road bike
    // go here
}
```

Java as a Programming Language

Inheritance

Inheritance:
Define new, more *specialized* classes from existing classes




```
class RoadBike extends Bicycle {
    // additional fields and
methods
    // that define a road bike
    // go here
}
```

Java as a Programming Language

```
class BicycleDemo {
     public static void main(String[] args) {
          // Create two different Bicycle objects
          Bicycle bike1 = new Bicycle();
          Bicycle bike2 = new Bicycle();

          // Invoke methods on these objects
          bike1.changeCadence(50);
          bike1.speedUp(10);
          bike1.changeGear(2);

          bike2.changeCadence(50);
          bike2.speedUp(10);
          bike2.changeGear(2);
          bike2.changeCadence(40);
          bike2.speedUp(10);
          bike2.changeGear(3);
       }
}
```

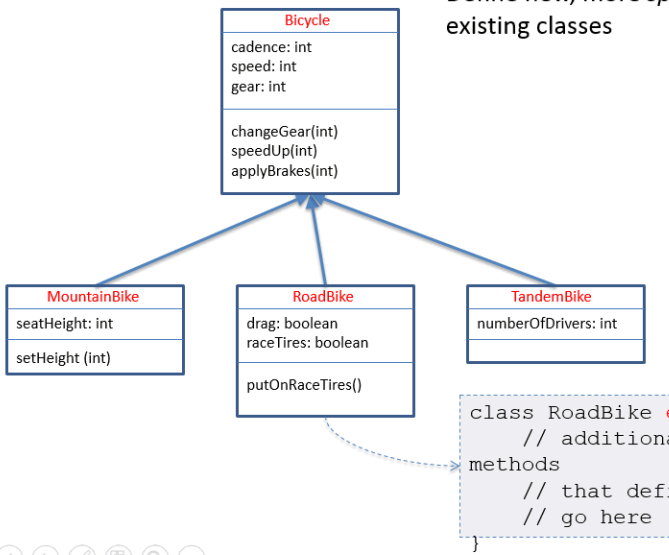
```
class Bicycle {
       int cadence = 0;
       int speed = 0;
       int gear = 1;

       void changeCadence(int newValue) {
            cadence = newValue;
       }

       void changeGear(int newValue) {
            gear = newValue;
       }

       void speedUp(int increment) {
            speed = speed + increment;
       }

       void applyBrakes(int decrement) {
            speed = speed - decrement;
       }
}
```

Source: [JTutorial]

Java as a Programming Language

Inheritance

Inheritance:
Define new, more *specialized* classes from existing classes




```
class RoadBike extends Bicycle {
    // additional fields and
methods
    // that define a road bike
    // go here
}
```

Java as a Programming Language

Interfaces

Interface:
Specify in an abstract way what a class implementing that interface should exhibit as behaviours (create blueprint for blueprints)

```
interface IBicycle {
       void changeCadence(int newValue);

       void changeGear(int newValue);

       void speedUp(int increment);

       void applyBrakes(int decrement);
}
```

```
class Bicycle implements IBicycle {
   // remainder of this class implemented as before
   // except that above methods must be public
}
```

see: [JTutorial]

Java as a Programming Language

Interfaces

Interface:
Specify in an abstract way what a class implementing that interface should exhibit as behaviours (create blueprint for blueprints)

```
interface IBicycle {
       void changeCadence(int newValue);

       void changeGear(int newValue);

       void speedUp(int increment);

       void applyBrakes(int decrement);
}
```

```
class Bicycle implements IBicycle {
   // remainder of this class implemented as before
   // except that above methods must be public
}
```

see: [JTutorial]

Interfaces

Interface:
Specify in an abstract way what a class implementing that interface should exhibit as behaviours (create blueprint for blueprints)

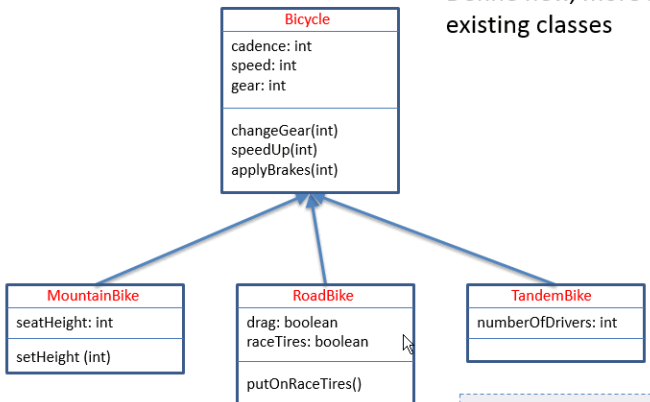
```
interface IBicycle {
       void changeCadence(int newValue);

       void changeGear(int newValue);

       void speedUp(int increment);

       void applyBrakes(int decrement);
}
```

```
class Bicycle implements IBicycle {
   // remainder of this class implemented as before
   // except that above methods must be public
}
```

see: [JTutorial]

Inheritance

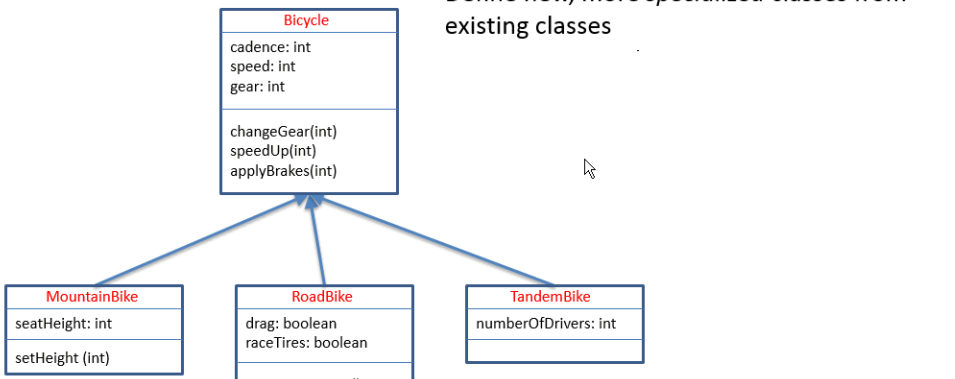
Inheritance:
Define new, more *specialized* classes from existing classes




```
class RoadBike extends Bicycle {
    // additional fields and
methods
    // that define a road bike
    // go here
}
```


Inheritance

Inheritance:
Define new, more *specialized* classes from existing classes




```
class RoadBike extends Bicycle {
    // additional fields and
methods
    // that define a road bike
    // go here
}
```


Interfaces

Interface:
Specify in an abstract way what a class implementing that interface should exhibit as behaviours (create blueprint for blueprints)

```
interface IBicycle {
       void changeCadence(int newValue);

       void changeGear(int newValue);

       void speedUp(int increment);

       void applyBrakes(int decrement);
}
```

```
class Bicycle implements IBicycle {
   // remainder of this class implemented as before
   // except that above methods must be public
}
```

see: [JTutorial]

Interfaces

Interface:

Specify in an abstract way what a class implementing that interface should exhibit as behaviours (create blueprint for blueprints)

```
interface IBicycle {
       void changeCadence(int newValue);

       void changeGear(int newValue);

       void speedUp(int increment);

       void applyBrakes(int decrement);
}
```

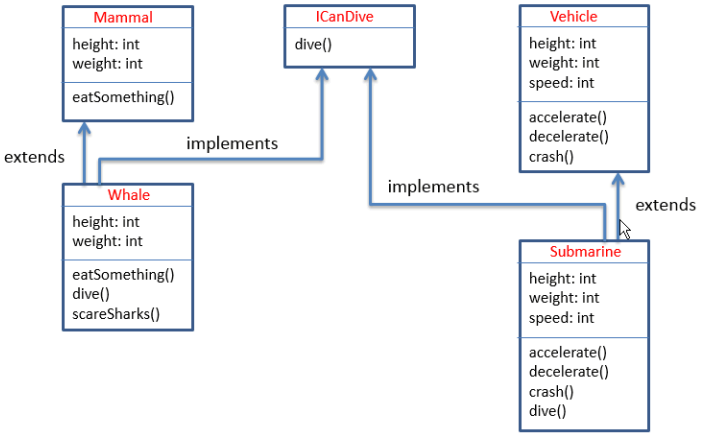
```
class Bicycle implements IBicycle {
   // remainder of this class implemented as before
   // except that above methods must be public
}
```

see: [JTutorial]

1 Java as a Programming Language

Interfaces

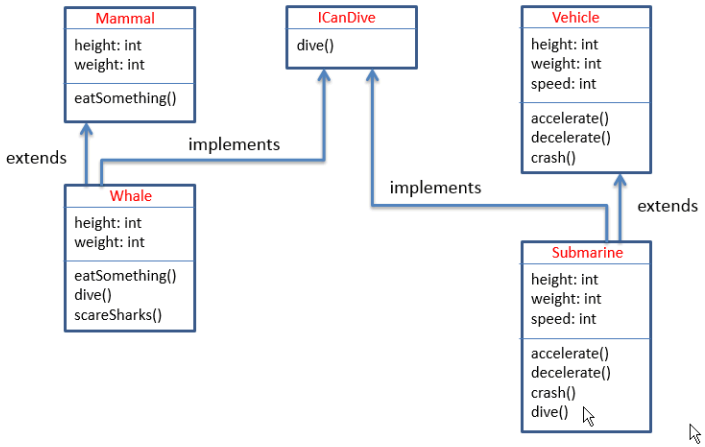
Example:

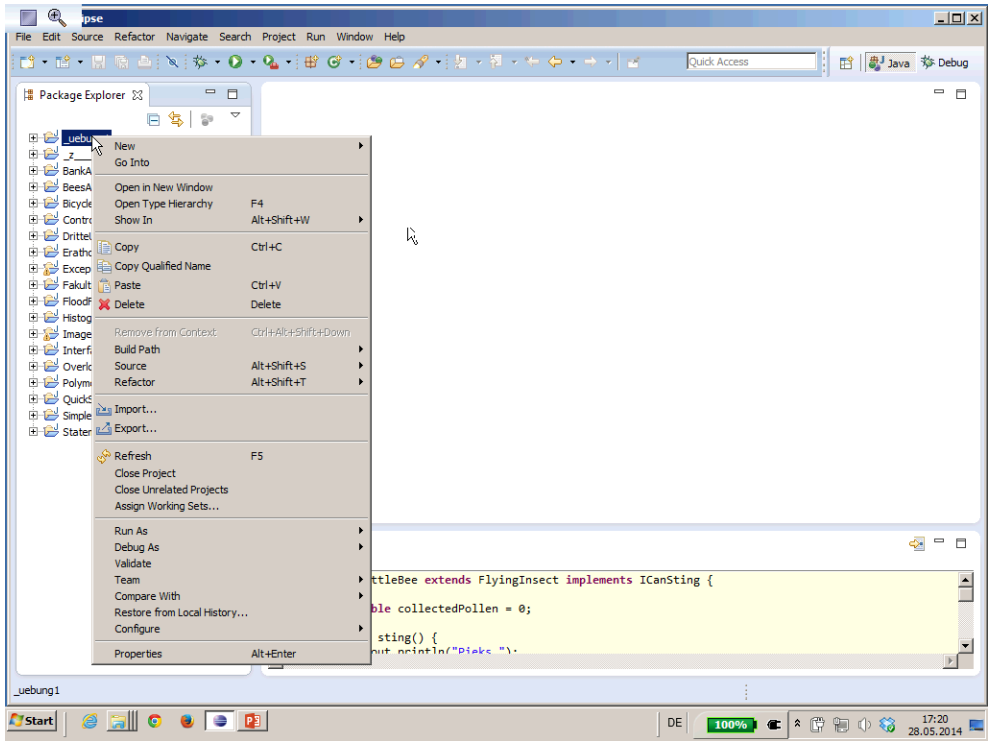


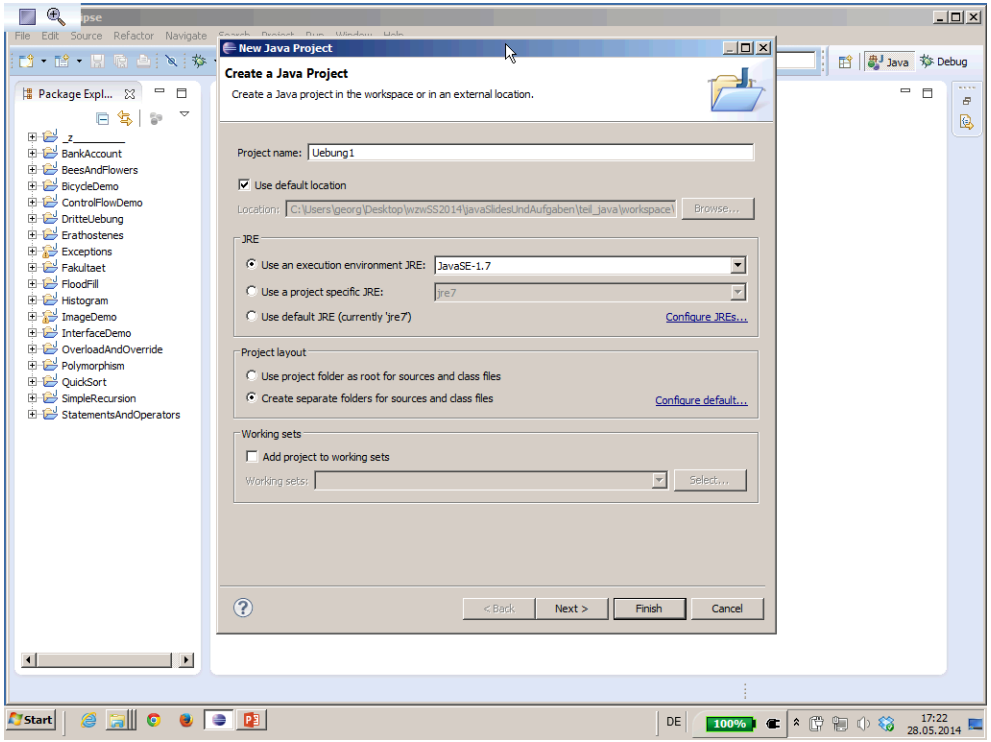
1 Java as a Programming Language

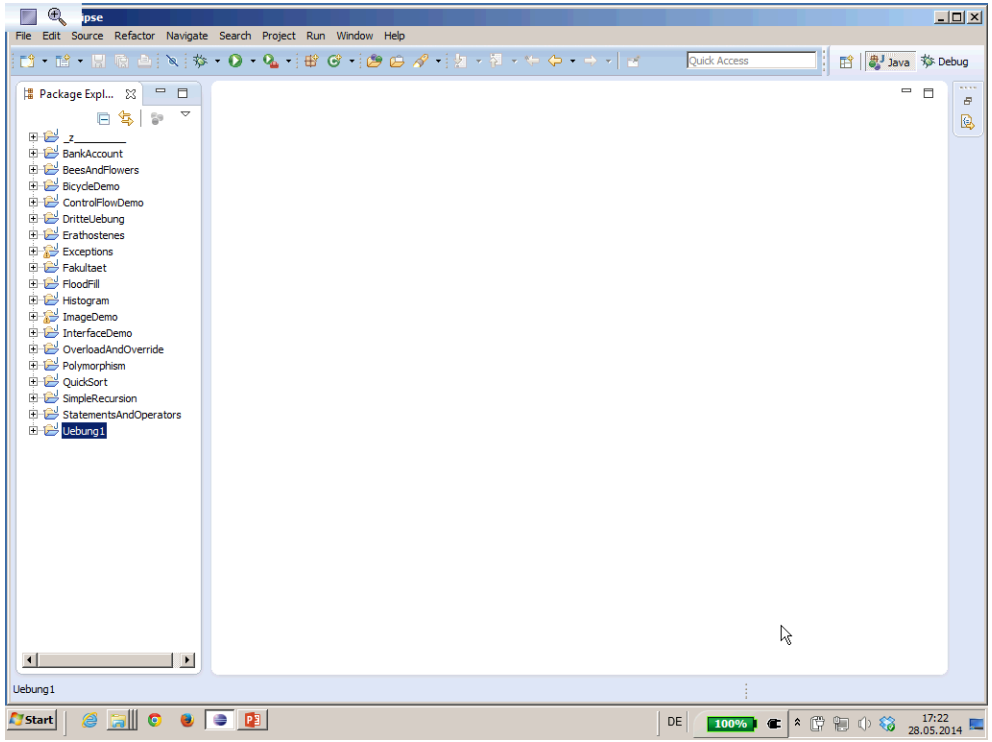
Interfaces

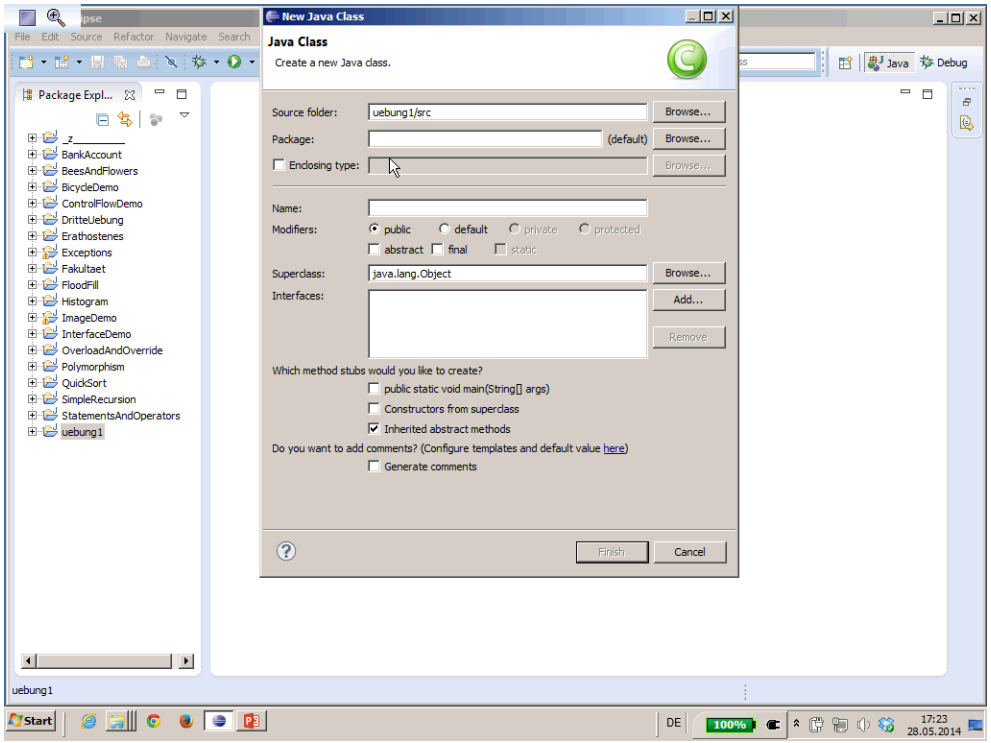
Example:

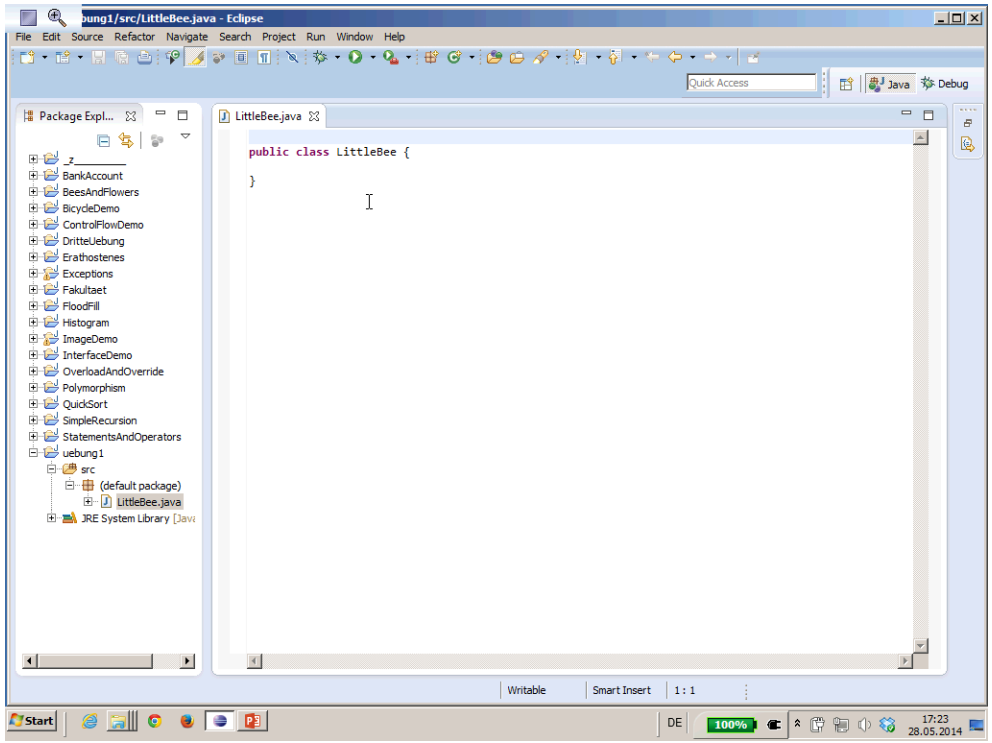


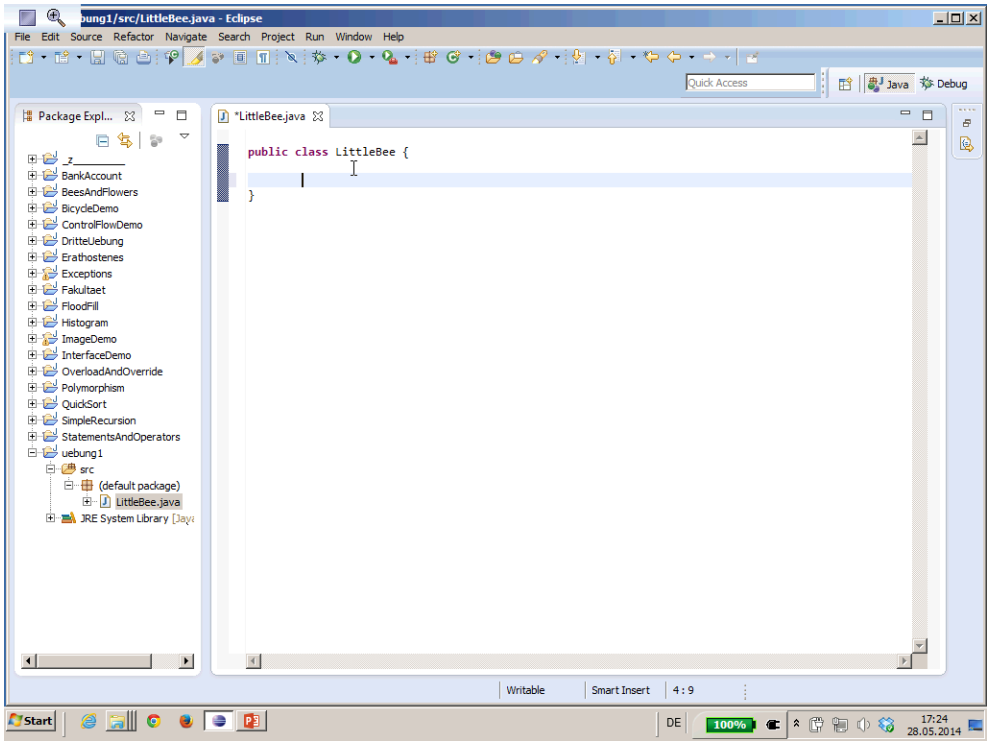


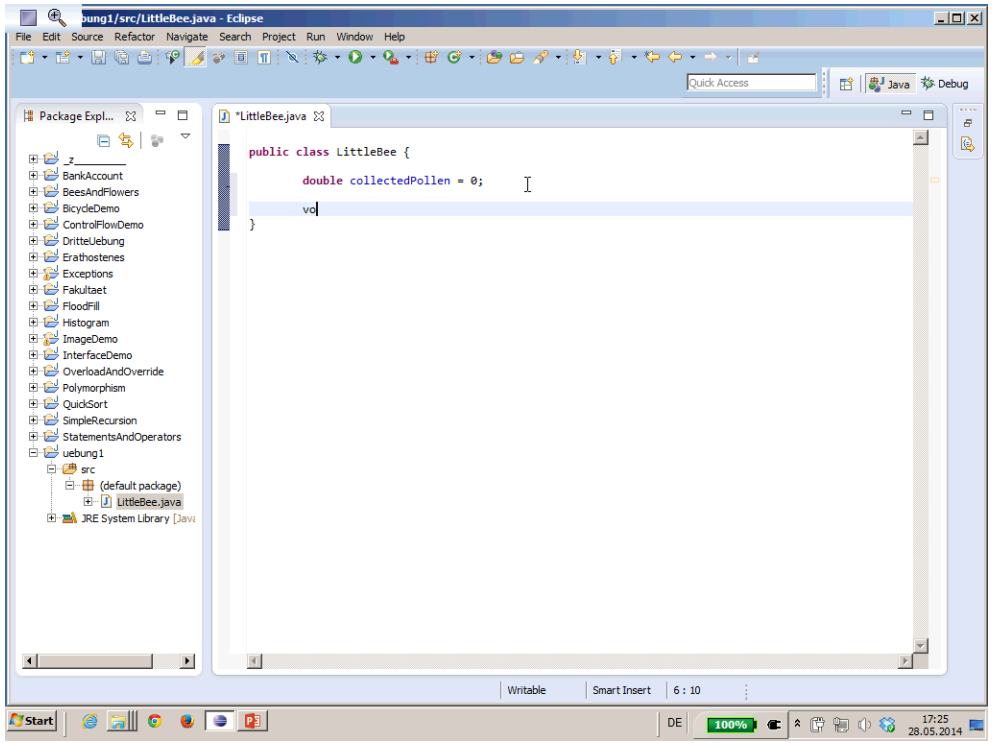

New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: Uebung1
Use default location
Location: C:\Users\georg\Desktop\wzwSS2014\javaSlidesUndAufgaben\teil_java\workspace\
Browse...
JRE
Use an execution environment JRE: JavaSE-1.7
Use a project specific JRE: jre7
Use default JRE (currently 'jre7')
Configure JREs...
Project layout
Use project folder as root for sources and class files
Create separate folders for sources and class files
Configure default...
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

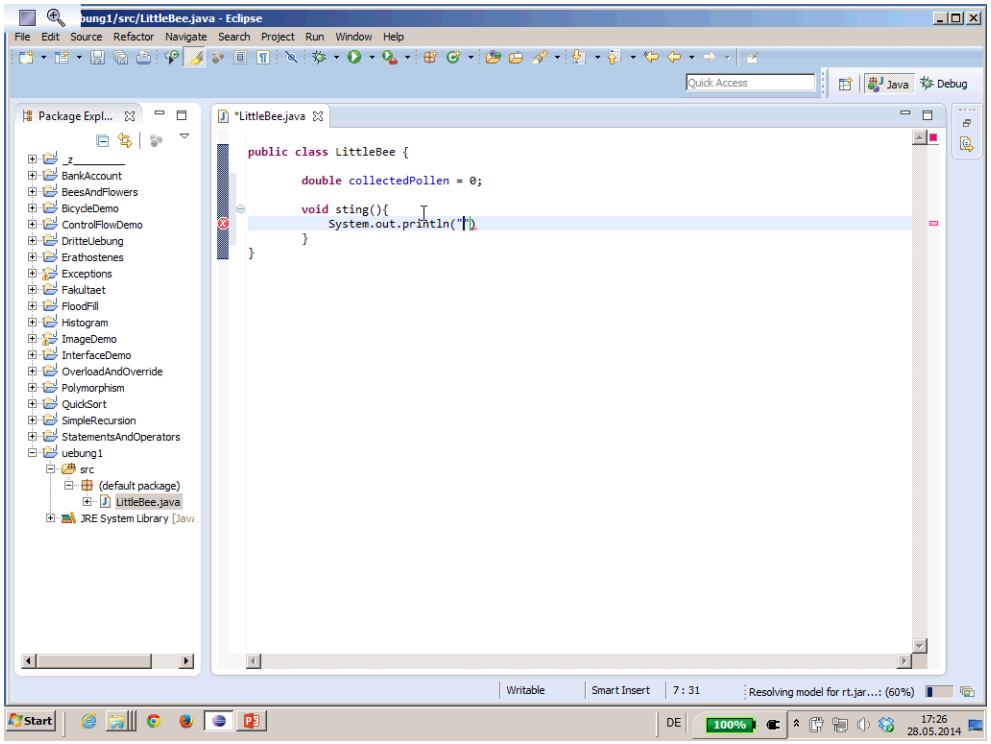

Uebung1

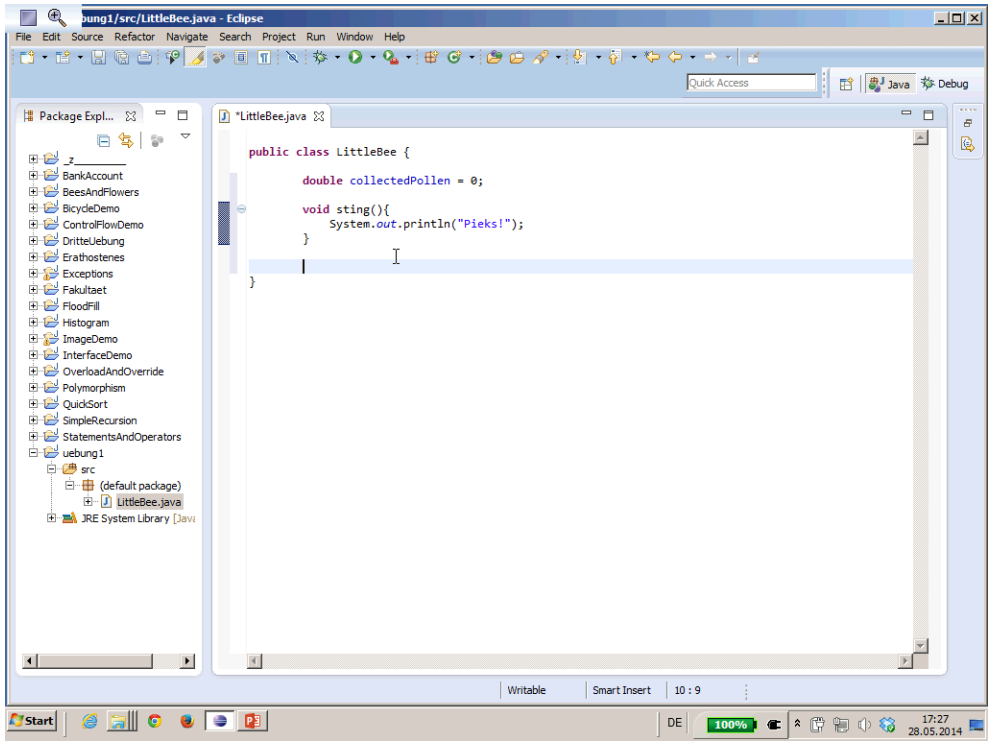

New Java Class
Java Class
Create a new Java class.
Source folder: uebung1/src
Package:
(default)
Enclosing type:
Name:
Modifiers:
public
default
private
protected
abstract
final
static
Superclass: java.lang.Object
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

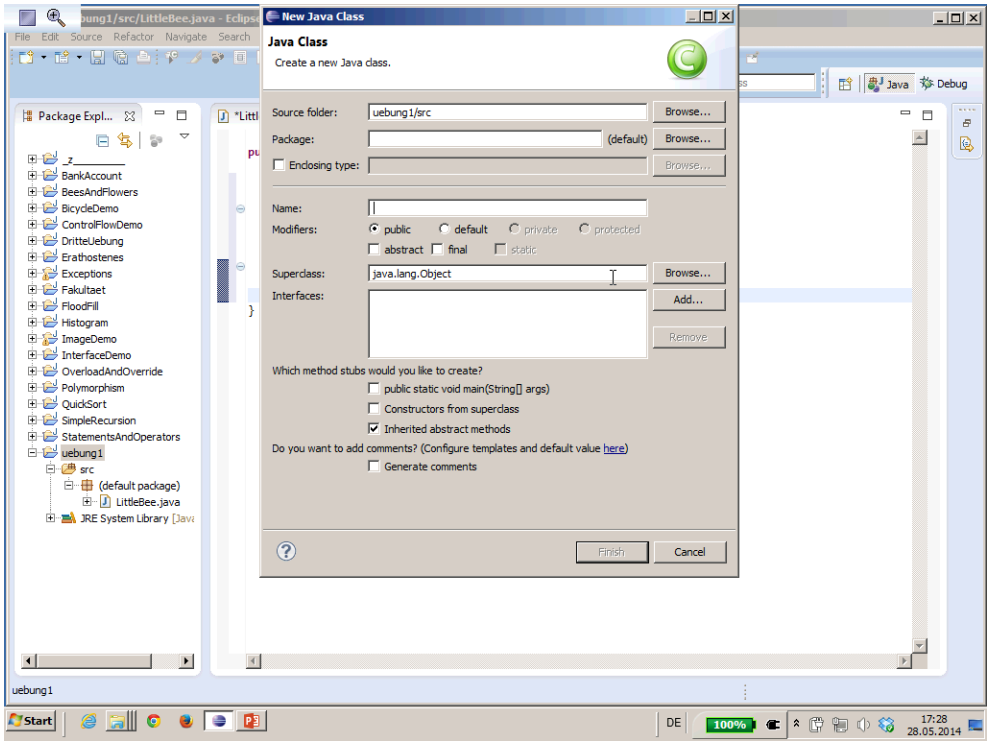

LittleBee.java
public class LittleBee {
}
Writable
Smart Insert
1 : 1

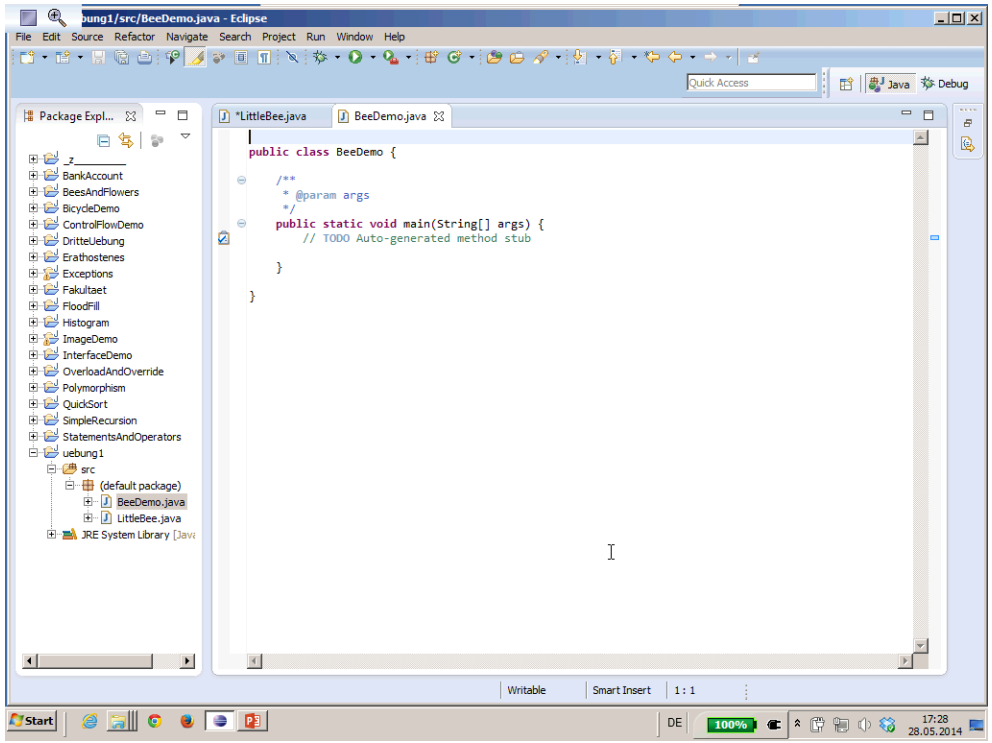

bung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
*LittleBee.java
public class LittleBee {
}
BankAccount
BeesAndFlowers
BicycleDemo
ControlFlowDemo
DritteUebung
Erathostenes
Exceptions
Fakultaet
FloodFill
Histogram
ImageDemo
InterfaceDemo
OverloadAndOverride
Polymorphism
QuickSort
SimpleRecursion
StatementsAndOperators
uebung1
src
(default package)
LittleBee.java
JRE System Library
Writable
Smart Insert
4 : 9
Start
DE
100%
17:24
28.05.2014

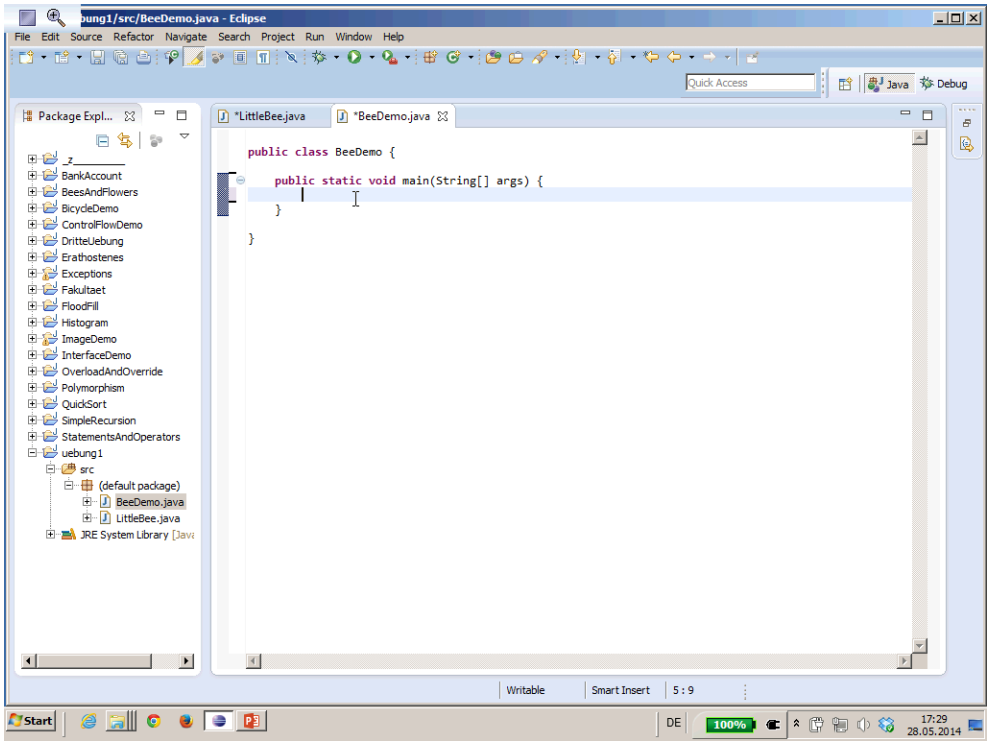

bung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
*LittleBee.java
public class LittleBee {
double collectedPollen = 0;
vo
}
BankAccount
BeesAndFlowers
BicycleDemo
ControlFlowDemo
DritteUebung
Erathostenes
Exceptions
Fakultaet
FloodFill
Histogram
ImageDemo
InterfaceDemo
OverloadAndOverride
Polymorphism
QuickSort
SimpleRecursion
StatementsAndOperators
uebung1
src
(default package)
LittleBee.java
JRE System Library
Writable
Smart Insert
6 : 10
Start
DE
100%
17:25
28.05.2014

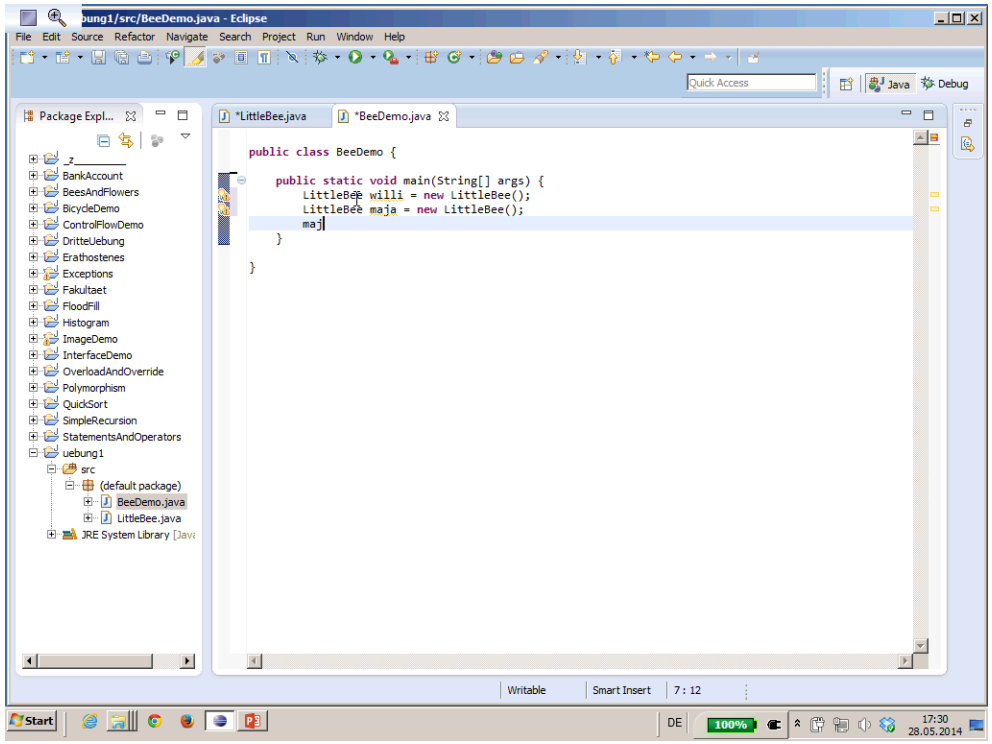

bung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
*LittleBee.java
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("
}
}
BankAccount
BeesAndFlowers
BicycleDemo
ControlFlowDemo
DritteUebung
Erathostenes
Exceptions
Fakultaet
FloodFill
Histogram
ImageDemo
InterfaceDemo
OverloadAndOverride
Polymorphism
QuickSort
SimpleRecursion
StatementsAndOperators
uebung1
src
(default package)
LittleBee.java
JRE System Library
Writable
Smart Insert
7 : 31
Resolving model for rt.jar...: (60%)
Start
DE
100%
17:26
28.05.2014

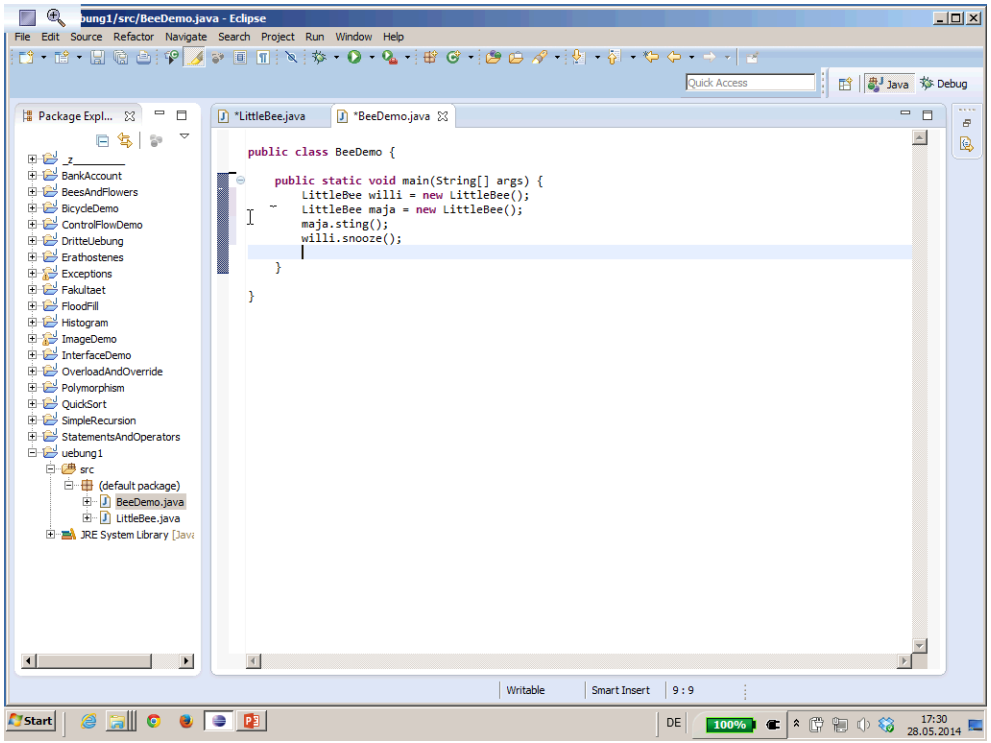

bung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
*LittleBee.java
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
}
BankAccount
BeesAndFlowers
BicycleDemo
ControlFlowDemo
DritteUebung
Erathostenes
Exceptions
Fakultaet
FloodFill
Histogram
ImageDemo
InterfaceDemo
OverloadAndOverride
Polymorphism
QuickSort
SimpleRecursion
StatementsAndOperators
uebung1
src
(default package)
LittleBee.java
JRE System Library
Writable
Smart Insert
10 : 9
Start
DE
100%
17:27
28.05.2014

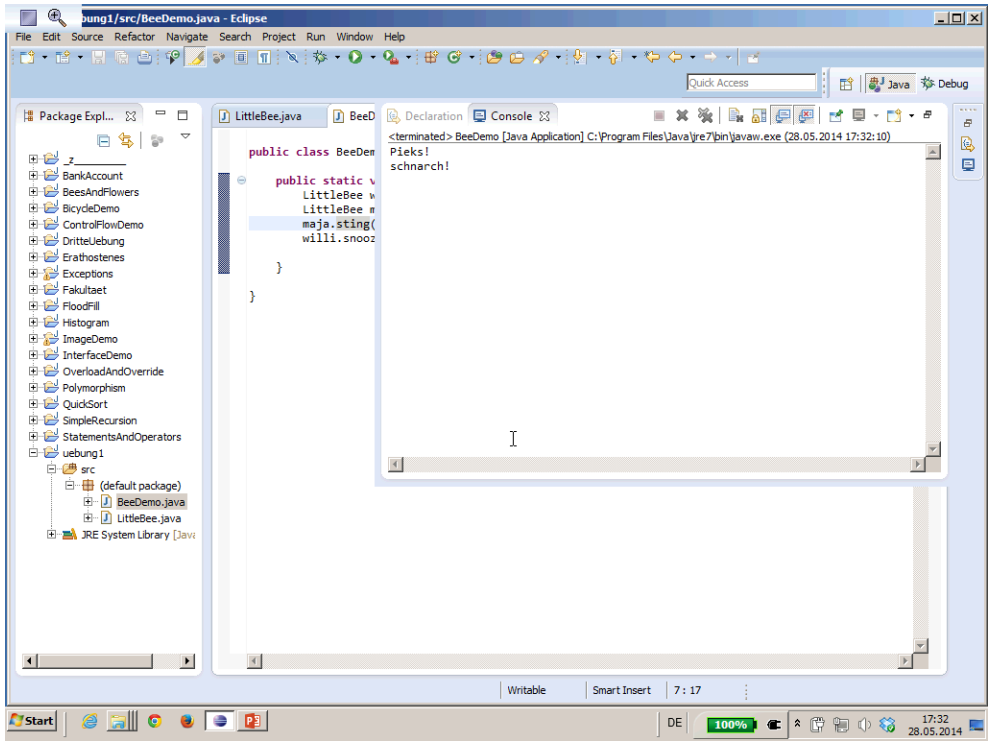

New Java Class
Java Class
Create a new Java class.
Source folder: uebung1/src
Package:
Enclosing type:
Name:
Modifiers: public default private protected
abstract final static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

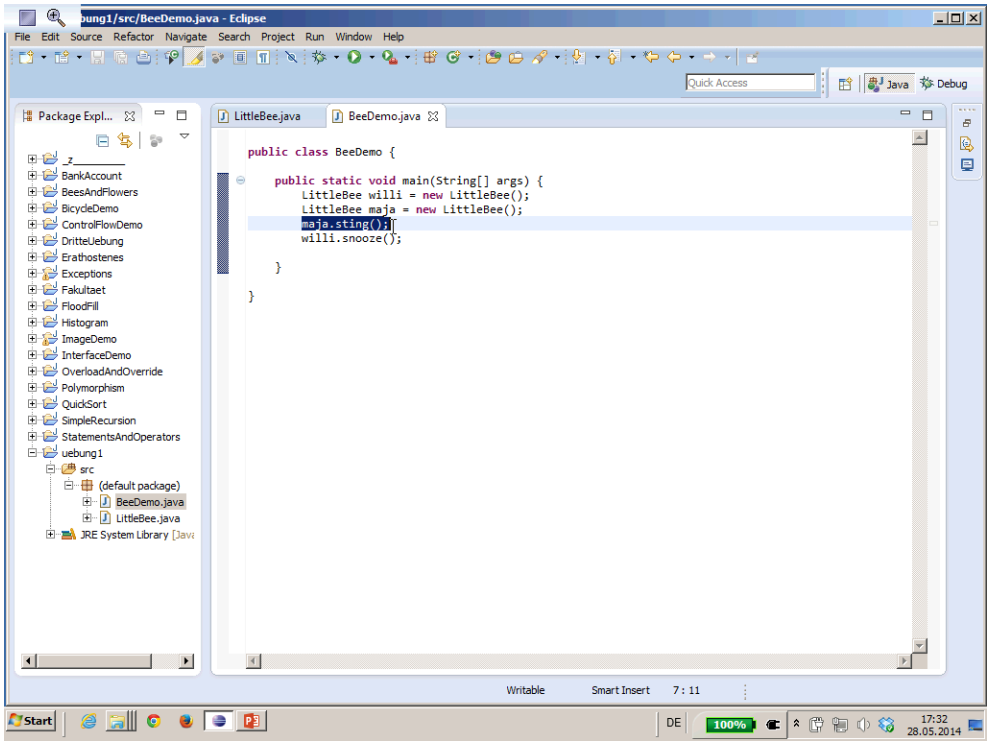

public class BeeDemo {
/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub
}
}

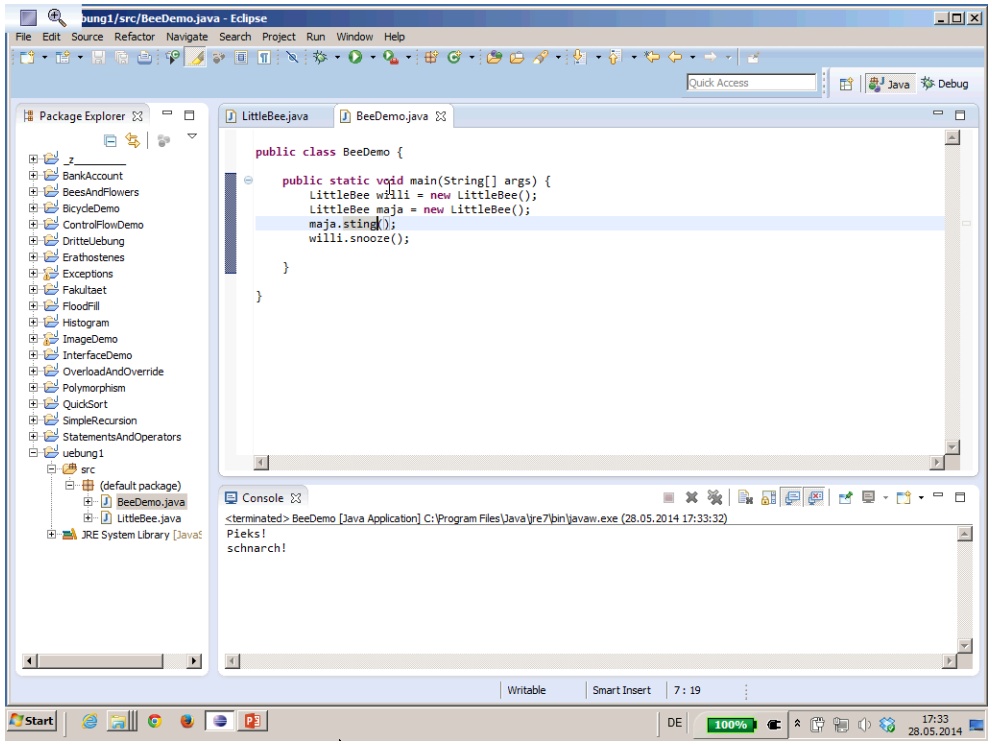

public class BeeDemo {
public static void main(String[] args) {
}
}

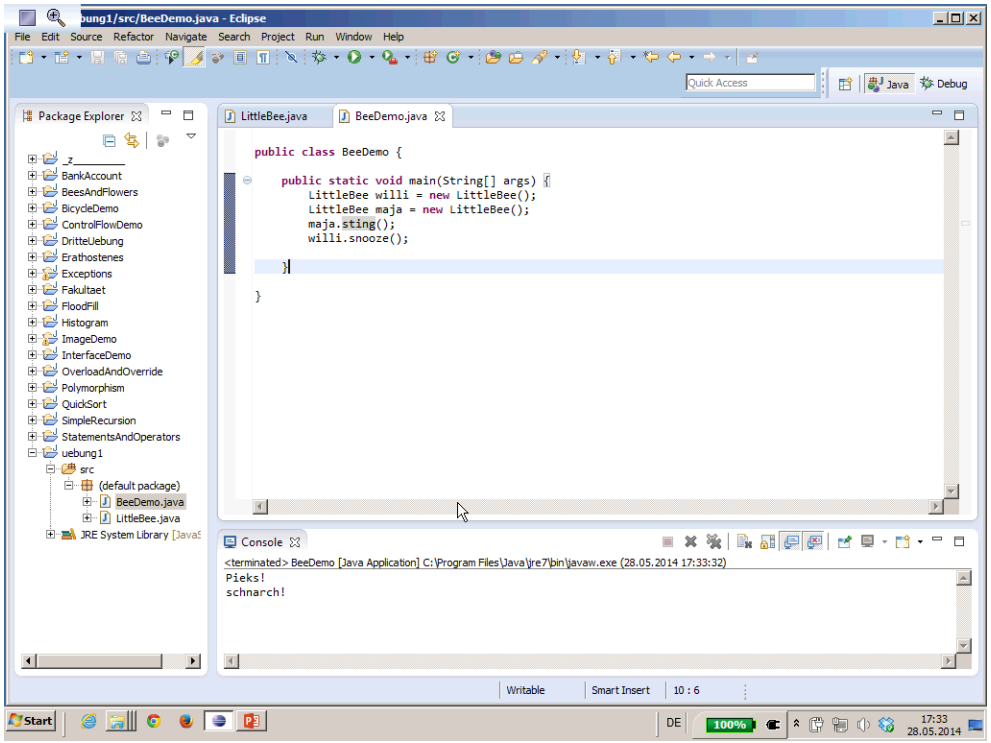

public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maj
}
}

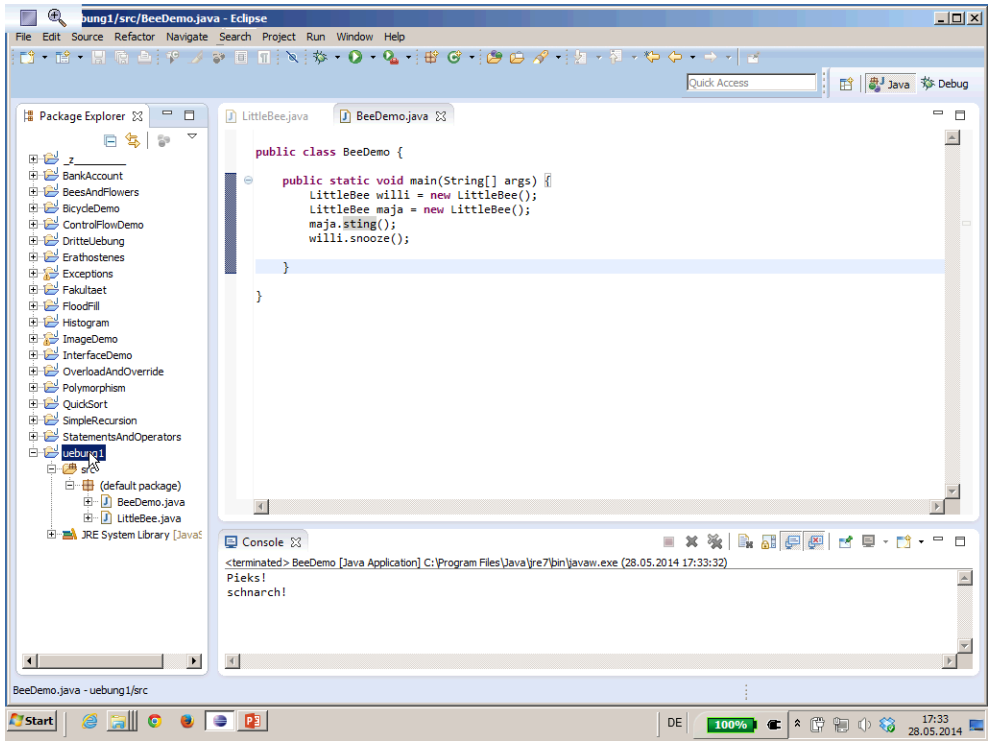

bung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
*LittleBee.java
*BeeDemo.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
}
}
BankAccount
BeesAndFlowers
BicycleDemo
ControlFlowDemo
DritteUebung
Erathostenes
Exceptions
Fakultaet
FloodFill
Histogram
ImageDemo
InterfaceDemo
OverloadAndOverride
Polymorphism
QuickSort
SimpleRecursion
StatementsAndOperators
uebung1
src
(default package)
BeeDemo.java
LittleBee.java
JRE System Library
Writable
Smart Insert
9 : 9
Start
DE
100%
17:30
28.05.2014

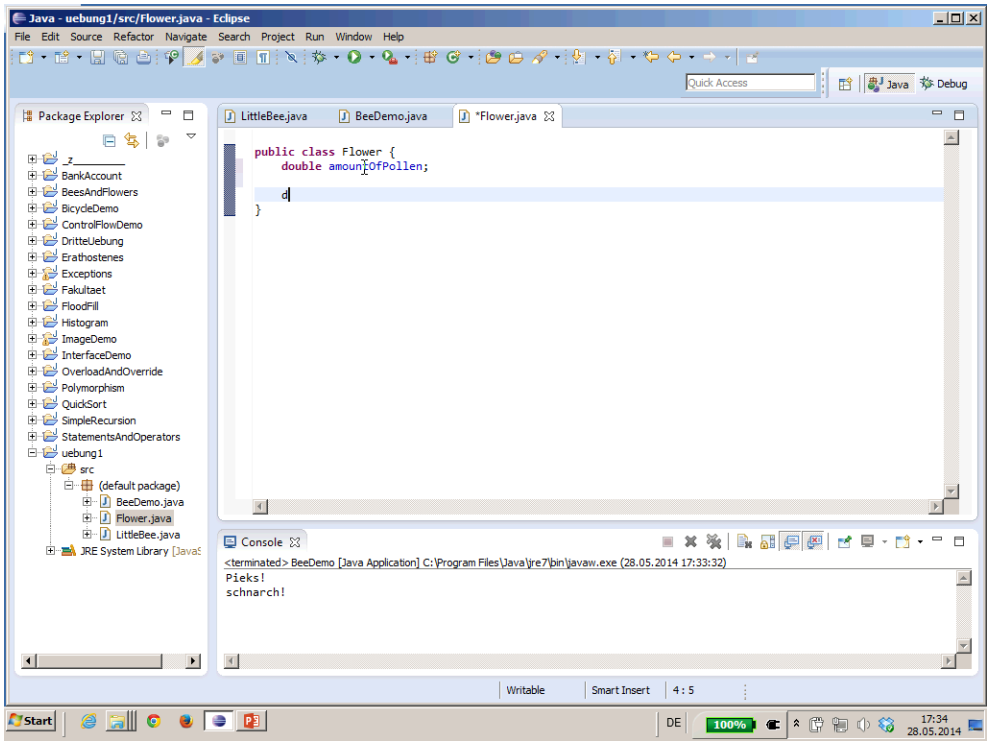

bung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
LittleBee.java
Declaration
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:32:10)
Pieks!
schnarch!
Writable
Smart Insert
7 : 17
Start
DE
100%
17:32
28.05.2014

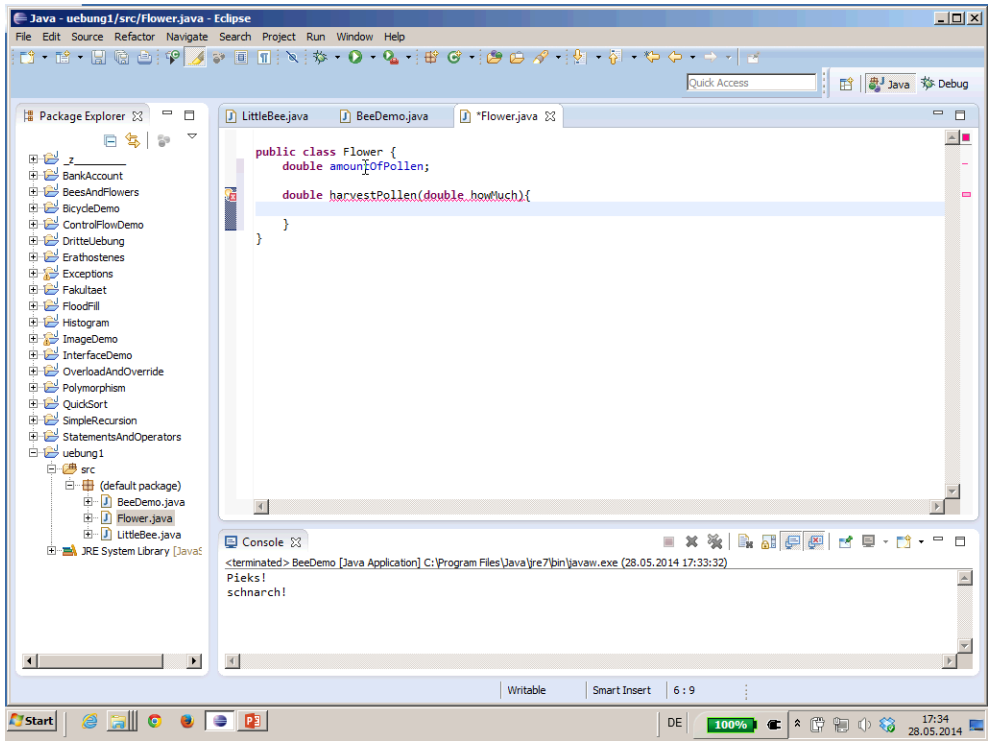

bung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Expl...
LittleBee.java
BeeDemo.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
}
}
Writable
Smart Insert
7 : 11
Start
DE
100%
17:32
28.05.2014

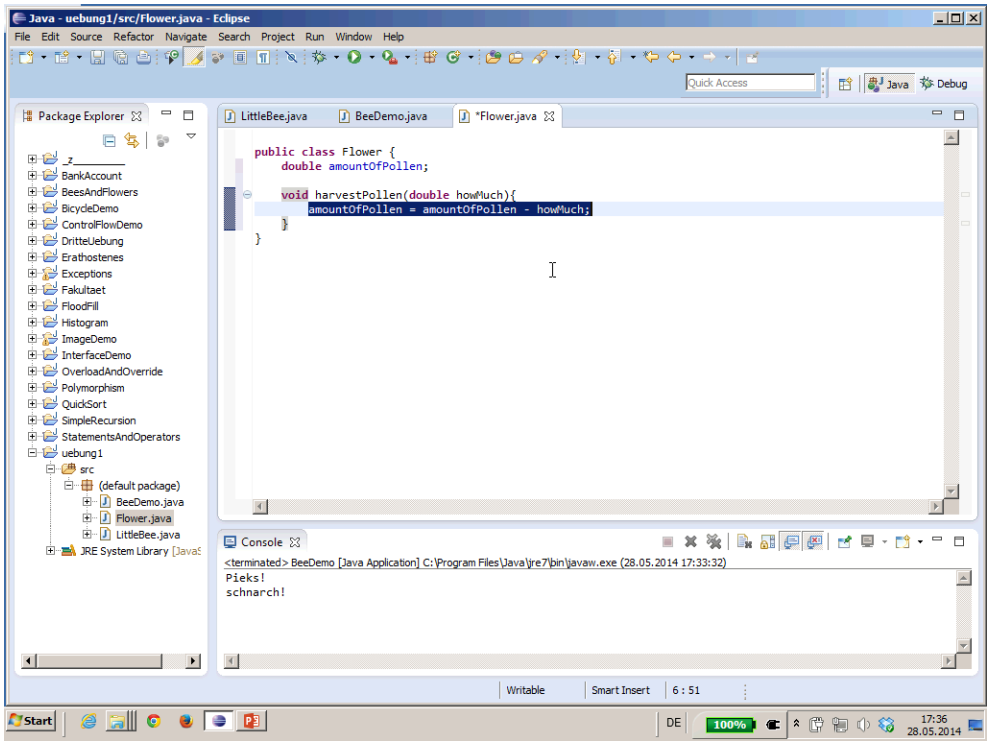

bung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
LittleBee.java
BeeDemo.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
7 : 19
Start
DE
100%
17:33
28.05.2014

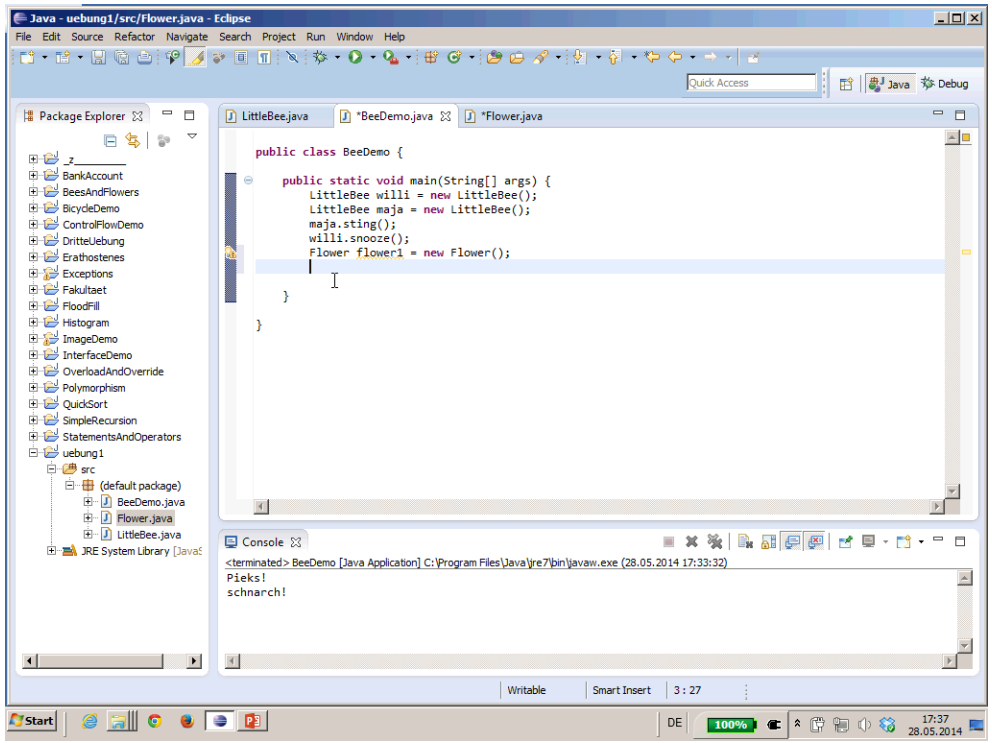

bung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
LittleBee.java
BeeDemo.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
10 : 6
Start
DE
100%
17:33
28.05.2014

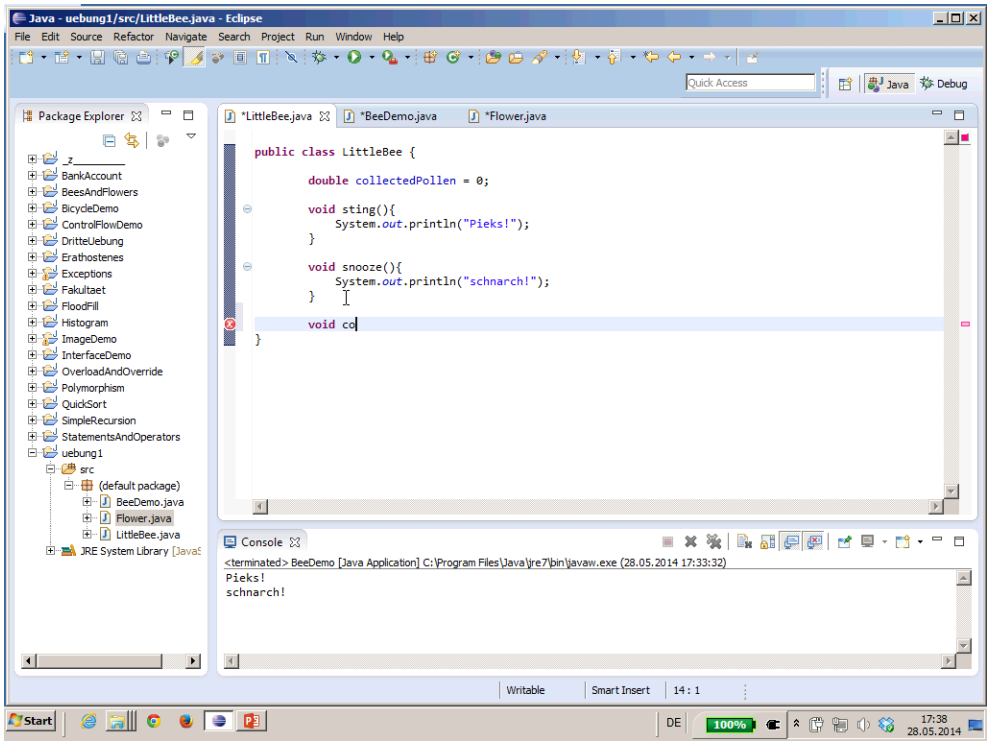

bung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
LittleBee.java
BeeDemo.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
BeeDemo.java - uebung1/src
Start
DE
100%
17:33
28.05.2014

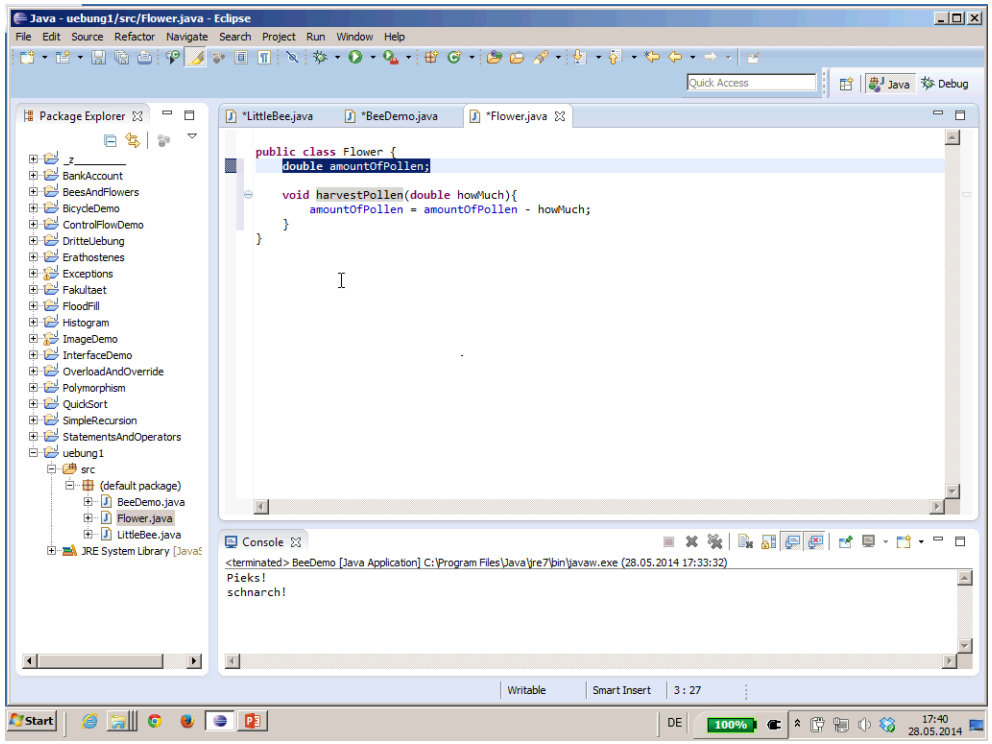

Java - uebung1/src/Flower.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
LittleBee.java
BeeDemo.java
*Flower.java
public class Flower {
double amountOfPollen;
d
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
4 : 5
Start
DE
100%
17:34
28.05.2014

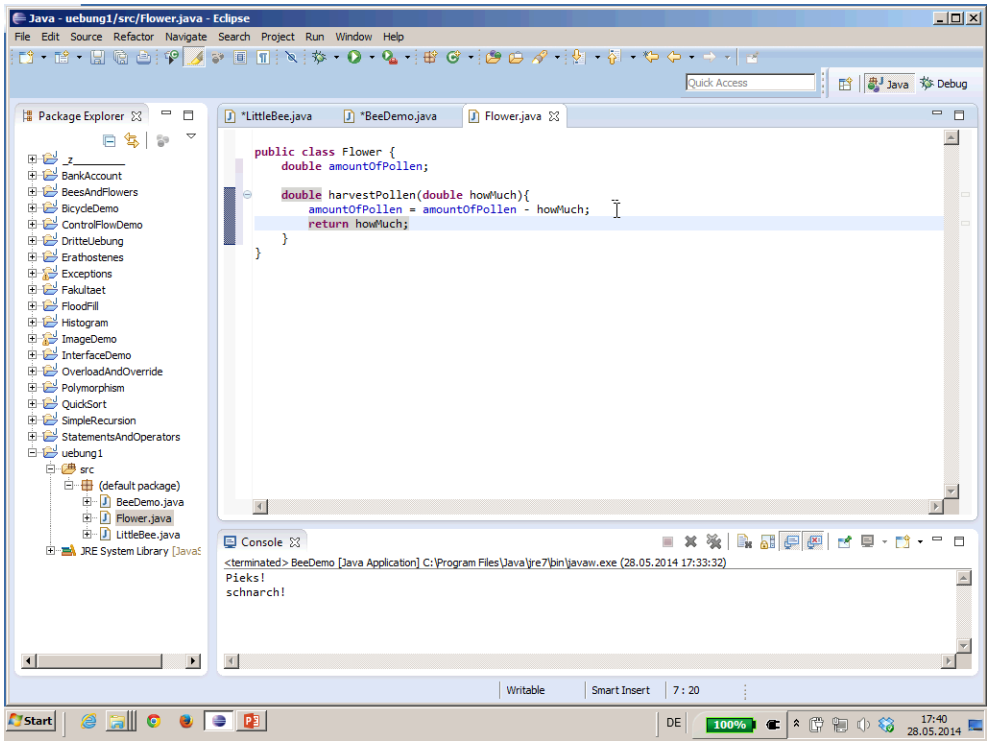

Java - uebung1/src/Flower.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Debug
Package Explorer
LittleBee.java
BeeDemo.java
*Flower.java
public class Flower {
double amountOfPollen;
double harvestPollen(double howMuch){
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
6 : 9
Start
DE
100%
17:34
28.05.2014

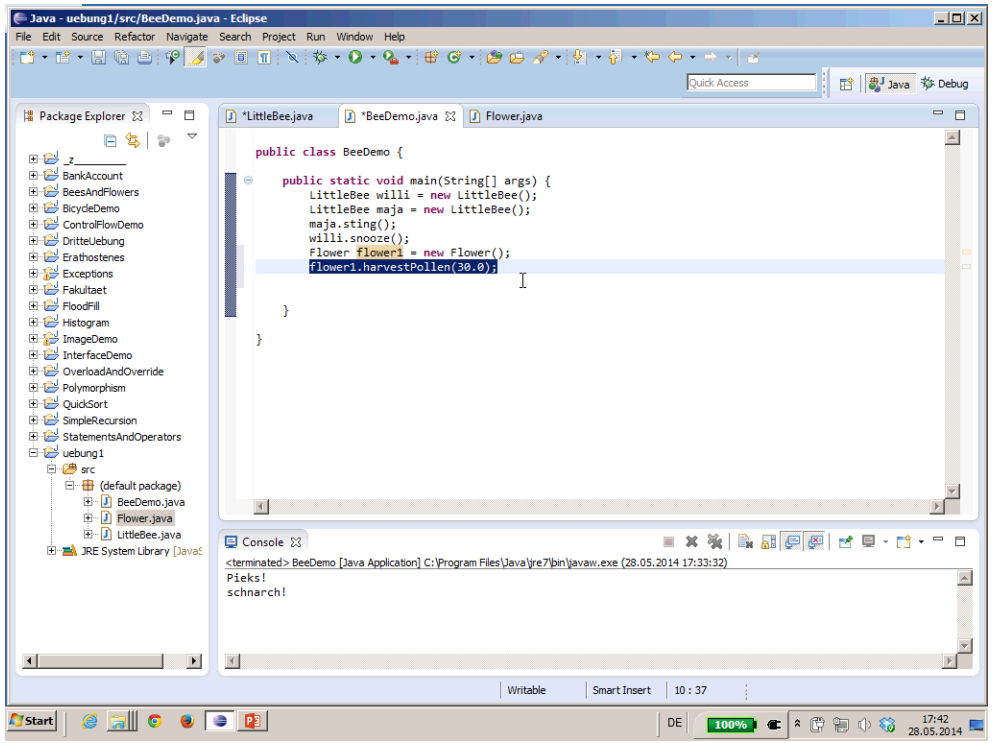

Java - uebung1/src/Flower.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
*Flower.java
public class Flower {
double amountOfPollen;
void harvestPollen(double howMuch){
amountOfPollen = amountOfPollen - howMuch;
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
6 : 51

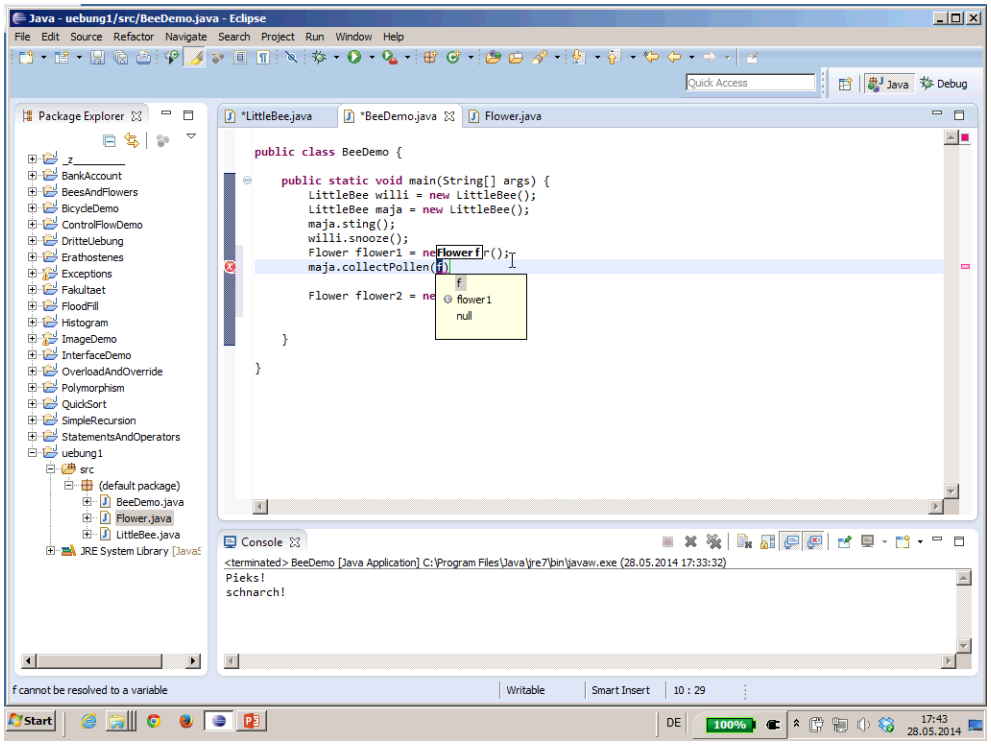

Java - uebung1/src/Flower.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
*BeeDemo.java
*Flower.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
3 : 27

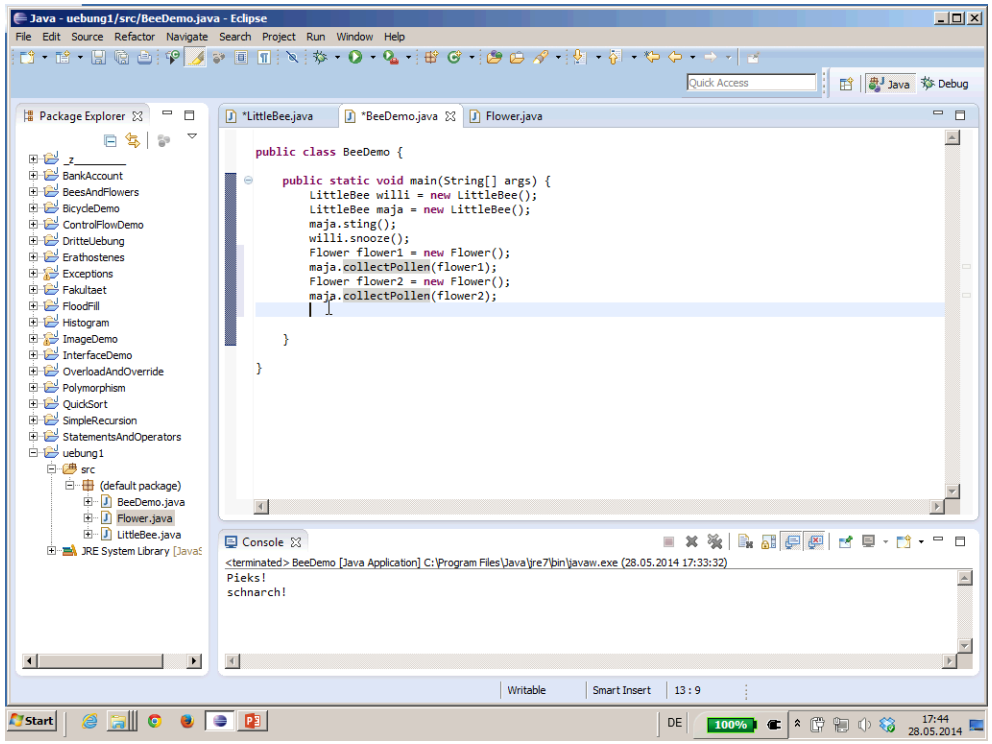

Java - uebung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
*Flower.java
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void co
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
14 : 1

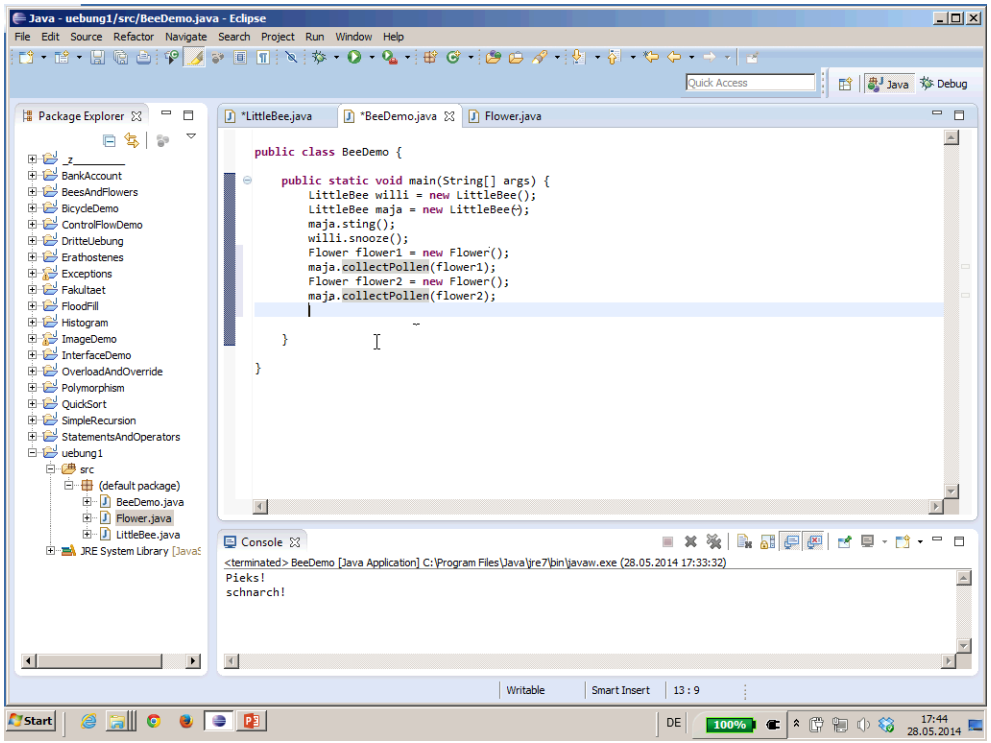

Java - uebung1/src/Flower.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
*Flower.java
public class Flower {
double amountOfPollen;
void harvestPollen(double howMuch){
amountOfPollen = amountOfPollen - howMuch;
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
3 : 27

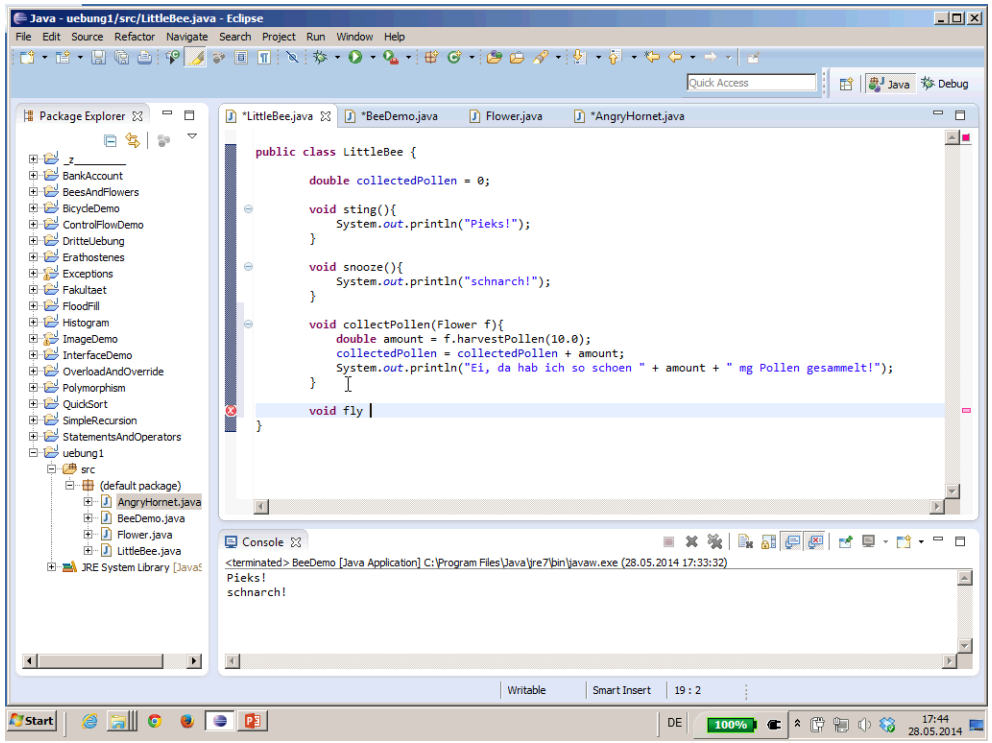

Java - uebung1/src/Flower.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
public class Flower {
double amountOfPollen;
double harvestPollen(double howMuch){
amountOfPollen = amountOfPollen - howMuch;
return howMuch;
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
7 : 20

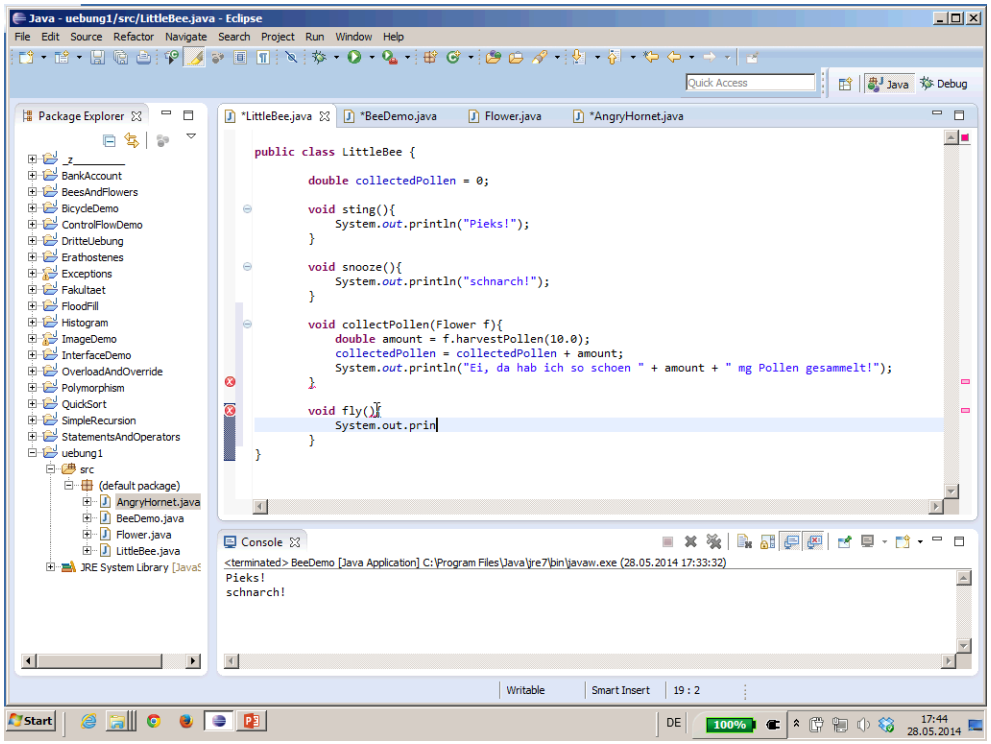

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
flower1.harvestPollen(30.0);
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
10 : 37

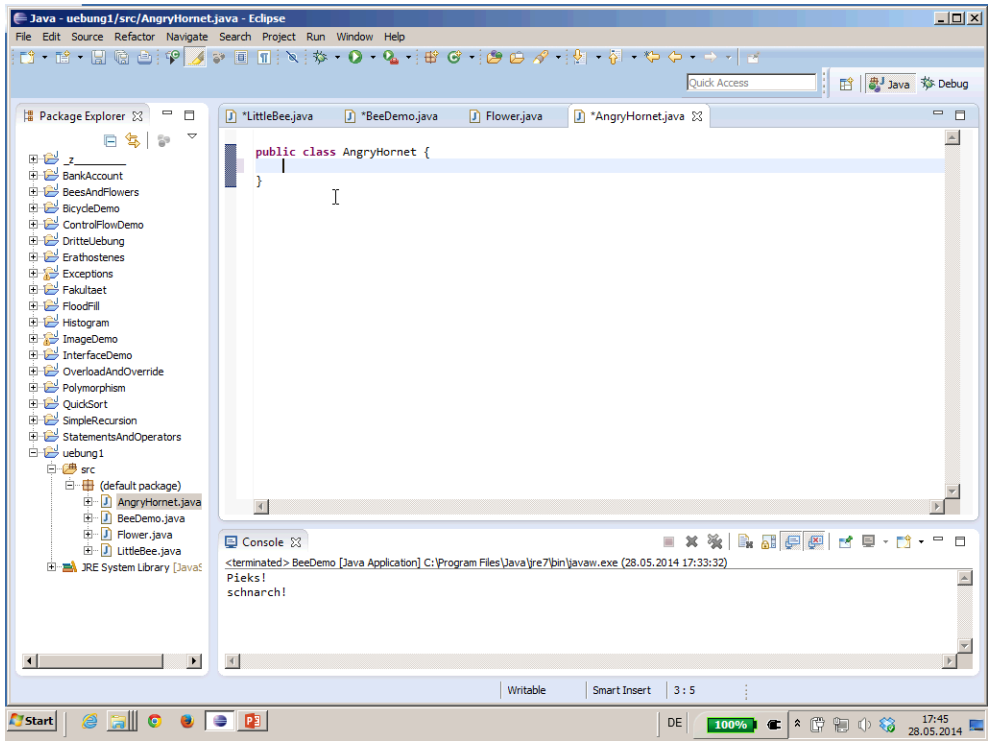

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(f);
Flower flower2 = new Flower();
}
}
f
flower1
null
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
f cannot be resolved to a variable
Writable
Smart Insert
10 : 29

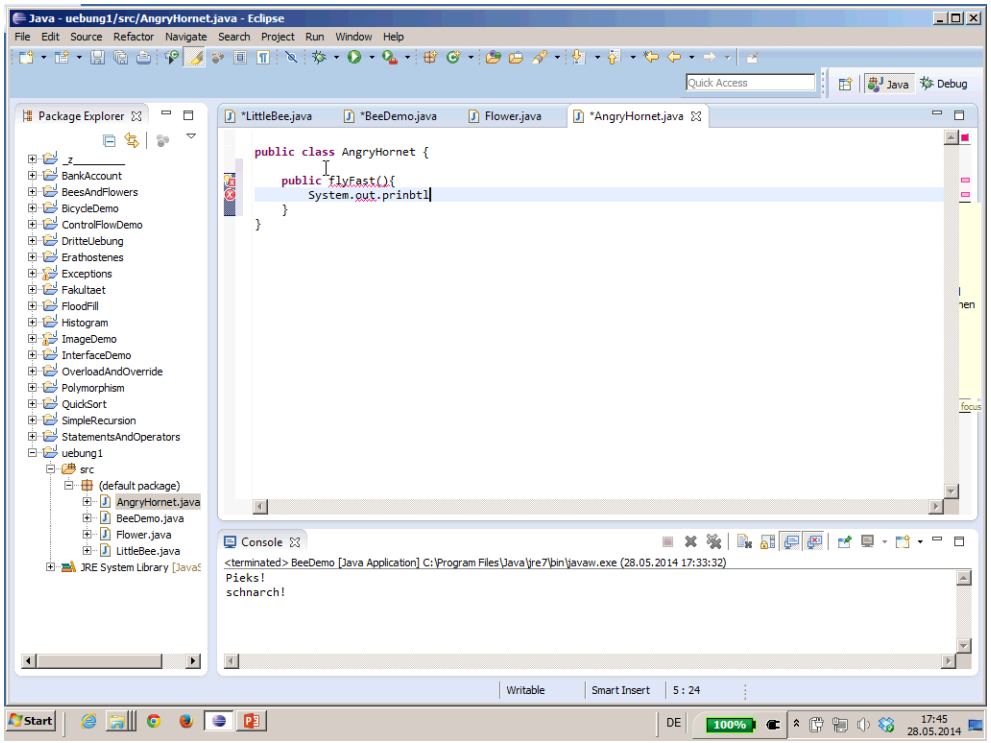

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
13 : 9

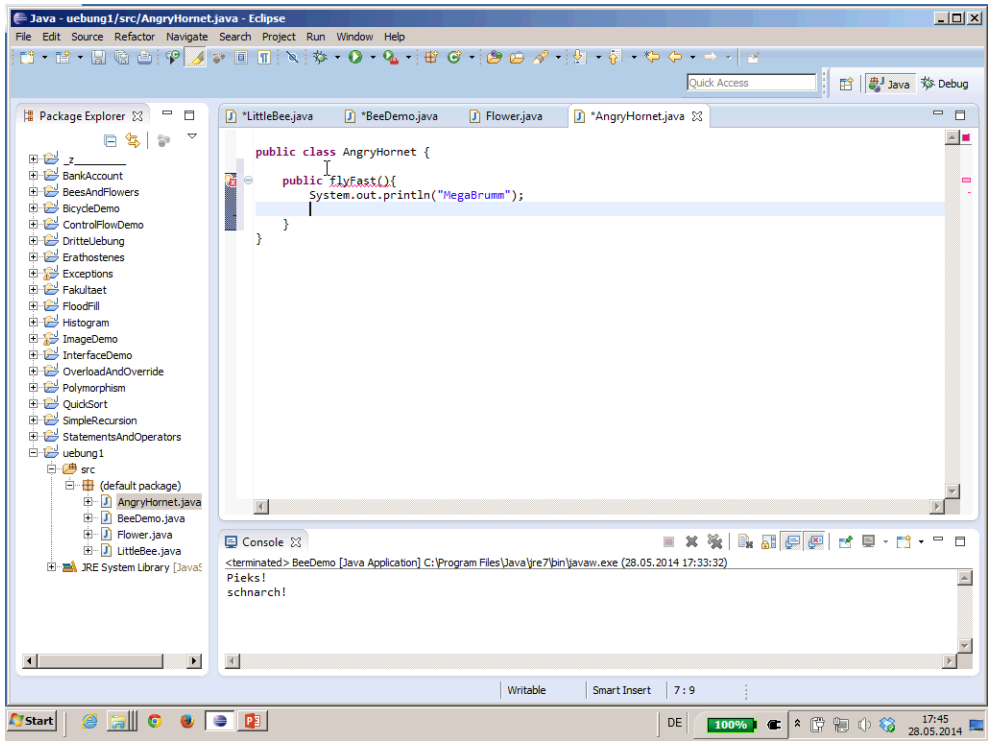

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
13 : 9

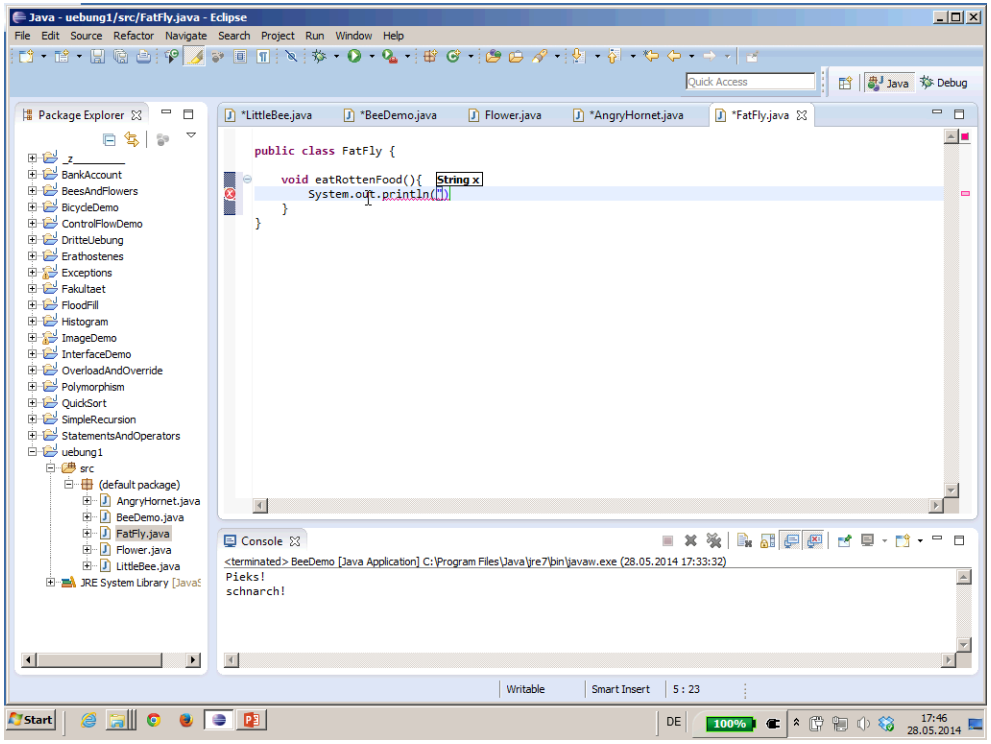

Java - uebung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
*AngryHornet.java
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schoen " + amount + " mg Pollen gesammelt!");
}
void fly
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
19 : 2

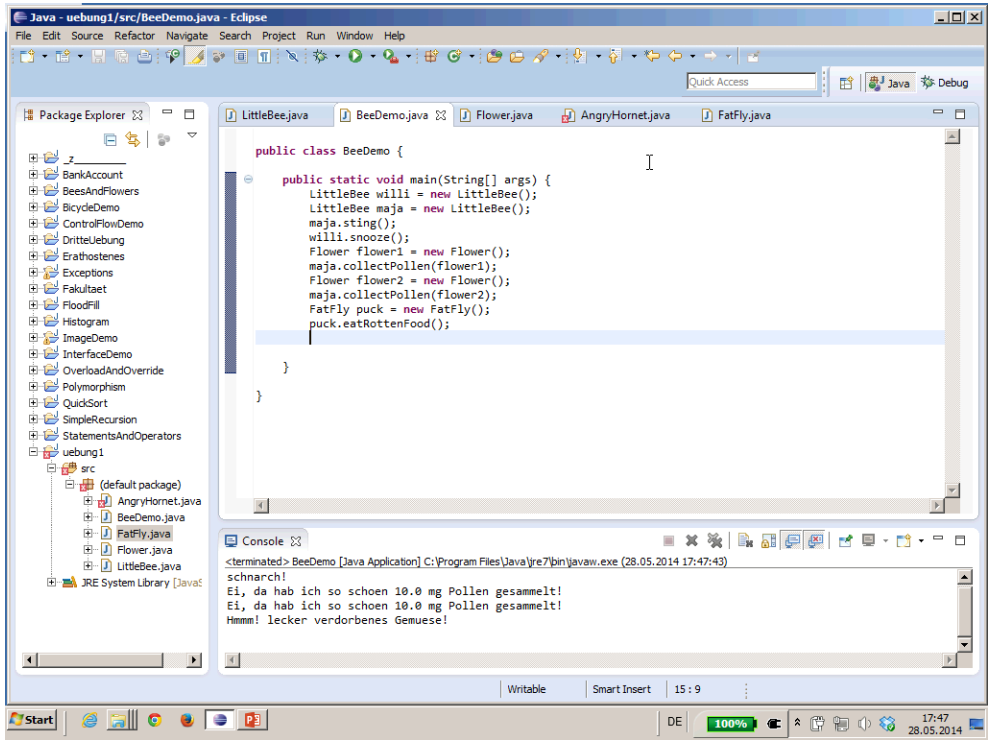

Java - uebung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
*AngryHornet.java
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schoen " + amount + " mg Pollen gesammelt!");
}
void fly(){
System.out.prin
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
19 : 2

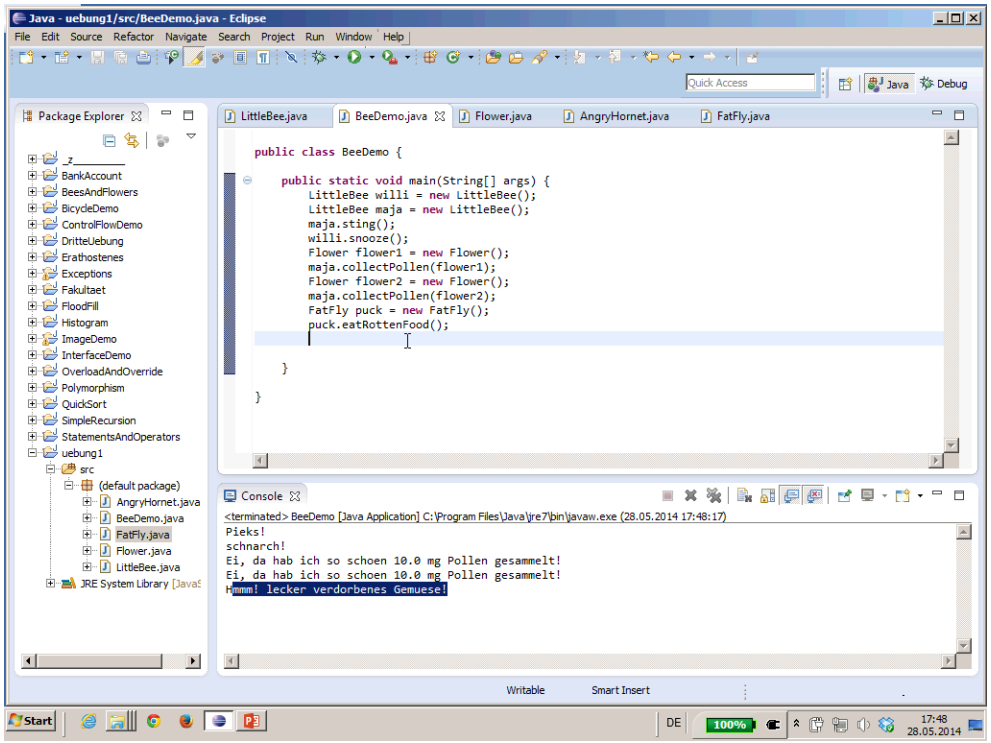

Java - uebung1/src/AngryHornet.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
*AngryHornet.java
public class AngryHornet {
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:33:32)
Pieks!
schnarch!
Writable
Smart Insert
3 : 5

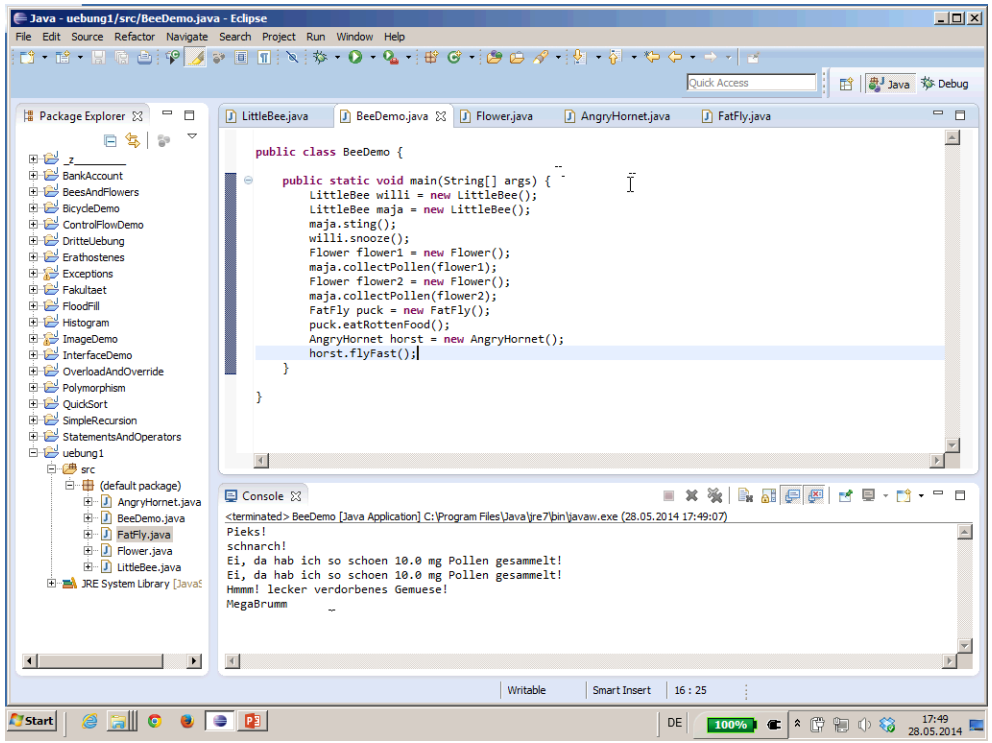

Java - uebung1/src/AngryHornet.java - Eclipse
public class AngryHornet {
public flyFast(){
System.out.prinbtl
}
}
Pieks!
schnarch!
Writable
Smart Insert
5 : 24

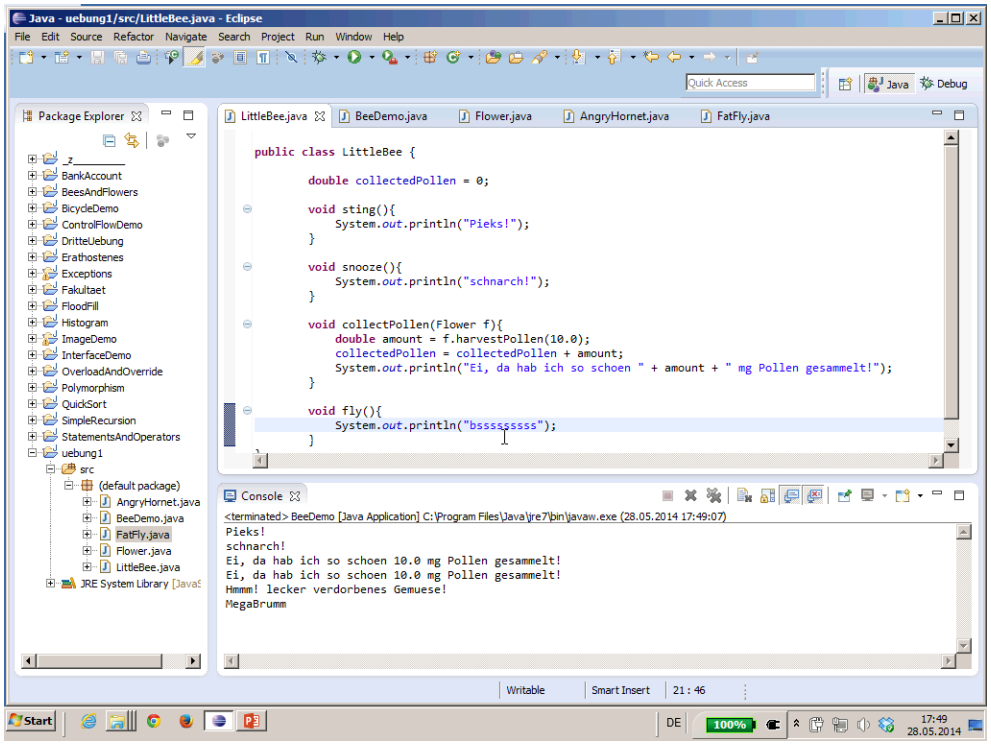

Java - uebung1/src/AngryHornet.java - Eclipse
public class AngryHornet {
public flyFast(){
System.out.println("MegaBrumm");
}
}
Pieks!
schnarch!
Writable
Smart Insert
7 : 9

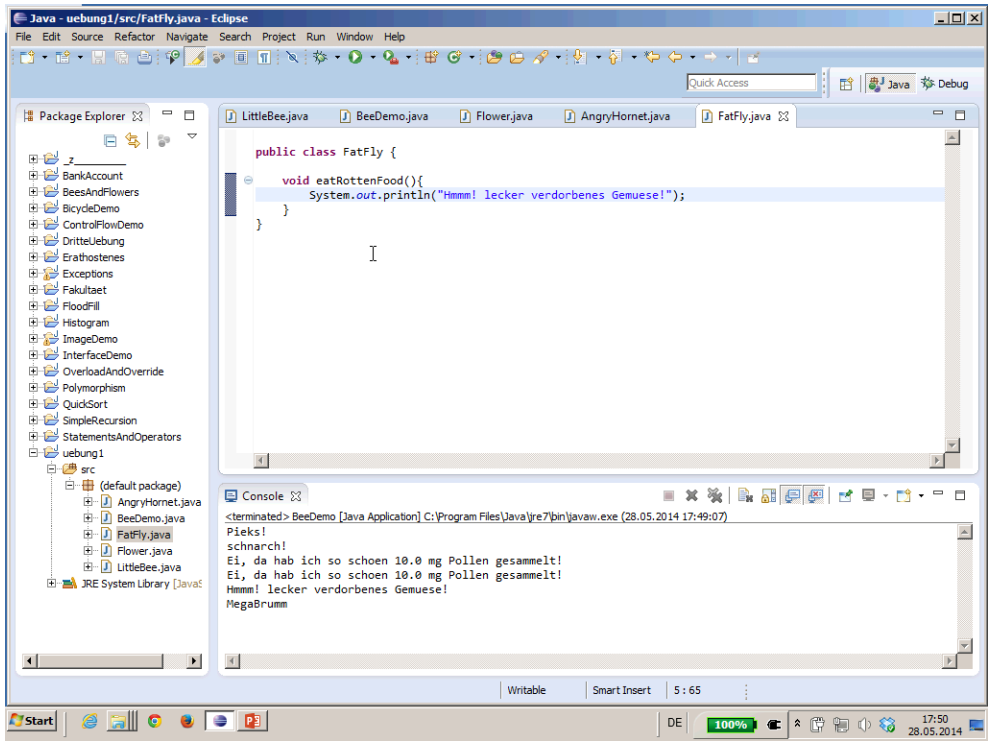

Java - uebung1/src/FatFly.java - Eclipse
public class FatFly {
void eatRottenFood(){ String x
System.out.println()
}
}
Pieks!
schnarch!
Writable
Smart Insert
5 : 23

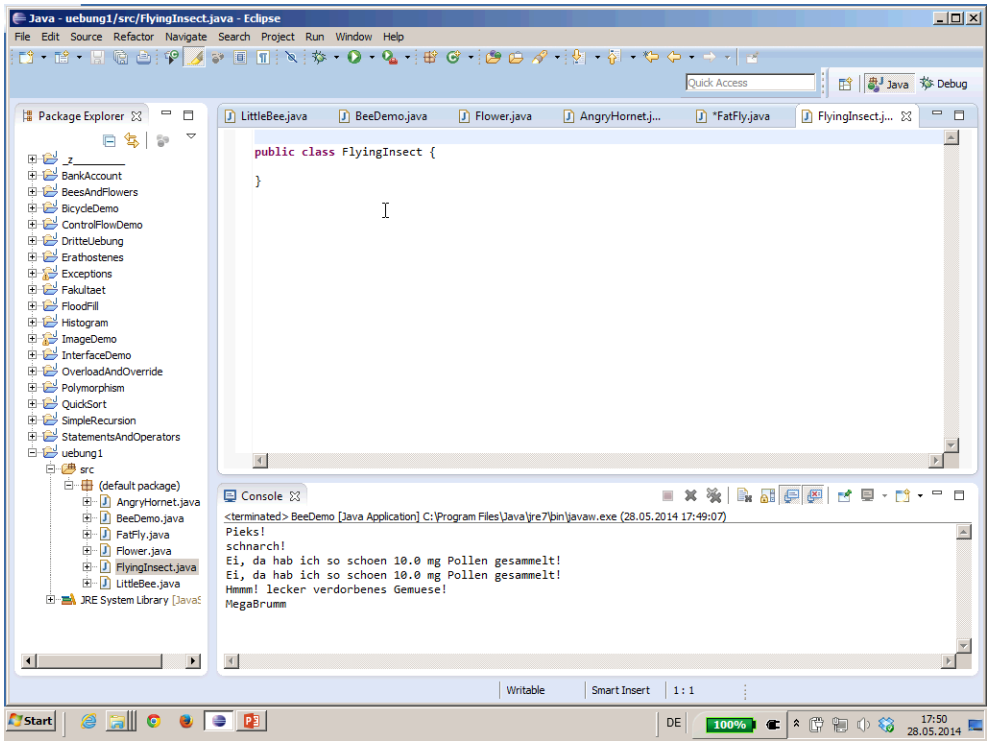

Java - uebung1/src/BeeDemo.java - Eclipse
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
}
}
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
Writable
Smart Insert
15 : 9

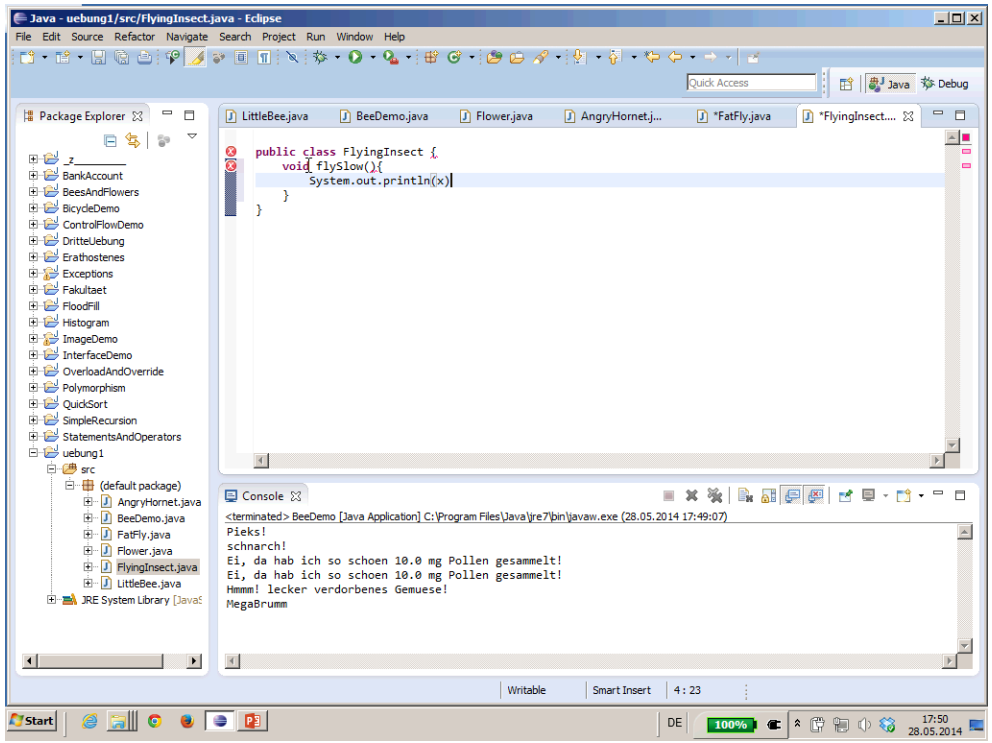

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.java
FatFly.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:48:17)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
Writable
Smart Insert

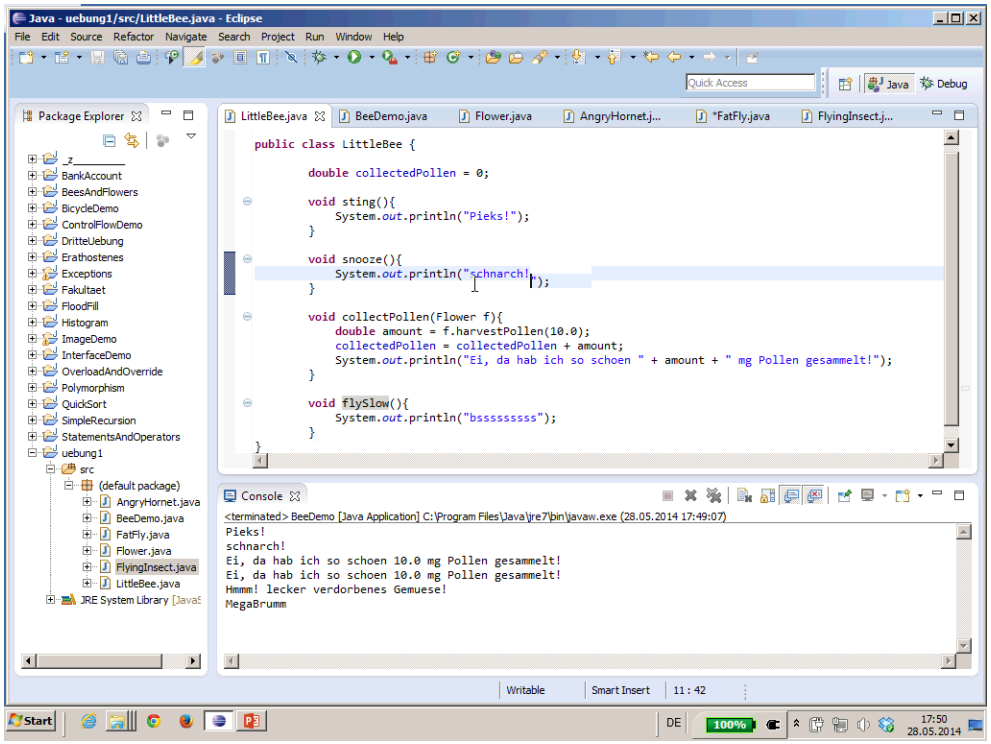

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.java
FatFly.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
16 : 25

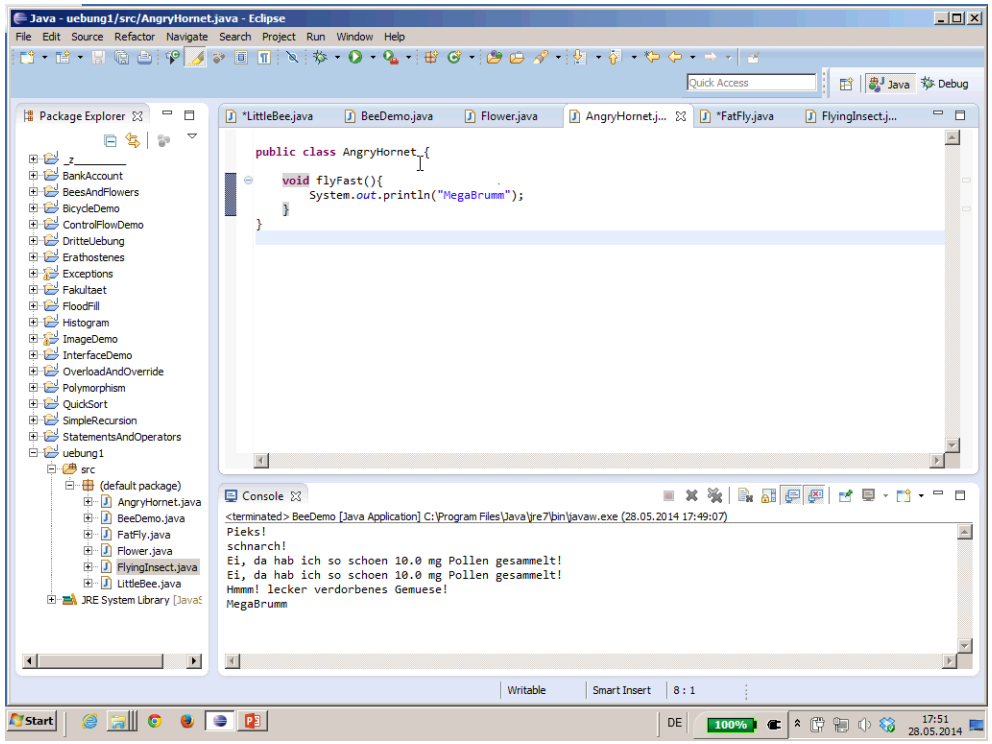

Java - uebung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.java
FatFly.java
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schoen " + amount + " mg Pollen gesammelt!");
}
void fly(){
System.out.println("bsssssssss");
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
21 : 46

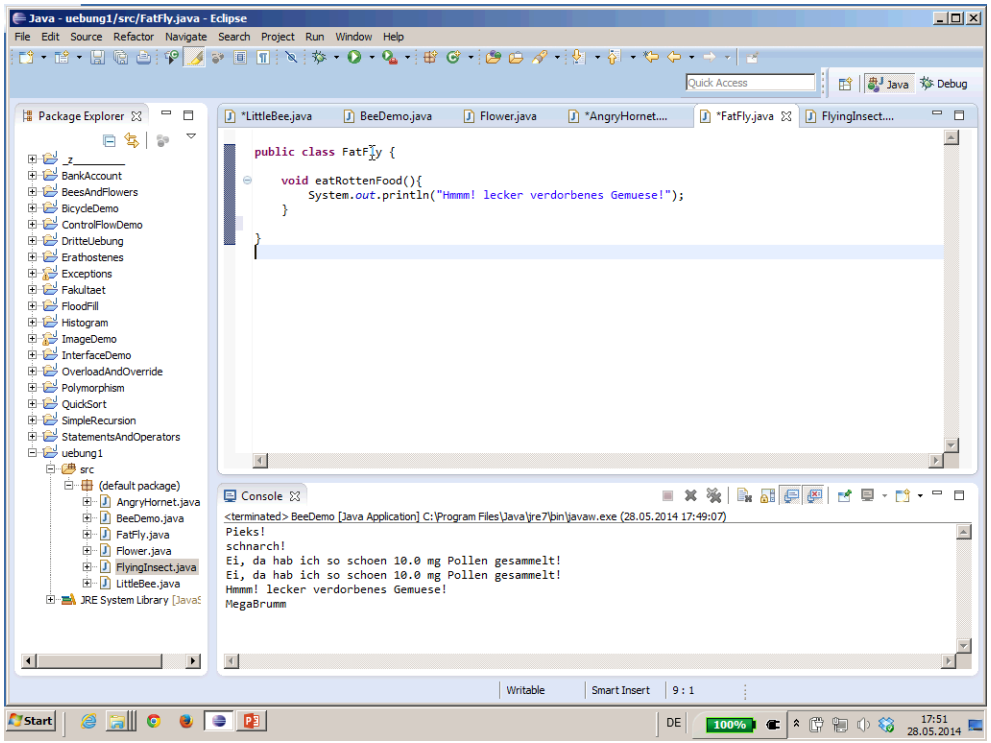

Java - uebung1/src/FatFly.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.java
FatFly.java
public class FatFly {
void eatRottenFood(){
System.out.println("Hmmm! lecker verdorbenes Gemuese!");
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
5 : 65

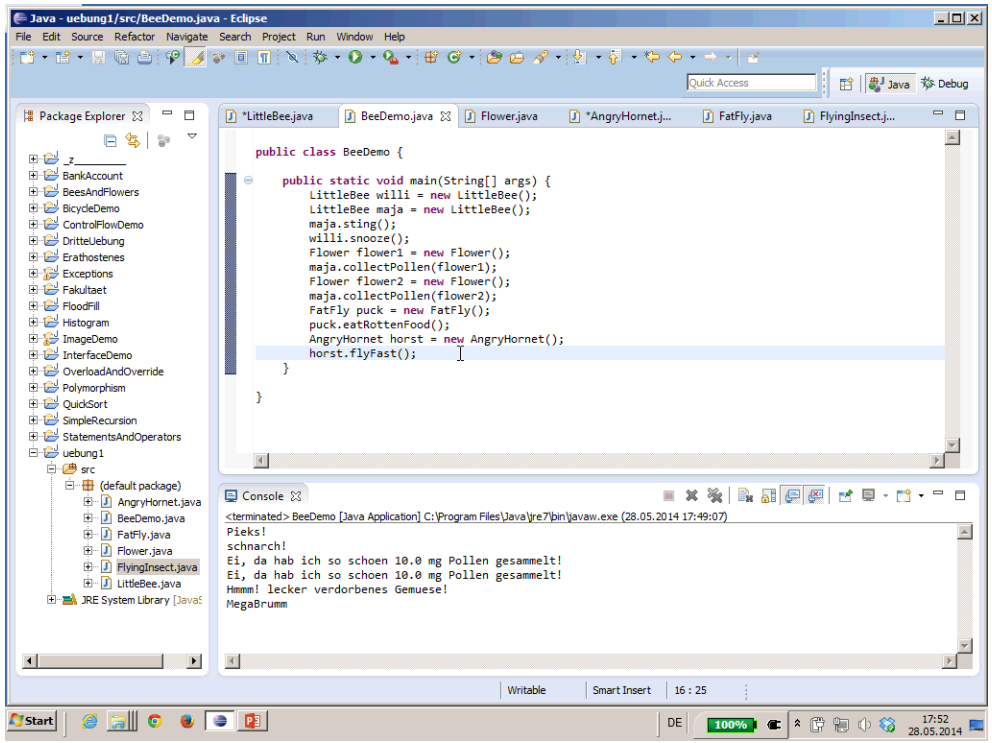

Java - uebung1/src/FlyingInsect.java - Eclipse
public class FlyingInsect {
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
1 : 1

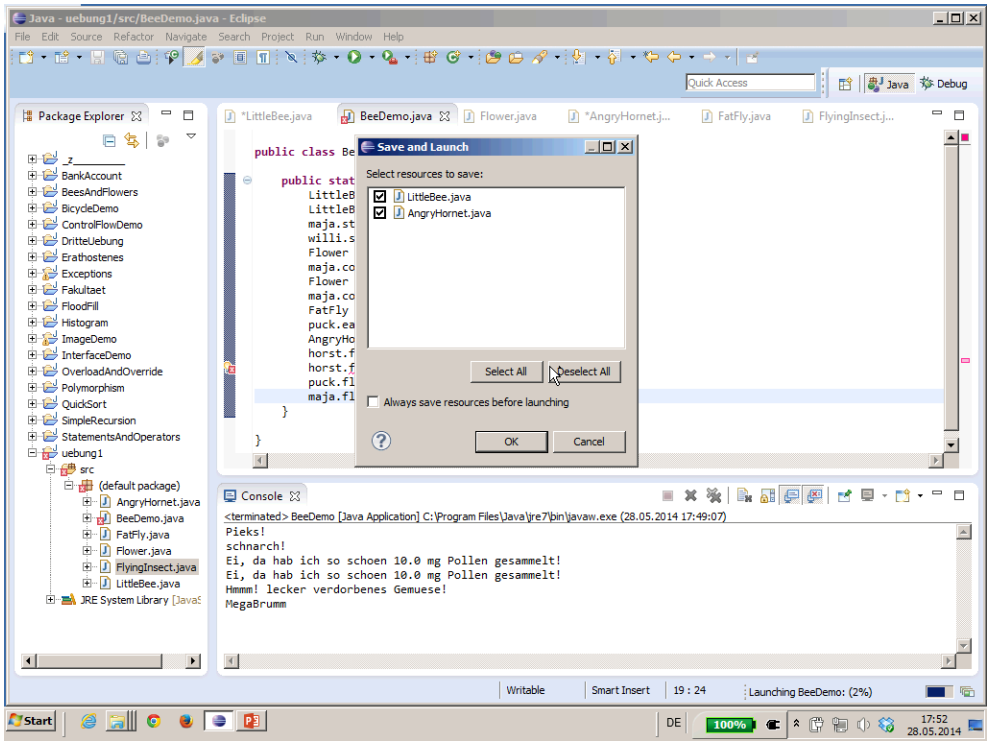

Java - uebung1/src/FlyingInsect.java - Eclipse
public class FlyingInsect {
void flySlow(){
System.out.println(x)
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
4 : 23

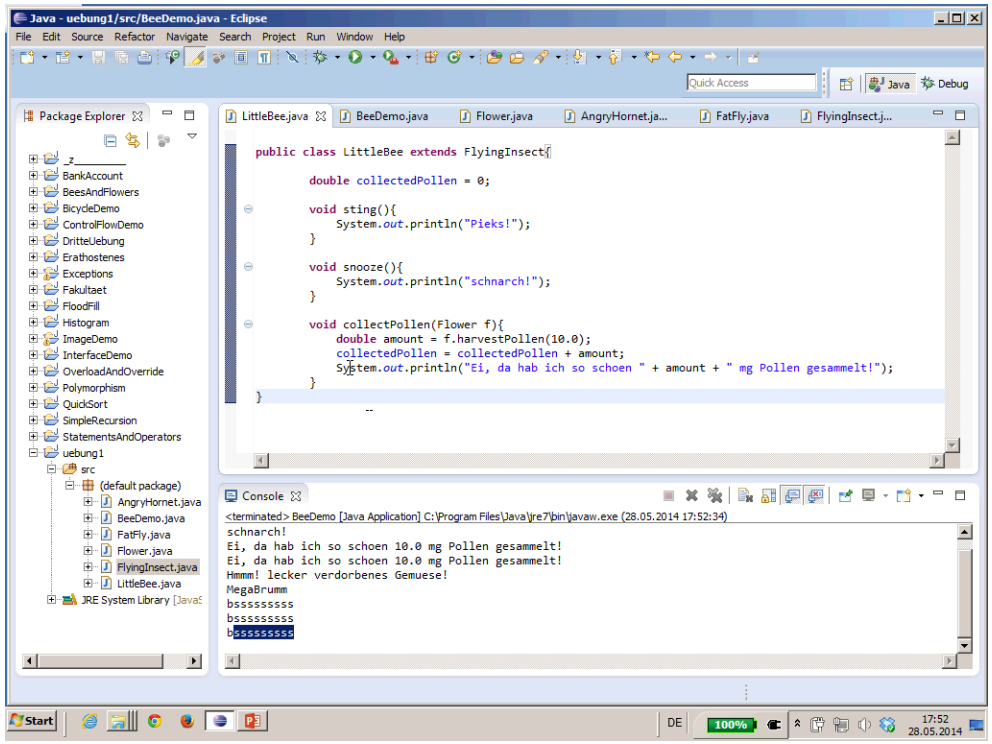

Java - uebung1/src/LittleBee.java - Eclipse
public class LittleBee {
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schoen " + amount + " mg Pollen gesammelt!");
}
void flySlow(){
System.out.println("bsssssssss");
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
11 : 42

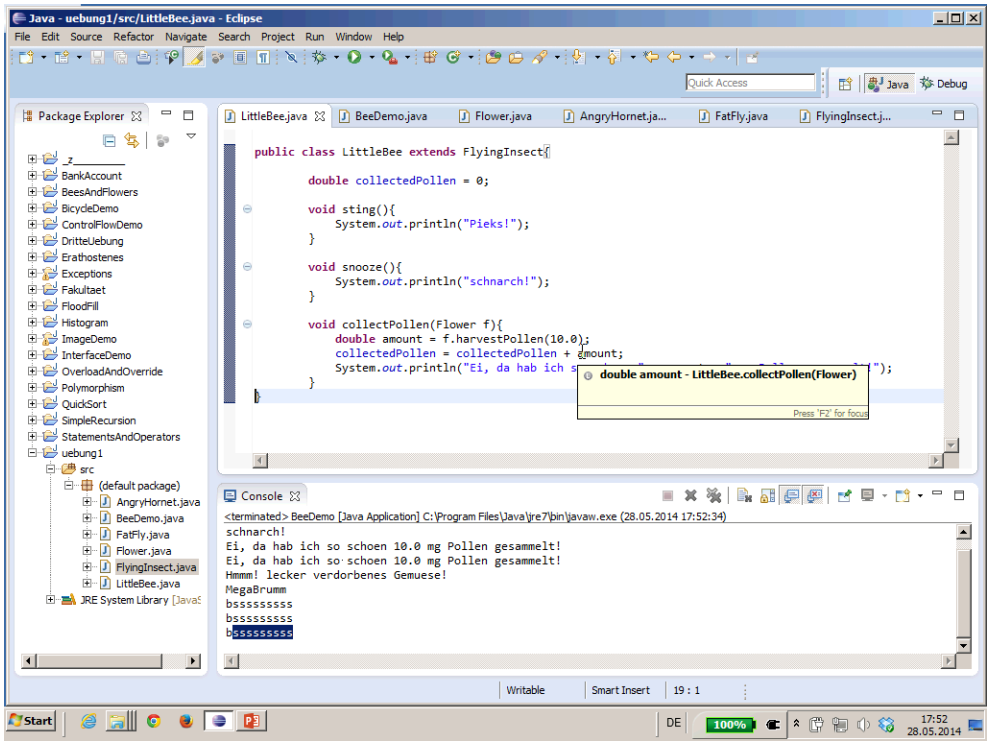

Java - uebung1/src/AngryHornet.java - Eclipse
public class AngryHornet {
void flyFast(){
System.out.println("MegaBrumm");
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
8 : 1

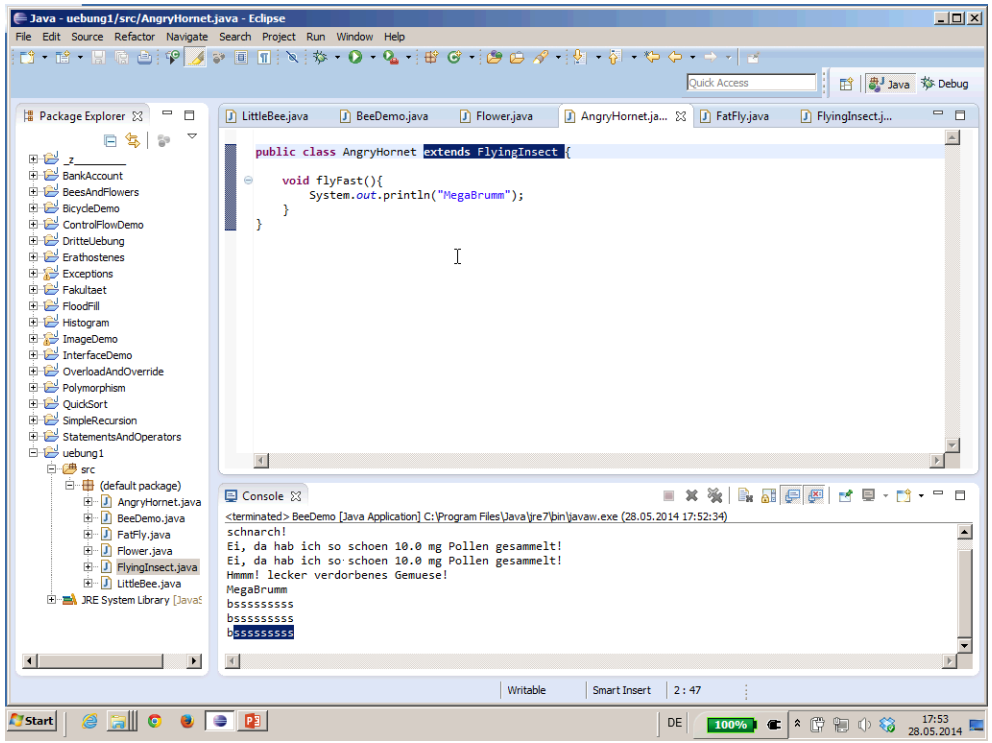

Java - uebung1/src/FatFly.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
BeeDemo.java
Flower.java
*AngryHornet....
*FatFly.java
FlyingInsect....
public class FatFly {
void eatRottenFood(){
System.out.println("Hmmm! lecker verdorbenes Gemuese!");
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
9 : 1

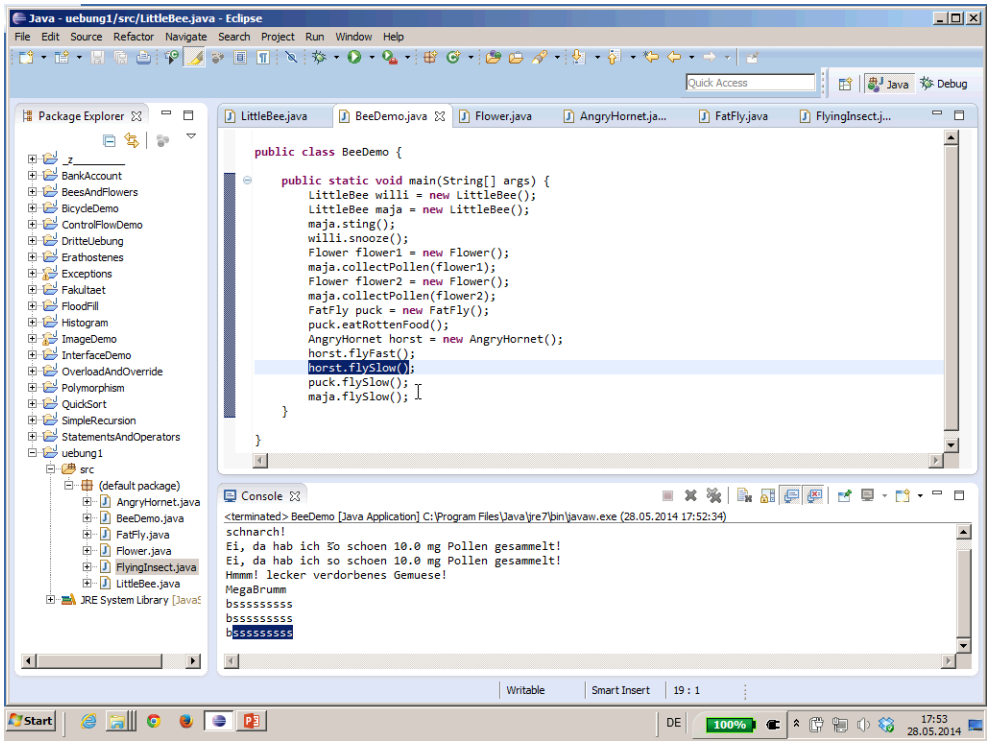

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
BeeDemo.java
Flower.java
*AngryHornet.j...
FatFly.java
FlyingInsect.j...
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
16 : 25

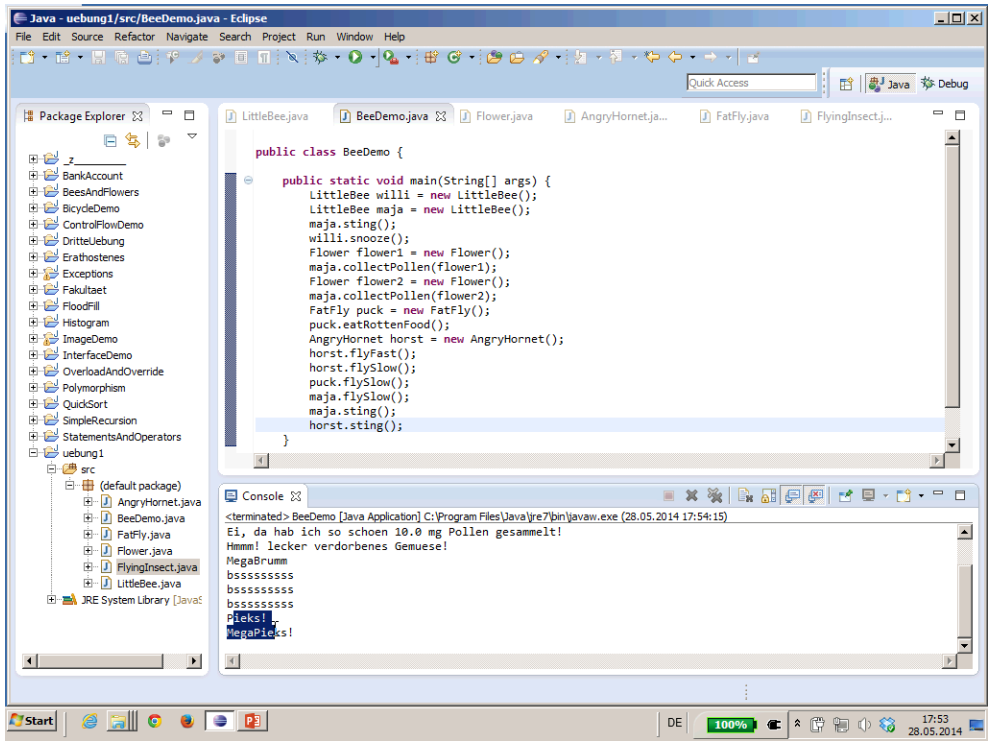

Java - uebung1/src/BeeDemo.java - Eclipse
Save and Launch
Select resources to save:
LittleBee.java
AngryHornet.java
Select All
Deselect All
Always save resources before launching
OK
Cancel
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:49:07)
Pieks!
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
Writable
Smart Insert
19 : 24
Launching BeeDemo: (2%)

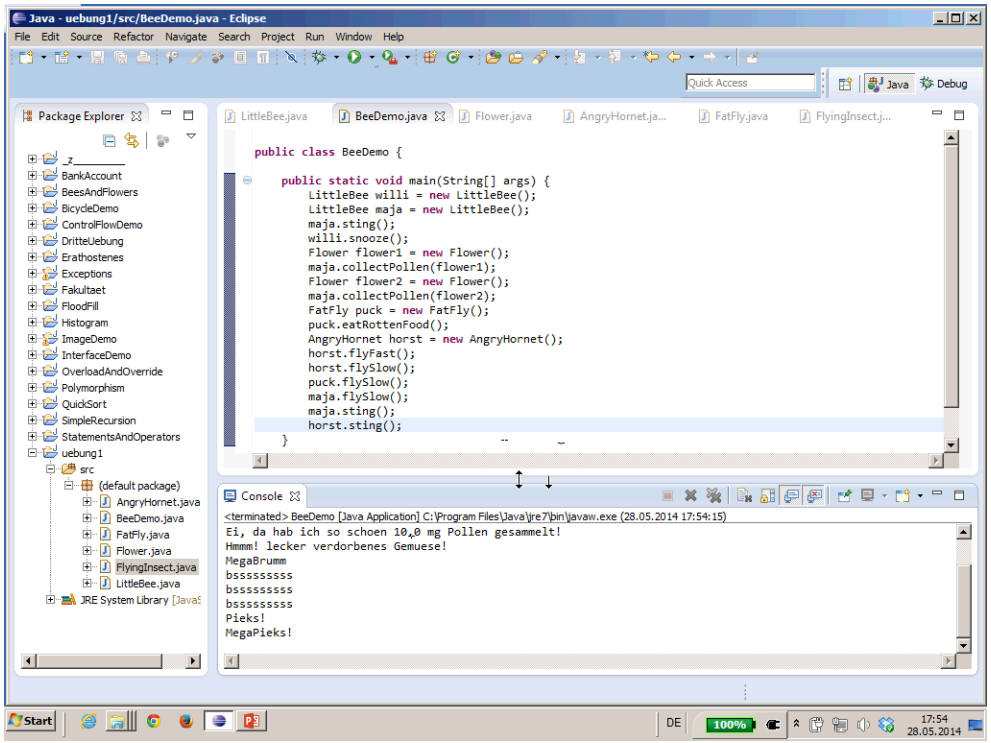

Java - uebung1/src/BeeDemo.java - Eclipse
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.ja...
FatFly.java
FlyingInsect.j...
public class LittleBee extends FlyingInsect{
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schoen " + amount + " mg Pollen gesammelt!");
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:52:34)
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssssss
bssssssssss
bssssssssss

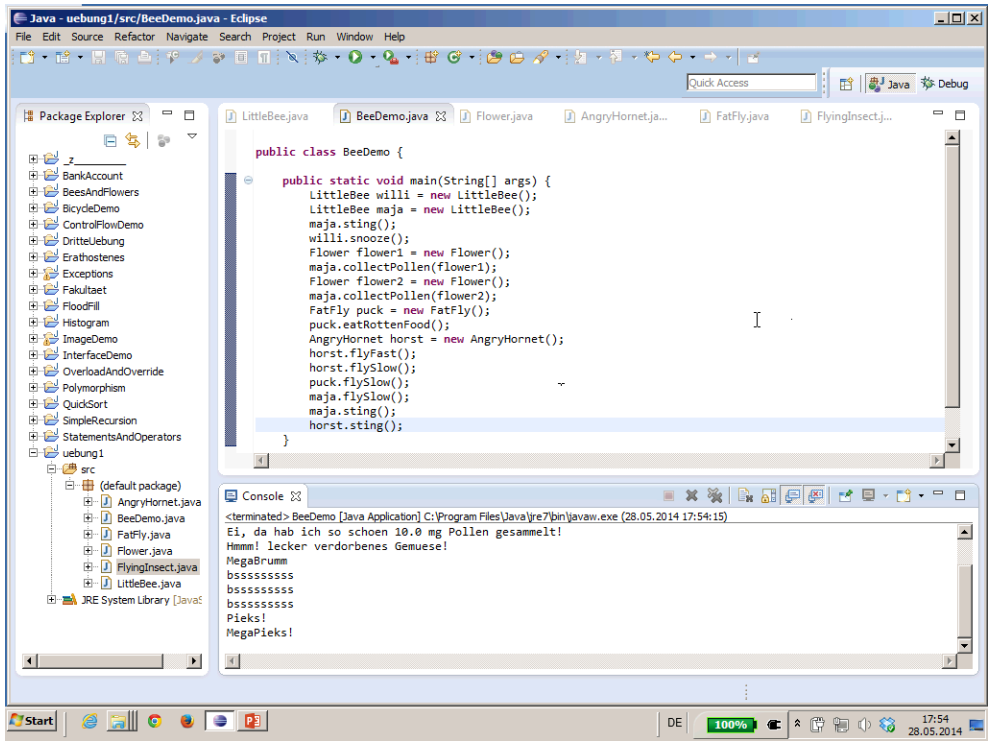

Java - uebung1/src/LittleBee.java - Eclipse
public class LittleBee extends FlyingInsect{
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich
}
}
double amount - LittleBee.collectPollen(Flower)
Press 'F2' for focus
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:52:34)
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Writable
Smart Insert
19 : 1

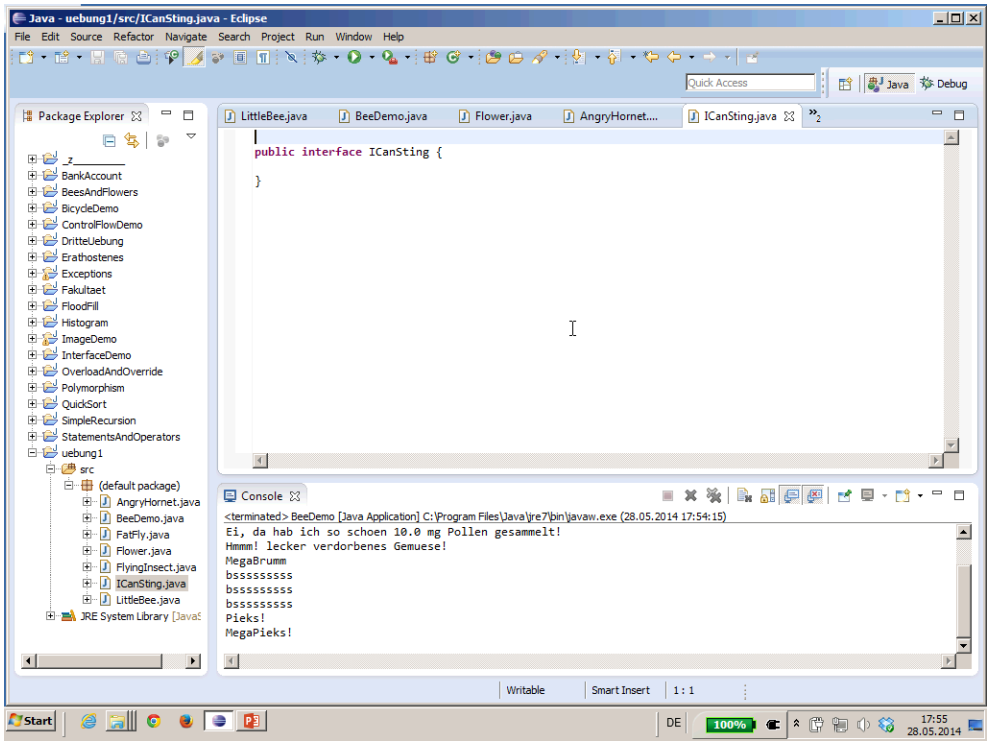

Java - uebung1/src/AngryHornet.java - Eclipse
public class AngryHornet extends FlyingInsect {
void flyFast(){
System.out.println("MegaBrumm");
}
}
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:52:34)
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Writable
Smart Insert
2 : 47

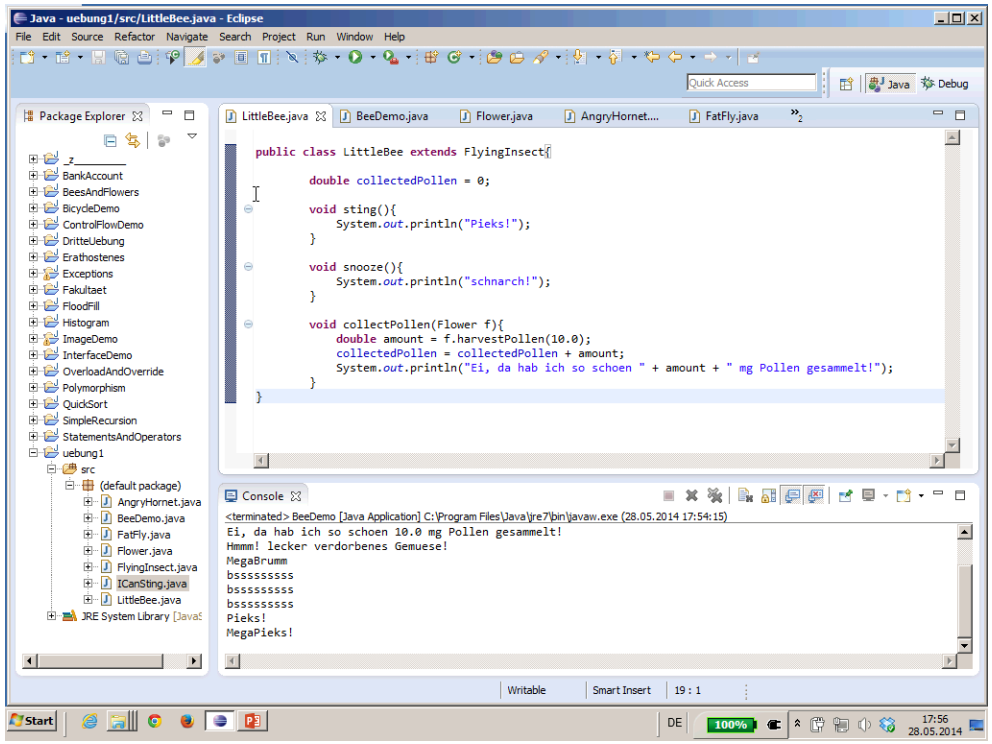

Java - uebung1/src/LittleBee.java - Eclipse
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
}
}
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:52:34)
schnarch!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Writable
Smart Insert
19 : 1

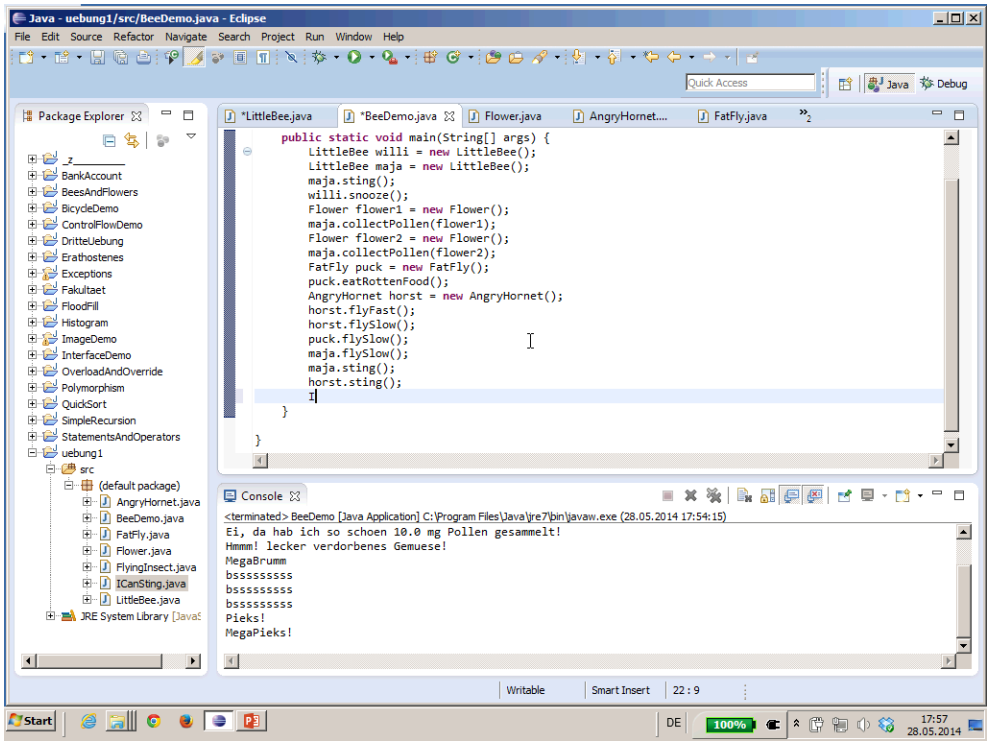

Java - uebung1/src/BeeDemo.java - Eclipse
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
maja.sting();
horst.sting();
}
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Pieks!
MegaPieks!

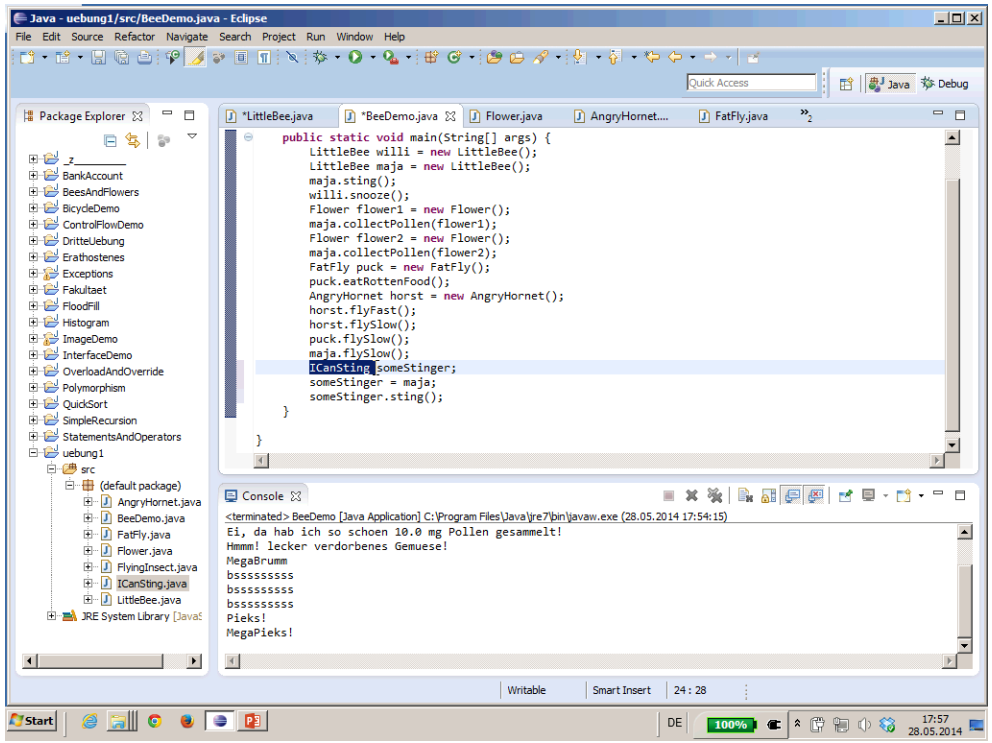

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.ja...
FatFly.java
FlyingInsect.j...
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
maja.sting();
horst.sting();
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10,0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssssss
bssssssssss
bssssssssss
Pieks!
MegaPieks!

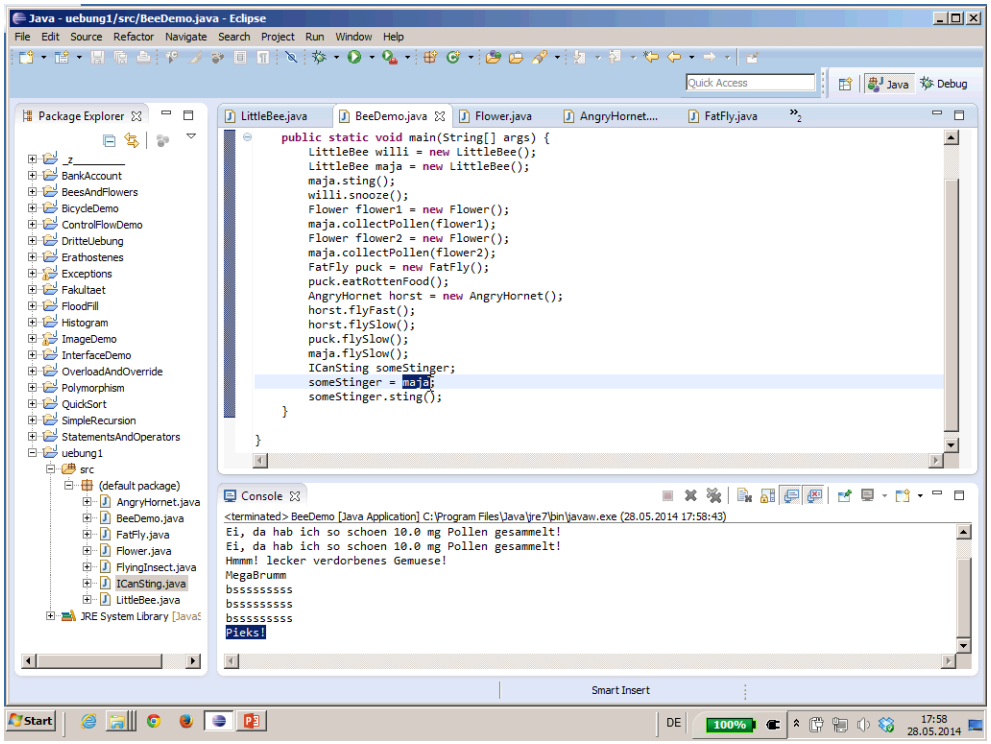

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet.ja...
FatFly.java
FlyingInsect.j...
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
maja.sting();
horst.sting();
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssssss
bssssssssss
bssssssssss
Pieks!
MegaPieks!

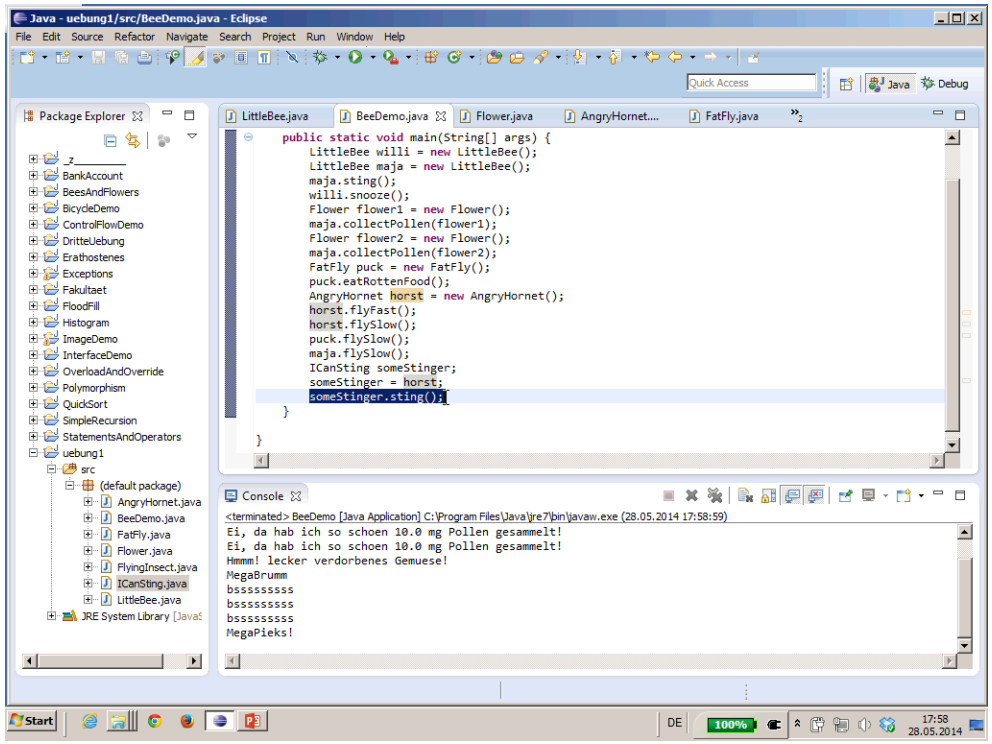

Java - uebung1/src/ICanSting.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
ICanSting.java
public interface ICanSting {
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssssss
bssssssssss
bssssssssss
Pieks!
MegaPieks!
Writable
Smart Insert
1 : 1

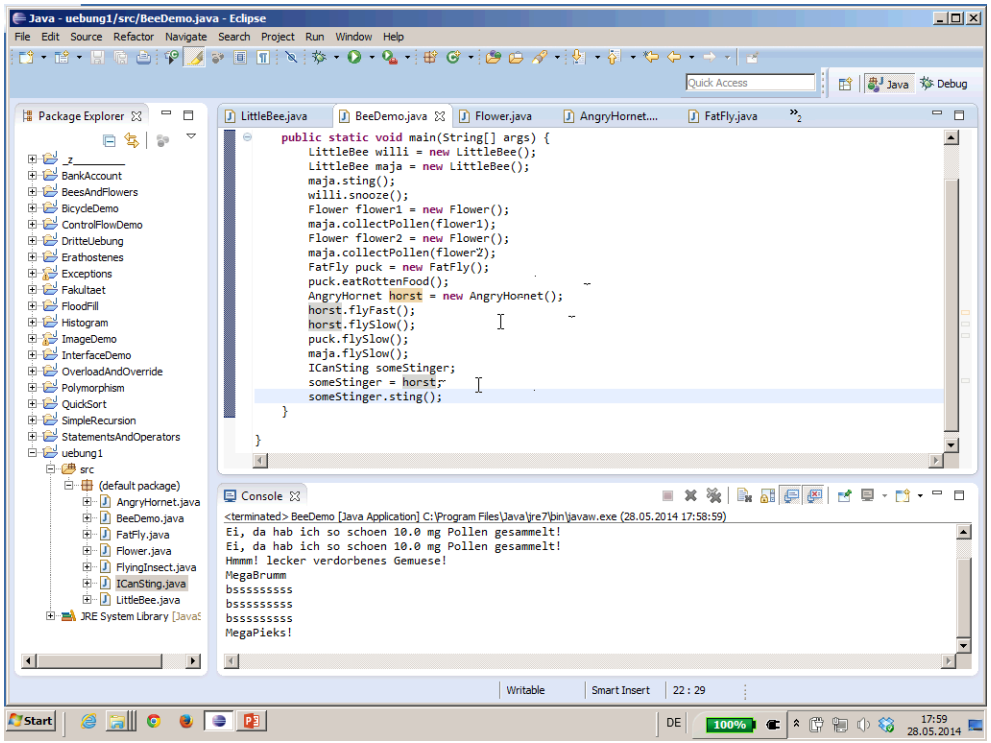

Java - uebung1/src/LittleBee.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public class LittleBee extends FlyingInsect{
double collectedPollen = 0;
void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schoen " + amount + " mg Pollen gesammelt!");
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssssss
bssssssssss
bssssssssss
Pieks!
MegaPieks!
Writable
Smart Insert
19 : 1

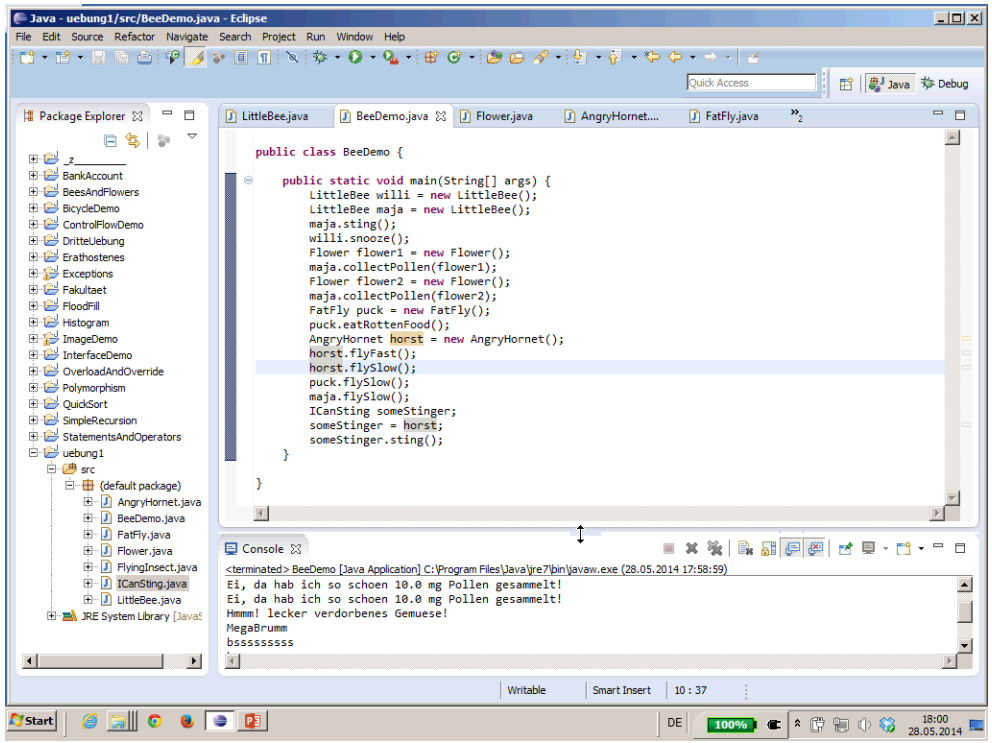

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
maja.sting();
horst.sting();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Pieks!
MegaPieks!
Writable
Smart Insert
22 : 9

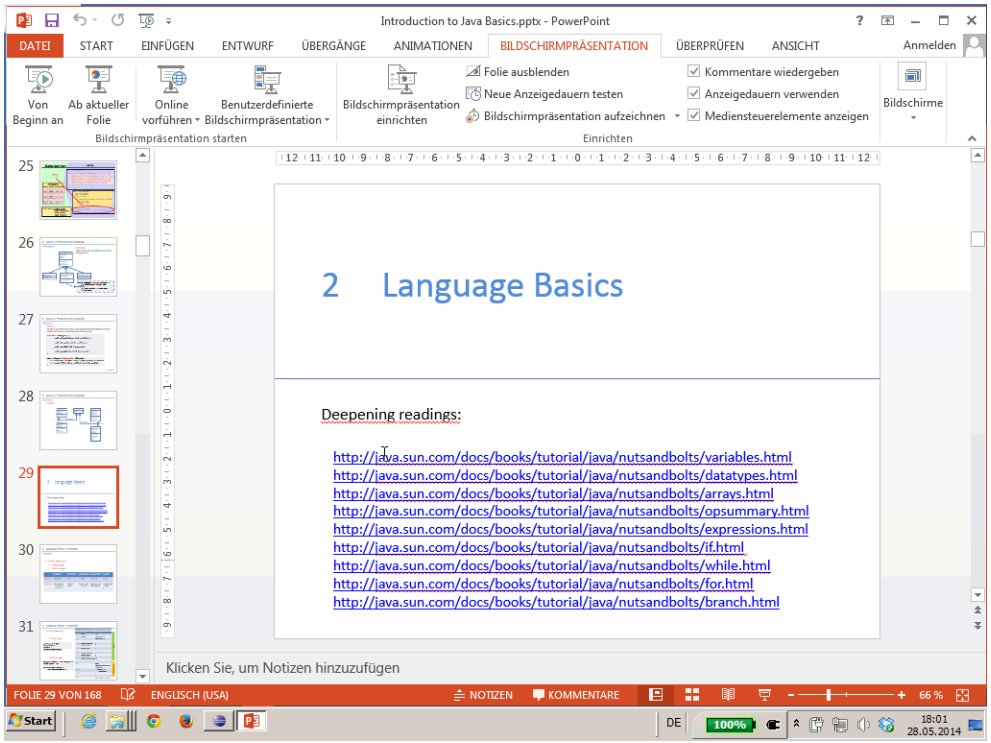

Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
*LittleBee.java
*BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
ICanSting someStinger;
someStinger = maja;
someStinger.sting();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:54:15)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Pieks!
MegaPieks!
Writable
Smart Insert
24 : 28


Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
ICanSting someStinger;
someStinger = maja;
someStinger.sting();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:58:43)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
Pieks!
Smart Insert


Java - uebung1/src/BeeDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
ICanSting someStinger;
someStinger = horst;
someStinger.sting();
}
}
Console
<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (28.05.2014 17:58:59)
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Ei, da hab ich so schoen 10.0 mg Pollen gesammelt!
Hmmm! lecker verdorbenes Gemuese!
MegaBrumm
bssssssss
bssssssss
bssssssss
MegaPieks!







2 Language Basics

Deepening readings:

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/variables.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/datatypes.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/arrays.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/opsummary.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/expressions.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/if.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/while.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/for.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/branch.html

2 Language Basics

Deepening readings:

http://java.sun.com/docs/books/tutorial/java/nutsandbolts/variables.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/datatypes.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/arrays.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/opsummary.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/expressions.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/if.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/while.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/for.html
http://java.sun.com/docs/books/tutorial/java/nutsandbolts/branch.html

2 Language Basics – Variables

Variables

- Variables have a type
  - Primitive type
  - Reference type

	Definition	Declaration	Instantiation	Manipulation	Equality
Primitive	predefined	int a;	a = 117;	a = b + 42;	a == b;

Reference	class Student {
	// Fields and
	// methods ...
	}
		Student heiner;	heiner = new Student();	heiner.age = 21;
				heiner.yawn();
					heiner.equals(
					sabine
					);

2 Language Basics – Variables

Variables

- Variables have a type
  - Primitive type
  - Reference type

	Definition	Declaration	Instantiation	Manipulation	Equality
Primitive	predefined	int a;	a = 117;	a = b + 42;	a == b;

Reference	class Student {
	// Fields and
	// methods ...
	}
		Student heiner;	heiner = new Student();	heiner.age = 21;
				heiner.yawn();
					heiner.equals(
					sabine
					);

2 Language Basics – Variables

Variables

- Variables have a type
  - Primitive type
  - Reference type

	Definition	Declaration	Instantiation	Manipulation	Equality
Primitive	predefined	int a;	a = 117;	a = b + 42;	a == b;

Reference	class Student {
	// Fields and
	// methods ...
	}
		Student heiner;	heiner = new Student();	heiner.age = 21;
				heiner.yawn();
					heiner.equals(
					sabine
					);

2 Language Basics – Variables

Variables

- Variables have a type
  - Primitive type
  - Reference type

	Definition	Declaration	Instantiation	Manipulation	Equality
Primitive	predefined	int a;	a = 117;	a = b + 42;	a == b;

Reference	class Student {
	// Fields and
	// methods ...
	}
		Student heiner;	heiner = new Student();	heiner.age = 21;
				heiner.yawn();
					heiner.equals(sabine);

2 Language Basics – Variables

Variables

- Variables have a type
  - Primitive type
  - Reference type

	Definition	Declaration	Instantiation	Manipulation	Equality
Primitive	predefined	int a;	a = 117;	a = b + 42;	a == b;

Reference	class Student {
	// Fields and
	// methods ...
	}
		Student heiner;	heiner = new Student();	heiner.age = 21;
				heiner.yawn();
					heiner.equals(sabine);

2 Language Basics – Variables

Variables

- Variables have a type
  - Primitive type
  - Reference type

	Definition	Declaration	Instantiation	Manipulation	Equality
Primitive	predefined	int a;	a = 117;	a = b + 42;	a == b;

Reference	class Student {
	// Fields and
	// methods ...
	}
		Student heiner;	heiner = new Student();	heiner.age = 21;
				heiner.yawn();
					heiner.equals(sabine);

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner =
235638465837465845;
```

  - Reference type

```
Bicycle bike1 = new Bicycle();
bike1.gear = 3;

MountainBike bike2 =
    new MountainBike();
```

memory (simplified model)

cell nr.	cell name	cell content	
1123	horst	101	data
1124	heiner	235638465	data
1125		837465845	data
...	...	...	
1150	bike1.cadence	0	data
1151	bike1.speed	0	data
1152	bike1.gear	3	data
...	...	...	
1330	bike2.cadence	0	data
1331	bike2.speed	0	data
1332	bike2.gear	1	data
1333	bike2.seatHeight	15	data
...	...	...	
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	
4035		int horst = 101;	

Primitive Types

- Primitive types (numeric):

byte	short	int	long	float	double
8 bit	16 bit	32 bit	64 bit	32 bit	64 bit

- Examples:

```
byte flags = 63;
short bbb  = 10133;
int heiner = 234103234;
long dong  = -83628735682345;
float fff  = 5464.00345;
float ggg  = -345545.34534E-12f;
double sss = 3245343455.555E67;
```

= -345545.34534 * 10^{-12}
= 3245343455.555 * 10^{67}

2 Language Basics – Variables

Primitive Types

- Primitive types (numeric):

byte	short	int	long	float	double
8 bit	16 bit	32 bit	64 bit	32 bit	64 bit

- Examples:

```
byte flags = 63;
short bbb  = 10133;
int heiner = 234103234;
long dong  = -83628735682345;
float fff  = 5464.00345;
float ggg  = -345545.34534E-12f;
double sss = 3245343455.555E67;
```

= -345545.34534 * 10^{-12}
= 3245343455.555 * 10^{67}

2 Language Basics – Variables

Primitive Types

- Primitive types (numeric):

byte	short	int	long	float	double
8 bit	16 bit	32 bit	64 bit	32 bit	64 bit

- Examples:

```
byte flags = 63;
short bbb  = 10133;
int heiner = 234103234;
long dong  = -83628735682345;
float fff  = 5464.00345;
float ggg  = -345545.34534E-12f;
double sss = 3245343455.555E67;
```

= -345545.34534 * 10^{-12}
= 3245343455.555 * 10^{67}

2 Language Basics – Variables

- Primitive types (numeric, boolean, character):

$\in \mathbb{Z}$ (byte, short, int, long) $\in \mathbb{R}$ (float, double) Unicode (UTF-16) (char)

byte	short	int	long	float	double	boolean	char
8 bit	16 bit	32 bit	64 bit	32 bit	64 bit	1 bit	16 bit
$[-2^7, 2^7-1]$ = [-128,127]	$[-2^{15}, 2^{15}-1]$ = [-32768,32767]	$[-2^{31}, 2^{31}-1]$ = [-2147483648, 2147483647]	$[-2^{63}, 2^{63}-1]$ = [-9223372036854775808, 9223372036854775807]	[+/- ~ $1.4*10^{-45}$, +/- ~ $3.4*10^{38}$]	[+/- ~ $4.9*10^{-324}$, +/- ~ $1.8\ 10^{308}$]	{ true, false }	{ … !, „, §, $, %, &, …., a, b, c, …., ∭,…,∰,…,≎, ガ,キ,ギ,…, ㋘,㋗,㋙,…, 樵,㯭, 蕺,… 얚,얛,얜 …,ششش, …,→,↓,■, …}

2 Language Basics – Variables

- More examples:

```
byte flags = 63;
short bbb  = 10133;
int heiner = 234103234;
long dng   = -83628735682345;
float fff  = 5464.00345f;
float ggg  = -345545.34534E-12f;
double sss = 3245343455.555E67d;

char ccc   = 'm';
char ccc2  = '\n';

boolean isCool = true;
```

byte typically used for bit-patterns

= -345545.34534 * 10^{-12} (float)
= 3245343455.555 * 10^{67} (double)

\n means "new line"

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner =
235638465837465845;
```

  - Reference type

```
Bicycle bike1 = new Bicycle();
bike1.gear = 3;

MountainBike bike2 =
    new MountainBike();
```

memory (simplified model)

cell nr.	cell name	cell content	
1123	horst	101	data
1124	heiner	235638465	data
1125		837465845	data
...	...	...	data
1150	bike1.cadence	0	data
1151	bike1.speed	0	data
1152	bike1.gear	3	data
...	...	...	data
1330	bike2.cadence	0	data
1331	bike2.speed	0	data
1332	bike2.gear	1	data
1333	bike2.seatHeight	15	data
...	...	...	data
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	instructions
4035		int horst = 101;	instructions

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();

boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == false
```

memory (simplified model)

cell nr	cell name	cell content	
...	...	...	data
1149	bike1	<1150>	data
1150	bike1.cadence	0	data
1151	bike1.speed	0	data
1152	bike1.gear	1	data
...	...	...	data
1327	bike2	<1405>	data
...	...	...	data
1405	bike2.cadence	0	data
1406	bike2.speed	0	data
1407	bike2.gear	1	data
...	...	...	data

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();

boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == false
```

memory (simplified model)

cell nr	cell name	cell content	
...	...	...	data
1149	bike1	<1150>	data
1150	bike1.cadence	0	data
1151	bike1.speed	0	data
1152	bike1.gear	1	data
...	...	...	data
1327	bike2	<1405>	data
...	...	...	data
1405	bike2.cadence	0	data
1406	bike2.speed	0	data
1407	bike2.gear	1	data
...	...	...	data

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();




boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == false
```

memory (simplified model)

cell nr	cell name	cell content
...	...	...
1149	bike1	<1150>
1150	bike1.cadence	0
1151	bike1.speed	0
1152	bike1.gear	1
...	...	...
1327	bike2	<1405>
...	...	...
1405	bike2.cadence	0
1406	bike2.speed	0
1407	bike2.gear	1
...	...	...

data

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();




boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == false
```

memory (simplified model)

cell nr	cell name	cell content
...	...	...
1149	bike1	<1150>
1150	bike1.cadence	0
1151	bike1.speed	0
1152	bike1.gear	1
...	...	...
1327	bike2	<1405>
...	...	...
1405	bike2.cadence	0
1406	bike2.speed	0
1407	bike2.gear	1
...	...	...

data

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();


bike1.gear = 3;


bike1 = bike2;


boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == true
```

memory (simplified model)

cell nr	cell name	cell content
...	...	...
1149	bike1	<1405>
1150	bike1.cadence	0
1151	bike1.speed	0
1152	bike1.gear	3
...	...	...
1327	bike2	<1405>
...	...	...
1405	bike2.cadence	0
1406	bike2.speed	0
1407	bike2.gear	1
...	...	...

data

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();


bike1.gear = 3;


bike1 = bike2;


boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == true
```

memory (simplified model)

cell nr	cell name	cell content
...	...	...
1149	bike1	<1405>
1150	bike1.cadence	0
1151	bike1.speed	0
1152	bike1.gear	3
...	...	...
1327	bike2	<1405>
...	...	...
1405	bike2.cadence	0
1406	bike2.speed	0
1407	bike2.gear	1
...	...	...

data

2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();


bike1.gear = 3;


bike1 = bike2;



boolean c;
c = bike1.equals(bike2);
    // c == true
c = (bike1 == bike2);
    // c == true
```

memory (simplified model)

cell nr	cell name	cell content
...	...	...
1149	bike1	<1405>
1150	bike1.cadence	0
1151	bike1.speed	0
1152	bike1.gear	3
...	...	...
1327	bike2	<1405>
...	...	...
1405	bike2.cadence	0
1406	bike2.speed	0
1407	bike2.gear	1
...	...	...

data