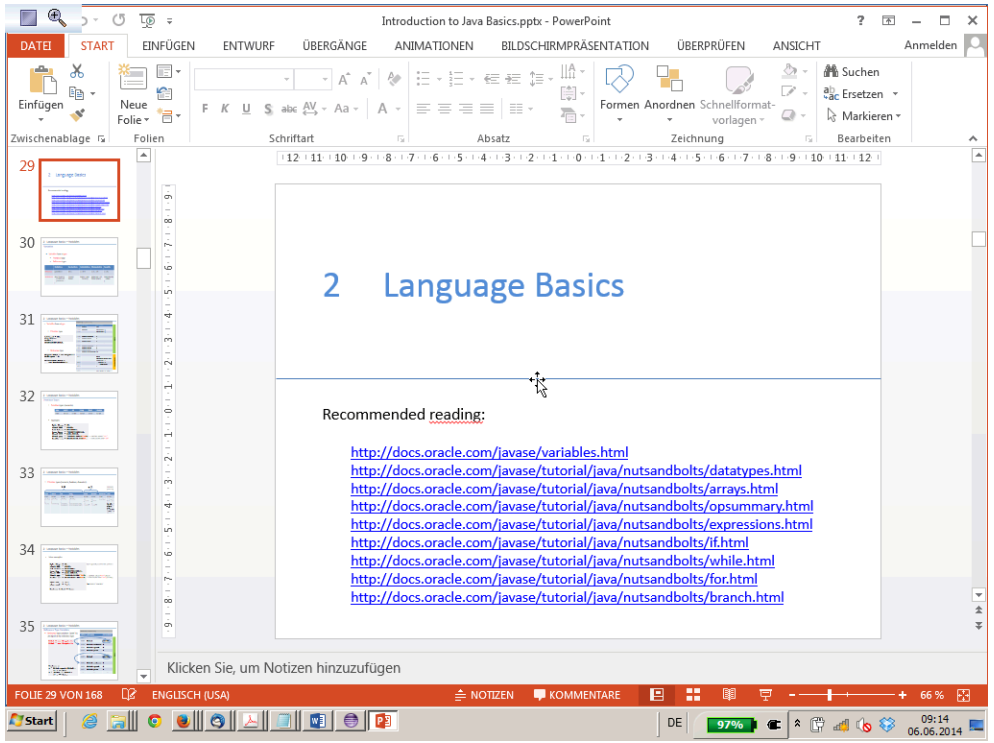
Script **generated by TTT**

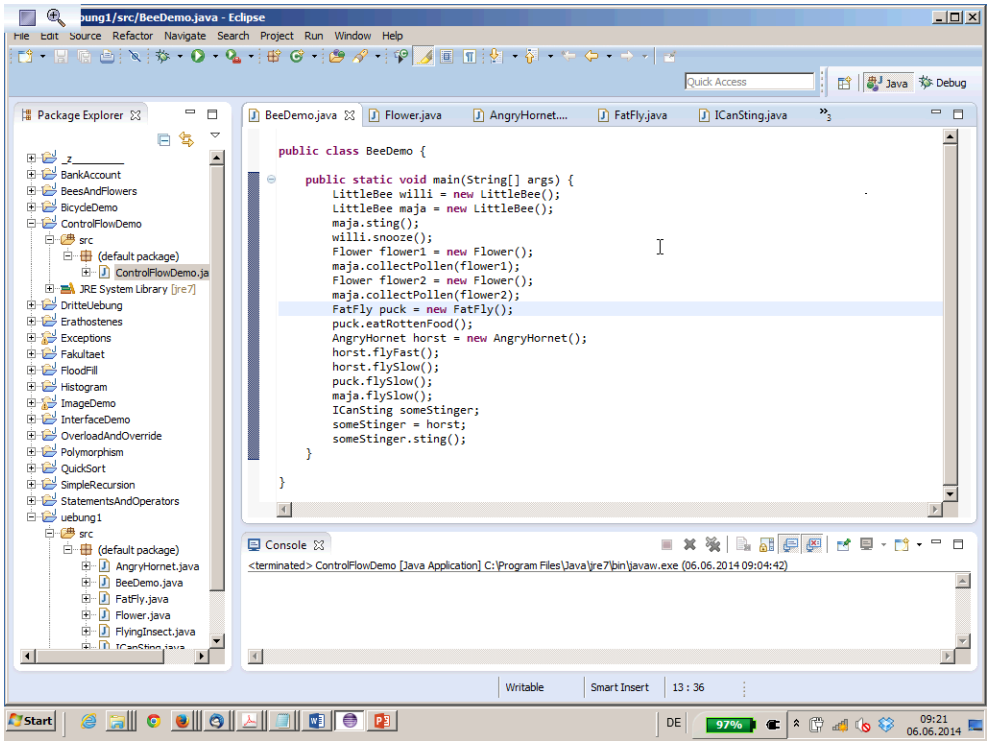
Title: groh: profile1 (06.06.2014)

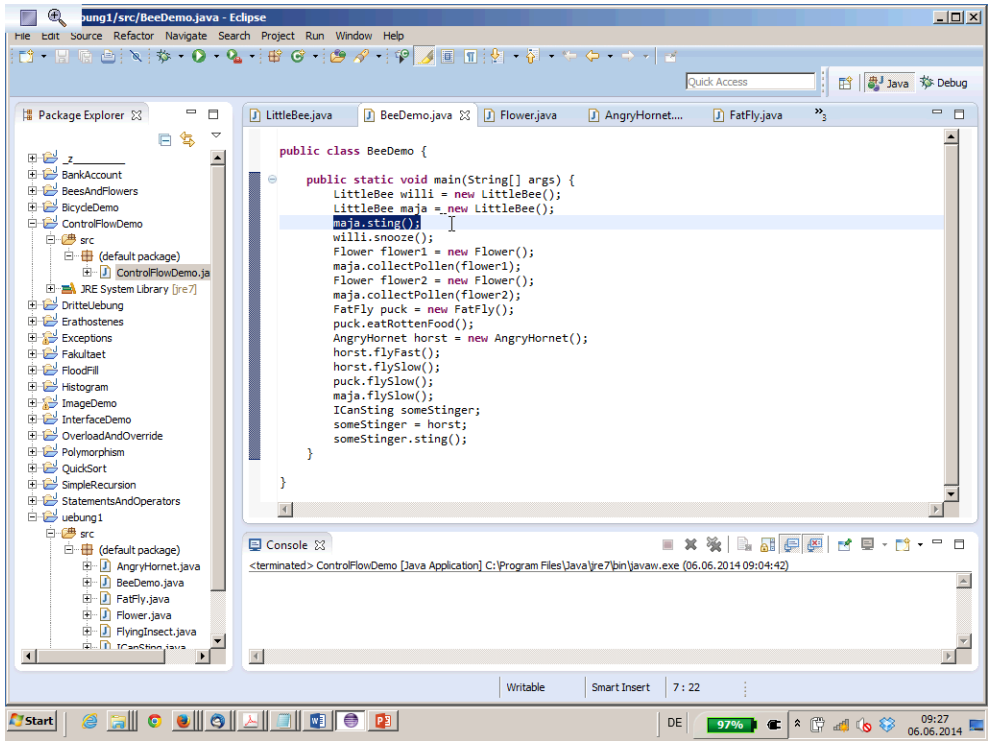
Date: Fri Jun 06 09:14:32 CEST 2014

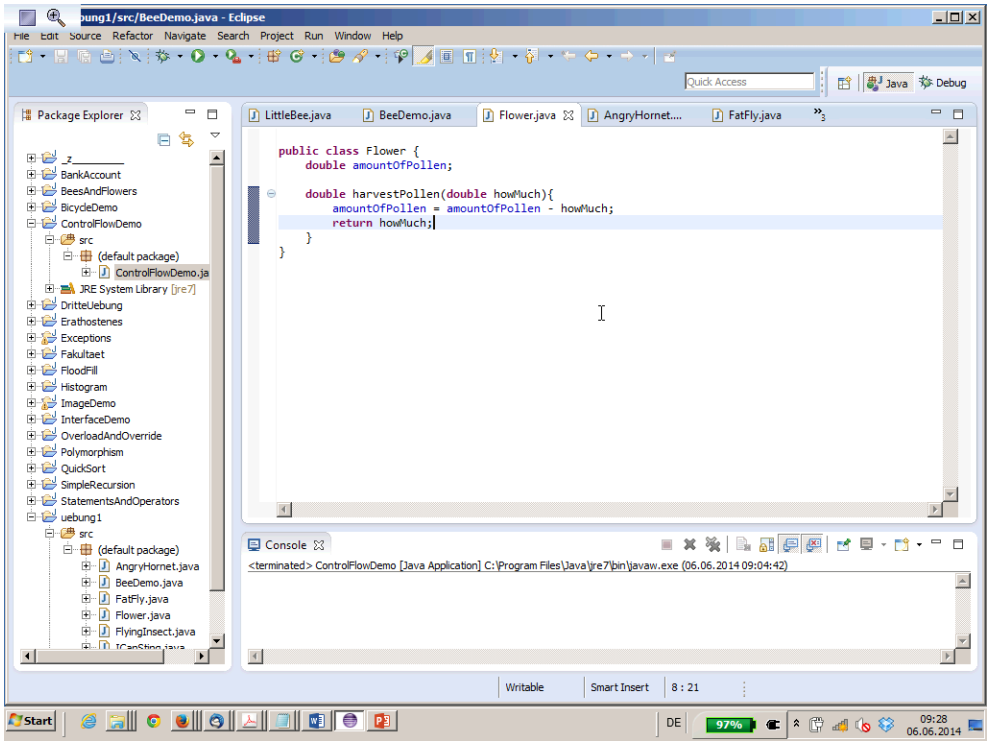
Duration: 92:34 min

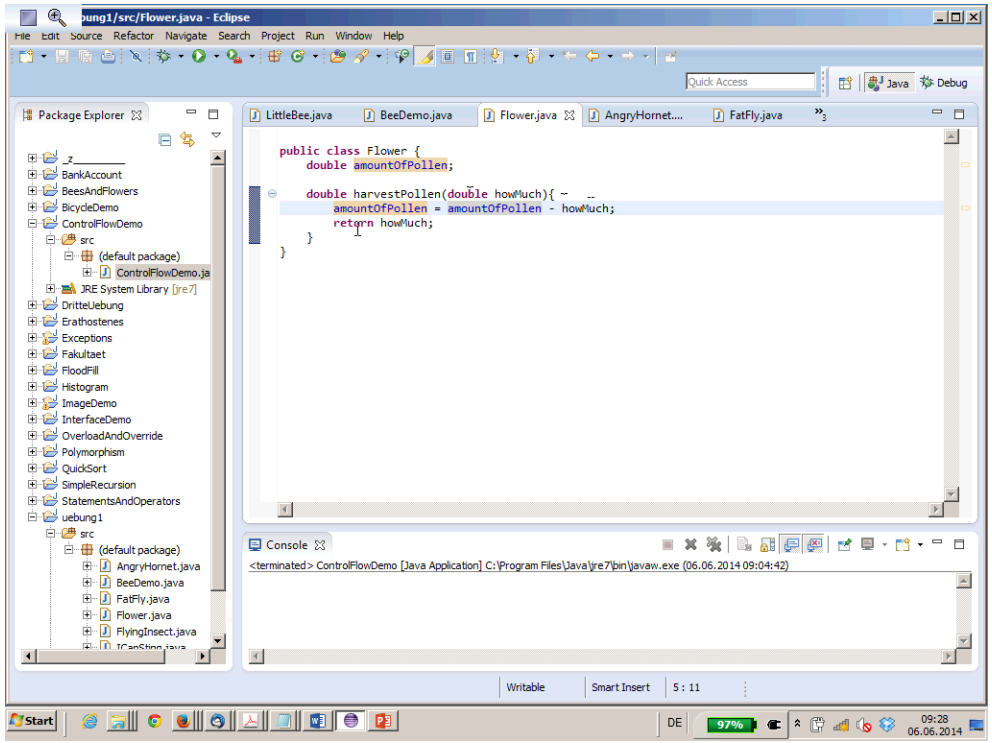
Pages: 54

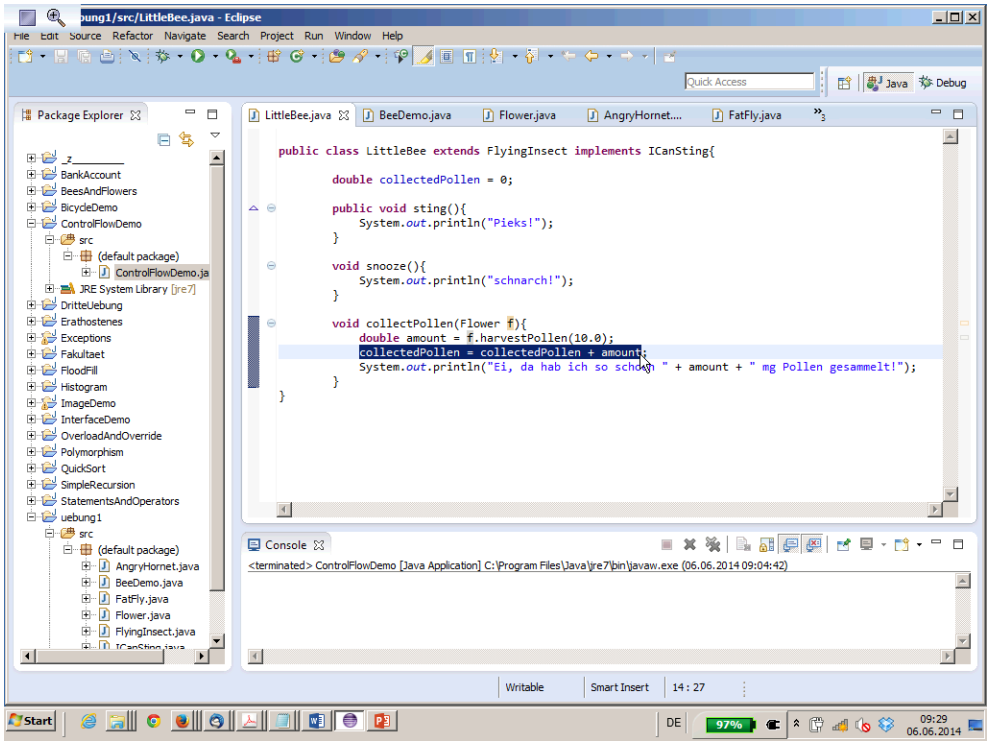


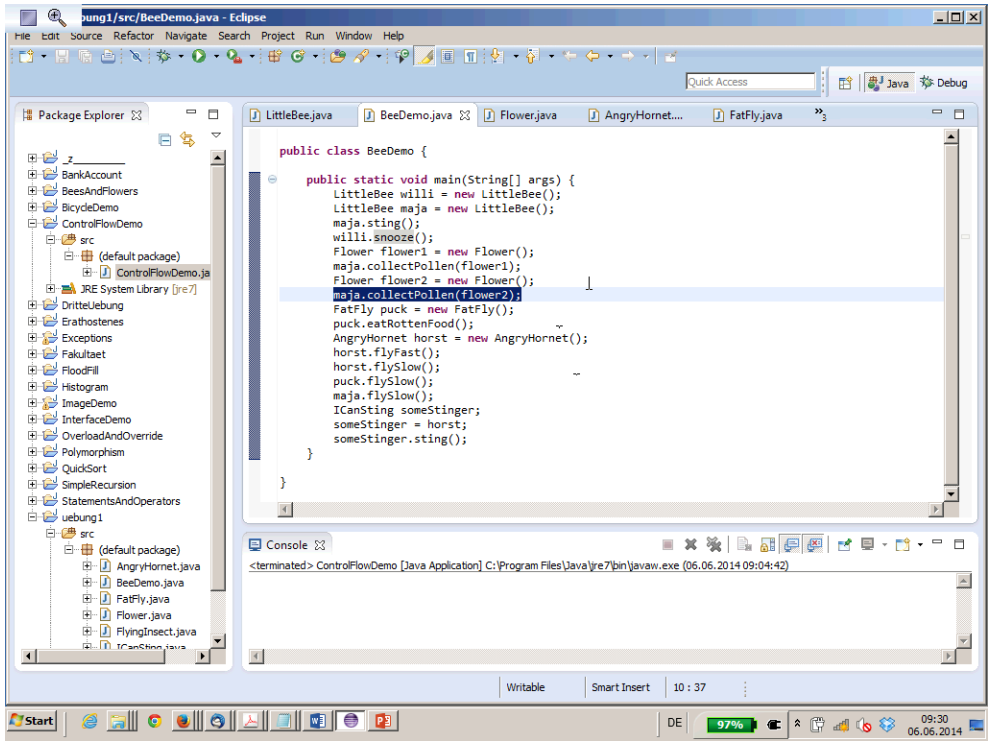


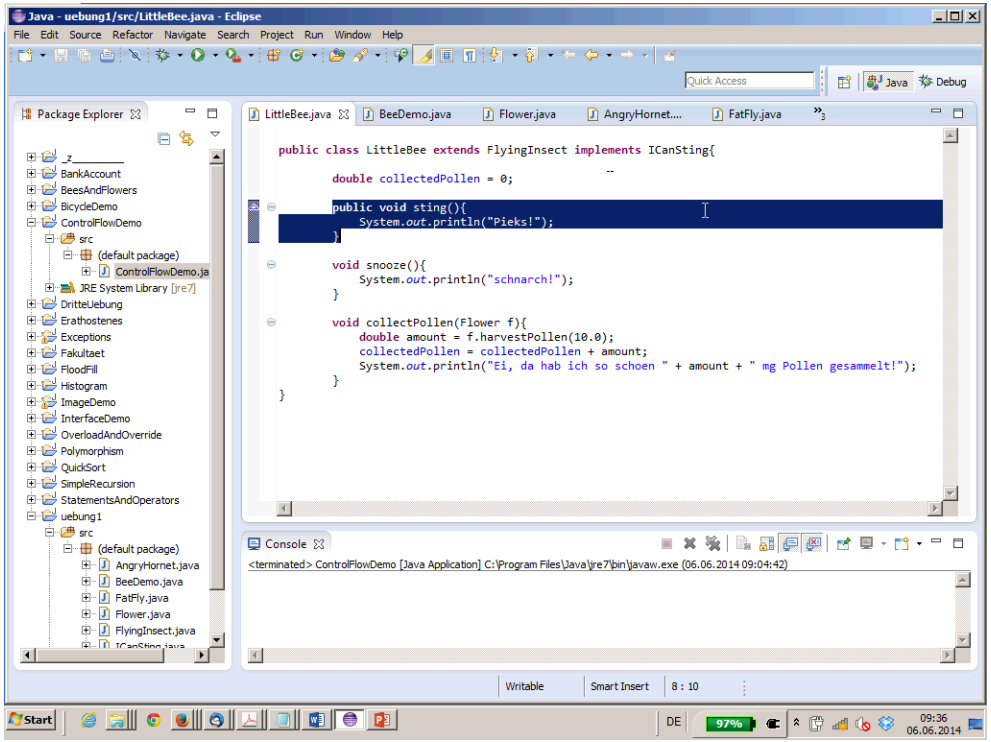


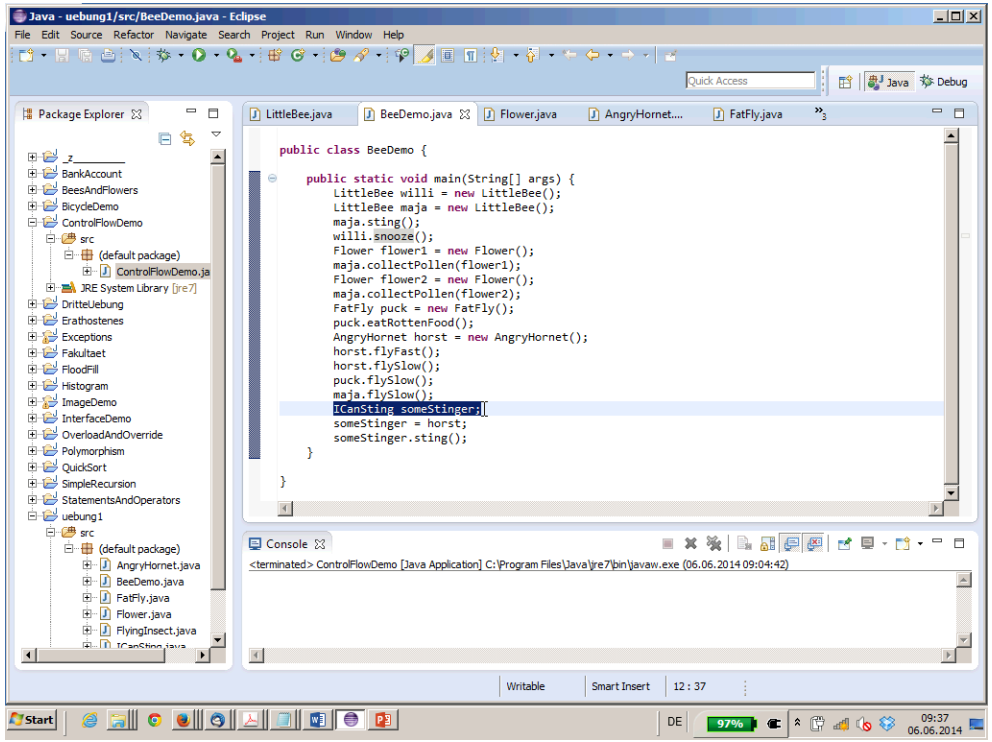

uebung1/src/BeeDemo.java - Eclipse
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public class Flower {
double amountOfPollen;
double harvestPollen(double howMuch){
amountOfPollen = amountOfPollen - howMuch;
return howMuch;
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
8 : 21

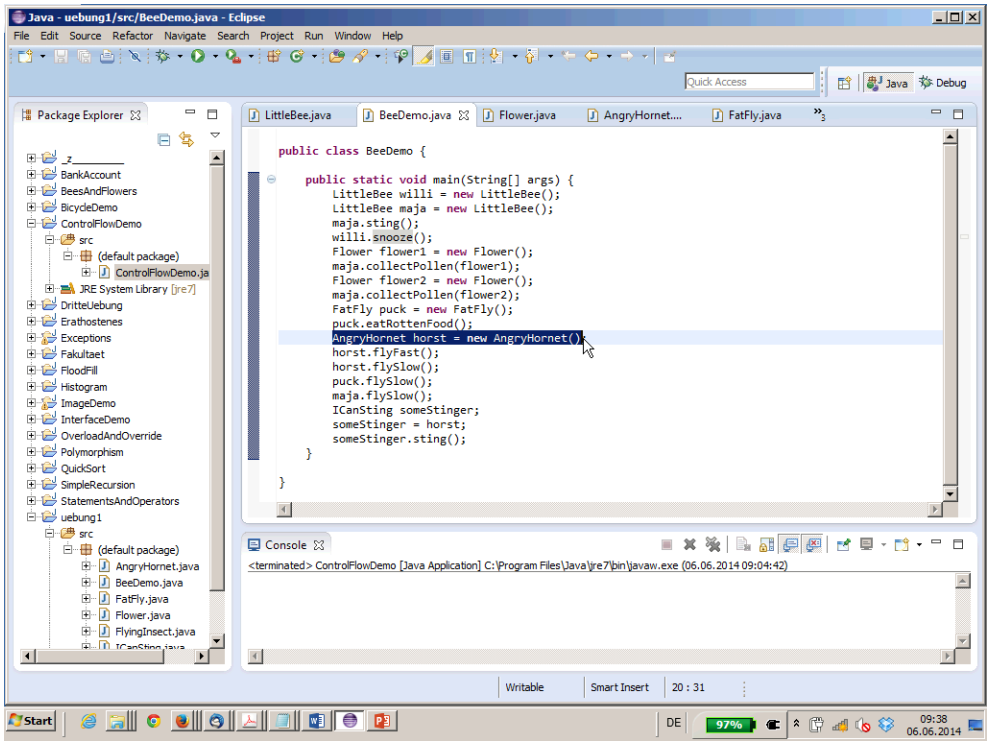

uebung1/src/Flower.java - Eclipse
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public class Flower {
double amountOfPollen;
double harvestPollen(double howMuch){
amountOfPollen = amountOfPollen - howMuch;
return howMuch;
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
5 : 11

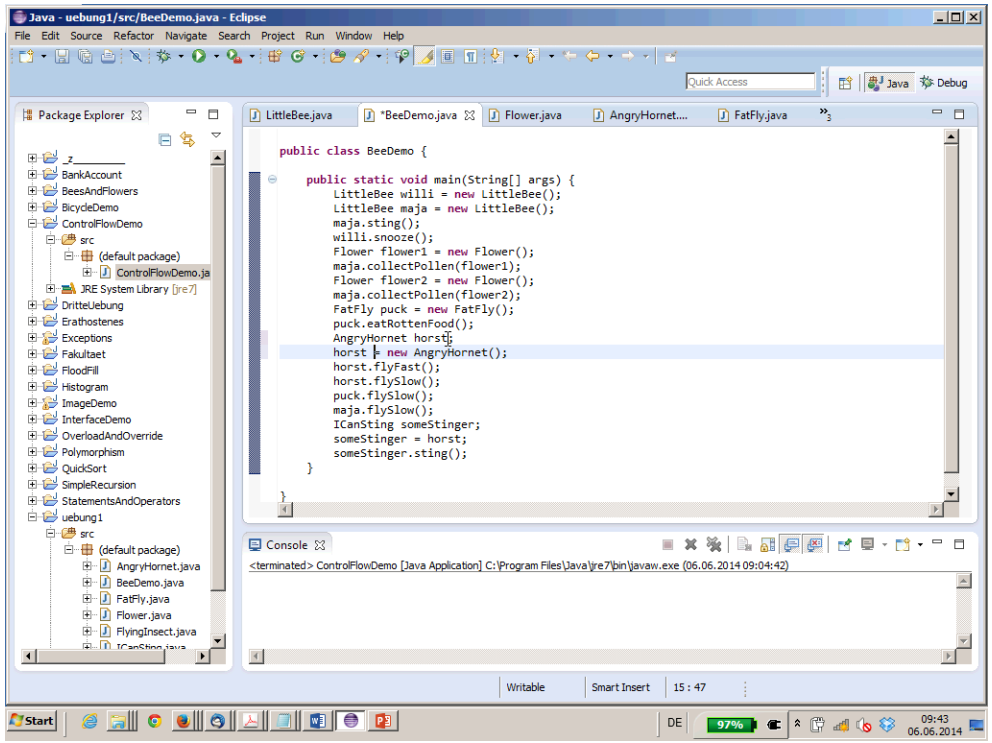

uebung1/src/LittleBee.java - Eclipse
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public class LittleBee extends FlyingInsect implements ICanSting{
double collectedPollen = 0;
public void sting(){
System.out.println("Pieks!");
}
void snooze(){
System.out.println("schnarch!");
}
void collectPollen(Flower f){
double amount = f.harvestPollen(10.0);
collectedPollen = collectedPollen + amount;
System.out.println("Ei, da hab ich so schön " + amount + " mg Pollen gesammelt!");
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
14 : 27

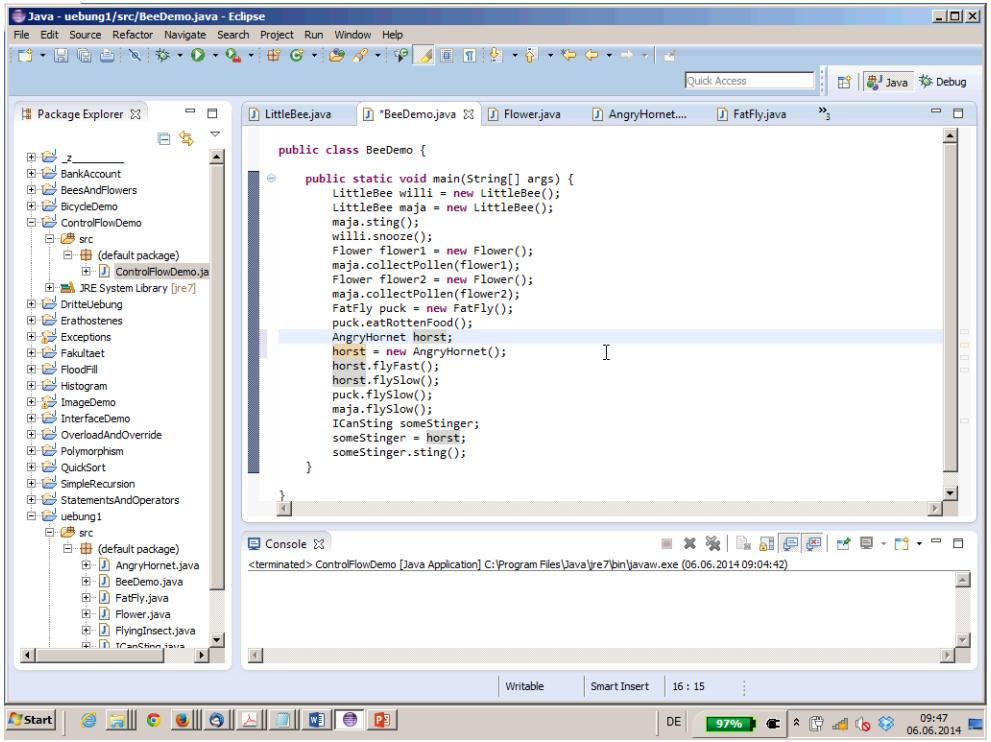

uebung1/src/BeeDemo.java - Eclipse
LittleBee.java
BeeDemo.java
Flower.java
AngryHornet....
FatFly.java
public class BeeDemo {
public static void main(String[] args) {
LittleBee willi = new LittleBee();
LittleBee maja = new LittleBee();
maja.sting();
willi.snooze();
Flower flower1 = new Flower();
maja.collectPollen(flower1);
Flower flower2 = new Flower();
maja.collectPollen(flower2);
FatFly puck = new FatFly();
puck.eatRottenFood();
AngryHornet horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
ICanSting someStinger;
someStinger = horst;
someStinger.sting();
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
10 : 37


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 implements ICanSting{
double collectedPollen = 0;
public 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> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
8 : 10


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 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();
ICanSting someStinger;
someStinger = horst;
someStinger.sting();
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
12 : 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
AngryHornet....
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();
horst.flySlow();
puck.flySlow();
maja.flySlow();
ICanSting someStinger;
someStinger = horst;
someStinger.sting();
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
20 : 31


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 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;
horst = new AngryHornet();
horst.flyFast();
horst.flySlow();
puck.flySlow();
maja.flySlow();
ICanSting someStinger;
someStinger = horst;
someStinger.sting();
}
}
Console
<terminated> ControlFlowDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 09:04:42)
Writable
Smart Insert
15 : 47



2 Language Basics

Recommended reading:

- http://docs.oracle.com/javase/variables.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/for.html
- http://docs.oracle.com/javase/tutorial/java/nutsandbolts/branch.html

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

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
...	...	...	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

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	
4035		int horst = 101;	

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	
4035		int horst = 101;	

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	
4035		int horst = 101;	

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	
4035		int horst = 101;	

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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	
1125		837465845	
...	...	...	
1150	bike1.cadence	0	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	instructions
4027		void changeCadence(int newValue) {	
4028		cadence = newValue;	
4029		}	
...	...	...	
4035		int horst = 101;	

2 Language Basics – Variables

- Variables have a type
  - Primitive type

```
int horst = 101;
long heiner;
heiner = 5638465837465845;
```

  - 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	
1125		837465845	
...	...	...	
1150	bike1.cadence	0	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	instructions
4027		void changeCadence(int newValue) {	
4028		cadence = newValue;	
4029		}	
...	...	...	
4035		int horst = 101;	

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]	[+/- ~ $1.4 \cdot 10^{-45}$, +/- ~ $3.4 \cdot 10^{38}$]	[+/- ~ $4.9 \cdot 10^{-324}$, +/- ~ $1.8\ 10^{308}$]	{ true, false }	{ … !, „, §, $, %, &, …., a, b, c, …., ∭,…., ∰,…., ガ, キ, ギ,…., ㋗, ㋘, ㋙,…., 樵, 樶, 蕺,… 액, 앴, 앤 …, ش ش, …, →, ↓, ■, …}

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

- 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 = 5638465837465845;
```

  - 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	
1151	bike1.speed	0	
1152	bike1.gear	3	
...	...	...	
1330	bike2.cadence	0	
1331	bike2.speed	0	
1332	bike2.gear	1	
1333	bike2.seatHeight	15	
...	...	...	
4027		void changeCadence(int newValue) {	instructions
4028		cadence = newValue;	instructions
4029		}	instructions
...	...	...	
4035		int horst = 101;	

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>	data
1150	bike1.cadence	0	data
1151	bike1.speed	0	data
1152	bike1.gear	1	data
...	...	...	
1327	bike2	<1405>	data
...	...	...	
1405	bike2.cadence	0	data
1406	bike2.speed	0	data
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>	data
1150	bike1.cadence	0	data
1151	bike1.speed	0	data
1152	bike1.gear	1	data
...	...	...	
1327	bike2	<1405>	data
...	...	...	
1405	bike2.cadence	0	data
1406	bike2.speed	0	data
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();

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();

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

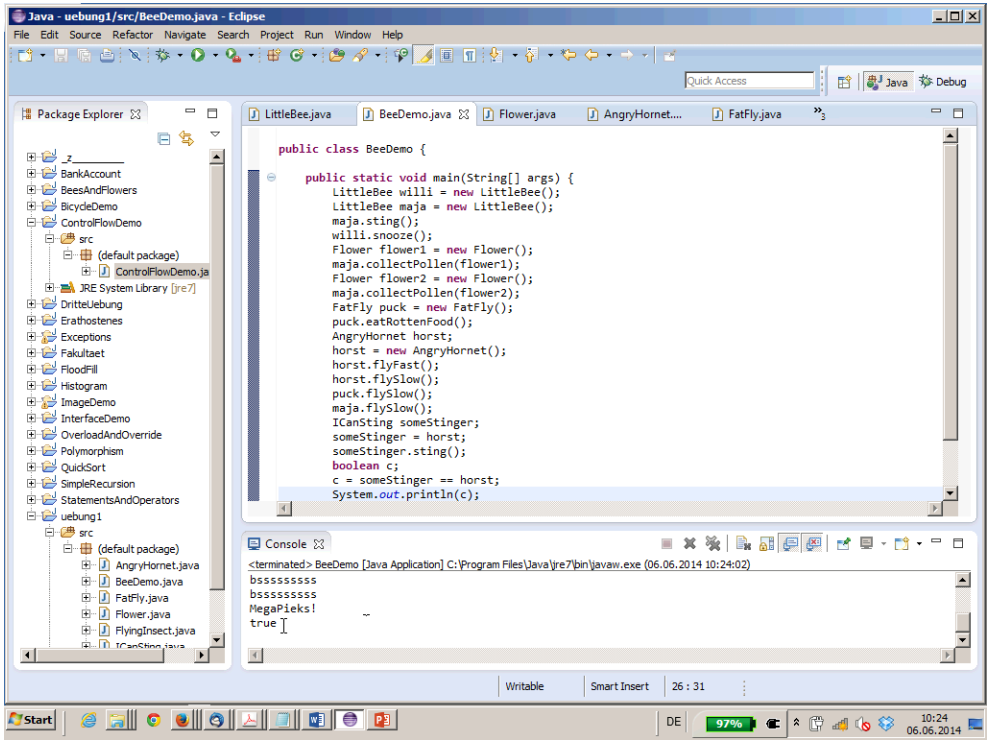
- Reference type variables "point" to an object of the reference type

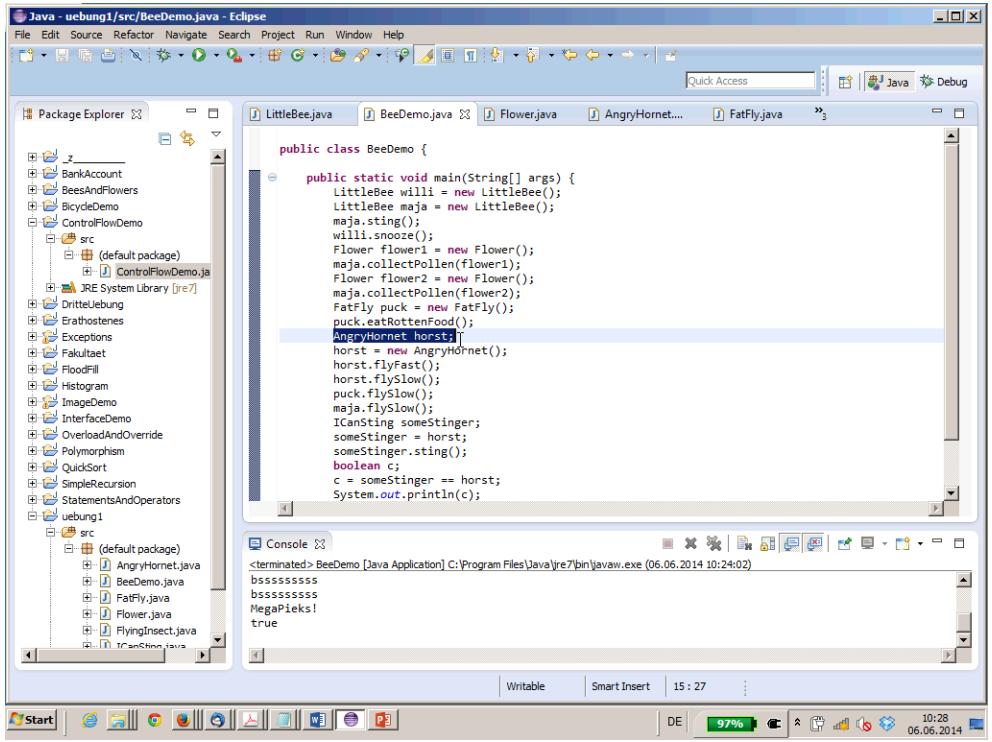
```
bike1 = new Bicycle();
bike2 = new Bicycle();

bike1.gear = 3;

bike1 = bike2;

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

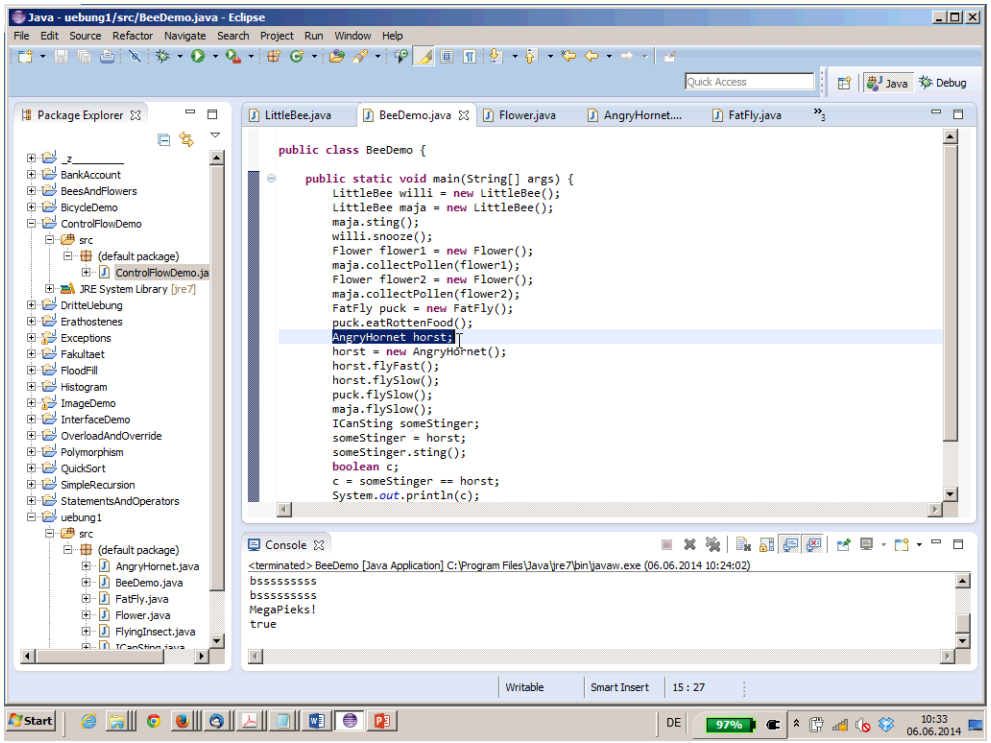
memory (simplified model)

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







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)

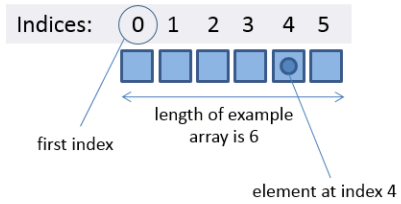
cell nr	cell name	cell content
...	...	...
1149	bike1	<1405>
1150	bike1.cadence	0
1151	bike1.speed	0
1152	bike1.gear	3
...	...	...
1327	bike2	<1405>
...	...	...
1405	bike2.cadence	0
1406	bike2.speed	0
1407	bike2.gear	1
...	...	...

data

2 Language Basics – Variables

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)




```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray = new AnyClass[45];
thirdArray[44] = new AnyClass();
thirdArray[22 * 2].someMethod();
```

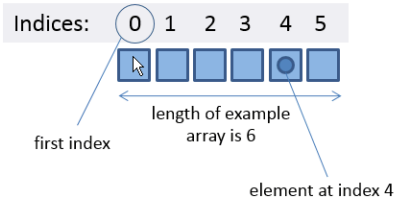
array of *primitive type* elements

array of *reference type* elements (objects)

2 Language Basics – Variables

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)




```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray = new AnyClass[45];
thirdArray[44] = new AnyClass();
thirdArray[22 * 2].someMethod();
```

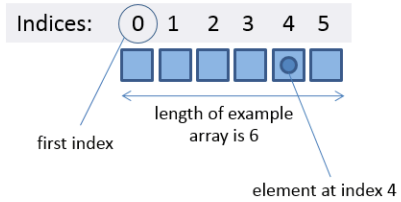
array of *primitive type* elements

array of *reference type* elements (objects)

2 Language Basics – Variables

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)




```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray = new AnyClass[45];
thirdArray[44] = new AnyClass();
thirdArray[22 * 2].someMethod();
```

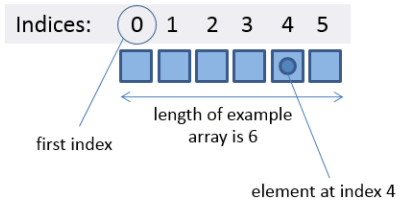
array of *primitive type* elements

array of *reference type* elements (objects)

2 Language Basics – Variables

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)




```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray = new AnyClass[45];
thirdArray[44] = new AnyClass();
thirdArray[22 * 2].someMethod();
```

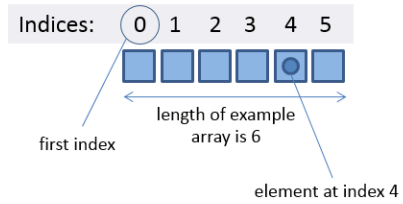
array of *primitive type* elements

array of *reference type* elements (objects)

2 Language Basics – Variables

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)




```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray = new AnyClass[45];
thirdArray[44] = new AnyClass();
thirdArray[22 * 2].someMethod();
```

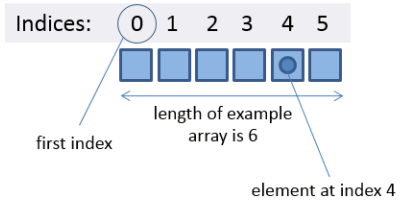
array of *primitive type* elements

array of *reference type* elements (objects)

2 Language Basics – Variables

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)




```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray = new AnyClass[45];
thirdArray[44] = new AnyClass();
thirdArray[22 * 2].someMethod();
```

array of *primitive type* elements

array of *reference type* elements (objects)