Script generated by TTT

Title: Lehmann: Uebung_Einf_HF (22.06.2012)

Date: Fri Jun 22 09:17:26 CEST 2012

Duration: 88:21 min

Pages: 73

Classes, Objects, Inheritance

Why do we need constructors?

- Ensure **complete** and **consistent** initialization after object creation
- Access superclass constructors:
  Construct object according to definition of superclass, then add specifics
- Provide several constructors for respective use-cases

```
class Bicycle {
  public int cadence;
  public int speed;
  public int gear;

  public Bicycle(int c, int s, int g) {
    cadence = c;
    speed = s;
    gear = g;
  }

  public Bicycle(int g) {
    cadence = 0;
    speed = 0;
    gear = g;
  }
}
```

```
class Tandem extends Bicycle {
  public int numberOfDrivers;

  public Tandem(int c, int s, int g, int n) {
    super(c, s, g);
    numberOfDrivers = n;
  }
}
```

Classes, Objects, Inheritance

Example:

```
class Person {
  String firstName;
  String lastName;
  long taxIdent;

  Person(String fName, String lName) {
    firstName = fName;
    lastName = lName;
    taxIdent = createUniqueTaxIdentifier();       // side-effect!
  }
}

class Student extends Person {
  long matrikelNr;

  //Student(String fName, String lName, long matrikel) {
  //  super(firstName, lastName);
  //  matrikelNr = matrikel;
  //}
}
```

```
// Manual initialization, easy to make a mistake (e.g. what about `taxIdent`?)
Student student1 = new Student();
student1.firstName = "Max";
student1.lastName = "Mustermann";
student1.matrikelNr = 1234567890;

// Complete, consistent, convenient ☺
Student student2 = new Student("Max", "Mustermann", 1234567890);
```

Example:

```
class Person {
  String firstName;
  String lastName;
  long taxIdent;

  Person(String fName, String lName) {
    firstName = fName;
    lastName = lName;
    taxIdent = createUniqueTaxIdentifier();       // side-effect!
  }
}

class Student extends Person {
  long matrikelNr;

  Student(String fName, String lName, long matrikel) {
    super(firstName, lastName);
    matrikelNr = matrikel;
  }
}
```

```
// Manual initialization, easy to make a mistake (e.g. what about `taxIdent`?)
Student student1 = new Student();
student1.firstName = "Max";
student1.lastName = "Mustermann";
student1.matrikelNr = 1234567890;

// Complete, consistent, convenient ☺
Student student2 = new Student("Max", "Mustermann", 1234567890);
```

Classes, Objects, Inheritance

Parameters

- *parameter list*: Passing parameters to methods or constructors

```
class SomeClass {
        public int doSomething(int primitiveParameter1,
                double primitiveParameter2,
                SomeClass referenceParameter)
        {
                int someInt = 17 + 9;
                primitiveParameter1 = 0;
                referenceParameter = null;
                return someInt;
        }
}
```

body

- Passing primitive type parameters: **Call By Value**
  Changes to parameter have no effect outside of method or constructor

```
int x = 1;
SomeClass someObject = new SomeClass();
int y = doSomething(x, 2.345, someObject);
// At this point, x still has value 1.
```

Classes, Objects, Inheritance

Example:

```
class Person {
  String firstName;
  String lastName;
  long taxIdent;

  Person(String fName, String lName) {
    firstName = fName;
    lastName = lName;
    taxIdent = createUniqueTaxIdentifier();       // side-effect!
  }
}

class Student extends Person {
  long matrikelNr;

  Student(String fName, String lName, long matrikel) {
    super(firstName, lastName);
    matrikelNr = matrikel;
  }
}
```

```
// Manual initialization, easy to make a mistake (e.g. what about `taxIdent`?)
Student student1 = new Student();
student1.firstName = "Max";
student1.lastName = "Mustermann";
student1.matrikelNr = 1234567890;

// Complete, consistent, convenient ☺
Student student2 = new Student("Max", "Mustermann", 1234567890);
```

Classes, Objects, Inheritance

Parameters

- *parameter list*: Passing parameters to methods or constructors

```
class SomeClass {
        public int doSomething(int primitiveParameter1,
                double primitiveParameter2,
                SomeClass referenceParameter)
        {
                int someInt = 17 + 9;
                primitiveParameter1 = 0;
                referenceParameter = null;
                return someInt;
        }
}
```

body

- Passing primitive type parameters: **Call By Value**
  Changes to parameter have no effect outside of method or constructor

```
int x = 1;
SomeClass someObject = new SomeClass();
int y = doSomething(x, 2.345, someObject);
// At this point, x still has value 1.
```

Parameters

- *parameter list*: Passing parameters to methods or constructors

```
class SomeClass {
        public int doSomething(int primitiveParameter1,
                double primitiveParameter2,
                SomeClass referenceParameter)
        {
                int someInt = 17 + 9;
                primitiveParameter1 = 0;
                referenceParameter = null;
                return someInt;
        }
}
```

body

- Passing reference type parameters: **ALSO Call By Value (!!)**
  Changes to parameter ITSELF have no effect outside of method or constructor

```
int x = 1;
SomeClass someObject = new SomeClass();
int y = doSomething(x, 2.345, someObject);
// At this point, someObject still references
// the same object (someObject != null).
```


Parameters

- However, passing reference type parameters can be used to modify objects or arrays with a lasting effect:

```
public void doSomethingElse(int[] refParameter) {
  for (int i=0; i<refParameter.length; i++) {
    refParameter[i] = 47;
  }
}
```

```
public void anotherMethod() {
  int[] someArray = { 1, 2, 3, 4, 5 };
  doSomethingElse(someArray);
  for (int i=0; i<someArray.length; i++) {
    System.out.print("#" + i + ": " + someArray[i]);
  }
}
```

⇨ output will be: `#0: 47 #1: 47 #2: 47 #3: 47`

Classes, Objects, Inheritance

Parameters

- However, passing reference type parameters can be used to modify objects or arrays with a lasting effect:

```
public void doSomethingElse(int[] refParameter) {
  for (int i=0; i<refParameter.length; i++) {
    refParameter[i] = 47;
  }
}
```

```
public void anotherMethod() {
  int[] someArray = { 1, 2, 3, 4, 5 };
  doSomethingElse(someArray);
  for (int i=0; i<someArray.length; i++) {
    System.out.print("#" + i + ": " + someArray[i]);
  }
}
```

⇨ output will be: `#0: 47 #1: 47 #2: 47 #3: 47`

Classes, Objects, Inheritance

Why is this so?

- Remember: Reference type variables **point to an object** of the reference type
- Call By Value means: When method or constructor is called, **copies** of corresponding variables' **values** are passed
- Once method returns: Copies are destroyed
- Reference type variables may be used to manipulate something OUTSIDE the method (or constructor).

➔ **"Side effect"**

memory (simplified model)

cell nr	cell name	cell content
...	...	...
1149	someArray	<1150>
1150		0
1151		0
1152		7
...	...	...
1327	**refParameter**	**<1150>**
...	...	...

The special value `null` :

- `null` points to "nothing"

```
Bicycle bike1 = new Bicycle();

Bicycle sameBike = bike1;
```

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	sameBike	<1150>
...	...	...

Classes, Objects, Inheritance

The special value `null` :

- `null` points to "nothing"

```
Bicycle bike1 = new Bicycle();

Bicycle sameBike = bike1;

sameBike = null;
// Has no effect on bike1.
```

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	sameBike	null
...	...	...

Classes, Objects, Inheritance

Returning values

- Methods may return a value (corresponding to declared return type):

```
public long fakultaet(int n) {
  long result = 1;
  for (int i=2; i<=n; i++) {
    result = result * i;
  }
  return result;
}

public static void main(String[] args) {
  long x = fakultaet(5);
  System.out.println("Faculty of 5 is " + x + ".");
}
```

- General form: `return` *expression*`;`

  Returns the **value** of *expression*

Classes, Objects, Inheritance

Returning values

- Aside from primitive types, references can be returned as well:

```
public Bicycle goGetABike() {
  return new Bicycle();
}

public double[] iWantRandomNumbers() {
  double[] result = new double[10];
  for (int i=0; i<result.length; i++) {
    result[i] = Math.random();
  }
  return result;
}

public static void main(String[] args) {
  Bicycle bike = goGetABike();
  double[] myShinyNewArray = iWantRandomNumbers();
}
```

- Corresponding objects/arrays are not "destroyed" (Remember: Reference type variables hold references to the objects, not the objects themselves!)

Returning values

- Aside from primitive types, references can be returned as well:

```
public Bicycle goGetABike() {
  return new Bicycle();
}

public double[] iWantRandomNumbers() {
  double[] result = new double[10];
  for (int i=0; i<result.length; i++) {
    result[i] = Math.random();
  }
  return result;
}

public static void main(String[] args) {
  Bicycle bike = goGetABike();
  double[] myShinyNewArray = iWantRandomNumbers();
}
```

- Corresponding objects/arrays are not "destroyed" (Remember: Reference type variables hold references to the objects, not the objects themselves!)

Classes, Objects, Inheritance

Returning values

- Aside from primitive types, references can be returned as well:

```
public Bicycle goGetABike() {
  return new Bicycle();
}

public double[] iWantRandomNumbers() {
  double[] result = new double[10];
  for (int i=0; i<result.length; i++) {
    result[i] = Math.random();
  }
  return result;
}

public static void main(String[] args) {
  Bicycle bike = goGetABike();
  double[] myShinyNewArray = iWantRandomNumbers();
}
```

- Corresponding objects/arrays are not "destroyed" (Remember: Reference type variables hold references to the objects, not the objects themselves!)

Classes, Objects, Inheritance

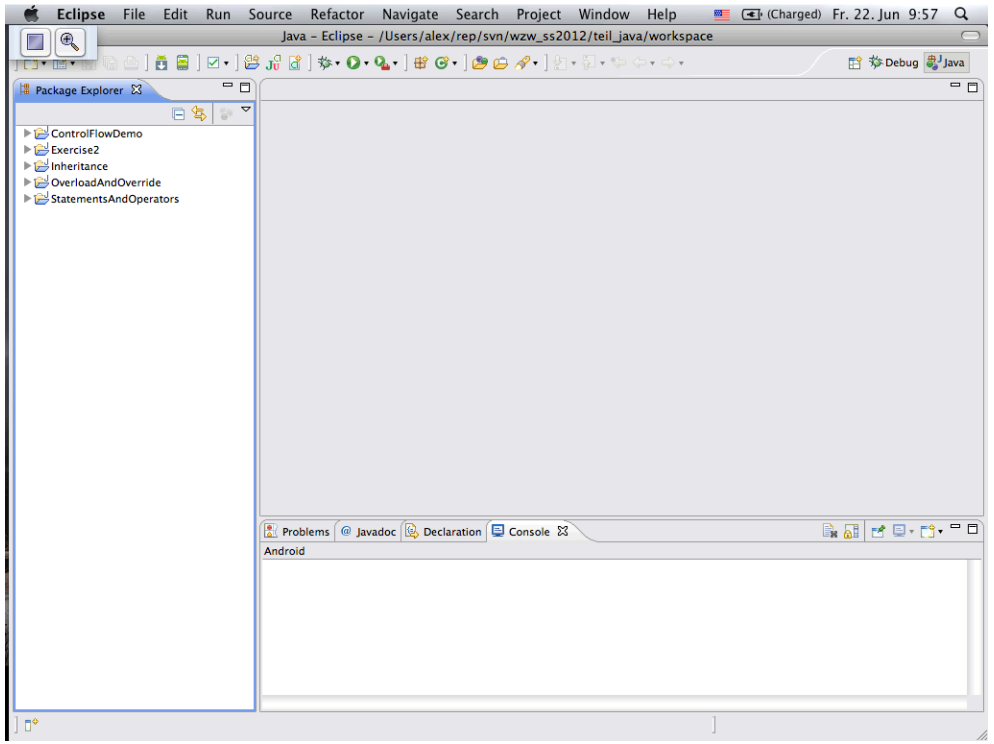
Calling methods

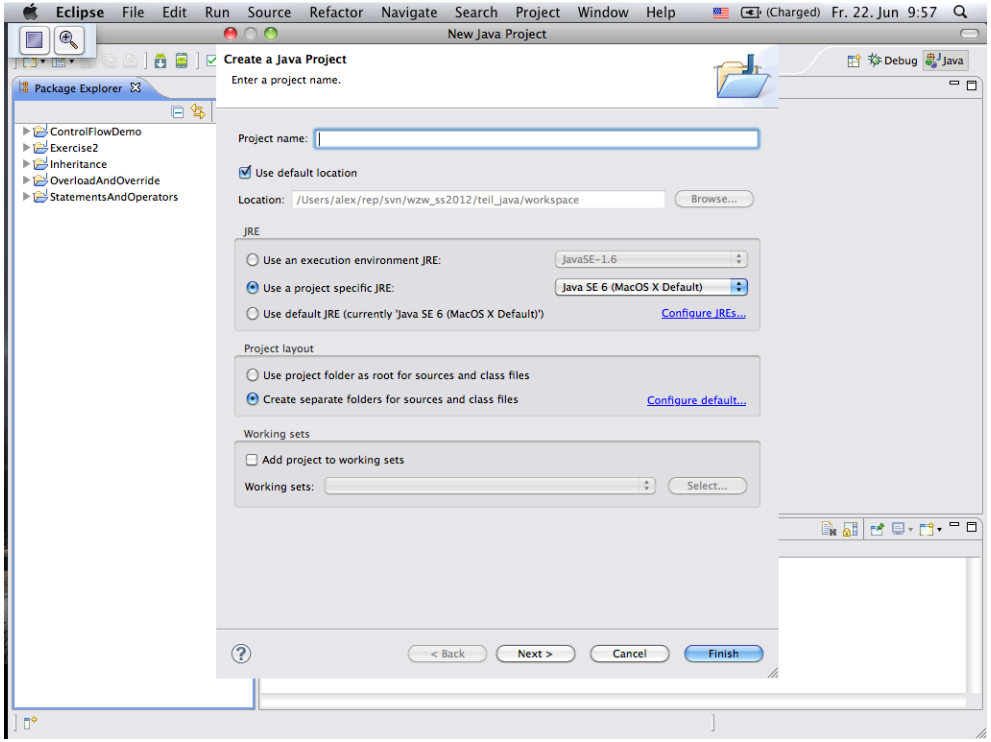
- Methods can be called from inside and outside a class:

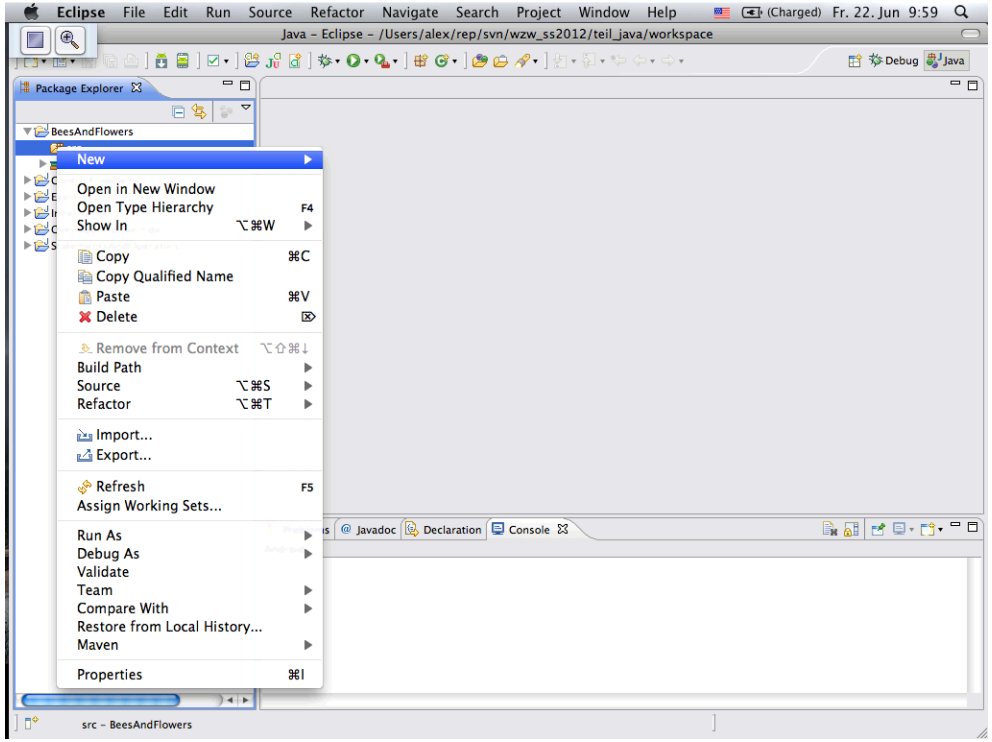
```
public class Bicycle {
  public int cadence = 0;

  public void changeCadence(int newCadence) {
    cadence = newCadence;                  // also: this.cadence
  }

  public void someArbitraryNonsenseMethod() {
    changeCadence(5);                      // also: this.changeCadence
  }
}
```

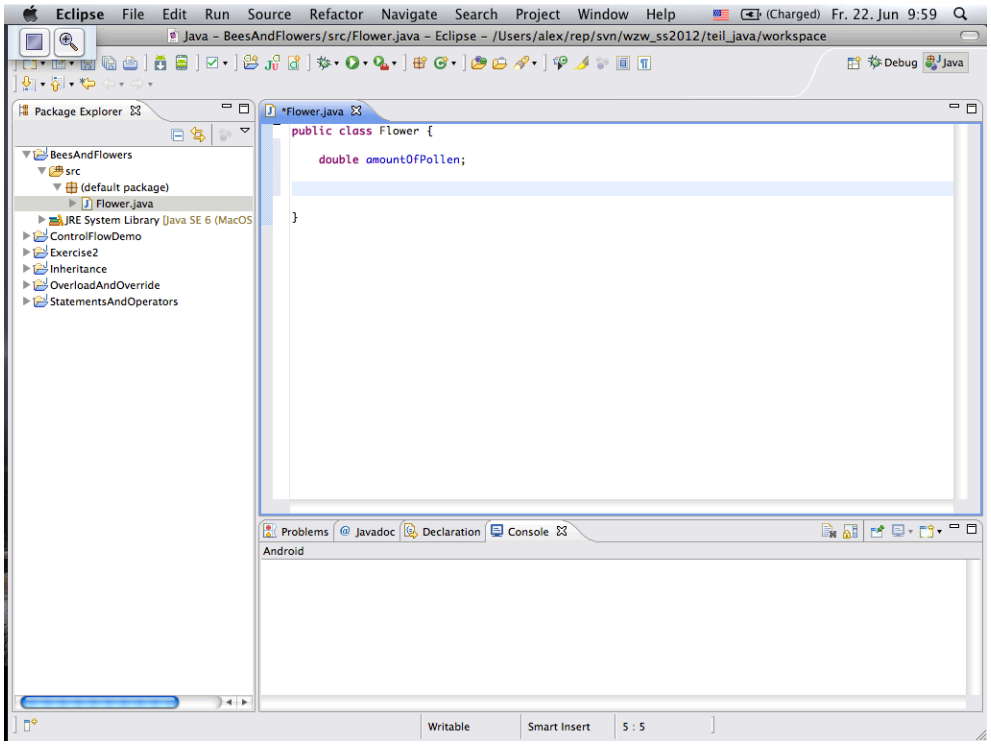
```
public static void main(String[] args) {
  Bicycle bike = new Bicycle();
  bike.changeCadence(10);
  // bike.cadence == 10;
  bike.someArbitraryNonsenseMethod();
  // bike.cadence == 5;
}
```

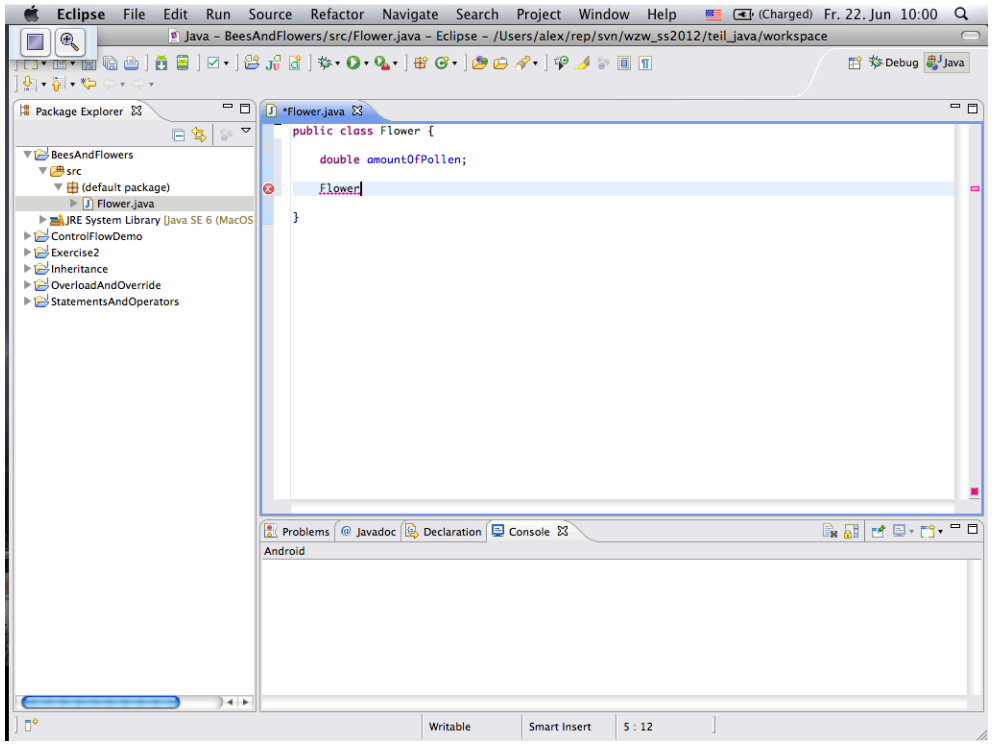
- If needed, objects may refer to themselves as `this`

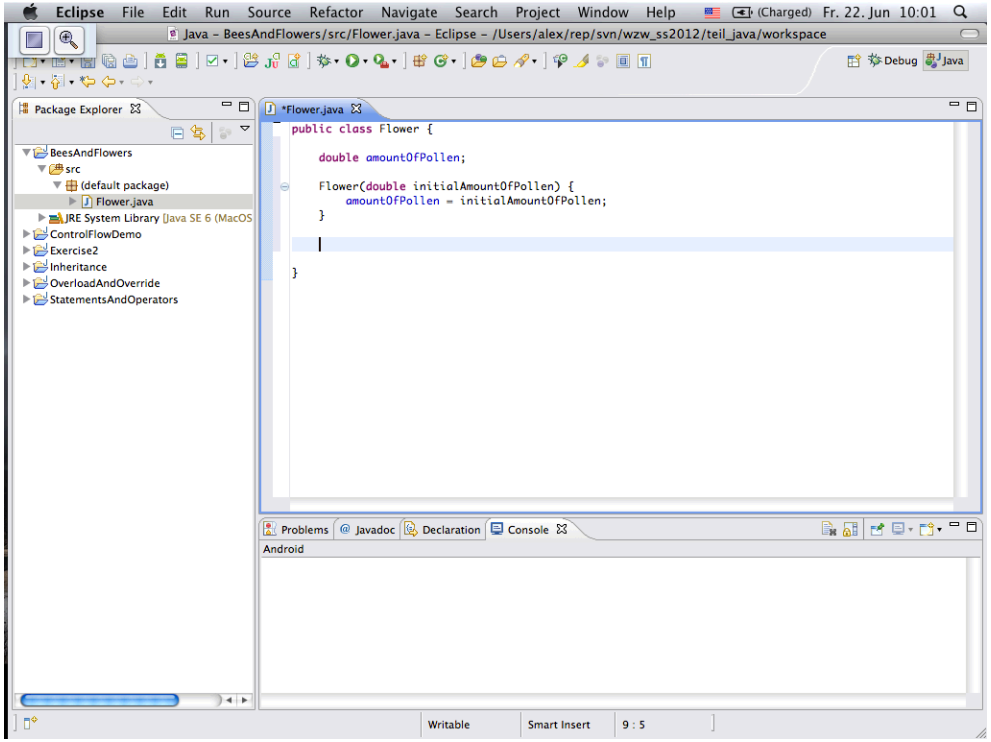


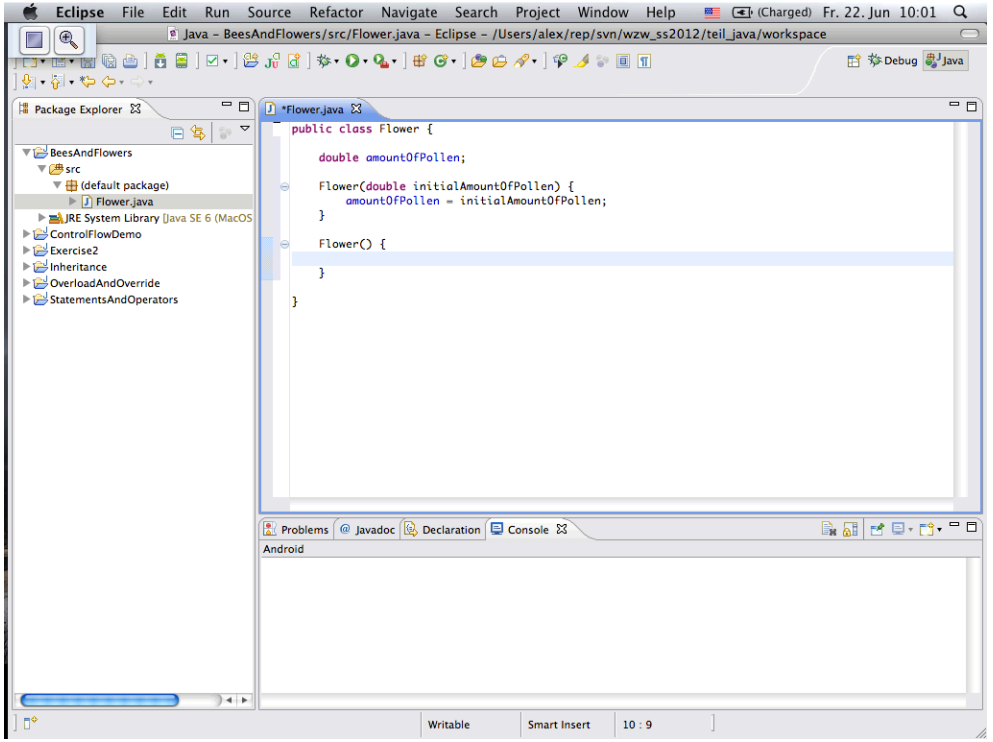

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 9:57
New Java Project
Create a Java Project
Enter a project name.
Package Explorer
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Project name:
Use default location
Location: /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Browse...
JRE
Use an execution environment JRE: JavaSE-1.6
Use a project specific JRE: Java SE 6 (MacOS X Default)
Use default JRE (currently 'Java SE 6 (MacOS X Default)')
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 >
Cancel
Finish

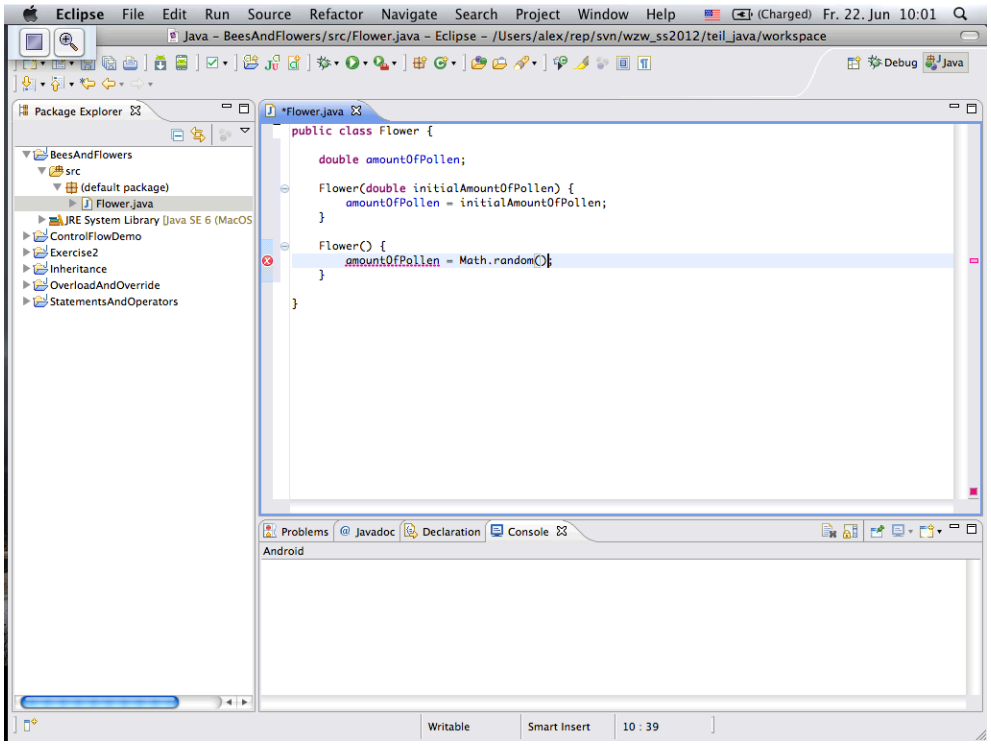

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 9:59
Java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
New
Open in New Window
Open Type Hierarchy F4
Show In ⌥⌘W
Copy ⌘C
Copy Qualified Name
Paste ⌘V
Delete
Remove from Context ⌥⇧⌘↓
Build Path
Source ⌥⌘S
Refactor ⌥⌘T
Import...
Export...
Refresh F5
Assign Working Sets...
Run As
Debug As
Validate
Team
Compare With
Restore from Local History...
Maven
Properties ⌘I
Javadoc Declaration Console
src – BeesAndFlowers

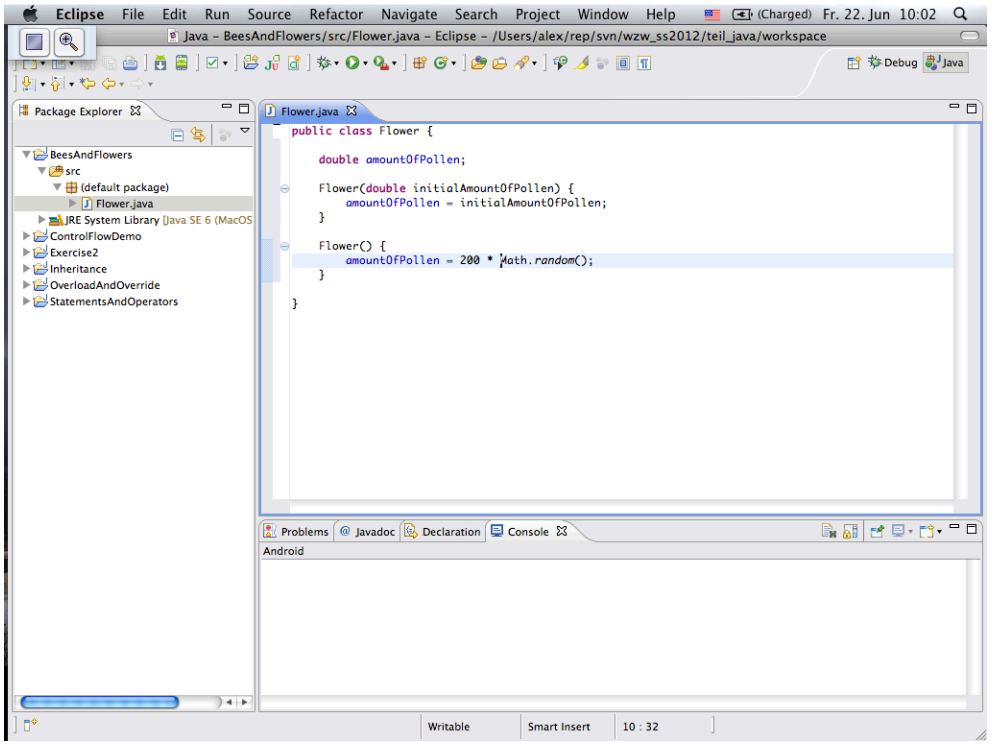

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 9:59
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
*Flower.java
public class Flower {
double amountOfPollen;
}
Problems Javadoc Declaration Console
Android
Writable
Smart Insert
5 : 5

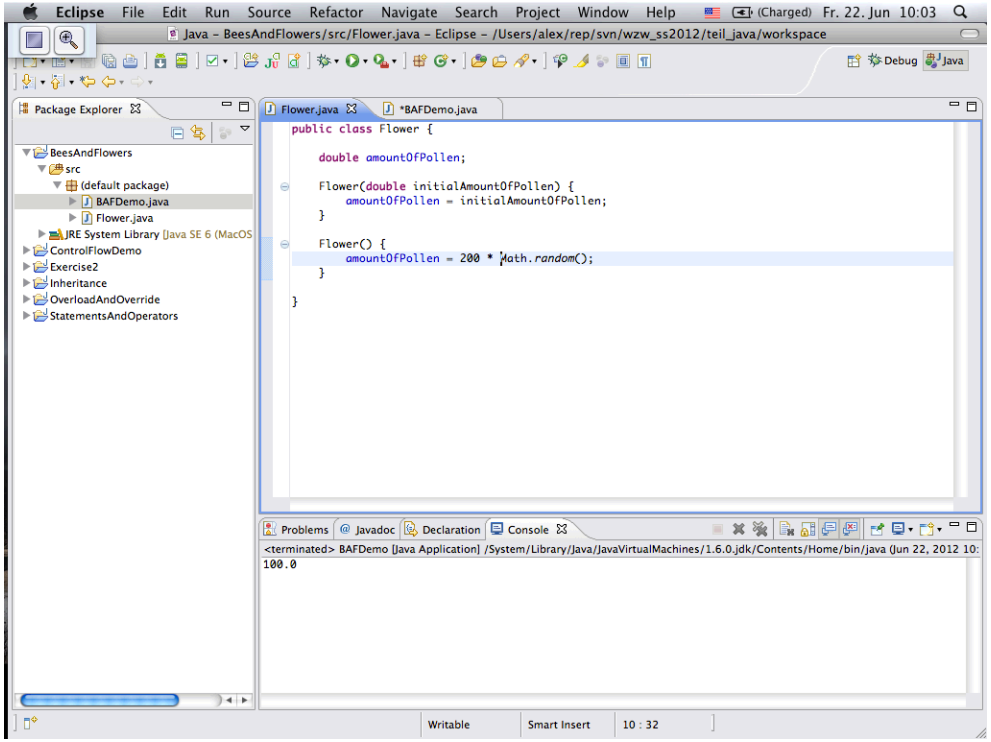

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:00
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
*Flower.java
public class Flower {
double amountOfPollen;
Flower
}
Problems Javadoc Declaration Console
Android
Writable
Smart Insert
5 : 12

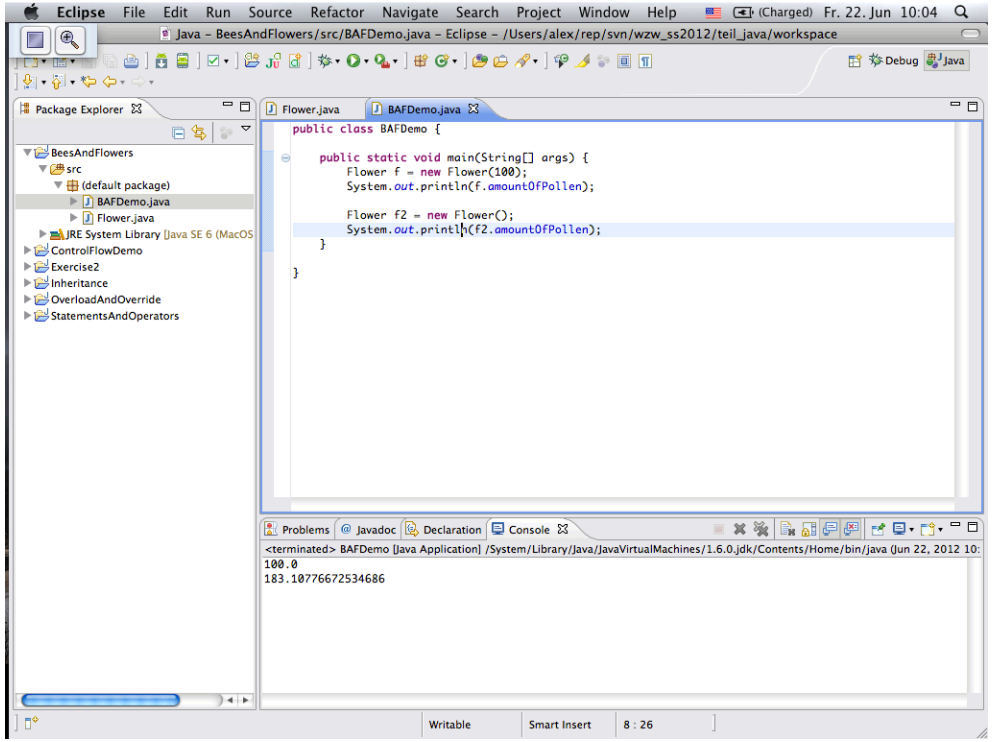

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:01
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
*Flower.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
}
Problems Javadoc Declaration Console
Android
Writable Smart Insert 9 : 5

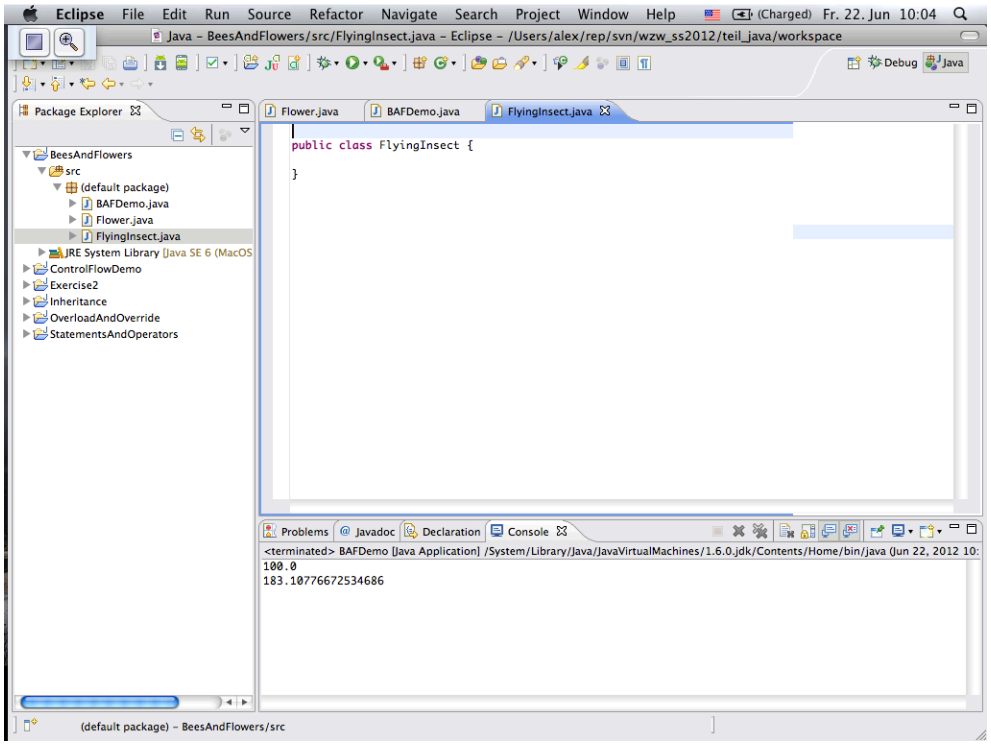

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:01
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
*Flower.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
}
}
Problems Javadoc Declaration Console
Android
Writable Smart Insert 10 : 9

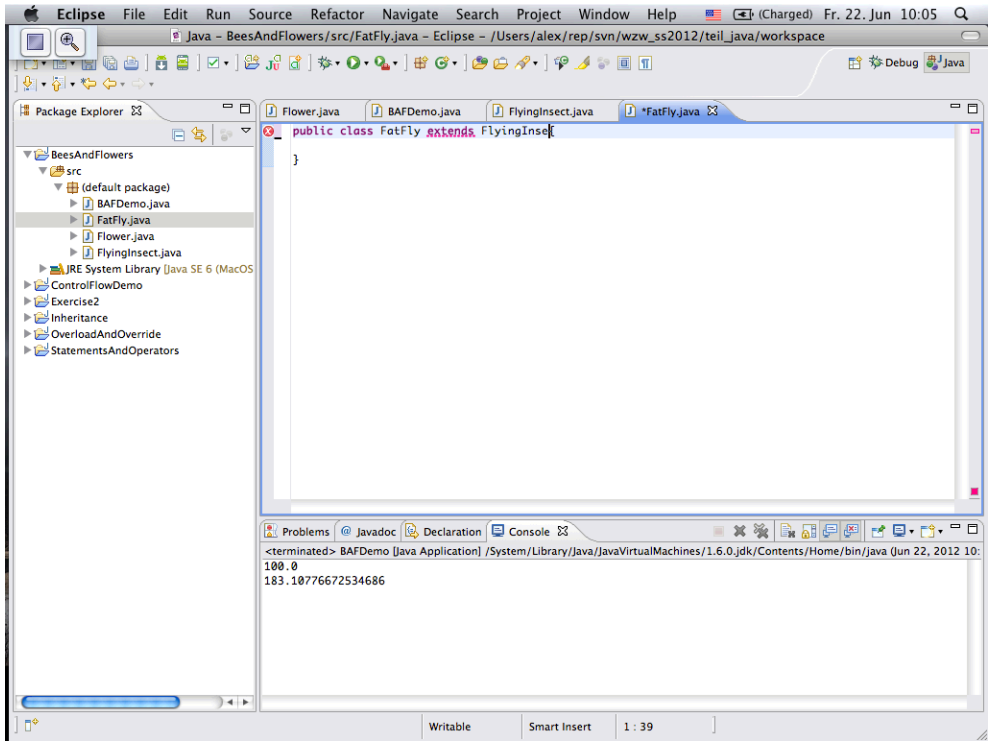

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:01
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
*Flower.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = Math.random();
}
}
Problems Javadoc Declaration Console
Android
Writable Smart Insert 10 : 39

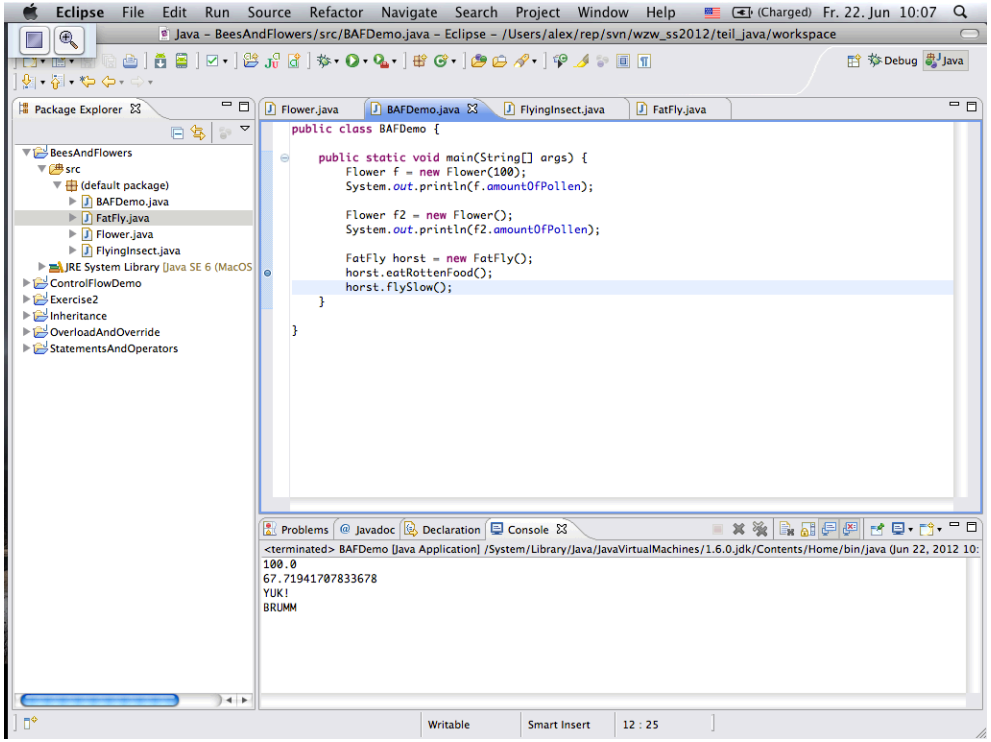

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:02
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
}
Problems Javadoc Declaration Console
Android
Writable Smart Insert 10 : 32

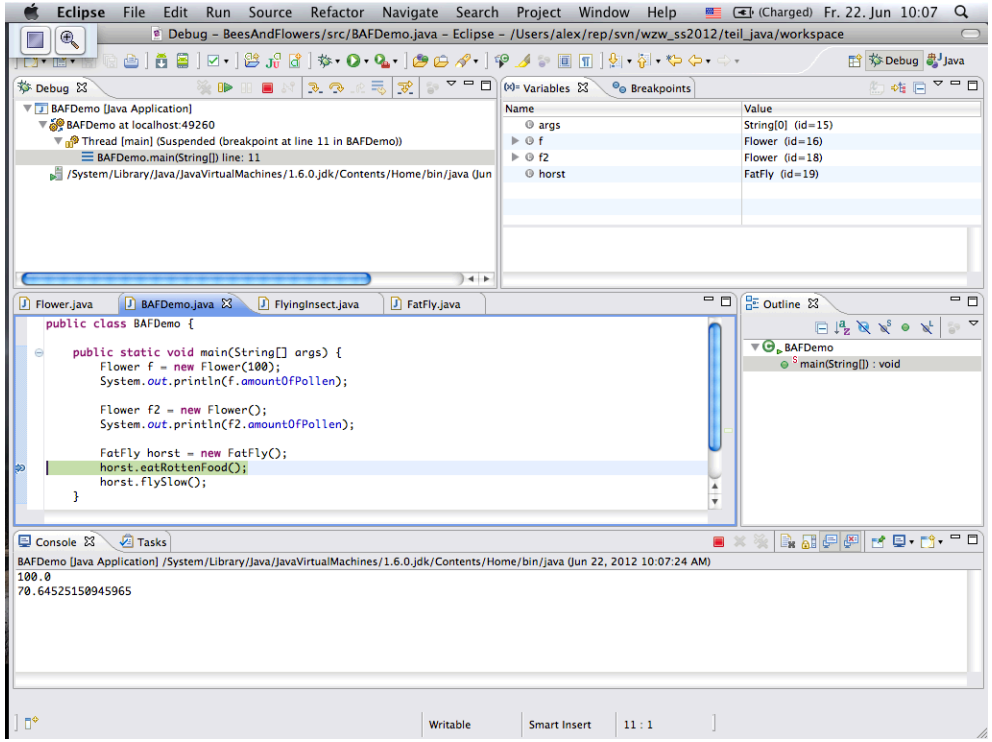

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:03
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
*BAFDemo.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
Writable Smart Insert 10 : 32

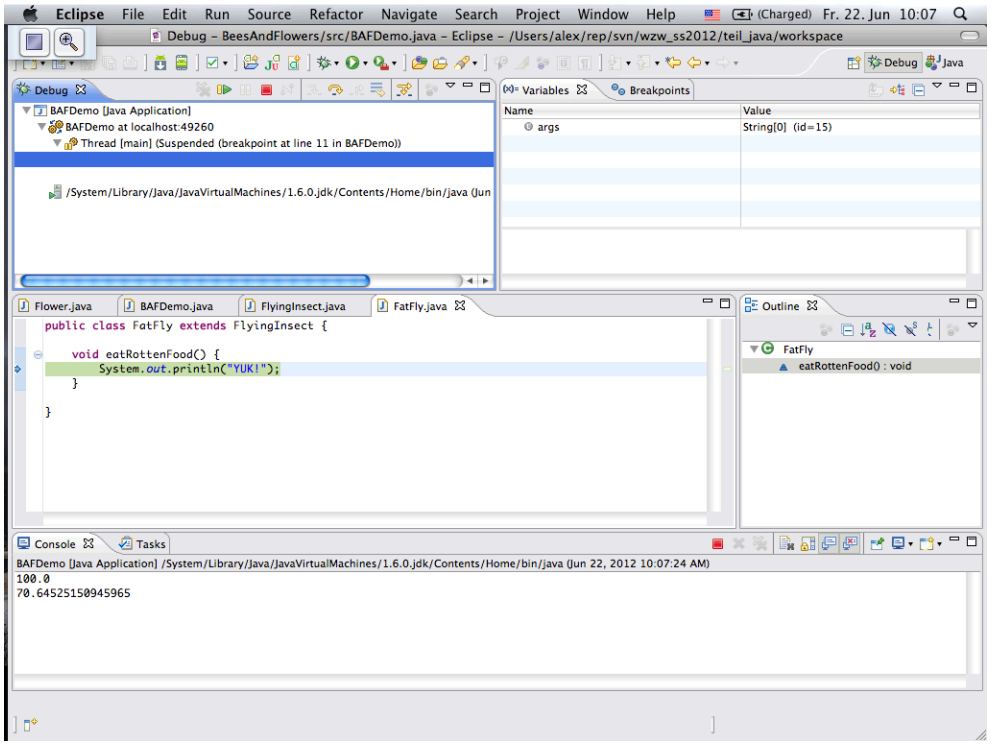

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:04
Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
Flower.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
183.10776672534686
Writable Smart Insert 8 : 26

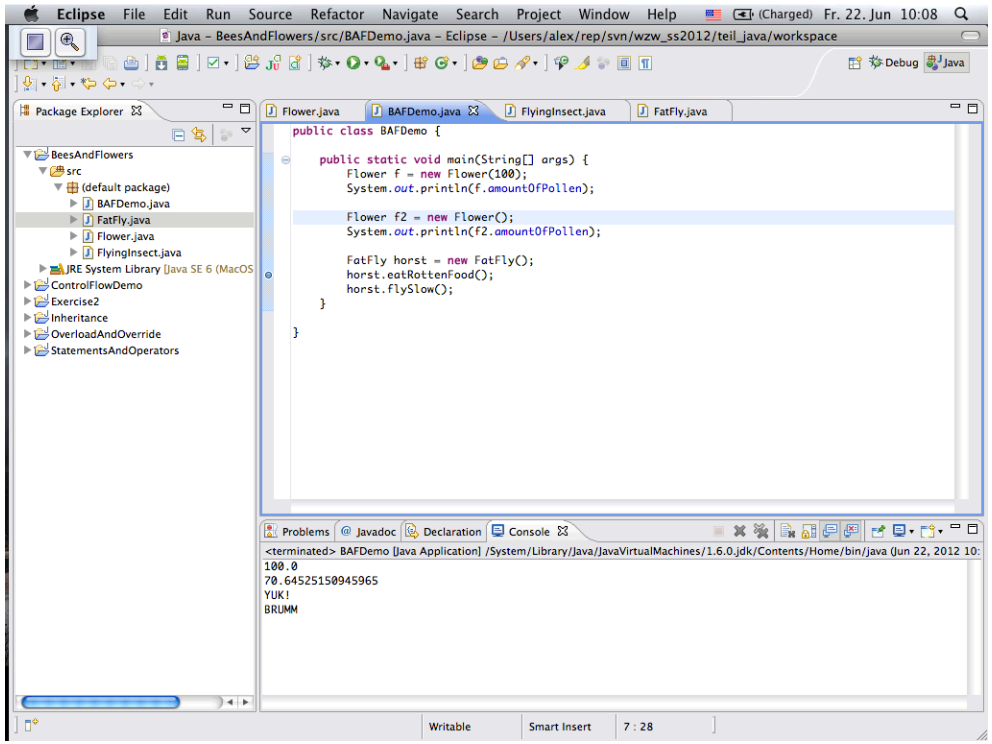

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:04
Java – BeesAndFlowers/src/FlyingInsect.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
Flower.java
FlyingInsect.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
public class FlyingInsect {
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
183.10776672534686
(default package) – BeesAndFlowers/src

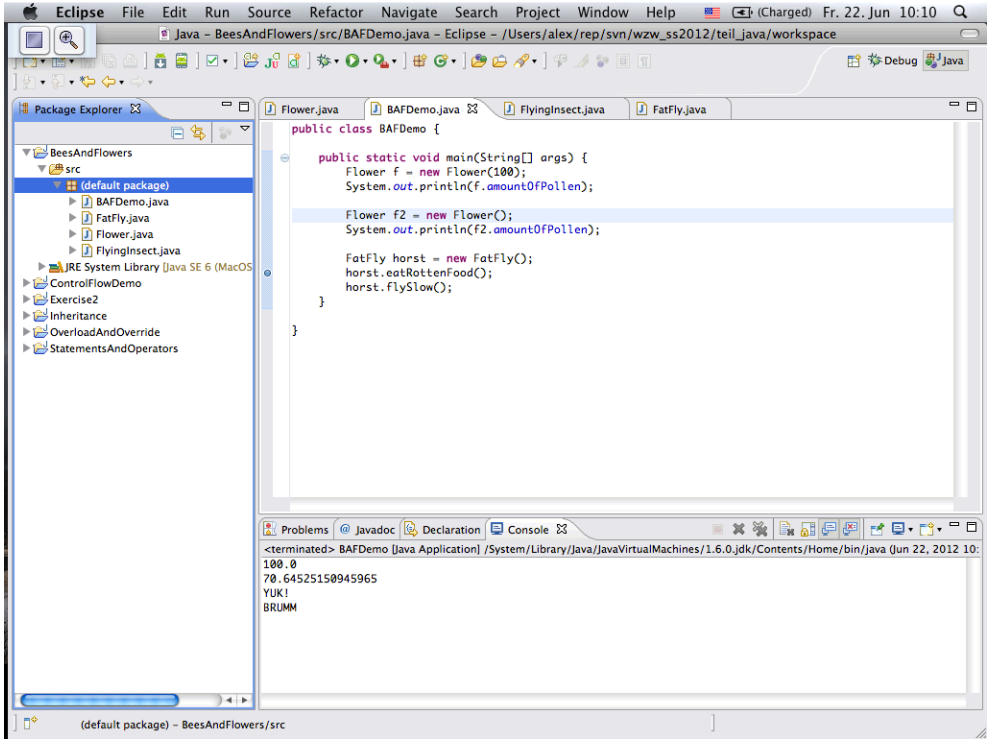

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:05
Java – BeesAndFlowers/src/FatFly.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
*FatFly.java
public class FatFly extends FlyingInse
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
183.10776672534686
Writable Smart Insert 1 : 39

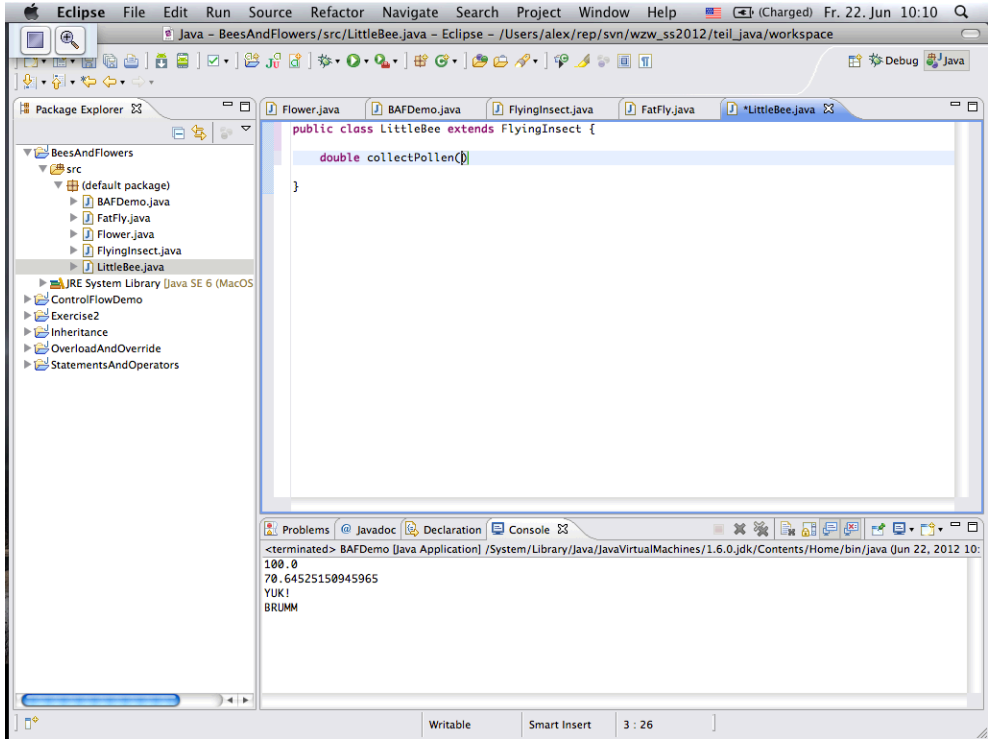

Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
}
}
Problems
Javadoc
Declaration
Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
67.71941707833678
YUK!
BRUMM
Writable
Smart Insert
12 : 25

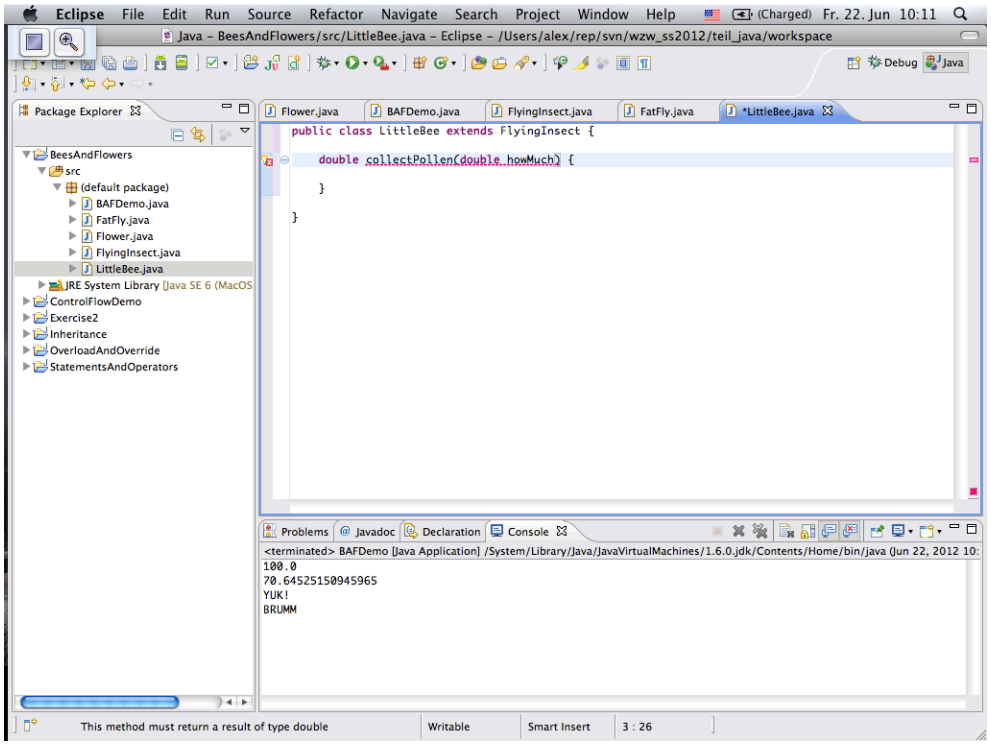

Debug – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Debug
BAFDemo [Java Application]
BAFDemo at localhost:49260
Thread [main] (Suspended (breakpoint at line 11 in BAFDemo))
BAFDemo.main(String[]) line: 11
/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun
Variables
Breakpoints
Name
Value
args
String[0] (id=15)
f
Flower (id=16)
f2
Flower (id=18)
horst
FatFly (id=19)
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
}
Outline
BAFDemo
main(String[]) : void
Console
Tasks
BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:07:24 AM)
100.0
70.64525150945965
Writable
Smart Insert
11 : 1

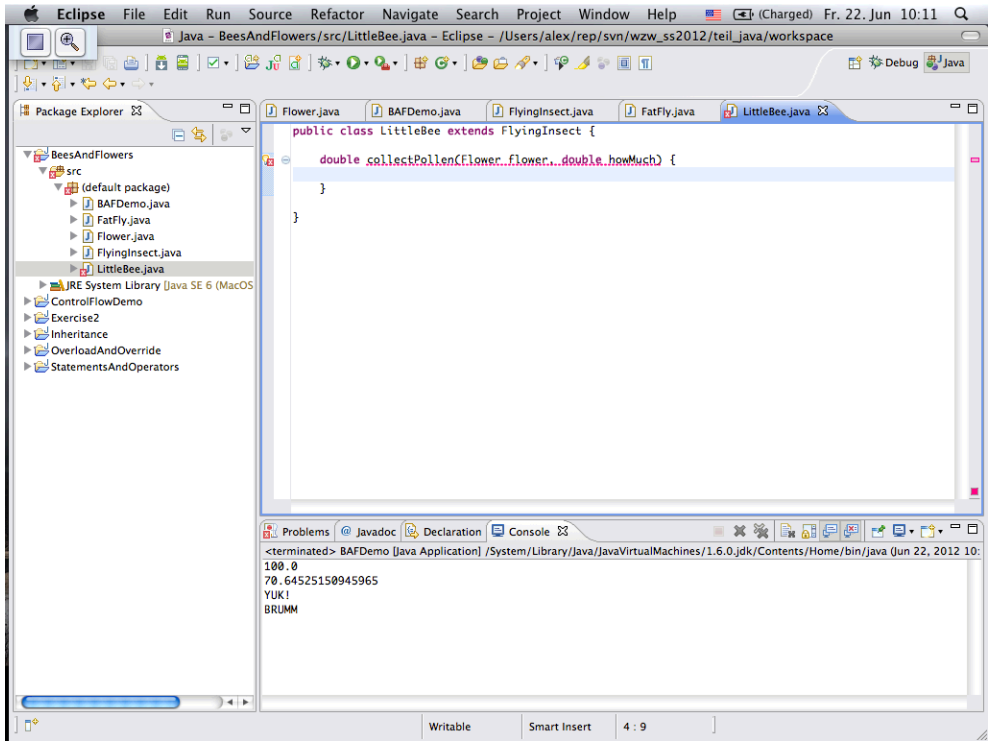

Debug – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Debug
BAFDemo [Java Application]
BAFDemo at localhost:49260
Thread [main] (Suspended (breakpoint at line 11 in BAFDemo))
/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun
Variables
Breakpoints
Name
Value
args
String[0] (id=15)
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
public class FatFly extends FlyingInsect {
void eatRottenFood() {
System.out.println("YUK!");
}
}
Outline
FatFly
eatRottenFood() : void
Console
Tasks
BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:07:24 AM)
100.0
70.64525150945965

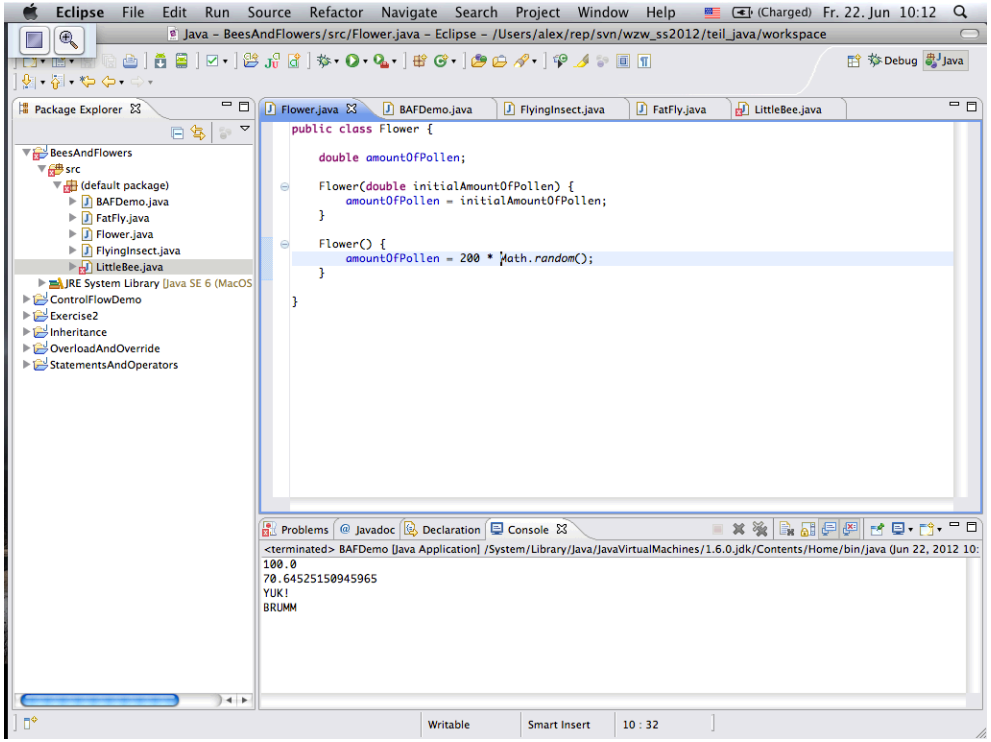

Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
}
}
Problems
Javadoc
Declaration
Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable
Smart Insert
7 : 28

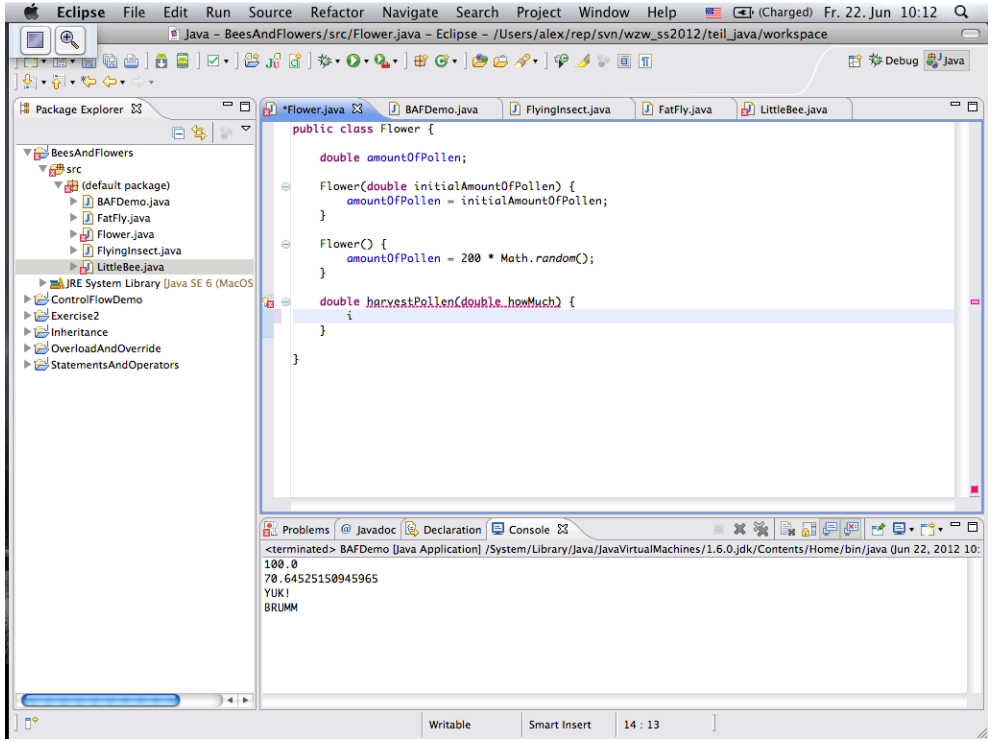

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:10
Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
(default package) – BeesAndFlowers/src

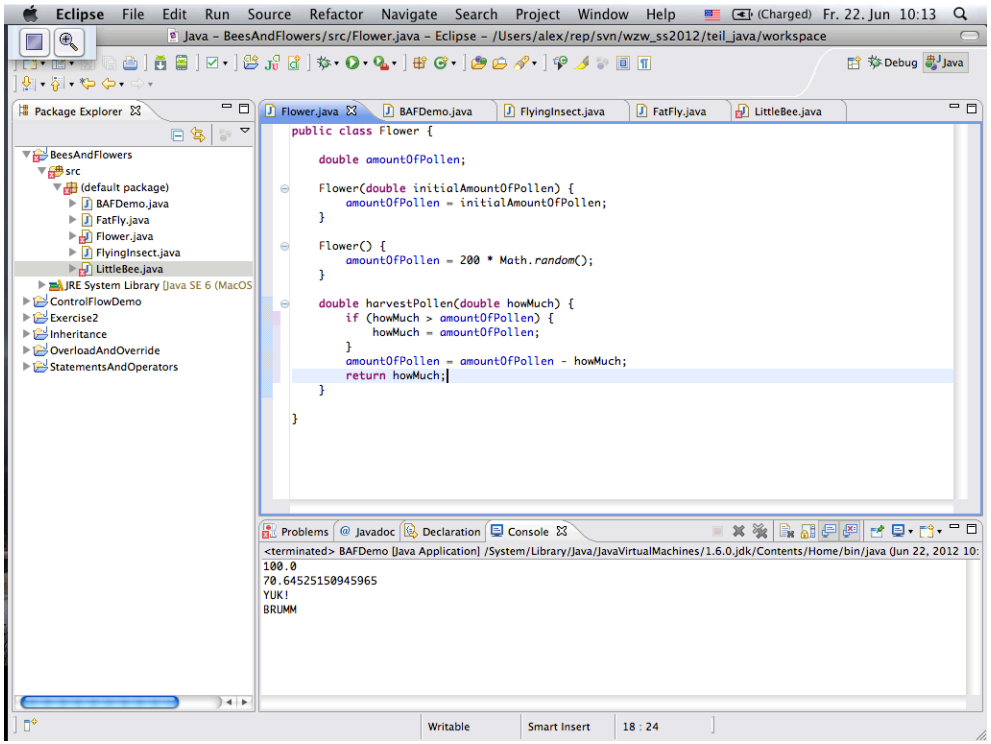

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:10
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
*LittleBee.java
public class LittleBee extends FlyingInsect {
double collectPollen()
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable
Smart Insert
3 : 26

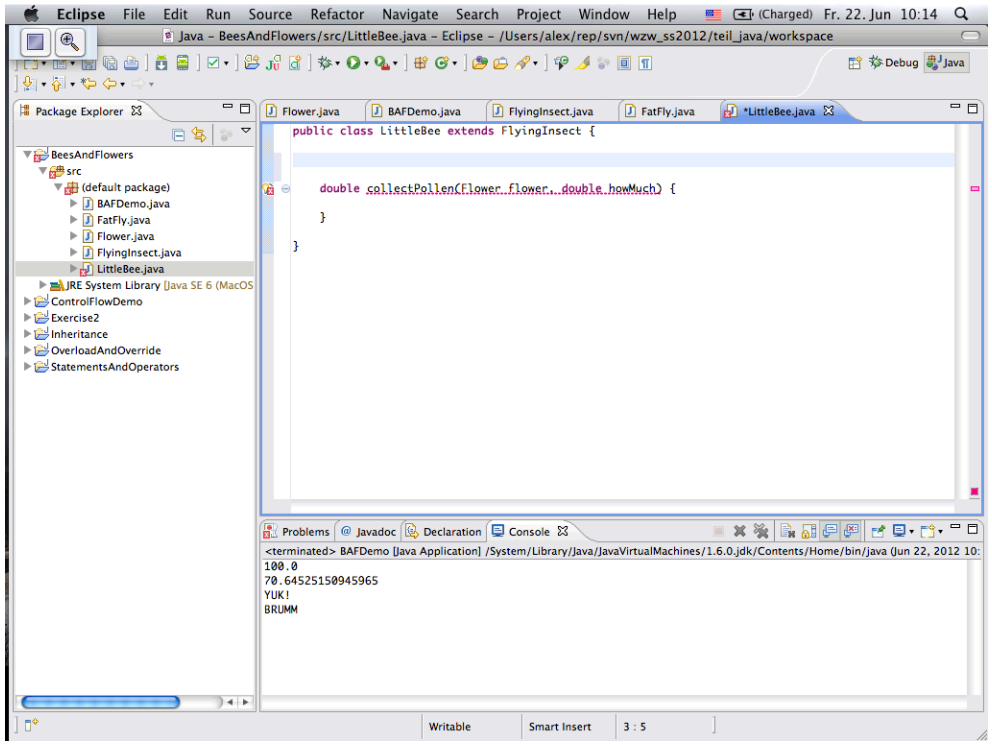

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:11
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
*LittleBee.java
public class LittleBee extends FlyingInsect {
double collectPollen(double howMuch) {
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
This method must return a result of type double
Writable
Smart Insert
3 : 26

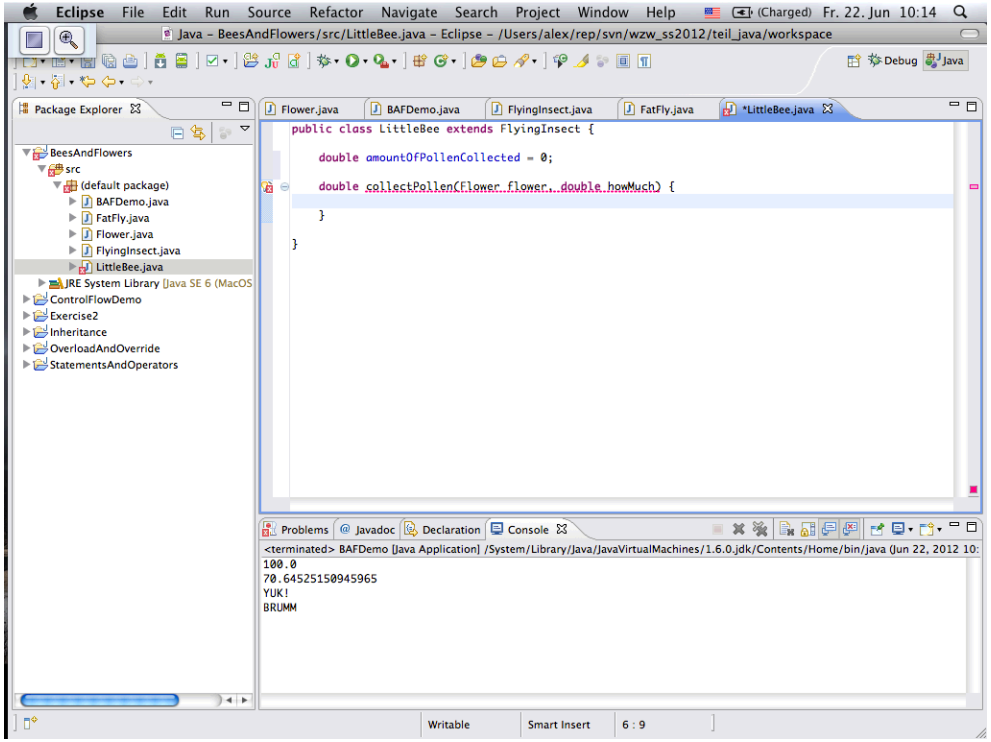

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:11
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
LittleBee.java
public class LittleBee extends FlyingInsect {
double collectPollen(Flower flower, double howMuch) {
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable
Smart Insert
4 : 9

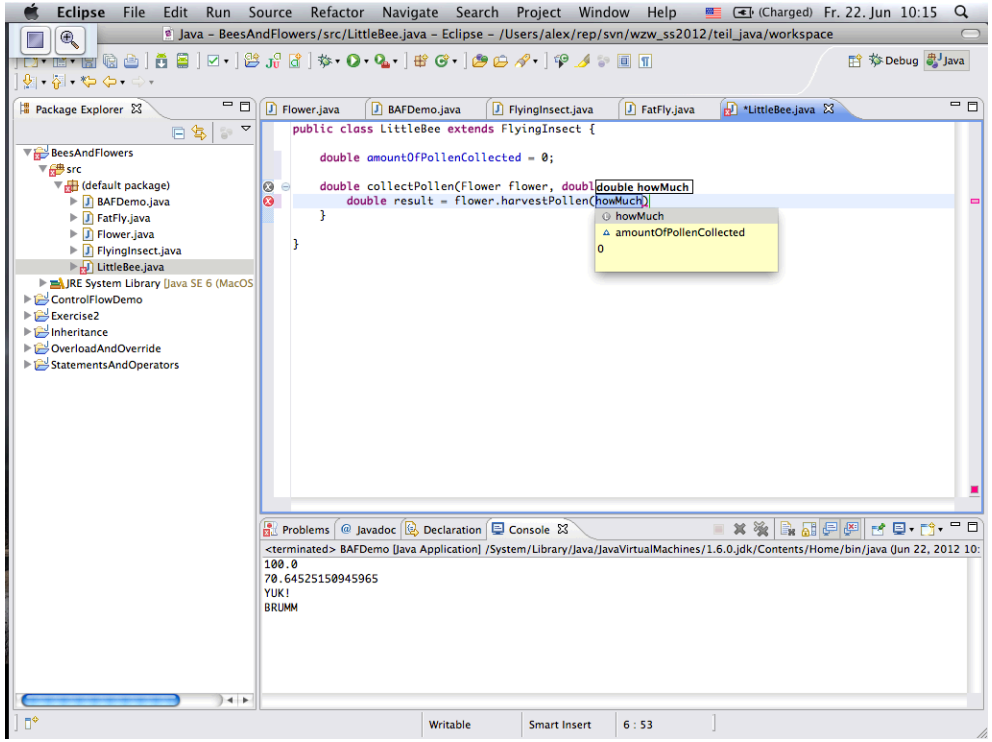

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:12
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
LittleBee.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable Smart Insert 10 : 32

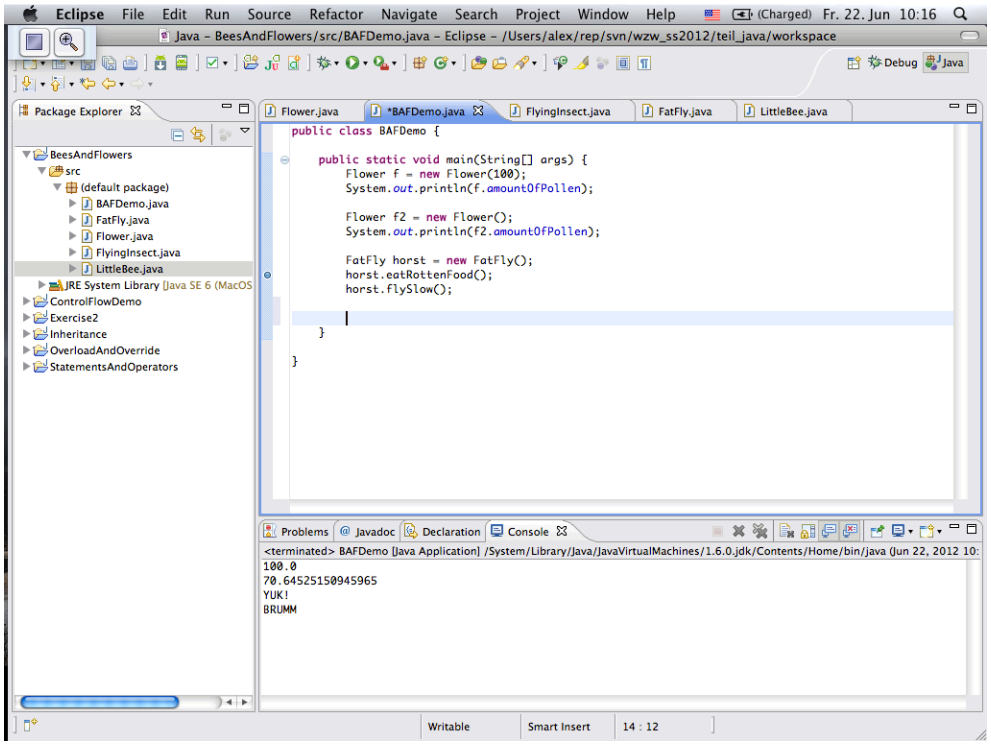

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:12
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
*Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
LittleBee.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
double harvestPollen(double howMuch) {
i
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable Smart Insert 14 : 13

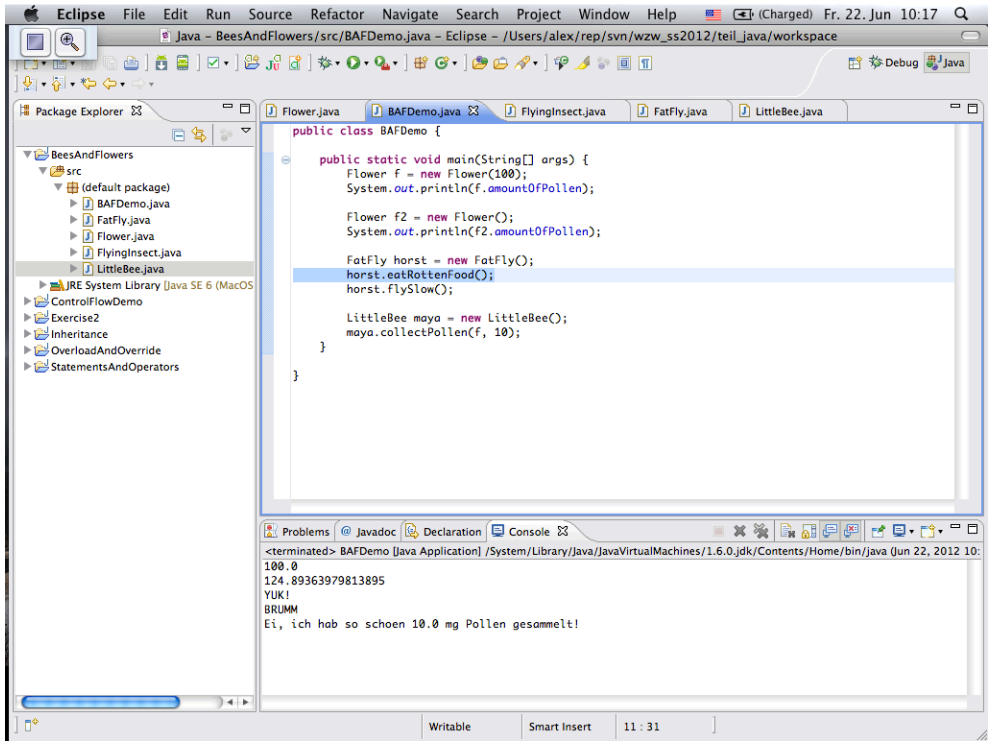

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:13
Java – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
LittleBee.java
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
double harvestPollen(double howMuch) {
if (howMuch > amountOfPollen) {
howMuch = amountOfPollen;
}
amountOfPollen = amountOfPollen - howMuch;
return howMuch;
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable Smart Insert 18 : 24

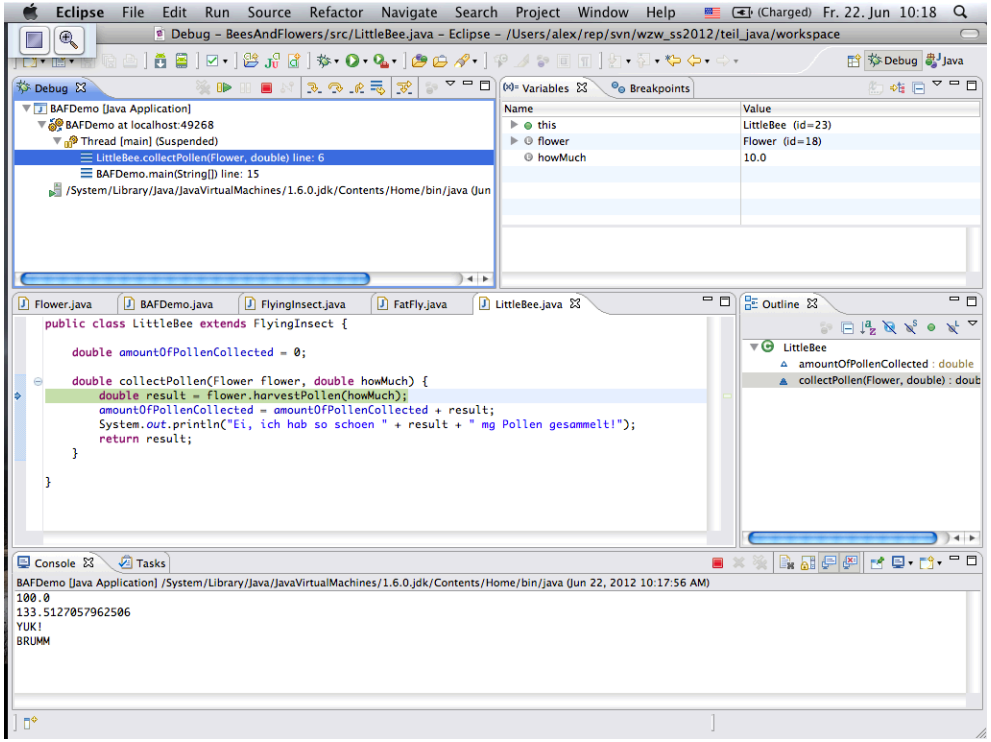

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:14
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
*LittleBee.java
public class LittleBee extends FlyingInsect {
double collectPollen(Flower flower, double howMuch) {
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable Smart Insert 3 : 5

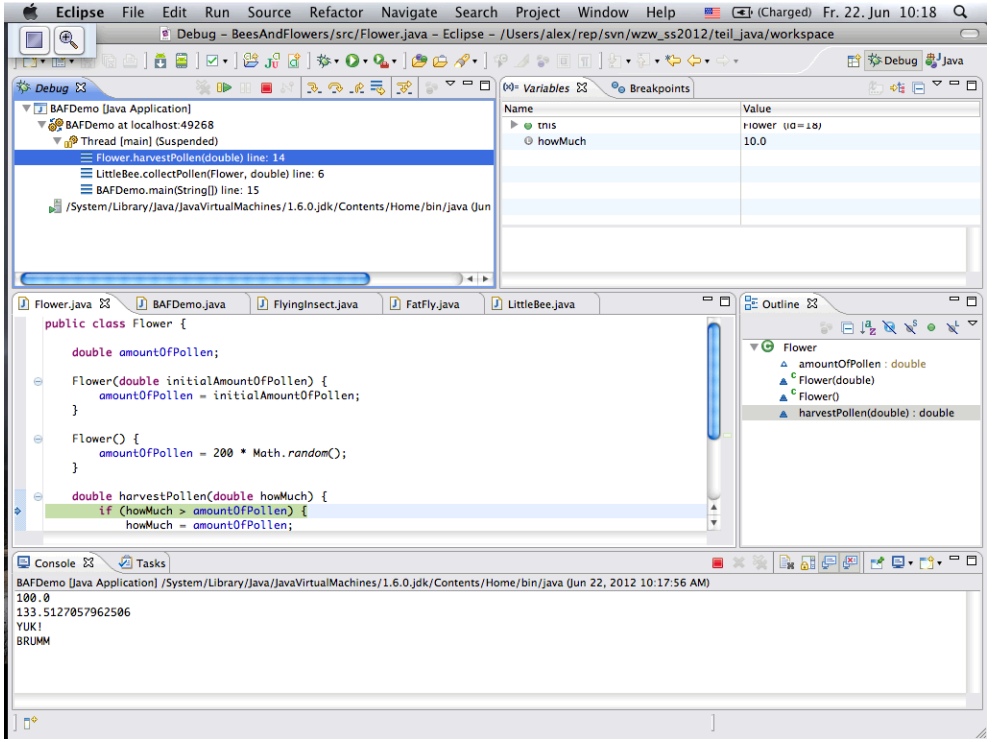

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:14
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
*LittleBee.java
public class LittleBee extends FlyingInsect {
double amountOfPollenCollected = 0;
double collectPollen(Flower flower, double howMuch) {
}
}
Problems
Javadoc
Declaration
Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable
Smart Insert
6 : 9

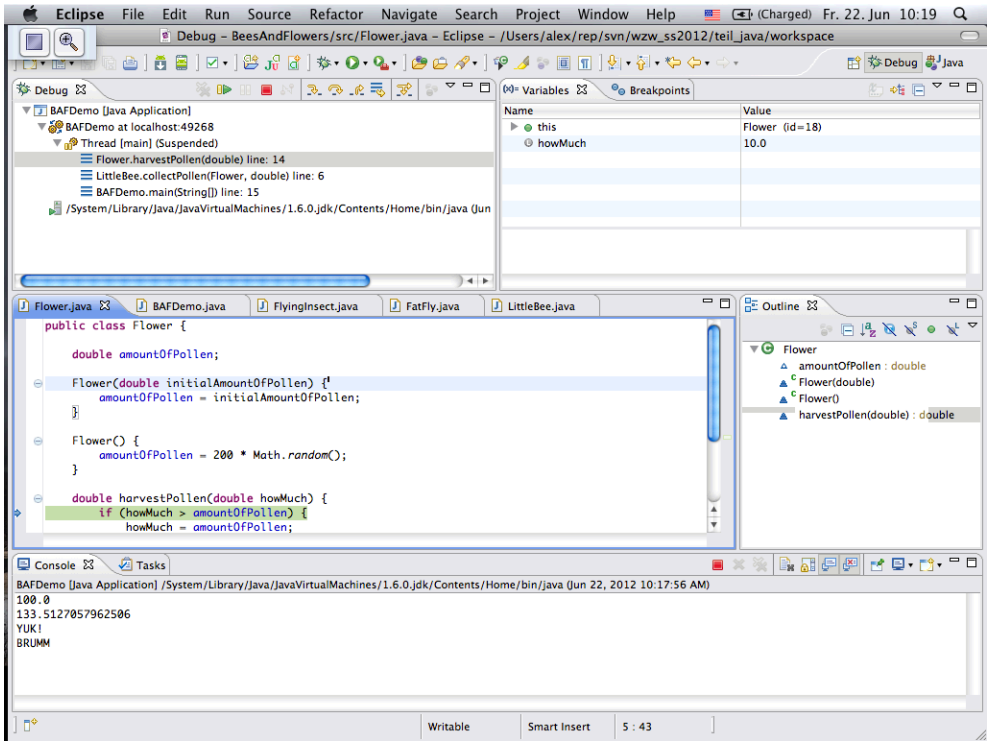

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:15
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
*LittleBee.java
public class LittleBee extends FlyingInsect {
double amountOfPollenCollected = 0;
double collectPollen(Flower flower, doubl
double howMuch
double result = flower.harvestPollen(howMuch);
howMuch
amountOfPollenCollected
0
}
Problems
Javadoc
Declaration
Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable
Smart Insert
6 : 53

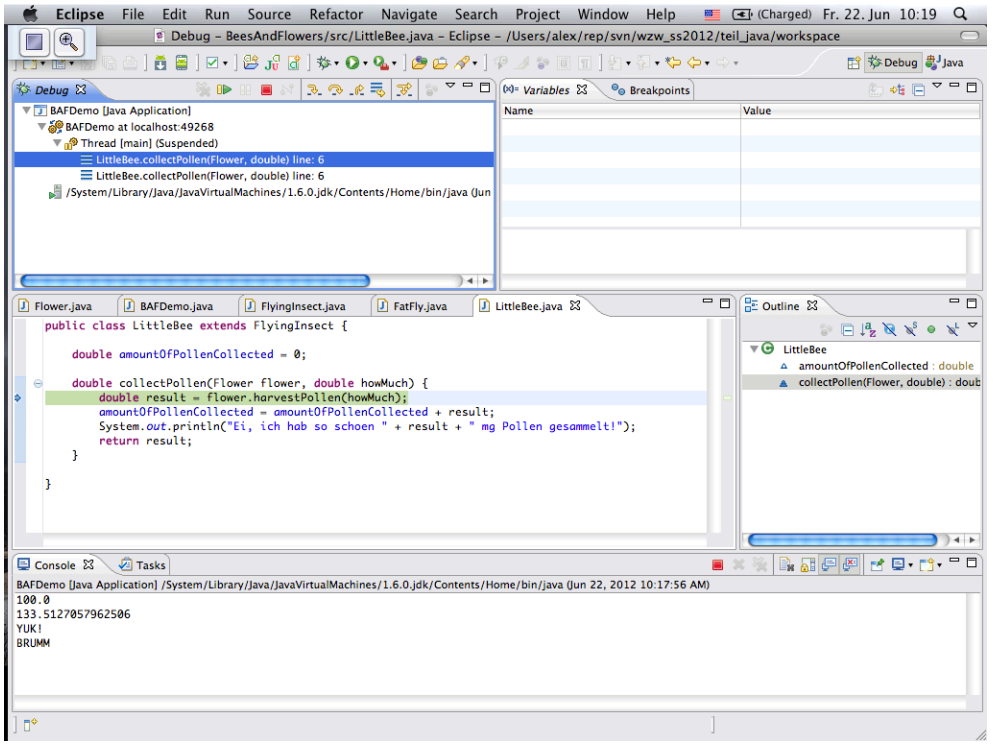

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:16
Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
*BAFDemo.java
FlyingInsect.java
FatFly.java
LittleBee.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
}
}
Problems
Javadoc
Declaration
Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
70.64525150945965
YUK!
BRUMM
Writable
Smart Insert
14 : 12

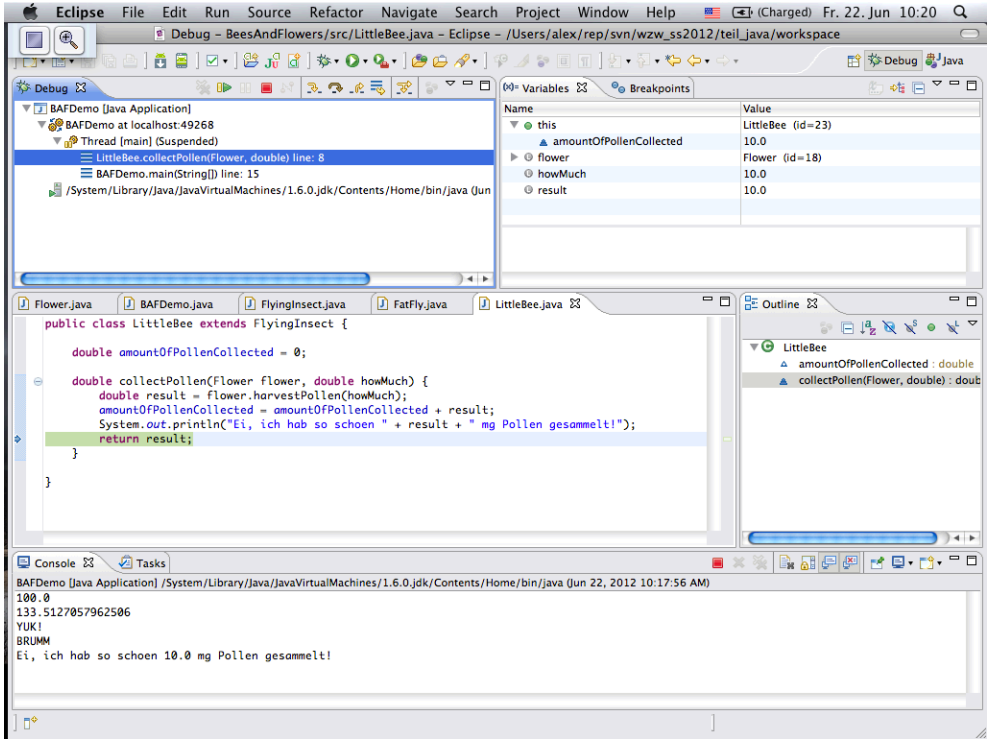

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:17
Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FlyingInsect.java
FatFly.java
LittleBee.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
LittleBee maya = new LittleBee();
maya.collectPollen(f, 10);
}
}
Problems
Javadoc
Declaration
Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
124.89363979813895
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Writable
Smart Insert
11 : 31

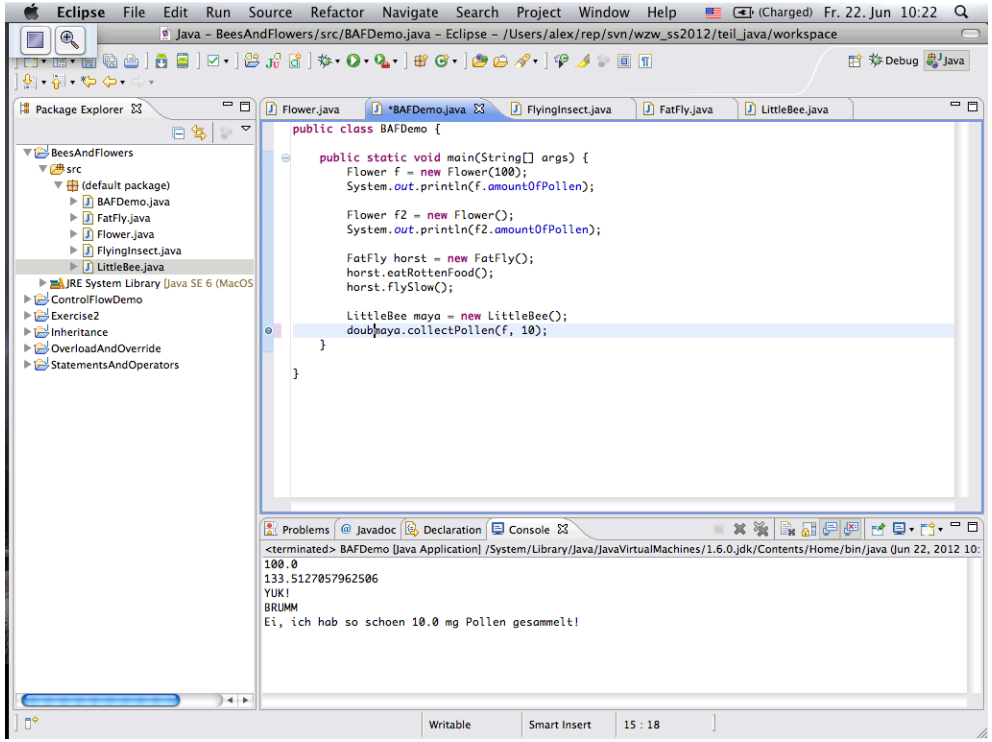

Debug – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
BAFDemo [Java Application]
BAFDemo at localhost:49268
Thread [main] (Suspended)
LittleBee.collectPollen(Flower, double) line: 6
BAFDemo.main(String[]) line: 15
this LittleBee (id=23)
flower Flower (id=18)
howMuch 10.0
public class LittleBee extends FlyingInsect {
double amountOfPollenCollected = 0;
double collectPollen(Flower flower, double howMuch) {
double result = flower.harvestPollen(howMuch);
amountOfPollenCollected = amountOfPollenCollected + result;
System.out.println("Ei, ich hab so schoen " + result + " mg Pollen gesammelt!");
return result;
}
}
100.0
133.5127057962506
YUK!
BRUMM

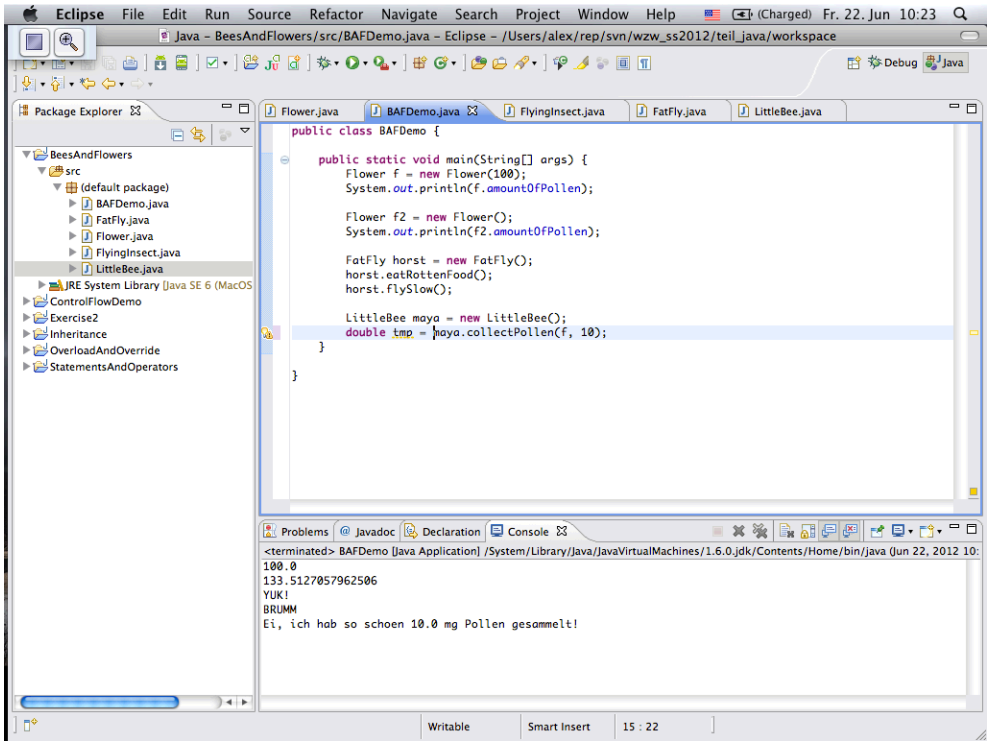

Debug – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
BAFDemo [Java Application]
BAFDemo at localhost:49268
Thread [main] (Suspended)
Flower.harvestPollen(double) line: 14
LittleBee.collectPollen(Flower, double) line: 6
BAFDemo.main(String[]) line: 15
howMuch 10.0
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
double harvestPollen(double howMuch) {
if (howMuch > amountOfPollen) {
howMuch = amountOfPollen;
100.0
133.5127057962506
YUK!
BRUMM

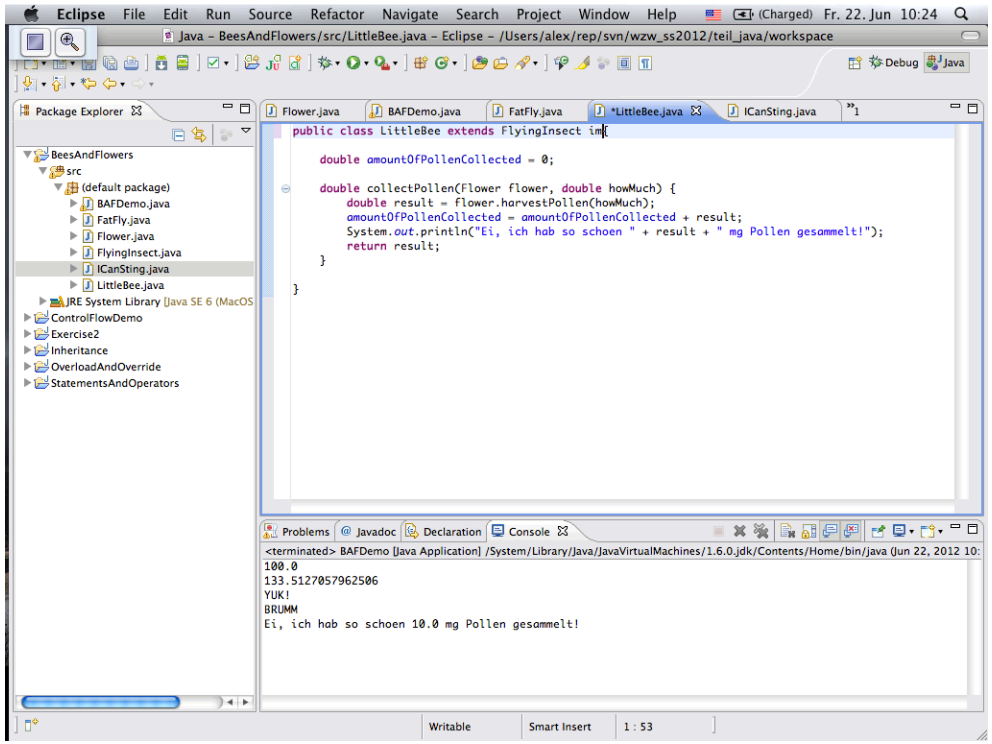

Debug – BeesAndFlowers/src/Flower.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
BAFDemo [Java Application]
BAFDemo at localhost:49268
Thread [main] (Suspended)
Flower.harvestPollen(double) line: 14
LittleBee.collectPollen(Flower, double) line: 6
BAFDemo.main(String[]) line: 15
this Flower (id=18)
howMuch 10.0
public class Flower {
double amountOfPollen;
Flower(double initialAmountOfPollen) {
amountOfPollen = initialAmountOfPollen;
}
Flower() {
amountOfPollen = 200 * Math.random();
}
double harvestPollen(double howMuch) {
if (howMuch > amountOfPollen) {
howMuch = amountOfPollen;
100.0
133.5127057962506
YUK!
BRUMM
Writable
Smart Insert
5 : 43

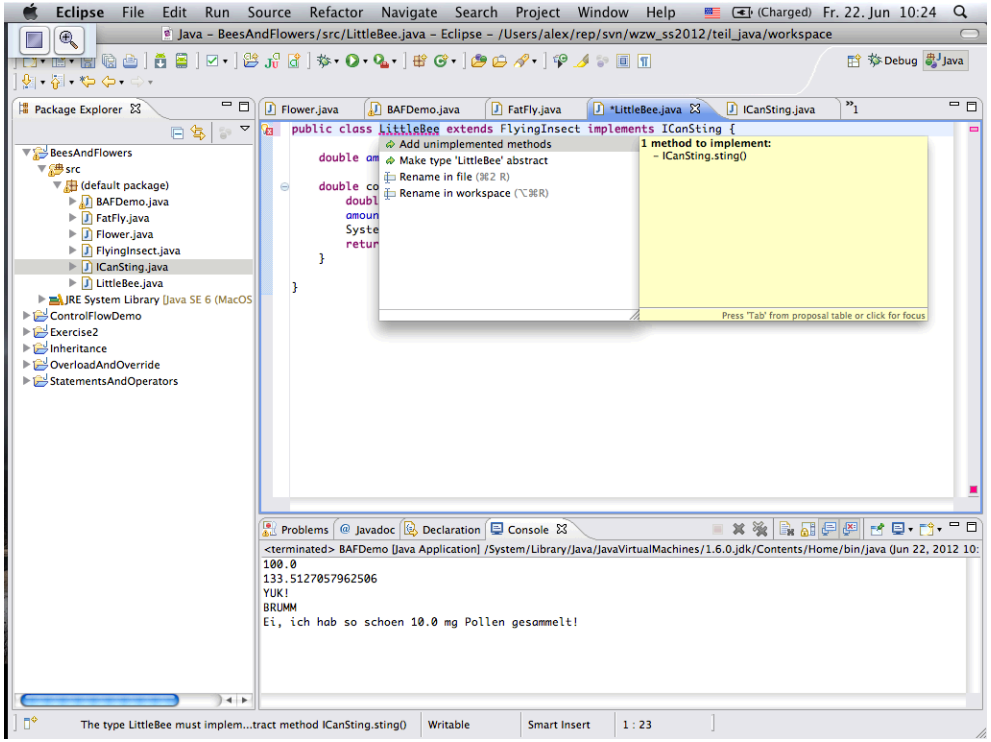

Debug – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
BAFDemo [Java Application]
BAFDemo at localhost:49268
Thread [main] (Suspended)
LittleBee.collectPollen(Flower, double) line: 6
LittleBee.collectPollen(Flower, double) line: 6
public class LittleBee extends FlyingInsect {
double amountOfPollenCollected = 0;
double collectPollen(Flower flower, double howMuch) {
double result = flower.harvestPollen(howMuch);
amountOfPollenCollected = amountOfPollenCollected + result;
System.out.println("Ei, ich hab so schoen " + result + " mg Pollen gesammelt!");
return result;
}
}
100.0
133.5127057962506
YUK!
BRUMM

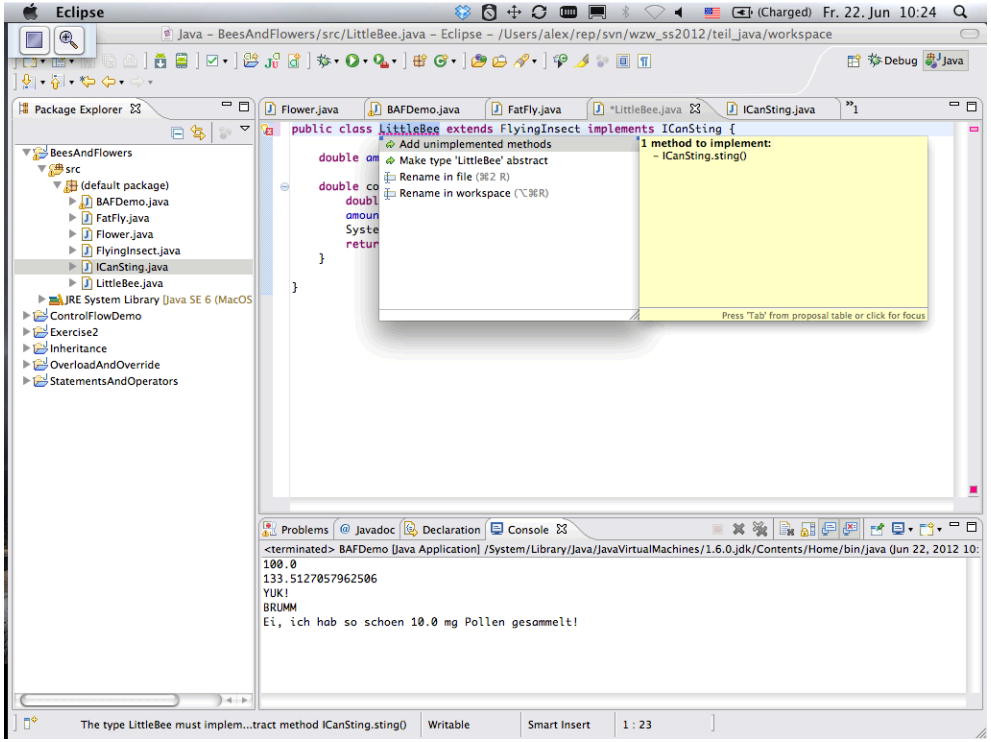

Debug – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
BAFDemo [Java Application]
BAFDemo at localhost:49268
Thread [main] (Suspended)
LittleBee.collectPollen(Flower, double) line: 8
BAFDemo.main(String[]) line: 15
Name
Value
this
LittleBee (id=23)
amountOfPollenCollected
10.0
flower
Flower (id=18)
howMuch
10.0
result
10.0
public class LittleBee extends FlyingInsect {
double amountOfPollenCollected = 0;
double collectPollen(Flower flower, double howMuch) {
double result = flower.harvestPollen(howMuch);
amountOfPollenCollected = amountOfPollenCollected + result;
System.out.println("Ei, ich hab so schoen " + result + " mg Pollen gesammelt!");
return result;
}
}
BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:17:56 AM)
100.0
133.5127057962506
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!

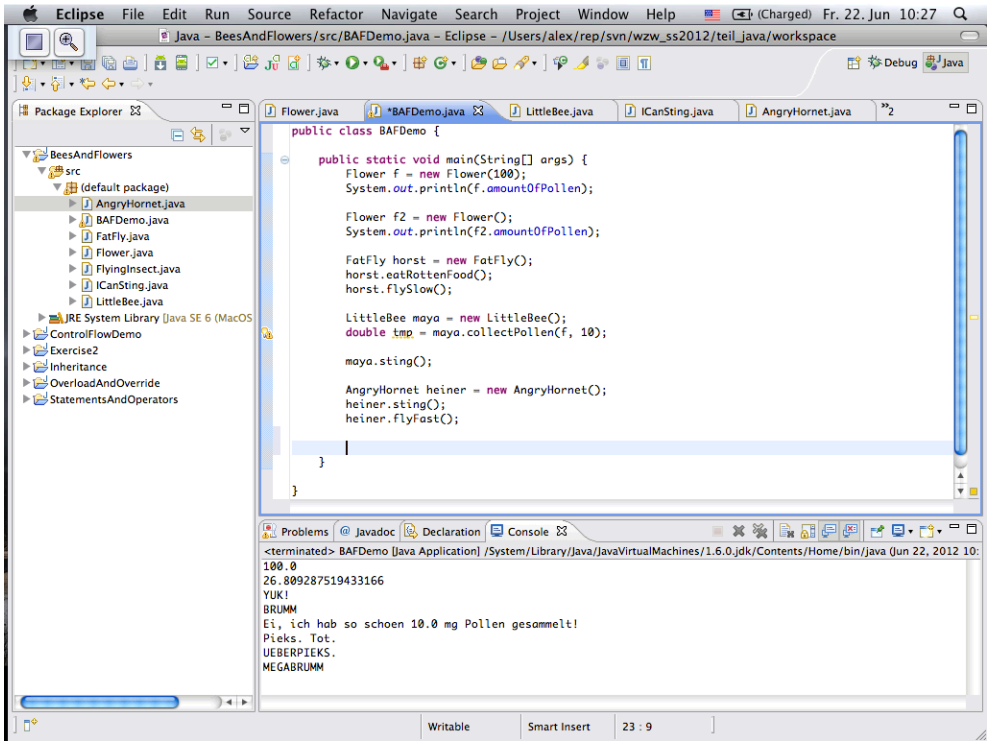

Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
LittleBee maya = new LittleBee();
doub maya.collectPollen(f, 10);
}
}
100.0
133.5127057962506
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Writable
Smart Insert
15 : 18

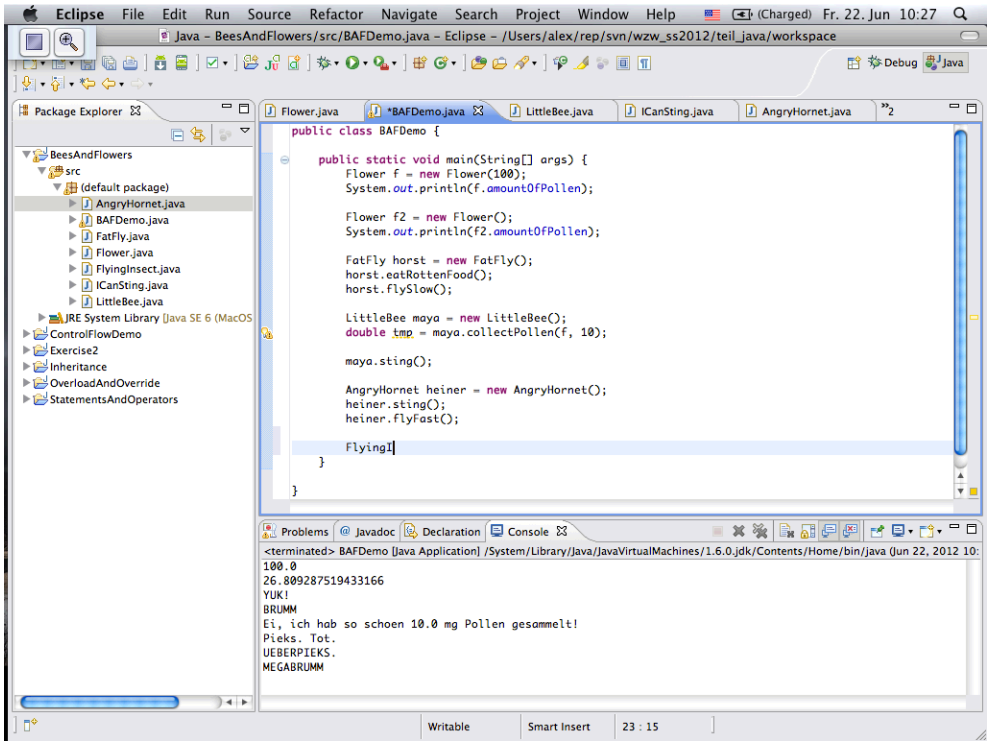

Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
LittleBee maya = new LittleBee();
double tmp = maya.collectPollen(f, 10);
}
}
100.0
133.5127057962506
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Writable
Smart Insert
15 : 22

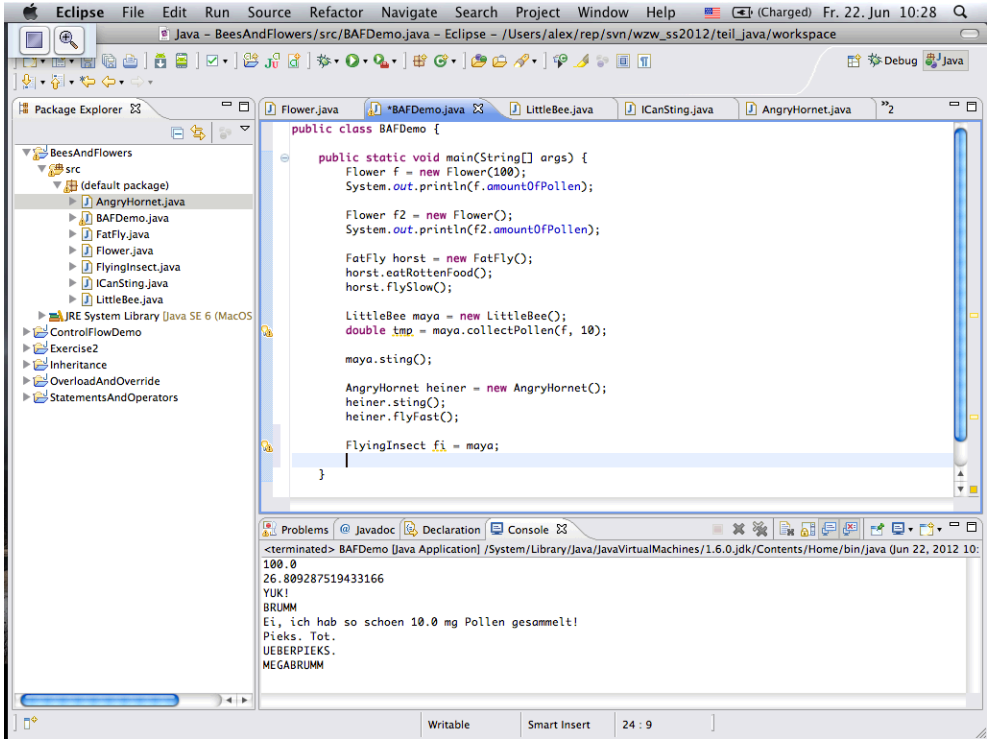

Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
public class LittleBee extends FlyingInsect im
double amountOfPollenCollected = 0;
double collectPollen(Flower flower, double howMuch) {
double result = flower.harvestPollen(howMuch);
amountOfPollenCollected = amountOfPollenCollected + result;
System.out.println("Ei, ich hab so schoen " + result + " mg Pollen gesammelt!");
return result;
}
}
100.0
133.5127057962506
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Writable
Smart Insert
1 : 53


Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:24
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
ICanSting.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FatFly.java
*LittleBee.java
ICanSting.java
public class LittleBee extends FlyingInsect implements ICanSting {
Add unimplemented methods
Make type 'LittleBee' abstract
Rename in file (⌘2 R)
Rename in workspace (⌥⌘R)
1 method to implement:
– ICanSting.sting()
Press 'Tab' from proposal table or click for focus
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
133.5127057962506
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
The type LittleBee must implem...tract method ICanSting.sting()
Writable
Smart Insert
1 : 23


Eclipse
Java – BeesAndFlowers/src/LittleBee.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
ICanSting.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
BAFDemo.java
FatFly.java
*LittleBee.java
ICanSting.java
public class LittleBee extends FlyingInsect implements ICanSting {
Add unimplemented methods
Make type 'LittleBee' abstract
Rename in file (⌘2 R)
Rename in workspace (⌥⌘R)
1 method to implement:
– ICanSting.sting()
Press 'Tab' from proposal table or click for focus
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
133.5127057962506
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
The type LittleBee must implem...tract method ICanSting.sting()
Writable
Smart Insert
1 : 23

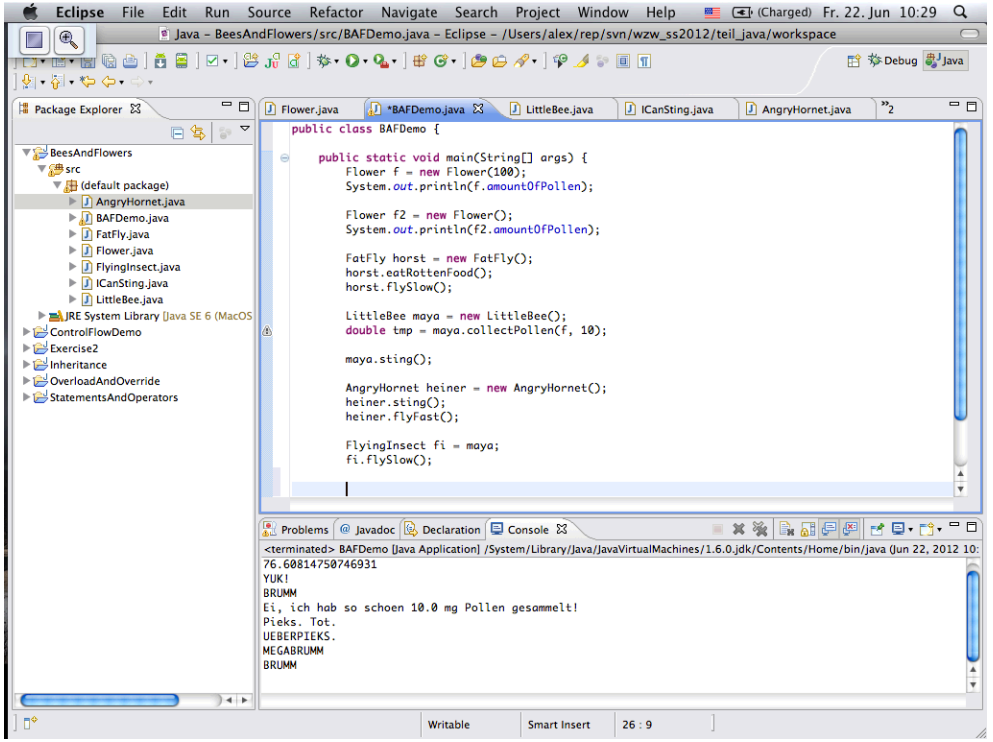

Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:27
Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
AngryHornet.java
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
ICanSting.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
*BAFDemo.java
LittleBee.java
ICanSting.java
AngryHornet.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
LittleBee maya = new LittleBee();
double tmp = maya.collectPollen(f, 10);
maya.sting();
AngryHornet heiner = new AngryHornet();
heiner.sting();
heiner.flyFast();
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
26.809287519433166
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Pieks. Tot.
UEBERPIEKS.
MEGABRUMM
Writable
Smart Insert
23 : 9


Eclipse File Edit Run Source Refactor Navigate Search Project Window Help (Charged) Fr. 22. Jun 10:27
Java – BeesAndFlowers/src/BAFDemo.java – Eclipse – /Users/alex/rep/svn/wzw_ss2012/teil_java/workspace
Package Explorer
BeesAndFlowers
src
(default package)
AngryHornet.java
BAFDemo.java
FatFly.java
Flower.java
FlyingInsect.java
ICanSting.java
LittleBee.java
JRE System Library [Java SE 6 (MacOS
ControlFlowDemo
Exercise2
Inheritance
OverloadAndOverride
StatementsAndOperators
Flower.java
*BAFDemo.java
LittleBee.java
ICanSting.java
AngryHornet.java
public class BAFDemo {
public static void main(String[] args) {
Flower f = new Flower(100);
System.out.println(f.amountOfPollen);
Flower f2 = new Flower();
System.out.println(f2.amountOfPollen);
FatFly horst = new FatFly();
horst.eatRottenFood();
horst.flySlow();
LittleBee maya = new LittleBee();
double tmp = maya.collectPollen(f, 10);
maya.sting();
AngryHornet heiner = new AngryHornet();
heiner.sting();
heiner.flyFast();
FlyingI
}
}
Problems Javadoc Declaration Console
<terminated> BAFDemo [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (Jun 22, 2012 10:
100.0
26.809287519433166
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Pieks. Tot.
UEBERPIEKS.
MEGABRUMM
Writable
Smart Insert
23 : 15



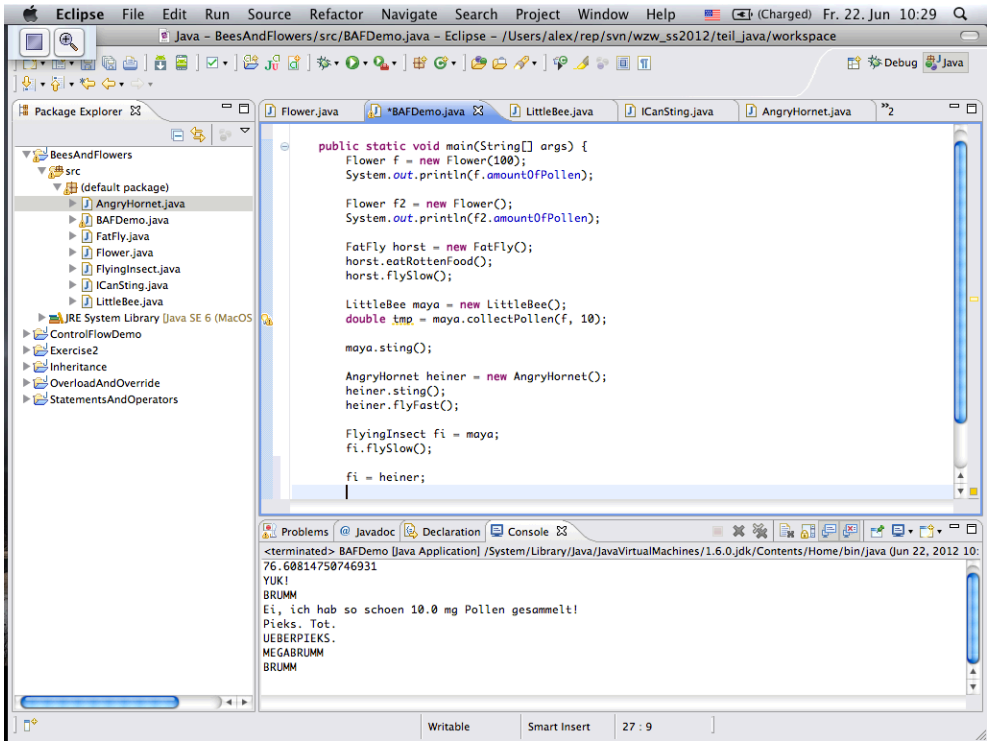
```
public class BAFDemo {

    public static void main(String[] args) {
        Flower f = new Flower(100);
        System.out.println(f.amountOfPollen);

        Flower f2 = new Flower();
        System.out.println(f2.amountOfPollen);

        FatFly horst = new FatFly();
        horst.eatRottenFood();
        horst.flySlow();

        LittleBee maya = new LittleBee();
        double tmp = maya.collectPollen(f, 10);

        maya.sting();

        AngryHornet heiner = new AngryHornet();
        heiner.sting();
        heiner.flyFast();

        FlyingInsect fi = maya;

    }
```

```
100.0
26.809287519433166
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Pieks. Tot.
UEBERPIEKS.
MEGABRUMM
```



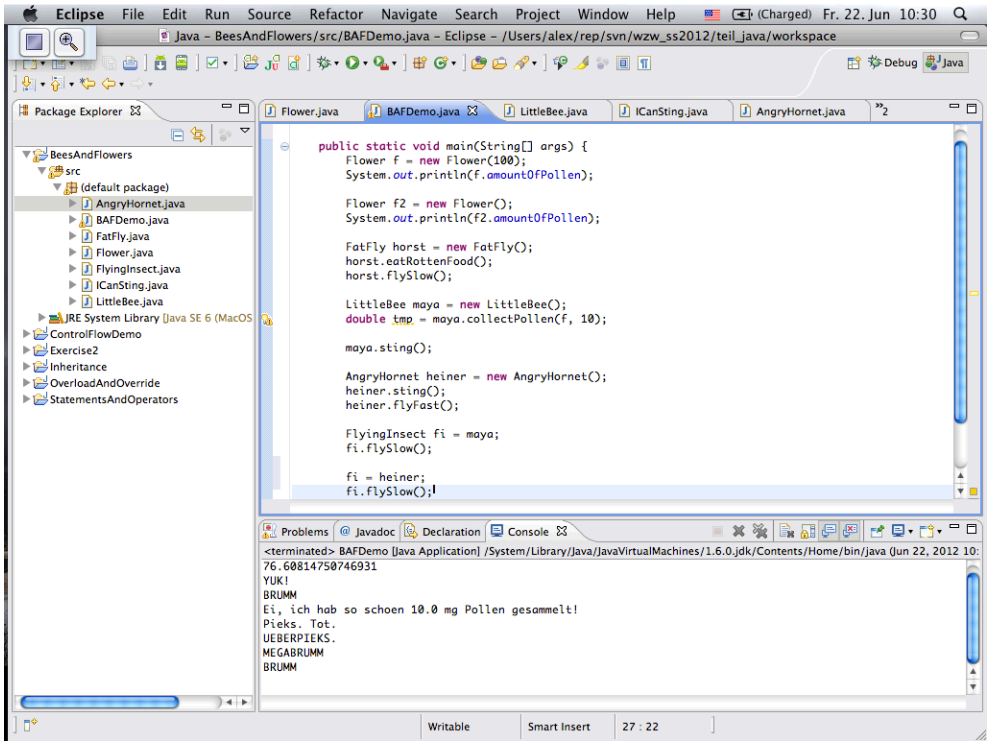
```
public class BAFDemo {

    public static void main(String[] args) {
        Flower f = new Flower(100);
        System.out.println(f.amountOfPollen);

        Flower f2 = new Flower();
        System.out.println(f2.amountOfPollen);

        FatFly horst = new FatFly();
        horst.eatRottenFood();
        horst.flySlow();

        LittleBee maya = new LittleBee();
        double tmp = maya.collectPollen(f, 10);

        maya.sting();

        AngryHornet heiner = new AngryHornet();
        heiner.sting();
        heiner.flyFast();

        FlyingInsect fi = maya;
        fi.flySlow();
```

```
76.60814750746931
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Pieks. Tot.
UEBERPIEKS.
MEGABRUMM
BRUMM
```



```
    public static void main(String[] args) {
        Flower f = new Flower(100);
        System.out.println(f.amountOfPollen);

        Flower f2 = new Flower();
        System.out.println(f2.amountOfPollen);

        FatFly horst = new FatFly();
        horst.eatRottenFood();
        horst.flySlow();

        LittleBee maya = new LittleBee();
        double tmp = maya.collectPollen(f, 10);

        maya.sting();

        AngryHornet heiner = new AngryHornet();
        heiner.sting();
        heiner.flyFast();

        FlyingInsect fi = maya;
        fi.flySlow();

        fi = heiner;
```

```
76.60814750746931
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Pieks. Tot.
UEBERPIEKS.
MEGABRUMM
BRUMM
```



```
    public static void main(String[] args) {
        Flower f = new Flower(100);
        System.out.println(f.amountOfPollen);

        Flower f2 = new Flower();
        System.out.println(f2.amountOfPollen);

        FatFly horst = new FatFly();
        horst.eatRottenFood();
        horst.flySlow();

        LittleBee maya = new LittleBee();
        double tmp = maya.collectPollen(f, 10);

        maya.sting();

        AngryHornet heiner = new AngryHornet();
        heiner.sting();
        heiner.flyFast();

        FlyingInsect fi = maya;
        fi.flySlow();

        fi = heiner;
        fi.flySlow();
```

```
76.60814750746931
YUK!
BRUMM
Ei, ich hab so schoen 10.0 mg Pollen gesammelt!
Pieks. Tot.
UEBERPIEKS.
MEGABRUMM
BRUMM
```

Overloading

- Overloading: Methods with same name but different parameters (types)

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

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

```
public static void main(String[] args) {
       OverloadingDemoClass odc = new OverloadingDemoClass();
       int result1 = odc.doSomething();
       int result2 = odc.doSomething(33);
}
```

- Method signature comprised of name and parameter types

Access Modifiers & Packages

- Access modifiers:
  - `public`: Can be accessed / invoked by anybody
  - `private`: Can only be accessed / invoked from within same class
  - `protected`: Can only be accessed / invoked from within same class and its subclasses
  - <no modifier>: Can be accessed / invoked from within same package

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

Classes, Objects, Inheritance

Access Modifiers & Packages

- Access modifiers:
  - `public`: Can be accessed / invoked by anybody
  - `private`: Can only be accessed / invoked from within same class
  - `protected`: Can only be accessed / invoked from within same class and its subclasses
  - <no modifier>: Can be accessed / invoked from within same package
- Packages:
  - Encapsulate a set of classes and interfaces
  - Hierarchical organization
  - Examples: `java.math`, `de.tum.wzw`

Classes, Objects, Inheritance

Access Modifiers & Packages

- Access modifiers:
  - `public`: Can be accessed / invoked by anybody
  - `private`: Can only be accessed / invoked from within same class
  - `protected`: Can only be accessed / invoked from within same class and its subclasses
  - <no modifier>: Can be accessed / invoked from within same package

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

Access Modifiers (`final`, `static`)

- Access modifiers:
  - **`static`**: field or method bound to class instead of object
    class-method, *class-variable* as opposed to *instance-method*, *instance-variable*
  - **`final`**:
    - fields: cannot be changed (constants)
    - methods: cannot be *overridden* (later)
    - classes: cannot be subclassed

```
final class MyClass {
        static int       sameValueForAllInstances = 3;
        final  int       constantValue = 5;
        static final int constantValueForAllInstances = 7;

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

Classes, Objects, Inheritance

Access Modifiers (`final`, `static`) **Example**

```
public final class MyClass {
        public static int       sameValueForAllInstances = 3;
        public final int        constantValue;
        public static final int constantValueForAllInstances = 7;

        public MyClass(int cv) { constantValue = cv; }

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

```
public static void main(String[] args) {
    MyClass m1 = new MyClass(20);
    MyClass m2 = new MyClass(30);
    m2.sameValueForAllInstances = 99;
    System.out.println(m1.sameValueForAllInstances);        // 99
    System.out.println(m2.sameValueForAllInstances);        // 99
    System.out.println(MyClass.sameValueForAllInstances);   // 99
    System.out.println(m1.constantValue);                   // 20
    m1.constantValue = 77;                                  // ERROR
    MyClass.methodOne();
    m1.methodTwo();
    MyClass.methodThree();
}
```

Classes, Objects, Inheritance

Access Modifiers (`final`, `static`) **Example**


```
public final class MyClass {
        public static int       sameValueForAllInstances = 3;
        public final int        constantValue;
        public static final int constantValueForAllInstances = 7;

        public MyClass(int cv) { constantValue = cv; }

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

```
public static void main(String[] args) {
    MyClass m1 = new MyClass(20);
    MyClass m2 = new MyClass(30);
    m2.sameValueForAllInstances = 99;
    System.out.println(m1.sameValueForAllInstances);        // 99
    System.out.println(m2.sameValueForAllInstances);        // 99
    System.out.println(MyClass.sameValueForAllInstances);   // 99
    System.out.println(m1.constantValue);                   // 20
    m1.constantValue = 77;                                  // ERROR
    MyClass.methodOne();
    m1.methodTwo();
    MyClass.methodThree();
}
```


Classes, Objects, Inheritance

Access Modifiers (`final`, `static`) **Example**


```
public final class MyClass {
        public static int       sameValueForAllInstances = 3;
        public final int        constantValue;
        public static final int constantValueForAllInstances = 7;

        public MyClass(int cv) { constantValue = cv; }

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

```
public static void main(String[] args) {
    MyClass m1 = new MyClass(20);
    MyClass m2 = new MyClass(30);
    m2.sameValueForAllInstances = 99;
    System.out.println(m1.sameValueForAllInstances);        // 99
    System.out.println(m2.sameValueForAllInstances);        // 99
    System.out.println(MyClass.sameValueForAllInstances);   // 99
    System.out.println(m1.constantValue);                   // 20
    m1.constantValue = 77;                                  // ERROR
    MyClass.methodOne();
    m1.methodTwo();
    MyClass.methodThree();
}
```

Access Modifiers (`final`, `static`)

Example

```
public final class MyClass {
        public static int       sameValueForAllInstances = 3;
        public final int        constantValue;
        public static final int constantValueForAllInstances = 7;

        public MyClass(int cv) { constantValue = cv; }

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

```
public static void main(String[] args) {
    MyClass m1 = new MyClass(20);
    MyClass m2 = new MyClass(30);
    m2.sameValueForAllInstances = 99;
    System.out.println(m1.sameValueForAllInstances);        // 99
    System.out.println(m2.sameValueForAllInstances);        // 99
    System.out.println(MyClass.sameValueForAllInstances);   // 99
    System.out.println(m1.constantValue);                   // 20
    m1.constantValue = 77;                                  // ERROR
    MyClass.methodOne();
    m1.methodTwo();
    MyClass.methodThree();
}
```

Classes, Objects, Inheritance

Access Modifiers (`final`, `static`)

Example

```
public final class MyClass {
        public static int       sameValueForAllInstances = 3;
        public final int        constantValue;
        public static final int constantValueForAllInstances = 7;

        public MyClass(int cv) { constantValue = cv; }

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

```
public static void main(String[] args) {
    MyClass m1 = new MyClass(20);
    MyClass m2 = new MyClass(30);
    m2.sameValueForAllInstances = 99;
    System.out.println(m1.sameValueForAllInstances);        // 99
    System.out.println(m2.sameValueForAllInstances);        // 99
    System.out.println(MyClass.sameValueForAllInstances);   // 99
    System.out.println(m1.constantValue);                   // 20
    m1.constantValue = 77;                                  // ERROR
    MyClass.methodOne();
    m1.methodTwo();
    MyClass.methodThree();
}
```