

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

The screenshot shows a Microsoft PowerPoint presentation window. The title bar reads "Introduction to Java Basics.pptx - PowerPoint". The ribbon menu is visible with tabs like DATEI, START, EINFÜGEN, ENTWURF, ÜBERGÄNGE, ANIMATIONEN, BILDSCHEINPRÄSENTATION, ÜBERPRÜFEN, and ANSICHT. A search bar at the top right says "Suchen". The main content area displays slide 29, which has a blue header "2 Language Basics". Below the header is a "Recommended reading:" section containing several hyperlinks to Oracle Java documentation. The status bar at the bottom shows "FOLIE 29 VON 168 ENGLISCH (USA)" and the date "06.06.2014".

The screenshot shows two Eclipse IDE windows side-by-side. Both windows have the title "BeeDemo.java - Eclipse". The left window shows the original code for BeeDemo.java, and the right window shows the modified code where the line "maja.string();" has been highlighted with a blue selection bar. The code is as follows:

```
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();  
    }  
}
```

The screenshot shows the Eclipse IDE interface with the title "BeeDemo.java - Eclipse". The code editor displays the original version of BeeDemo.java, identical to the one shown in the adjacent window. The package explorer on the left shows various Java projects and source files. The status bar at the bottom indicates "DE 97% 06.06.2014".

Screenshot of Eclipse IDE showing the code for `Flower.java`:

```
public class Flower {
    double amountOfPollen;

    double harvestPollen(double howMuch){
        amountOfPollen = amountOfPollen - howMuch;
        return howMuch;
    }
}
```

Screenshot of Eclipse IDE showing the code for `Flower.java` (same code as above):

```
public class Flower {
    double amountOfPollen;

    double harvestPollen(double howMuch){
        amountOfPollen = amountOfPollen - howMuch;
        return howMuch;
    }
}
```

Screenshot of Eclipse IDE showing the code for `LittleBee.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 sch&gt; " + amount + " mg Pollen gesammelt!");
    }
}
```

Screenshot of Eclipse IDE showing the code for `BeeDemo.java`:

```
public static void main(String[] args) {
    LittleBee willi = new LittleBee();
    LittleBee maja = new LittleBee();
    willi.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();
}
```

Java - uebung1/src/LittleBee.java - Eclipse

```
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!");
    }
}
```

Java - uebung1/src/BeeDemo.java - Eclipse

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

Java - uebung1/src/BeeDemo.java - Eclipse

```
public class BeeDemo {
    public static void main(String[] args) {
        LittleBee willi = new LittleBee();
        LittleBee maja = new LittleBee();
        maja.sting();
        willi.snooze();
        Flower flower1 = new Flower();
        maja.collectPollen(flower1);
        Flower flower2 = new Flower();
        maja.collectPollen(flower2);
        Fatfly puck = new Fatfly();
        puck.eatRottenFood();
        AngryHornet horst = new AngryHornet();
        horst.flyFast();
        horst.flySlow();
        puck.flySlow();
        maja.flySlow();
        ICanSting someStinger;
        someStinger = horst;
        someStinger.sting();
    }
}
```

Java - uebung1/src/BeeDemo.java - Eclipse

```
public class BeeDemo {
    public static void main(String[] args) {
        LittleBee willi = new LittleBee();
        LittleBee maja = new LittleBee();
        maja.sting();
        willi.snooze();
        Flower flower1 = new Flower();
        maja.collectPollen(flower1);
        Flower flower2 = new Flower();
        maja.collectPollen(flower2);
        Fatfly puck = new Fatfly();
        puck.eatRottenFood();
        AngryHornet horst;
        horst = new AngryHornet();
        horst.flyFast();
        horst.flySlow();
        puck.flySlow();
        maja.flySlow();
        ICanSting someStinger;
        someStinger = horst;
        someStinger.sting();
    }
}
```

The screenshot shows the Eclipse IDE interface. The left side features the Package Explorer with several Java projects and source files listed. The central area displays the code for `LittleBee.java`, which includes imports for `BankAccount`, `BeesAndFlowers`, `BicycleDemo`, `ControlFlowDemo`, `DritteJelbung`, `Erathostenes`, `Exceptions`, `Fakultaet`, `Floodfill`, `Histogram`, `ImageDemo`, `InterfaceDemo`, `OverloadAndOverride`, `Polymorphism`, `Quicksort`, `SimpleRecursion`, `StatementsAndOperators`, and `uebung1`. The code itself defines a `BeeDemo` class with a `main` method that creates instances of `LittleBee`, `Flower`, and `FatFly`, and calls their methods like `sting`, `snooze`, `collectPollen`, `eatRottenFood`, `flyFast`, `flySlow`, and `sting`. The right side shows a Console tab with the output of the application's execution.

2 Language Basics – Variables

- **Variables have a type**

- **Primitive type**

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

- **Reference type**

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

MountainBike bike2 =
    new MountainBike();
```

memory (simplified model)		
cell nr.	cell name	cell content
1123	horst	101
1124	heiner	235638465845]
1125		837465845]
...
1150	bikel.cadence	0
1151	bikel.speed	0
1152	bikel.gear	3
...
1330	bike2.cadence	0
1331	bike2.speed	0
1332	bike2.gear	1
1333	bike2.seatHeight	15
...
4027		void changeCadence(int newValue) {
4028		cadence = newValue;
4029		}
...
4035		int horst = 101;



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

- **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 = 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
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
...
4027		void changeCadence(int newValue) {
4028		cadence = newValue;
4029		}
...
4035		int horst = 101;

data

instructions

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
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
...
4027		void changeCadence(int newValue) {
4028		cadence = newValue;
4029		}
...
4035		int horst = 101;

data

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
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
...
4027		void changeCadence(int newValue) {
4028		cadence = newValue;
4029		}
...
4035		int horst = 101;

data

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
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
...
4027		void changeCadence(int newValue) {
4028		cadence = newValue;
4029		}
...
4035		int horst = 101;

- **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
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
...
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
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
...
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
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
...
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()
```

2 Language Basics – Variables

- Primitive types (numeric, boolean, character):

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()
```

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-1;
double sss = 3245343455.555E0;
char ccc = 'm';
char ccc2 = '\n';
```

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 * 10⁶⁷ (double)

\n means "new line"



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

- 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
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
...
4027		void changeCadence(int newValue) { cadence = newValue; }
4028		cadence = newValue;
4029		}
...
4035		int horst = 101;

data

instructions

2 Language Basics – Variables

Reference Type Variables

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

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

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
...

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
...

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
...

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
...

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
...

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
...

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
...

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
...

Java - uebung1/src/BeeDemo.java - Eclipse

```

public class BeeDemo {
    public static void main(String[] args) {
        LittleBee willi = new LittleBee();
        LittleBee maja = new LittleBee();
        maja sting();
        willi.snooze();
        Flower flower1 = new Flower();
        maja.collectPollen(flower1);
        Flower flower2 = new Flower();
        maja.collectPollen(flower2);
        FatFly puck = new FatFly();
        puck.eatRottenFood();
        AngryHornet horst;
        horst = new AngryHornet();
        horst.flyFast();
        horst.flySlow();
        puck.flySlow();
        maja.flySlow();
        ICanSting someStinger;
        someStinger = horst;
        someStinger.sting();
        boolean c;
        c = someStinger == horst;
        System.out.println(c);
    }
}

```

Console

```

<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 10:24:02)
bssssssss
bssssssss
MegaPieks!
true

```

Java - uebung1/src/BeeDemo.java - Eclipse

```

public class BeeDemo {
    public static void main(String[] args) {
        LittleBee willi = new LittleBee();
        LittleBee maja = new LittleBee();
        maja sting();
        willi.snooze();
        Flower flower1 = new Flower();
        maja.collectPollen(flower1);
        Flower flower2 = new Flower();
        maja.collectPollen(flower2);
        FatFly puck = new FatFly();
        puck.eatRottenFood();
        AngryHornet horst;
        horst = new AngryHornet();
        horst.flyFast();
        horst.flySlow();
        puck.flySlow();
        maja.flySlow();
        ICanSting someStinger;
        someStinger = horst;
        someStinger.sting();
        boolean c;
        c = someStinger == horst;
        System.out.println(c);
    }
}

```

Console

```

<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 10:24:02)
bssssssss
bssssssss
MegaPieks!
true

```

Java - uebung1/src/BeeDemo.java - Eclipse

```

public class BeeDemo {
    public static void main(String[] args) {
        LittleBee willi = new LittleBee();
        LittleBee maja = new LittleBee();
        maja sting();
        willi.snooze();
        Flower flower1 = new Flower();
        maja.collectPollen(flower1);
        Flower flower2 = new Flower();
        maja.collectPollen(flower2);
        FatFly puck = new FatFly();
        puck.eatRottenFood();
        AngryHornet horst;
        horst = new AngryHornet();
        horst.flyFast();
        horst.flySlow();
        puck.flySlow();
        maja.flySlow();
        ICanSting someStinger;
        someStinger = horst;
        someStinger.sting();
        boolean c;
        c = someStinger == horst;
        System.out.println(c);
    }
}

```

Console

```

<terminated> BeeDemo [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (06.06.2014 10:24:02)
bssssssss
bssssssss
MegaPieks!
true

```

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
...

2 Language Basics – Variables

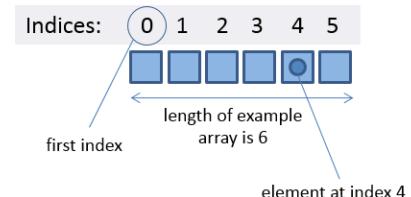
Arrays

- 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

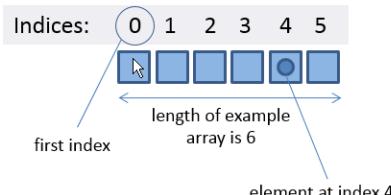
array of reference type elements (objects)

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)

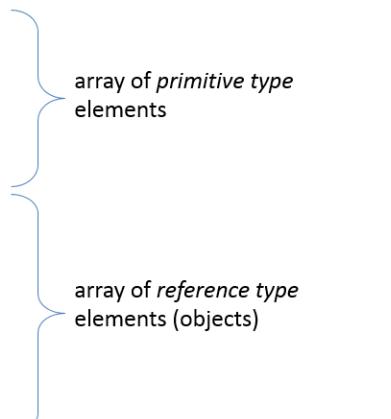
```
int[] someArray;
someArray = new int[6];
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];
```



array of primitive type elements

```
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 reference type elements (objects)



2 Language Basics – Variables

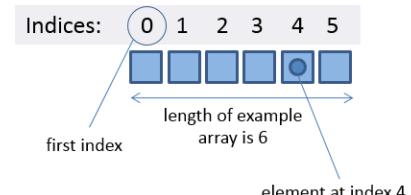
Arrays

- 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)

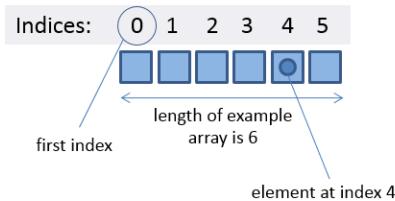


2 Language Basics – Variables

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

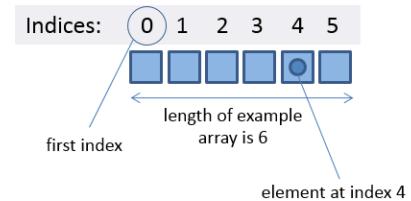
array of reference type elements (objects)

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

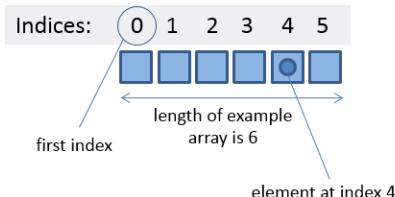
array of reference type elements (objects)

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)