

Script generated by TTT

Title: groh: profile1 (19.06.2015)

Date: Fri Jun 19 09:20:31 CEST 2015

Duration: 84:09 min

Pages: 79

```

algebra1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer
ICanString.java BeeDemo.java AngryHornet... Flower.java ControlStruct...
BankAccount BeesAndFlowers BicycleDemo ControlFlowDemo DritteJubung Erathostenes Exceptions Fakultaet FloodFill Histogram ImageDemo InterfaceDemo javaUebung1 KlausurJuli2014 OverloadAndOverride Polymorphism QuickSort SimpleRecursion StatementsAndOperators uebung1 uebung2 uebung3 uebung4 zzz_EinfWL
public class ControlStructuresDemo {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        ControlStructuresDemo eins = new ControlStructuresDemo();
        long x = 8;
        long y = 9;
        System.out.println(eins.minimum(x,y));
    }
    public long minimum(long a, long b){
        if(a < b){
            return a;
        } else {
            return b;
        }
    }
    public double exp (double x){
        double result = 1.0d;
        double y = 1.0d;
        for(int i=0; i<31; i++){
            y = x * y;
            result = result + (y / fakultaet(i));
        }
        return result;
    }
}

```

Darstellung / Approximation von reellen Zahlen

$\left. \begin{array}{l} \text{float ggg} = -345545.34534\text{E-12f;} \\ \text{double sss} = 3245343455.555\text{E67;} \end{array} \right\} \in \mathbb{Q}, \approx \mathbb{R}$

- mit **beschränkter Anzahl Bits**: Nur **Approximation** von reellen Zahlen (bspw. π) bzw. rationalen Zahlen mit nicht abbrechender Dezimalbruchentwicklung (bspw. $1/3$) möglich. (Standard: IEEE 754).
- Folgen: **Numerische Fehler** möglich. Bsp:

```

21000 double e = Math.pow(2.0d,1000.0d); //1.0715086071862673E301
25000 double f = Math.pow(2.0d,5000.0d); //Infinity (Zahlbereichsüberschreitung)
double g = 0.1d + 0.1d + 0.1d; //0.30000000000000004 (Rundungsfehler)
siehe auch: http://introcs.cs.princeton.edu/java/91float/

```

- **Test auf Gleichheit** zweier float oder double Werte x, y nicht mit $x == y$ sondern mit $|x - y| < \epsilon$;
viele weitere **Konsequenzen** → numerische Mathematik
- Es gibt in Java auch Möglichkeiten mit **beliebiger Genauigkeit** zu rechnen (bspw. mit Klasse BigDecimal)

$\left. \begin{array}{l} \text{float ggg} = -345545.34534\text{E-12f;} \\ \text{double sss} = 3245343455.555\text{E67;} \end{array} \right\} \in \mathbb{Q}, \approx \mathbb{R}$

- mit **beschränkter Anzahl Bits**: Nur **Approximation** von reellen Zahlen (bspw. π) bzw. rationalen Zahlen mit nicht abbrechender Dezimalbruchentwicklung (bspw. $1/3$) möglich. (Standard: IEEE 754).
- Folgen: **Numerische Fehler** möglich. Bsp:

```

21000 double e = Math.pow(2.0d,1000.0d); //1.0715086071862673E301
25000 double f = Math.pow(2.0d,5000.0d); //Infinity (Zahlbereichsüberschreitung)
double g = 0.1d + 0.1d + 0.1d; //0.30000000000000004 (Rundungsfehler)
siehe auch: http://introcs.cs.princeton.edu/java/91float/

```

- **Test auf Gleichheit** zweier float oder double Werte x, y nicht mit $x == y$ sondern mit $|x - y| < \epsilon$;
viele weitere **Konsequenzen** → numerische Mathematik
- Es gibt in Java auch Möglichkeiten mit **beliebiger Genauigkeit** zu rechnen (bspw. mit Klasse BigDecimal)

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
4
5  public static void main(String[] args) {
6      // TODO Auto-generated method stub
7      ControlStructuresDemo eins = new ControlStructuresDemo();
8      long x = 8;
9      long y = 9;
10     System.out.println(eins.minimum(x,y));
11 }
12
13@ public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21@ public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31 }
32 }
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
4
5  public static void main(String[] args) {
6      // TODO Auto-generated method stub
7      ControlStructuresDemo eins = new ControlStructuresDemo();
8      long x = 8;
9      long y = 9;
10     System.out.println(eins.minimum(x,y));
11 }
12
13@ public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21@ public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31@ public |
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
4
5  public static void main(String[] args) {
6      // TODO Auto-generated method stub
7      ControlStructuresDemo eins = new ControlStructuresDemo();
8      long x = 8;
9      long y = 9;
10     System.out.println(eins.minimum(x,y));
11 }
12
13@ public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21@ public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31 }
```

The method fakultaet(int n) is undefined for the type ControlStructuresDemo

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
8@ long x = 8;
9@ long y = 9;
10@ System.out.println(eins.minimum(x,y));
11 }
12
13@ public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21@ public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31@ public long fakultaet(int n){
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
10     System.out.println(eins.minimum(x,y));
11 }
12
13 public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21 public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31 public long fakultaet(int n){
32     long result = 1;
33     while(n>1){
34         result = result * n;
35         n = n - 1;
36     }
37     return result;
38 }
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
10     System.out.println(eins.minimum(x,y));
11 }
12
13 public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21 public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31 public long fakultaet(int n){
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
10     System.out.println(eins.minimum(x,y));
11 }
12
13 public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21 public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31 public long fakultaet(int n){
32     long result = 1;
33     while(n>1){
34         result = result * n;
35         n = n - 1;
36     }
37     return result;
38 }
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
10     System.out.println(eins.minimum(x,y));
11 }
12
13 public long minimum(long a, long b){
14     if(a < b){
15         return a;
16     } else {
17         return b;
18     }
19 }
20
21 public double exp (double x){
22     double result = 1.0d;
23     double y = 1.0d;
24     for(int i=0; i<31; i++){
25         y = x * y;
26         result = result + (y / fakultaet(i));
27     }
28     return result;
29 }
30
31 public long fakultaet(int n){
```

Console

No consoles to display at this time.

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
public class ControlStructuresDemo {
    public static void main(String[] args) {
        ControlStructuresDemo eins = new ControlStructuresDemo();
        long x = 8;
        long y = 9;
        System.out.println(eins.minimum(x,y));
        double xx = 0.0d;
    }
}

public long minimum(long a, long b){
    if(a < b){
        return a;
    } else {
        return b;
    }
}

public double exp (double x){
    double result = 1.0d;
    double y = 1.0d;
    for(int i=0; i<31; i++){
        y = x * y;
        result = result + (y / fakultaet(i));
    }
    return result;
}
```

Console

```
No consoles to display at this time.
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
public class ControlStructuresDemo {
    public static void main(String[] args) {
        ControlStructuresDemo herbert = new ControlStructuresDemo();
        long x = 8;
        long y = 9;
        System.out.println(herbert.minimum(x,y));
        double xx = 0.0d;
        double yy = herbert.exp(xx);
        System.out.println(yy);
    }
}

public long minimum(long a, long b){
    if(a < b){
        return a;
    } else {
        return b;
    }
}

public double exp (double x){
    double result = 1.0d;
    double y = 1.0d;
    for(int i=0; i<31; i++){
        y = x * y;
        result = result + (y / fakultaet(i));
    }
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:39:50)
at javaUebung1.ControlStructuresDemo.main(ControlStructuresDemo.java:13)
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
public class ControlStructuresDemo {
    public static void main(String[] args) {
        ControlStructuresDemo herbert = new ControlStructuresDemo();
        long x = 8;
        long y = 9;
        System.out.println(herbert.minimum(x,y));
        double xx = 1.0d;
        double yy = herbert.exp(xx);
        System.out.println(yy);
    }
}

public long minimum(long a, long b){
    if(a < b){
        return a;
    } else {
        return b;
    }
}

public double exp (double x){
    double result = 1.0d;
    double y = 1.0d;
    for(int i=0; i<31; i++){
        y = x * y;
        result = result + (y / fakultaet(i));
    }
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:40:08)
8
3.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
public class ControlStructuresDemo {
    public static void main(String[] args) {
        ControlStructuresDemo herbert = new ControlStructuresDemo();
        long x = 8;
        long y = 9;
        System.out.println(herbert.minimum(x,y));
        double xx = 1.0d;
        double yy = herbert.exp(xx);
        System.out.println(yy);
    }
}

public long minimum(long a, long b){
    if(a < b){
        return a;
    } else {
        return b;
    }
}

public double exp (double x){
    double result = 1.0d;
    double y = 1.0d;
    for(int i=0; i<31; i++){
        y = x * y;
        result = result + (y / fakultaet(i));
    }
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:40:36)
8
61.25649079078201
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
16    public long minimum(long a, long b){  
17        if(a < b){  
18            return a;  
19        } else {  
20            return b;  
21        }  
22    }  
23  
24    public double exp (double x){  
25        double result = 1.0d;  
26        double y = 1.0d;  
27        for(int i=0; i<31; i++){  
28            y = x * y;  
29            result = result + (y / fakultaet(i));  
30        }  
31        return result;  
32    }  
33  
34    public long fakultaet(int n){  
35        long result = 1;  
36        while(n>1){  
37            result = result * n;  
38            n = n - 1;  
39        }  
40        return result;  
41    }  
42
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:40:36)  
8  
61.25649079078201
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
16    public long minimum(long a, long b){  
17        if(a < b){  
18            return a;  
19        } else {  
20            return b;  
21        }  
22    }  
23  
24    public double exp (double x){  
25        double result = 1.0d;  
26        double y = 1.0d;  
27        for(int i=0; i<31; i++){  
28            y = x * y;  
29            result = result + (y / fakultaet(i));  
30        }  
31        return result;  
32    }  
33  
34    public long fakultaet(int n){  
35        long result = 1;  
36        while(n>1){  
37            result = result * n;  
38            n = n - 1;  
39        }  
40        return result;  
41    }  
42
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:40:36)  
8  
61.25649079078201
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
16    public long minimum(long a, long b){  
17        if(a < b){  
18            return a;  
19        } else {  
20            return b;  
21        }  
22    }  
23  
24    public double exp (double x){  
25        double result = 1.0d;  
26        double y = 1.0d;  
27        for(int i=0; i<31; i++){  
28            y = x * y;  
29            result = result + (y / fakultaet(i));  
30        }  
31        return result;  
32    }  
33  
34    public long fakultaet(int n){  
35        long result = 1;  
36        while(n>1){  
37            result = result * n;  
38            n = n - 1;  
39        }  
40        return result;  
41    }  
42
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:40:36)  
8  
61.25649079078201
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
16    public long minimum(long a, long b){  
17        if(a < b){  
18            return a;  
19        } else {  
20            return b;  
21        }  
22    }  
23  
24    public double exp (double x){  
25        double result = 1.0d;  
26        double y = 1.0d;  
27        for(int i=0; i<31; i++){  
28            y = x * y;  
29            result = result + (y / fakultaet(i));  
30        }  
31        return result;  
32    }  
33  
34    public long fakultaet(int n){  
35        long result = 1;  
36        while(n>1){  
37            result = result * n;  
38            n = n - 1;  
39        }  
40        return result;  
41    }  
42
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:43:59)  
8  
3.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
18     return a;
19 } else {
20     return b;
21 }

24 public double exp (double x){
25     double result = 1.0d;
26     double y = 1.0d;
27     for(int i=0; i<31; i++){
28         y = x * y;
29         result = result + (y / fakultaet(i));
30     }
31     return result;
32 }

34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:43:59)
8
3.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
18     return a;
19 } else {
20     return b;
21 }

24 public double exp (double x){
25     double result = 1.0d;
26     double y = 1.0d;
27     for(int i=1; i<31; i++){
28         y = x * y;
29         result = result + (y / fakultaet(i));
30     }
31     return result;
32 }

34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:43:59)
8
3.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
18     return a;
19 } else {
20     return b;
21 }

24 public double exp (double x){
25     double result = 1.0d;
26     double y = 1.0d;
27     for(int i=1; i<31; i++){
28         y = x * y;
29         result = result + (y / fakultaet(i));
30     }
31     return result;
32 }

34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:43:59)
8
3.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
18     return a;
19 } else {
20     return b;
21 }

24 public double exp (double x){
25     double result = 1.0d;
26     double y = 1.0d;
27     for(int i=1; i<31; i++){
28         y = x * y;
29         result = result + (y / fakultaet(i));
30     }
31     return result;
32 }

34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:43:59)
8
3.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSting.java BeeDemo.java AngryHornet... Flower.java ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
21 }
22 }
23 }
24 public double exp (double x){
25     double result = 1.0d;
26     double y = 1.0d;
27     for(int i=1; i<31; i++){
28         y = x * y;
29         result = result + (y / fakultaet(i));
30     }
31     return result;
32 }
33 }
34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40     return result;
41 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 41:6

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSting.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
26 double y = 1.0d;
27 for(int i=1; i<31; i++){
28     y = x * y;
29     result = result + (y / fakultaet(i));
30 }
31 return result;
32 }
33 }
34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40     return result;
41 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 43:5

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSting.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
26 double result = 1.0d;
27 double y = 1.0d;
28 for(int i=1; i<31; i++){
29     y = x * y;
30     result = result + (y / fakultaet(i));
31 }
32 return result;
33 }
34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40     return result;
41 }
42
43 public double dotProduct(){
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Syntax error on token ")", { expected after this token

Writable Smart Insert 43:21

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSting.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
26 double result = 1.0d;
27 double y = 1.0d;
28 for(int i=1; i<31; i++){
29     y = x * y;
30     result = result + (y / fakultaet(i));
31 }
32 return result;
33 }
34 public long fakultaet(int n){
35     long result = 1;
36     while(n>1){
37         result = result * n;
38         n = n - 1;
39     }
40     return result;
41 }
42
43 public double dotProduct(double[] a, double[] b){
44     double result
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 44:9

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanString.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

```
private double result = 1.0d;
double y = 1.0d;
for(int i=1; i<31; i++){
    y = x * y;
    result = result + (y / fakultaet(i));
}
return result;
```

```
public long fakultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 29 : 48

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanString.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

```
private double result = 1.0d;
double y = 1.0d;
for(int i=1; i<31; i++){
    y = x * y;
    result = result + (y / fakultaet(i));
}
return result;
```

```
public long fakultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++)
        result += a[i] * b[i];
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 45 : 26

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanString.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

```
private double result = 1.0d;
double y = 1.0d;
for(int i=1; i<31; i++){
    y = x * y;
    result = result + (y / fakultaet(i));
}
return result;
```

```
public long fakultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++)
        result += a[i] * b[i];
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 45 : 26

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanString.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

```
private double result = 1.0d;
double y = 1.0d;
for(int i=1; i<31; i++){
    y = x * y;
    result = result + (y / fakultaet(i));
}
return result;
```

```
public long fakultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++)
        result += a[i] * b[i];
    return result;
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

Writable Smart Insert 44 : 1

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSing.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
30     }  
31     return result;  
32 }  
33  
34 public long fakultaet(int n){  
35     long result = 1;  
36     while(n>1){  
37         result = result * n;  
38         n = n - 1;  
39     }  
40     return result;  
41 }  
42  
43 public double dotProduct(double[] a, double[] b){  
44     double result = 0.0;  
45     for(int i=0; i < a.length;){  
46         result = result + a[i] * b[i];  
47     }  
48     return result;  I
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)  
8  
2.7182818284590455
```

Writable Smart Insert 46 : 18

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSing.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
30     }  
31     return result;  
32 }  
33  
34 public long fakultaet(int n){  
35     long result = 1;  
36     while(n>1){  
37         result = result * n;  
38         n = n - 1;  
39     }  
40     return result;  
41 }  
42  
43 public double dotProduct(double[] a, double[] b){  
44     double result = 0.0;  
45     for(int i=0; i < a.length;){  
46         result = result + a[i] * b[i];  
47     }  
48     return result;  I
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)  
8  
2.7182818284590455
```

Writable Smart Insert 46 : 18

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
ICanSing.java BeeDemo.java AngryHornet... Flower.java *ControlStru... 2
```

File Edit Source Refactor Navigate Project Run Window Help

Package Explorer

```
1 package javaUebung1;  
2  
3 public class ControlStructuresDemo {  
4  
5     public static void main(String[] args) {  
6         // TODO Auto-generated method stub  
7         ControlStructuresDemo herbert = new ControlStructuresDemo();  
8         long x = 8;  
9         long y = 9;  
10        System.out.println(herbert.minimum(x,y));  
11        double xx = 1.0d;  
12        double yy = herbert.exp(xx);  
13        System.out.println(yy);  I  
14    }  
15  
16    public long minimum(long a, long b){  
17        if(a < b){  
18            return a;  
19        } else {  
20            return b;  
21        }  
22    }  
23  
24    public double exp(double x){  
25        double result = 1.0d;  
26        for(int i=1; i <= x; i++){  
27            result = result * x / i;  
28        }  
29        return result;  
30    }  
31  
32    public static void main(String[] args) {  
33        // TODO Auto-generated method stub  
34        ControlStructuresDemo herbert = new ControlStructuresDemo();  
35        long x = 8;  
36        long y = 9;  
37        System.out.println(herbert.minimum(x,y));  
38        double xx = 1.0d;  
39        double yy = herbert.exp(xx);  
40        System.out.println(yy);  I  
41    }  
42  
43    public long minimum(long a, long b){  
44        if(a < b){  
45            return a;  
46        } else {  
47            return b;  
48        }  
49    }  
50  
51    public static void main(String[] args) {  
52        // TODO Auto-generated method stub  
53        ControlStructuresDemo herbert = new ControlStructuresDemo();  
54        long x = 8;  
55        long y = 9;  
56        System.out.println(herbert.minimum(x,y));  
57        double xx = 1.0d;  
58        double yy = herbert.exp(xx);  
59        System.out.println(yy);  I  
60    }  
61  
62    public long minimum(long a, long b){  
63        if(a < b){  
64            return a;  
65        } else {  
66            return b;  
67        }  
68    }  
69  
70    public static void main(String[] args) {  
71        // TODO Auto-generated method stub  
72        ControlStructuresDemo herbert = new ControlStructuresDemo();  
73        long x = 8;  
74        long y = 9;  
75        System.out.println(herbert.minimum(x,y));  
76        double xx = 1.0d;  
77        double yy = herbert.exp(xx);  
78        System.out.println(yy);  I  
79    }  
80  
81    public long minimum(long a, long b){  
82        if(a < b){  
83            return a;  
84        } else {  
85            return b;  
86        }  
87    }  
88  
89    public static void main(String[] args) {  
90        // TODO Auto-generated method stub  
91        ControlStructuresDemo herbert = new ControlStructuresDemo();  
92        long x = 8;  
93        long y = 9;  
94        System.out.println(herbert.minimum(x,y));  
95        double xx = 1.0d;  
96        double yy = herbert.exp(xx);  
97        System.out.println(yy);  I  
98    }  
99  
100   public long minimum(long a, long b){  
101      if(a < b){  
102          return a;  
103      } else {  
104          return b;  
105      }  
106  }  
107  
108  public static void main(String[] args) {  
109      // TODO Auto-generated method stub  
110      ControlStructuresDemo herbert = new ControlStructuresDemo();  
111      long x = 8;  
112      long y = 9;  
113      System.out.println(herbert.minimum(x,y));  
114      double xx = 1.0d;  
115      double yy = herbert.exp(xx);  
116      System.out.println(yy);  I  
117  }  
118  
119  public long minimum(long a, long b){  
120      if(a < b){  
121          return a;  
122      } else {  
123          return b;  
124      }  
125  }  
126  
127  public static void main(String[] args) {  
128      // TODO Auto-generated method stub  
129      ControlStructuresDemo herbert = new ControlStructuresDemo();  
130      long x = 8;  
131      long y = 9;  
132      System.out.println(herbert.minimum(x,y));  
133      double xx = 1.0d;  
134      double yy = herbert.exp(xx);  
135      System.out.println(yy);  I  
136  }  
137  
138  public long minimum(long a, long b){  
139      if(a < b){  
140          return a;  
141      } else {  
142          return b;  
143      }  
144  }  
145  
146  public static void main(String[] args) {  
147      // TODO Auto-generated method stub  
148      ControlStructuresDemo herbert = new ControlStructuresDemo();  
149      long x = 8;  
150      long y = 9;  
151      System.out.println(herbert.minimum(x,y));  
152      double xx = 1.0d;  
153      double yy = herbert.exp(xx);  
154      System.out.println(yy);  I  
155  }  
156  
157  public long minimum(long a, long b){  
158      if(a < b){  
159          return a;  
160      } else {  
161          return b;  
162      }  
163  }  
164  
165  public static void main(String[] args) {  
166      // TODO Auto-generated method stub  
167      ControlStructuresDemo herbert = new ControlStructuresDemo();  
168      long x = 8;  
169      long y = 9;  
170      System.out.println(herbert.minimum(x,y));  
171      double xx = 1.0d;  
172      double yy = herbert.exp(xx);  
173      System.out.println(yy);  I  
174  }  
175  
176  public long minimum(long a, long b){  
177      if(a < b){  
178          return a;  
179      } else {  
180          return b;  
181      }  
182  }  
183  
184  public static void main(String[] args) {  
185      // TODO Auto-generated method stub  
186      ControlStructuresDemo herbert = new ControlStructuresDemo();  
187      long x = 8;  
188      long y = 9;  
189      System.out.println(herbert.minimum(x,y));  
190      double xx = 1.0d;  
191      double yy = herbert.exp(xx);  
192      System.out.println(yy);  I  
193  }  
194  
195  public long minimum(long a, long b){  
196      if(a < b){  
197          return a;  
198      } else {  
199          return b;  
200      }  
201  }  
202  
203  public static void main(String[] args) {  
204      // TODO Auto-generated method stub  
205      ControlStructuresDemo herbert = new ControlStructuresDemo();  
206      long x = 8;  
207      long y = 9;  
208      System.out.println(herbert.minimum(x,y));  
209      double xx = 1.0d;  
210      double yy = herbert.exp(xx);  
211      System.out.println(yy);  I  
212  }  
213  
214  public long minimum(long a, long b){  
215      if(a < b){  
216          return a;  
217      } else {  
218          return b;  
219      }  
220  }  
221  
222  public static void main(String[] args) {  
223      // TODO Auto-generated method stub  
224      ControlStructuresDemo herbert = new ControlStructuresDemo();  
225      long x = 8;  
226      long y = 9;  
227      System.out.println(herbert.minimum(x,y));  
228      double xx = 1.0d;  
229      double yy = herbert.exp(xx);  
230      System.out.println(yy);  I  
231  }  
232  
233  public long minimum(long a, long b){  
234      if(a < b){  
235          return a;  
236      } else {  
237          return b;  
238      }  
239  }  
240  
241  public static void main(String[] args) {  
242      // TODO Auto-generated method stub  
243      ControlStructuresDemo herbert = new ControlStructuresDemo();  
244      long x = 8;  
245      long y = 9;  
246      System.out.println(herbert.minimum(x,y));  
247      double xx = 1.0d;  
248      double yy = herbert.exp(xx);  
249      System.out.println(yy);  I  
250  }  
251  
252  public long minimum(long a, long b){  
253      if(a < b){  
254          return a;  
255      } else {  
256          return b;  
257      }  
258  }  
259  
260  public static void main(String[] args) {  
261      // TODO Auto-generated method stub  
262      ControlStructuresDemo herbert = new ControlStructuresDemo();  
263      long x = 8;  
264      long y = 9;  
265      System.out.println(herbert.minimum(x,y));  
266      double xx = 1.0d;  
267      double yy = herbert.exp(xx);  
268      System.out.println(yy);  I  
269  }  
270  
271  public long minimum(long a, long b){  
272      if(a < b){  
273          return a;  
274      } else {  
275          return b;  
276      }  
277  }  
278  
279  public static void main(String[] args) {  
280      // TODO Auto-generated method stub  
281      ControlStructuresDemo herbert = new ControlStructuresDemo();  
282      long x = 8;  
283      long y = 9;  
284      System.out.println(herbert.minimum(x,y));  
285      double xx = 1.0d;  
286      double yy = herbert.exp(xx);  
287      System.out.println(yy);  I  
288  }  
289  
290  public long minimum(long a, long b){  
291      if(a < b){  
292          return a;  
293      } else {  
294          return b;  
295      }  
296  }  
297  
298  public static void main(String[] args) {  
299      // TODO Auto-generated method stub  
300      ControlStructuresDemo herbert = new ControlStructuresDemo();  
301      long x = 8;  
302      long y = 9;  
303      System.out.println(herbert.minimum(x,y));  
304      double xx = 1.0d;  
305      double yy = herbert.exp(xx);  
306      System.out.println(yy);  I  
307  }  
308  
309  public long minimum(long a, long b){  
310      if(a < b){  
311          return a;  
312      } else {  
313          return b;  
314      }  
315  }  
316  
317  public static void main(String[] args) {  
318      // TODO Auto-generated method stub  
319      ControlStructuresDemo herbert = new ControlStructuresDemo();  
320      long x = 8;  
321      long y = 9;  
322      System.out.println(herbert.minimum(x,y));  
323      double xx = 1.0d;  
324      double yy = herbert.exp(xx);  
325      System.out.println(yy);  I  
326  }  
327  
328  public long minimum(long a, long b){  
329      if(a < b){  
330          return a;  
331      } else {  
332          return b;  
333      }  
334  }  
335  
336  public static void main(String[] args) {  
337      // TODO Auto-generated method stub  
338      ControlStructuresDemo herbert = new ControlStructuresDemo();  
339      long x = 8;  
340      long y = 9;  
341      System.out.println(herbert.minimum(x,y));  
342      double xx = 1.0d;  
343      double yy = herbert.exp(xx);  
344      System.out.println(yy);  I  
345  }  
346  
347  public long minimum(long a, long b){  
348      if(a < b){  
349          return a;  
350      } else {  
351          return b;  
352      }  
353  }  
354  
355  public static void main(String[] args) {  
356      // TODO Auto-generated method stub  
357      ControlStructuresDemo herbert = new ControlStructuresDemo();  
358      long x = 8;  
359      long y = 9;  
360      System.out.println(herbert.minimum(x,y));  
361      double xx = 1.0d;  
362      double yy = herbert.exp(xx);  
363      System.out.println(yy);  I  
364  }  
365  
366  public long minimum(long a, long b){  
367      if(a < b){  
368          return a;  
369      } else {  
370          return b;  
371      }  
372  }  
373  
374  public static void main(String[] args) {  
375      // TODO Auto-generated method stub  
376      ControlStructuresDemo herbert = new ControlStructuresDemo();  
377      long x = 8;  
378      long y = 9;  
379      System.out.println(herbert.minimum(x,y));  
380      double xx = 1.0d;  
381      double yy = herbert.exp(xx);  
382      System.out.println(yy);  I  
383  }  
384  
385  public long minimum(long a, long b){  
386      if(a < b){  
387          return a;  
388      } else {  
389          return b;  
390      }  
391  }  
392  
393  public static void main(String[] args) {  
394      // TODO Auto-generated method stub  
395      ControlStructuresDemo herbert = new ControlStructuresDemo();  
396      long x = 8;  
397      long y = 9;  
398      System.out.println(herbert.minimum(x,y));  
399      double xx = 1.0d;  
400      double yy = herbert.exp(xx);  
401      System.out.println(yy);  I  
402  }  
403  
404  public long minimum(long a, long b){  
405      if(a < b){  
406          return a;  
407      } else {  
408          return b;  
409      }  
410  }  
411  
412  public static void main(String[] args) {  
413      // TODO Auto-generated method stub  
414      ControlStructuresDemo herbert = new ControlStructuresDemo();  
415      long x = 8;  
416      long y = 9;  
417      System.out.println(herbert.minimum(x,y));  
418      double xx = 1.0d;  
419      double yy = herbert.exp(xx);  
420      System.out.println(yy);  I  
421  }  
422  
423  public long minimum(long a, long b){  
424      if(a < b){  
425          return a;  
426      } else {  
427          return b;  
428      }  
429  }  
430  
431  public static void main(String[] args) {  
432      // TODO Auto-generated method stub  
433      ControlStructuresDemo herbert = new ControlStructuresDemo();  
434      long x = 8;  
435      long y = 9;  
436      System.out.println(herbert.minimum(x,y));  
437      double xx = 1.0d;  
438      double yy = herbert.exp(xx);  
439      System.out.println(yy);  I  
440  }  
441  
442  public long minimum(long a, long b){  
443      if(a < b){  
444          return a;  
445      } else {  
446          return b;  
447      }  
448  }  
449  
450  public static void main(String[] args) {  
451      // TODO Auto-generated method stub  
452      ControlStructuresDemo herbert = new ControlStructuresDemo();  
453      long x = 8;  
454      long y = 9;  
455      System.out.println(herbert.minimum(x,y));  
456      double xx = 1.0d;  
457      double yy = herbert.exp(xx);  
458      System.out.println(yy);  I  
459  }  
460  
461  public long minimum(long a, long b){  
462      if(a < b){  
463          return a;  
464      } else {  
465          return b;  
466      }  
467  }  
468  
469  public static void main(String[] args) {  
470      // TODO Auto-generated method stub  
471      ControlStructuresDemo herbert = new ControlStructuresDemo();  
472      long x = 8;  
473      long y = 9;  
474      System.out.println(herbert.minimum(x,y));  
475      double xx = 1.0d;  
476      double yy = herbert.exp(xx);  
477      System.out.println(yy);  I  
478  }  
479  
480  public long minimum(long a, long b){  
481      if(a < b){  
482          return a;  
483      } else {  
484          return b;  
485      }  
486  }  
487  
488  public static void main(String[] args) {  
489      // TODO Auto-generated method stub  
490      ControlStructuresDemo herbert = new ControlStructuresDemo();  
491      long x = 8;  
492      long y = 9;  
493      System.out.println(herbert.minimum(x,y));  
494      double xx = 1.0d;  
495      double yy = herbert.exp(xx);  
496      System.out.println(yy);  I  
497  }  
498  
499  public long minimum(long a, long b){  
500      if(a < b){  
501          return a;  
502      } else {  
503          return b;  
504      }  
505  }  
506  
507  public static void main(String[] args) {  
508      // TODO Auto-generated method stub  
509      ControlStructuresDemo herbert = new ControlStructuresDemo();  
510      long x = 8;  
511      long y = 9;  
512      System.out.println(herbert.minimum(x,y));  
513      double xx = 1.0d;  
514      double yy = herbert.exp(xx);  
515      System.out.println(yy);  I  
516  }  
517  
518  public long minimum(long a, long b){  
519      if(a < b){  
520          return a;  
521      } else {  
522          return b;  
523      }  
524  }  
525  
526  public static void main(String[] args) {  
527      // TODO Auto-generated method stub  
528      ControlStructuresDemo herbert = new ControlStructuresDemo();  
529      long x = 8;  
530      long y = 9;  
531      System.out.println(herbert.minimum(x,y));  
532      double xx = 1.0d;  
533      double yy = herbert.exp(xx);  
534      System.out.println(yy);  I  
535  }  
536  
537  public long minimum(long a, long b){  
538      if(a < b){  
539          return a;  
540      } else {  
541          return b;  
542      }  
543  }  
544  
545  public static void main(String[] args) {  
546      // TODO Auto-generated method stub  
547      ControlStructuresDemo herbert = new ControlStructuresDemo();  
548      long x = 8;  
549      long y = 9;  
550      System.out.println(herbert.minimum(x,y));  
551      double xx = 1.0d;  
552      double yy = herbert.exp(xx);  
553      System.out.println(yy);  I  
554  }  
555  
556  public long minimum(long a, long b){  
557      if(a < b){  
558          return a;  
559      } else {  
560          return b;  
561      }  
562  }  
563  
564  public static void main(String[] args) {  
565      // TODO Auto-generated method stub  
566      ControlStructuresDemo herbert = new ControlStructuresDemo();  
567      long x = 8;  
568      long y = 9;  
569      System.out.println(herbert.minimum(x,y));  
570      double xx = 1.0d;  
571      double yy = herbert.exp(xx);  
572      System.out.println(yy);  I  
573  }  
574  
575  public long minimum(long a, long b){  
576      if(a < b){  
577          return a;  
578      } else {  
579          return b;  
580      }  
581  }  
582  
583  public static void main(String[] args) {  
584      // TODO Auto-generated method stub  
585      ControlStructuresDemo herbert = new ControlStructuresDemo();  
586      long x = 8;  
587      long y = 9;  
588      System.out.println(herbert.minimum(x,y));  
589      double xx = 1.0d;  
590      double yy = herbert.exp(xx);  
591      System.out.println(yy);  I  
592  }  
593  
594  public long minimum(long a, long b){  
595      if(a < b){  
596          return a;  
597      } else {  
598          return b;  
599      }  
600  }  
601  
602  public static void main(String[] args) {  
603      // TODO Auto-generated method stub  
604      ControlStructuresDemo herbert = new ControlStructuresDemo();  
605      long x = 8;  
606      long y = 9;  
607      System.out.println(herbert.minimum(x,y));  
608      double xx = 1.0d;  
609      double yy = herbert.exp(xx);  
610      System.out.println(yy);  I  
611  }  
612  
613  public long minimum(long a, long b){  
614      if(a < b){  
615          return a;  
616      } else {  
617          return b;  
618      }  
619  }  
620  
621  public static void main(String[] args) {  
622      // TODO Auto-generated method stub  
623      ControlStructuresDemo herbert = new ControlStructuresDemo();  
624      long x = 8;  
625      long y = 9;  
626      System.out.println(herbert.minimum(x,y));  
627      double xx = 1.0d;  
628      double yy = herbert.exp(xx);  
629      System.out.println(yy);  I  
630  }  
631  
632  public long minimum(long a, long b){  
633      if(a < b){  
634          return a;  
635      } else {  
636          return b;  
637      }  
638  }  
639  
640  public static void main(String[] args) {  
641      // TODO Auto-generated method stub  
642      ControlStructuresDemo herbert = new ControlStructuresDemo();  
643      long x = 8;  
644      long y = 9;  
645      System.out.println(herbert.minimum(x,y));  
646      double xx = 1.0d;  
647      double yy = herbert.exp(xx);  
648      System.out.println(yy);  I  
649  }  
650  
651  public long minimum(long a, long b){  
652      if(a < b){  
653          return a;  
654      } else {  
655          return b;  
656      }  
657  }  
658  
659  public static void main(String[] args) {  
660      // TODO Auto-generated method stub  
661      ControlStructuresDemo herbert = new ControlStructuresDemo();  
662      long x = 8;  
663      long y = 9;  
664      System.out.println(herbert.minimum(x,y));  
665      double xx = 1.0d;  
666      double yy = herbert.exp(xx);  
667      System.out.println(yy);  I  
668  }  
669  
670  public long minimum(long a, long b){  
671      if(a < b){  
672          return a;  
673      } else {  
674          return b;  
675      }  
676  }  
677  
678  public static void main(String[] args) {  
679      // TODO Auto-generated method stub  
680      ControlStructuresDemo herbert = new ControlStructuresDemo();  
681      long x = 8;  
682      long y = 9;  
683      System.out.println(herbert.minimum(x,y));  
684      double xx = 1.0d;  
685      double yy = herbert.exp(xx);  
686      System.out.println(yy);  I  
687  }  
688  
689  public long minimum(long a, long b){  
690      if(a < b){  
691          return a;  
692      } else {  
693          return b;  
694      }  
695  }  
696  
697  public static void main(String[] args) {  
698      // TODO Auto-generated method stub  
699      ControlStructuresDemo herbert = new ControlStructuresDemo();  
700      long x = 8;  
701      long y = 9;  
702      System.out.println(herbert.minimum(x,y));  
703      double xx = 1.0d;  
704      double yy = herbert.exp(xx);  
705      System.out.println(yy);  I  
706  }  
707  
708  public long minimum(long a, long b){  
709      if(a < b){  
710          return a;  
711      } else {  
712          return b;  
713      }  
714  }  
715  
716  public static void main(String[] args) {  
717      // TODO Auto-generated method stub  
718      ControlStructuresDemo herbert = new ControlStructuresDemo();  
719      long x = 8;  
720      long y = 9;  
721      System.out.println(herbert.minimum(x,y));  
722      double xx = 1.0d;  
723      double yy = herbert.exp(xx);  
724      System.out.println(yy);  I  
725  }  
726  
727  public long minimum(long a, long b){  
728      if(a < b){  
729          return a;  
730      } else {  
731          return b;  
732      }  
733  }  
734  
735  public static void main(String[] args) {  
736      // TODO Auto-generated method stub  
737      ControlStructuresDemo herbert = new ControlStructuresDemo();  
738      long x = 8;  
739      long y = 9;  
740      System.out.println(herbert.minimum(x,y));  
741      double xx = 1.0d;  
742      double yy = herbert.exp(xx);  
743      System.out.println(yy);  I  
744  }  
745  
746  public long minimum(long a, long b){  
747      if(a < b){  
748          return a;  
749      } else {  
750          return b;  
751      }  
752  }  
753  
754  public static void main(String[] args) {  
755      // TODO Auto-generated method stub  
756      ControlStructuresDemo herbert = new ControlStructuresDemo();  
757      long x = 8;  
758      long y = 9;  
759      System.out.println(herbert.minimum(x,y));  
760      double xx = 1.0d;  
761      double yy = herbert.exp(xx);  
762      System.out.println(yy);  I  
763  }  
764  
765  public long minimum(long a, long b){  
766      if(a < b){  
767          return a;  
768      } else {  
769          return b;  
770      }  
771  }  
772  
773  public static void main(String[] args) {  
774      // TODO Auto-generated method stub  
775      ControlStructuresDemo herbert = new ControlStructuresDemo();  
776      long x = 8;  
777      long y = 9;  
778      System.out.println(herbert.minimum(x,y));  
779      double xx = 1.0d;  
780      double yy = herbert.exp(xx);  
781      System.out.println(yy);  I  
782  }  
783  
784  public long minimum(long a, long b){  
785      if(a < b){  
786          return a;  
787      } else {  
788          return b;  
789      }  
790  }  
791  
792  public static void main(String[] args) {  
793      // TODO Auto-generated method stub  
794      ControlStructuresDemo herbert = new ControlStructuresDemo();  
795      long x = 8;  
796      long y = 9;  
797      System.out.println(herbert.minimum(x,y));  
798      double xx = 1.0d;  
799      double yy = herbert.exp(xx);  
800      System.out.println(yy);  I  
801  }  
802  
803  public long minimum(long a, long b){  
804      if(a < b){  
805          return a;  
806      } else {  
807          return b;  
808      }  
809  }  
810  
811  public static void main(String[] args) {  
812      // TODO Auto-generated method stub  
813      ControlStructuresDemo herbert = new ControlStructuresDemo();  
814      long x = 8;  
815      long y = 9;  
816      System.out.println(herbert.minimum(x,y));  
817      double xx = 1.0d;  
818      double yy = herbert.exp(xx);  
819      System.out.println(yy);  I  
820  }  
821  
822  public long minimum(long a, long b){  
823      if(a < b){  
824          return a;  
825      } else {  
826          return b;  
827      }  
828  }  
829  
830  public static void main(String[] args) {  
831      // TODO Auto-generated method stub  
832      ControlStructuresDemo herbert = new ControlStructuresDemo();  
833      long x = 8;  
834      long y = 9;  
835      System.out.println(herbert.minimum(x,y));  
836      double xx = 1.0d;  
837      double yy = herbert.exp(xx);  
838      System.out.println(yy);  I  
839  }  
840  
841  public long minimum(long a, long b){  
842      if(a < b){  
843          return a;  
844      } else {  
845          return b;  
846      }  
847  }  
848  
849  public static void main(String[] args) {  
850      // TODO Auto-generated method stub  
851      ControlStructuresDemo herbert = new ControlStructuresDemo();  
852      long x = 8;  
853      long y = 9;  
854      System.out.println(herbert.minimum(x,y));  
855      double xx = 1.0d;  
856      double yy = herbert.exp(xx);  
857      System.out.println(yy);  I  
858  }  
859  
860  public long minimum(long a, long b){  
861      if(a < b){  
862          return a;  
863      } else {  
864          return b;  
865      }  
866  }  
867  
868  public static void main(String[] args) {  
869      // TODO Auto-generated method stub  
870      ControlStructuresDemo herbert = new ControlStructuresDemo();  
871      long x = 8;  
872      long y = 9;  
873      System.out.println(herbert.minimum(x,y));  
874      double xx = 1.0d;  
875      double yy = herbert.exp(xx);  
876      System.out.println(yy);  I  
877  }  
878  
879  public long minimum(long a, long b){  
880      if(a < b){  
881          return a;  

```

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, etc.
- Quick Access:** A search bar at the top right.
- Java Perspective Buttons:** ICAnString.java, BeeDemo.java, AngryHornet..., Flower.java, *ControlStru..., 2
- Package Explorer:** Shows a tree view of Java packages and files, including BankAccount, BeeAndFlowers, BicycleDemo, ControlFlowDemo, DritteUebung, Erathostenes, Exceptions, Fakultat, FloodFill, Histogram, ImageDemo, InterfaceDemo, javaUebung1, KlausurJuli2014, OverloadAndOverride, Polymorphism, QuickSort, SimpleRecursion, StatementsAndOperators, uebung1, uebung2, uebung3, uebung4, zzz_EinfWL.
- Code Editor:** Displays the Java code for ControlStructuresDemo.java. The code includes a main method and a minimum function. A cursor is positioned at the start of the double declaration in line 14.
- Console:** Shows the output of the application's run command. It includes the application name, path, date, time, and the result of the minimum function call: 2.7182818284590455.
- Bottom Status Bar:** Writable, Smart Insert, 14 : 38, and system icons.
- System Taskbar:** Shows the Start button, task icons, battery level (99%), signal strength, and the date/time (19.06.2015).

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Quick Access:** A search bar at the top right.
- Java Perspective:** Indicated by the Java icon in the title bar.
- Package Explorer:** Shows a tree view of Java packages and classes, including a package named `aUebung1` containing classes like `BankAccount`, `BeesAndFlowers`, `BicycleDemo`, etc., and a package named `javaUebung1` containing `ControlStructuresDemo`.
- Editor:** The main editor window displays the Java code for `ControlStructuresDemo`. The code includes imports for `java.util`, a main method, variable declarations `x` and `y`, and a call to `herbert.minimum(x,y)`. It also contains a `minimum` method definition.
- Console:** The bottom window shows the output of the application. It starts with the application's name and path, followed by the value `8`, and then the result of the `minimum` method call, which is `2.7182818284590455`.
- Bottom Status Bar:** Shows the status "Writable", "Smart Insert", the current time "18:41", and battery level "99%".

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbars:** Standard, Java, Debug
- Quick Access:** A search bar at the top right.
- Package Explorer:** Shows a tree view of Java packages and classes, including BankAccount, BeesAndFlowers, BicycleDemo, ControlFlowDemo, DritteUebung, Erathostenes, Exceptions, Fakultaet, FloodFill, Histogram, imageUemo, InterfaceDemo, javaUebung1, KlausurJuli2014, OverloadAndOverride, Polymorphism, Quicksort, SimpleKuerkson, StatementsAndOperators, uebung1, uebung2, uebung3, uebung4, and zzz_EinfBWL.
- Editor:** The main editor window displays the Java code for ControlStructuresDemo.java. The code includes a main method that creates a Herbert object and calls its dotProduct and minimum methods. It also contains a static minimum method that returns the smaller of two long values.
- Console:** The console output shows the result of running the application, which is 2.7182818284590455.
- Bottom Status Bar:** Shows Writable, Smart Insert, 19:12, battery level (99%), signal strength, and the date/time (19.06.2015 09:47:26).

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard icons for file operations, search, and navigation.
- Quick Access:** A search bar at the top right.
- Java Perspective:** Indicated by the Java icon in the title bar.
- Package Explorer:** Shows the project structure with packages like BankAccount, BeesAndFlowers, BicycleDemo, ControlFlowDemo, DritteUebung, Erathostenes, Exceptions, Fakultaet, FloodFill, Histogram, ImageDemo, InterfaceDemo, javaUebung1, KlausurJuli2014, OverloadAndOverride, Polymorphism, QuidSort, SimpleRekursion, StatementsAndOperators, uebung1, uebung2, uebung3, uebung4, and zzz_EinfWL.
- Editor:** The main window displays the Java code for `ControlStructuresDemo`. The code includes imports for `java.util`, a main method, a `minimum` static method, and a `dotProduct` instance method. It uses various data types like long, double, and arrays.
- Console:** Shows the output of the application's execution, indicating it terminated successfully with a return value of 0 and the timestamp 19.06.2015 09:47:26. The output also includes the calculated result 2.7182818284590455.
- Bottom Status Bar:** Shows the status of the editor (Writable), the current line (49 : 52), battery level (99%), and system information (DE, 10:00, 19.06.2015).

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
32     double y = 1.0d;
33     for(int i=1; i<31; i++){
34         y = x * y;
35         result = result + (y / fakultaet(i));
36     }
37     return result;
38 }
39
40 public long fakultaet(int n){
41     long result = 1;
42     while(n>1){
43         result = result * n;
44         n = n - 1;
45     }
46     return result;
47 }
48
49 public double dotProduct(double[] a, double[] b){
50     double result = 0.0;
51     for(int i=0; i < a.length; i++)
52         result = result + a[i] * b[i];
53     return result;
54 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 09:47:26)
8
2.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
2
3 public class ControlStructuresDemo {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         ControlStructuresDemo herbert = new ControlStructuresDemo();
8         long x = 8;
9         long y = 9;
10        System.out.println(herbert.minimum(x,y));
11        double xx = 1.0d;
12        double yy = herbert.exp(xx);
13        System.out.println(yy);
14        double[] xxx = new double[3];
15        xxx[0]=2.0d;
16        xxx[1]=1.0d;
17        xxx[2]=0.0d;
18        double[] yyy = {1.0d,1.0d,1.0d};
19        yy = herbert.dotProduct(xxx, yyy);
20        System.out.println(yy);
21    }
22
23    public long minimum(long a, long b){
24        if(a < b){
25            return a;
26        } else {
27            return b;
28        }
29 }
```

Console

```
ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:01:48)
8
2.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
33     double y = 1.0d;
34     for(int i=1; i<31; i++){
35         y = x * y;
36         result = result + (y / fakultaet(i));
37     }
38     return result;
39 }
40
41 public long fakultaet(int n){
42     long result = 1;
43     while(n>1){
44         result = result * n;
45         n = n - 1;
46     }
47     return result;
48 }
49
50 public double dotProduct(double[] a, double[] b){
51     double result = 0.0;
52     for(int i=0; i < a.length; i++)
53         result = result + a[i] * b[i];
54     return result;
55 }
```

Console

```
ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:03:57)
8
2.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
33     double y = 1.0d;
34     for(int i=1; i<31; i++){
35         y = x * y;
36         result = result + (y / fakultaet(i));
37     }
38     return result;
39 }
40
41 public long fakultaet(int n){
42     long result = 1;
43     while(n>1){
44         result = result * n;
45         n = n - 1;
46     }
47     return result;
48 }
49
50 public double dotProduct(double[] a, double[] b){
51     double result = 0.0;
52     for(int i=0; i < a.length; i++)
53         result = result + a[i] * b[i];
54     return result;
55 }
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:03:57)
8
2.7182818284590455
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
7 // TODO Auto-generated method stub
8 ControlStructuresDemo herbert = new ControlStructuresDemo();
9 long x = 8;
10 long y = 9;
11 System.out.println(herbert.minimum(x,y));
12 double xx = 1.0d;
13 double yy = herbert.exp(xx);
14 System.out.println(yy);
15 double[] xxx = new double[3];
16 xxx[0]=2.0d;
17 xxx[0]=1.0d;
18 xxx[0]=0.0d;
19 double[] yyy = {1.0d,1.0d,1.0d};
20 yy = herbert.dotProduct(xxx, yyy);
21 System.out.println(yy);
22 }
23
24 public long minimum(long a, long b){
25     if(a < b){
26         return a;
27     } else {
28         return b;
29     }
30 }
31
32 public double exp (double x){
33 }
```

Console

```
No consoles to display|Structs time. application C:\P
8
2.7182818284590455
0.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
6 // TODO Auto-generated method stub
7 ControlStructuresDemo herbert = new ControlStructuresDemo();
8 long x = 8;
9 long y = 9;
10 System.out.println(herbert.minimum(x,y));
11 double xx = 1.0d;
12 double yy = herbert.exp(xx);
13 System.out.println(yy);
14 double[] xxx = new double[3];
15 xxx[0]=2.0d;
16 xxx[0]=1.0d;
17 xxx[0]=0.0d;
18 double[] yyy = {1.0d,1.0d,1.0d};
19 yy = herbert.dotProduct(xxx, yyy);
20 System.out.println(yy);
21 }
22
23
24 public long minimum(long a, long b){
25     if(a < b){
26         return a;
27     } else {
28         return b;
29     }
30 }
31
32 public double exp (double x){
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:04:47)
8
2.7182818284590455
0.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
6 // TODO Auto-generated method stub
7 ControlStructuresDemo herbert = new ControlStructuresDemo();
8 long x = 8;
9 long y = 9;
10 System.out.println(herbert.minimum(x,y));
11 double xx = 1.0d;
12 double yy = herbert.exp(xx);
13 System.out.println(yy);
14 double[] xxx = new double[3];
15 xxx[0]=2.0d;
16 xxx[0]=1.0d;
17 xxx[0]=0.0d;
18 double[] yyy = {1.0d,1.0d,1.0d};
19 yy = herbert.dotProduct(xxx, yyy);
20 System.out.println(yy);
21 }
22
23
24 public long minimum(long a, long b){
25     if(a < b){
26         return a;
27     } else {
28         return b;
29     }
30 }
31
32 public double exp (double x){
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:04:47)
8
2.7182818284590455
0.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
6 // TODO Auto-generated method stub
7 ControlStructuresDemo herbert = new ControlStructuresDemo();
8 long x = 8;
9 long y = 9;
10 System.out.println(herbert.minimum(x,y));
11 double xx = 1.0d;
12 double yy = herbert.exp(xx);
13 System.out.println(yy);
14 double[] xxx = new double[3];
15 xxx[0]=2.0d;
16 xxx[1]=1.0d;
17 xxx[2]=0.0d;
18 double[] yyy = {1.0d,1.0d,1.0d};
19 yy = herbert.dotProduct(xxx, yyy);
20 System.out.println(yy);
21 }
22
23
24 public long minimum(long a, long b){
25     if(a < b){
26         return a;
27     } else {
28         return b;
29     }
30 }
31
32 public double exp (double x){
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
double y = 1.0d;
for(int i=1; i<31; i++){
    y = x * y;
    result = result + (y / faktultaet(i));
}
return result;
```

```
public long faktultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++){
        result = result + a[i] * b[i];
    }
    return result;
}
```

```
public double multMatrixVector()
```

Syntax error on token ")", { expected after this token

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
double y = 1.0d;
for(int i=1; i<31; i++){
    y = x * y;
    result = result + (y / faktultaet(i));
}
return result;
```

```
public long faktultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++){
        result = result + a[i] * b[i];
    }
    return result;
}
```

```
public double multMatrixVector()
```

Syntax error on token ")", { expected after this token

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
}
```

```
public long faktultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++){
        result = result + a[i] * b[i];
    }
    return result;
}
```

```
public double multMatrixVector(double[][] a, double[] b){
```

Syntax error on token ")", { expected after this token

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
}
```

```
public long faktultaet(int n){
    long result = 1;
    while(n>1){
        result = result * n;
        n = n - 1;
    }
    return result;
}
```

```
public double dotProduct(double[] a, double[] b){
    double result = 0.0;
    for(int i=0; i < a.length; i++){
        result = result + a[i] * b[i];
    }
    return result;
}
```

```
public double multMatrixVector(double[][] a, double[] b){
```

Syntax error on token ")", { expected after this token

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
37     }  
38     return result;  
39 }  
40  
public long fakultaet(int n){  
    long result = 1;  
    while(n>1){  
        result = result * n;  
        n = n - 1;  
    }  
    return result;  
}  
49  
50@ public double dotProduct(double[] a, double[] b){  
    double result = 0.0;  
    for(int i=0; i < a.length; i++){  
        result = result + a[i] * b[i];  
    }  
    return result;  
}  
58@ public double multMatrixVector(double[][] a, double[] b){  
    double result = 0.0d;  
      
    return result;  
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)  
8  
2.7182818284590455  
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
37     }  
38     return result;  
39 }  
40  
public long fakultaet(int n){  
    long result = 1;  
    while(n>1){  
        result = result * n;  
        n = n - 1;  
    }  
    return result;  
}  
49  
50@ public double dotProduct(double[] a, double[] b){  
    double result = 0.0;  
    for(int i=0; i < a.length; i++){  
        result = result + a[i] * b[i];  
    }  
    return result;  
}  
58@ public double multMatrixVector(double[][] a, double[] b){  
    double result = 0.0d;  
      
    return result;  
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)  
8  
2.7182818284590455  
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
37     }  
38     return result;  
39 }  
40  
public long fakultaet(int n){  
    long result = 1;  
    while(n>1){  
        result = result * n;  
        n = n - 1;  
    }  
    return result;  
}  
49  
50@ public double dotProduct(double[] a, double[] b){  
    double result = 0.0;  
    for(int i=0; i < a.length; i++){  
        result = result + a[i] * b[i];  
    }  
    return result;  
}  
58@ public double[] multMatrixVector(double[][] a, double[] b){  
    double result[];  
      
    return result;  
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)  
8  
2.7182818284590455  
3.0
```

aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse

```
37     }  
38     return result;  
39 }  
40  
public long fakultaet(int n){  
    long result = 1;  
    while(n>1){  
        result = result * n;  
        n = n - 1;  
    }  
    return result;  
}  
49  
50@ public double dotProduct(double[] a, double[] b){  
    double result = 0.0;  
    for(int i=0; i < a.length; i++){  
        result = result + a[i] * b[i];  
    }  
    return result;  
}  
58@ public double[] multMatrixVector(double[][] a, double[] b){  
    double result[] = new double[3];  
      
    return result;  
}
```

Console

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)  
8  
2.7182818284590455  
3.0
```

Eclipse IDE screenshot showing the Java code for `ControlStructuresDemo.java`. The code includes methods for calculating factorials, dot products, and matrix-vector products.

```
38     return result;
39 }
40
41 public long fakultaet(int n){
42     long result = 1;
43     while(n>1){
44         result = result * n;
45         n = n - 1;
46     }
47     return result;
48 }
49
50 public double dotProduct(double[] a, double[] b){
51     double result = 0.0;
52     for(int i=0; i < a.length; i++){
53         result = result + a[i] * b[i];
54     }
55     return result;
56 }
57
58 public double[] multMatrixVector(double[][] a, double[] b){
59     double result[] = new double[3];
60     for(int i=0; i<3; i++){
61         result[i] = 0.0d;
62         for(int j=0; j<3; j++){
63             result[i] = result[i] + a[i][j] * b[j];
64         }
65     }
66     return result;
67 }
```

The console output shows the results of the factorial and dot product calculations:

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

Eclipse IDE screenshot showing the Java code for `ControlStructuresDemo.java`. The code includes methods for calculating factorials, dot products, and matrix-vector products.

```
38     return result;
39 }
40
41 public long fakultaet(int n){
42     long result = 1;
43     while(n>1){
44         result = result * n;
45         n = n - 1;
46     }
47     return result;
48 }
49
50 public double dotProduct(double[] a, double[] b){
51     double result = 0.0;
52     for(int i=0; i < a.length; i++){
53         result = result + a[i] * b[i];
54     }
55     return result;
56 }
57
58 public double[] multMatrixVector(double[][] a, double[] b){
59     double result[] = new double[3];
60     for(int i=0; i<3; i++){
61         result[i] = 0.0d;
62         for(int j=0; j<3; j++){
63             result[i] = result[i] + a[i][j] * b[j];
64         }
65     }
66     return result;
67 }
```

The console output shows the results of the factorial and dot product calculations:

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

Eclipse IDE screenshot showing the Java code for `ControlStructuresDemo.java`. The code includes methods for calculating factorials, dot products, and matrix-vector products.

```
38     return result;
39 }
40
41 public long fakultaet(int n){
42     long result = 1;
43     while(n>1){
44         result = result * n;
45         n = n - 1;
46     }
47     return result;
48 }
49
50 public double dotProduct(double[] a, double[] b){
51     double result = 0.0;
52     for(int i=0; i < a.length; i++){
53         result = result + a[i] * b[i];
54     }
55     return result;
56 }
57
58 public double[] multMatrixVector(double[][] a, double[] b){
59     double result[] = new double[3];
60     for(int i=0; i<3; i++){
61         result[i] = 0.0d;
62         for(int j=0; j<3; j++){
63             result[i] = result[i] + a[i][j] * b[j];
64         }
65     }
66     return result;
67 }
```

The console output shows the results of the factorial and dot product calculations:

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

Eclipse IDE screenshot showing the Java code for `ControlStructuresDemo.java`. The code includes methods for calculating factorials, dot products, and matrix-vector products.

```
38     result = result * n;
39     n = n - 1;
40 }
41     return result;
42 }
43
44 public double dotProduct(double[] a, double[] b){
45     double result = 0.0;
46     for(int i=0; i < a.length; i++){
47         result = result + a[i] * b[i];
48     }
49     return result;
50 }
51
52 public double[] multMatrixVector(double[][] a, double[] b){
53     double result[] = new double[3];
54     for(int i=0; i<3; i++){
55         result[i] = 0.0d;
56         for(int j=0; j<3; j++){
57             result[i] = result[i] + a[i][j] * b[j];
58         }
59     }
60     return result;
61 }
62
63 }
```

The console output shows the results of the factorial and dot product calculations:

```
<terminated> ControlStructuresDemo (1) [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (19.06.2015 10:07:05)
8
2.7182818284590455
3.0
```

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Includes icons for New, Open, Save, Cut, Copy, Paste, Find, Select All, etc.
- Quick Access:** A search bar at the top right.
- Java Perspective:** Indicated by the Java and Debug buttons in the top right.
- Package Explorer:** Shows a tree view of Java packages and classes, including:
 - BankAccount
 - BeeAndFlowers
 - BicycleDemo
 - ControlFlowDemo
 - DritteUebung
 - Eratosthenes
 - Exceptions
 - Fakultaet
 - FloodFill
 - Histogram
 - ImageDemo
 - InterfaceDemo
 - javaUebung1
 - KlausurJuli2014
 - OverloadAndOverride
 - Polyorphism
 - Quicksort
 - SimpleRecursion
 - StatementsAndOperators
 - uebung1
 - uebung2
 - uebung3
 - uebung4
 - zzz_EinfBWL
- Editor:** The main window displays the source code for `ControlStructuresDemo.java`. The code includes methods for calculating dot products and matrix-vector multiplication.
- Console:** Shows the terminal output of the application, including the result of a calculation: 2.7182818284590455.
- Bottom Status Bar:** Shows Writable, Smart Insert, 63: 56, and battery status (99%).
- Windows Taskbar:** Shows icons for Start, Internet Explorer, File Explorer, and others.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar icons.
- Quick Access:** A search bar at the top right.
- Package Explorer:** Shows a tree view of Java packages and files. Visible nodes include BankAccount, BeesAndFlowers, BicyclesDemo, ControlFlowDemo, DritteUebung, Eratosthenes, Exceptions, Fakultaet, Floodfill, Histogram, ImageDemo, InterfaceDemo, javaUebung1, KlausurJuuli2014, OverloadAndOverride, Polymorphism, Quicksort, SimpleRecursion, StatementsAndOperators, uebung1, uebung2, uebung3, uebung4, and zzz_EinfWL.
- Editor:** The main code editor displays the file ICanString.java. The code implements methods for dotProduct and multMatrixVector, both utilizing nested loops to calculate results.
- Console:** The bottom console window shows the output of the application, which is the result of the dotProduct method: 3.0.
- System Tray:** Shows battery level (99%), signal strength, and system status.

The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** aUebung1/src/javaUebung1/ControlStructuresDemo.java - Eclipse
- Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbars:** Standard, Java, Debug
- Quick Access:** A search bar at the top right.
- Package Explorer:** Shows various Java projects and files, including BankAccount, BeesAndFlowers, BicycleDemo, ControlFlowDemo, DritteUebung, Erathostenes, Exceptions, Fakultaet, FloodFill, Histogram, ImageUemo, InterfaceDemo, javaUebung1, KlausurJuli2014, OverloadAndOverride, Polymorphism, Quicksort, SimpleKuerkson, StatementsAndOperators, uebung1, uebung2, uebung3, uebung4, and zzz_EinfBWL.
- Editor:** The main editor window displays the Java code for ControlStructuresDemo.java. The code includes methods for calculating factorials, dot products, and matrix-vector multiplication.
- Console:** The bottom window shows the terminal output of the application. It prints the value of PI (3.141592653589793) and the result of a matrix-vector multiplication (2.7182818284590455).
- Bottom Status Bar:** Shows battery level (99%), signal strength, and system status.

DATEI START EINFÜGEN ENTWURF ÜBERGÄNGE ANIMATIONEN BILDSCRIMPRÄSENTATION UBERPRÜFEN ANSICHT Anmelden

Von Ab aktueller Folie Online Benutzerdefinierte Bildschirmpäsentation starten

Folie ausblenden Neue Anzeigedauern testen Bildschirmpäsentation aufzeichnen Mediensteuerelemente anzeigen

Bildschirme

55

56

57

58

59

12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | Einrichten

Darstellung / Approximation von reellen Zahlen

`float ggg = -345545.34534E-12f; = -345545.34534 * 10-12` } ∈ ℚ „≈“
`doublesss = 3245343455.555E67; = 3245343455.555 * 1067`

- mit **beschränkter Anzahl Bits**: Nur **Approximation** von reellen Zahlen (bspw. π) bzw. rationalen Zahlen mit nicht abbrechender Dezimalbruchentwicklung (bspw. $1/3$) möglich. (Standard: IEEE 754).
- Folgen: **Numerische Fehler** möglich. Bsp:

```
21000 double e = Math.pow(2.0d,1000.0d); //1.0715086071862673E301
25000 double f = Math.pow(2.0d,5000.0d); //Infinity (Zahlbereichsüberschreitung)
double g = 0.1d + 0.1d + 0.1d; //0.30000000000000004 (Rundungsfehler)
```

siehe auch: <http://introcs.cs.princeton.edu/java/91float/>

- **Test auf Gleichheit** zweier `float` oder `double` Werte x, y nicht mit $x == y$, sondern mit $|x - y| < \epsilon$;
- viele weitere **Konsequenzen** → numerische Mathematik
- Es gibt in Java auch Möglichkeiten mit **beliebiger Genauigkeit** zu rechnen (bspw. mit Klasse `BigDecimal`)

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 | 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 | 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 | 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 | 401 | 402 | 403 | 404 | 405 | 406 | 407 | 408 | 409 | 410 | 411 | 412 | 413 | 414 | 415 | 416 | 417 | 418 | 419 | 420 | 421 | 422 | 423 | 424 | 425 | 426 | 427 | 428 | 429 | 430 | 431 | 432 | 433 | 434 | 435 | 436 | 437 | 438 | 439 | 440 | 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 450 | 451 | 452 | 453 | 454 | 455 | 456 | 457 | 458 | 459 | 460 | 461 | 462 | 463 | 464 | 465 | 466 | 467 | 468 | 469 | 470 | 471 | 472 | 473 | 474 | 475 | 476 | 477 | 478 | 479 | 480 | 481 | 482 | 483 | 484 | 485 | 486 | 487 | 488 | 489 | 490 | 491 | 492 | 493 | 494 | 495 | 496 | 497 | 498 | 499 | 500 | 501 | 502 | 503 | 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 | 512 | 513 | 514 | 515 | 516 | 517 | 518 | 519 | 520 | 521 | 522 | 523 | 524 | 525 | 526 | 527 | 528 | 529 | 530 | 531 | 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 560 | 561 | 562 | 563 | 564 | 565 | 566 | 567 | 568 | 569 | 570 | 571 | 572 | 573 | 574 | 575 | 576 | 577 | 578 | 579 | 580 | 581 | 582 | 583 | 584 | 585 | 586 | 587 | 588 | 589 | 590 | 591 | 592 | 593 | 594 | 595 | 596 | 597 | 598 | 599 | 600 | 601 | 602 | 603 | 604 | 605 | 606 | 607 | 608 | 609 | 610 | 611 | 612 | 613 | 614 | 615 | 616 | 617 | 618 | 619 | 620 | 621 | 622 | 623 | 624 | 625 | 626 | 627 | 628 | 629 | 630 | 631 | 632 | 633 | 634 | 635 | 636 | 637 | 638 | 639 | 640 | 641 | 642 | 643 | 644 | 645 | 646 | 647 | 648 | 649 | 650 | 651 | 652 | 653 | 654 | 655 | 656 | 657 | 658 | 659 | 660 | 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 670 | 671 | 672 | 673 | 674 | 675 | 676 | 677 | 678 | 679 | 680 | 681 | 682 | 683 | 684 | 685 | 686 | 687 | 688 | 689 | 690 | 691 | 692 | 693 | 694 | 695 | 696 | 697 | 698 | 699 | 700 | 701 | 702 | 703 | 704 | 705 | 706 | 707 | 708 | 709 | 710 | 711 | 712 | 713 | 714 | 715 | 716 | 717 | 718 | 719 | 720 | 721 | 722 | 723 | 724 | 725 | 726 | 727 | 728 | 729 | 720 | 721 | 722 | 723 | 724 | 725 | 726 | 727 | 728 | 729 | 730 | 731 | 732 | 733 | 734 | 735 | 736 | 737 | 738 | 739 | 730 | 731 | 732 | 733 | 734 | 735 | 736 | 737 | 738 | 739 | 740 | 741 | 742 | 743 | 744 | 745 | 746 | 747 | 748 | 749 | 740 | 741 | 742 | 743 | 744 | 745 | 746 | 747 | 748 | 749 | 750 | 751 | 752 | 753 | 754 | 755 | 756 | 757 | 758 | 759 | 750 | 751 | 752 | 753 | 754 | 755 | 756 | 757 | 758 | 759 | 760 | 761 | 762 | 763 | 764 | 765 | 766 | 767 | 768 | 769 | 760 | 761 | 762 | 763 | 764 | 765 | 766 | 767 | 768 | 769 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 780 | 781 | 782 | 783 | 784 | 785 | 786 | 787 | 788 | 789 | 780 | 781 | 782 | 783 | 784 | 785 | 786 | 787 | 788 | 789 | 790 | 791 | 792 | 793 | 794 | 795 | 796 | 797 | 798 | 799 | 790 | 791 | 792 | 793 | 794 | 795 | 796 | 797 | 798 | 799 | 800 | 801 | 802 | 803 | 804 | 805 | 806 | 807 | 808 | 809 | 800 | 801 | 802 | 803 | 804 | 805 | 806 | 807 | 808 | 809 | 810 | 811 | 812 | 813 | 814 | 815 | 816 | 817 | 818 | 819 | 810 | 811 | 812 | 813 | 814 | 815 | 816 | 817 | 818 | 819 | 820 | 821 | 822 | 823 | 824 | 825 | 826 | 827 | 828 | 829 | 820 | 821 | 822 | 823 | 824 | 825 | 826 | 827 | 828 | 829 | 830 | 831 | 832 | 833 | 834 | 835 | 836 | 837 | 838 | 839 | 830 | 831 | 832 | 833 | 834 | 835 | 836 | 837 | 838 | 839 | 840 | 841 | 842 | 843 | 844 | 845 | 846 | 847 | 848 | 849 | 840 | 841 | 842 | 843 | 844 | 845 | 846 | 847 | 848 | 849 | 850 | 851 | 852 | 853 | 854 | 855 | 856 | 857 | 858 | 859 | 850 | 851 | 852 | 853 | 854 | 855 | 856 | 857 | 858 | 859 | 860 | 861 | 862 | 863 | 864 | 865 | 866 | 867 | 868 | 869 | 860 | 861 | 862 | 863 | 864 | 865 | 866 | 867 | 868 | 869 | 870 | 871 | 872 | 873 | 874 | 875 | 876 | 877 | 878 | 879 | 870 | 871 | 872 | 873 | 874 | 875 | 876 | 877 | 878 | 879 | 880 | 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 880 | 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 890 | 891 | 892 | 893 | 894 | 895 | 896 | 897 | 898 | 899 | 890 | 891 | 892 | 893 | 894 | 895 | 896 | 897 | 898 | 899 | 900 | 901 | 902 | 903 | 904 | 905 | 906 | 907 | 908 | 909 | 900 | 901 | 902 | 903 | 904 | 905 | 906 | 907 | 908 | 909 | 910 | 911 | 912 | 913 | 914 | 915 | 916 | 917 | 918 | 919 | 910 | 911 | 912 | 913 | 914 | 915 | 916 | 917 | 918 | 919 | 920 | 921 | 922 | 923 | 924 | 925 | 926 | 927 | 928 | 929 | 920 | 921 | 922 | 923 | 924 | 925 | 926 | 927 | 928 | 929 | 930 | 931 | 932 | 933 | 934 | 935 | 936 | 937 | 938 | 939 | 930 | 931 | 932 | 933 | 934 | 935 | 936 | 937 | 938 | 939 | 940 | 941 | 942 | 943 | 944 | 945 | 946 | 947 | 948 | 949 | 940 | 941 | 942 | 943 | 944 | 945 | 946 | 947 | 948 | 949 | 950 | 951 | 952 | 953 | 954 | 955 | 956 | 957 | 958 | 959 | 950 | 951 | 952 | 953 | 954 | 955 | 956 | 957 | 958 | 959 | 960 | 961 | 962 | 963 | 964 | 965 | 966 | 967 | 968 | 969 | 960 | 961 | 962 | 963 | 964 | 965 | 966 | 967 | 968 | 969 | 970 | 971 | 972 | 973 | 974 | 975 | 976 | 977 | 978 | 979 | 970 | 971 | 972 | 973 | 974 | 975 | 976 | 977 | 978 | 979 | 980 | 981 | 982 | 983 | 984 | 985 | 986 | 987 | 988 | 989 | 980 | 981 | 982 | 983 | 984 | 985 | 986 | 987 | 988 | 989 | 990 | 991 | 992 | 993 | 994 | 995 | 996 | 997 | 998 | 999 | 990 | 991 | 992 | 993 | 994 | 995 | 996 | 997 | 998 | 999 | 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 | 1008 | 1009 | 1000 | 1001 | 1002 | 1003 | 1004 | 1005 | 1006 | 1007 | 1008 | 1009 | 1010 | 1011 | 1012 | 1013 | 1014 | 1015 | 1016 | 1017 | 1018 | 1019 | 1010 | 1011 | 1012 | 1013 | 1014 | 1015 | 1016 | 1017 | 1018 | 1019 | 1020 | 1021 | 1022 | 1023 | 1024 | 1025 | 1026 | 1027 | 1028 | 1029 | 1020 | 1021 | 1022 | 1023 | 1024 | 1025 | 1026 | 1027 | 1028 | 1029 | 1030 | 1031 | 1032 | 1033 | 1034 | 1035 | 1036 | 1037 | 1038 | 1039 | 1030 | 1031 | 1032 | 1033 | 1034 | 1035 | 1036 | 1037 | 1038 | 1039 | 1040 | 1041 | 1042 | 1043 | 1044 | 1045 | 1046 | 1047 | 1048 | 1049 | 1040 | 1041 | 1042 | 1043 | 1044 | 1045 | 1046 | 1047 | 1048 | 1049 | 1050 | 1051 | 1052 | 1053 | 1054 | 1055 | 1056 | 1057 | 1058 | 1059 | 1050 | 1051 | 1052 | 1053 | 1054 | 1055 | 1056 | 1057 | 1058 | 1059 | 1060 | 1061 | 1062 | 1063 | 1064 | 1065 | 1066 | 1067 | 1068 | 1069 | 1060 | 1061 | 1062 | 1063 | 1064 | 1065 | 1066 | 1067 | 1068 | 1069 | 1070 | 1071 | 1072 | 1073 | 1074 | 1075 | 1076 | 1077 | 1078 | 1079 | 1070 | 1071 | 1072 | 1073 | 1074 | 1075 | 1076 | 1077 | 1078 | 1079 | 1080 | 1081 | 1082 | 1083 | 1084 | 1085 | 1086 | 1087 | 1088 | 1089 | 1080 | 1081 | 1082 | 1083 | 1084 | 1085 | 1086 | 1087 | 1088 | 1089 | 1090 | 1091 | 1092 | 1093 | 1094 | 1095 | 1096 | 1097 | 1098 | 1099 | 1090 | 1091 | 1092 | 1093 | 1094 | 1095 | 1096 | 1097 | 1098 | 1099 | 1100 | 1101 | 1102 | 1103 | 1104 | 1105 | 1106 | 1107 | 1108 | 1109 | 1100 | 1101 | 1102 | 1103 | 1104 | 1105 | 1106 | 1107 | 1108 | 1109 | 1110 | 1111 | 1112 | 1113 | 1114 | 1115 | 1116 | 1117 | 1118 | 1119 | 1110 | 1111 | 1112 | 1113 | 1114 | 1115 | 1116 | 1117 | 1118 | 1119 | 1120 | 1121 | 1122 | 1123 | 1124 | 1125 | 1126 | 1127 | 1128 | 1129 | 1120 | 1121 | 1122 | 1123 | 1124 | 1125 | 1126 | 1127 | 1128 | 1129 | 1130 | 1131 | 1132 | 1133 | 1134 | 1135 | 1136 | 1137 | 1138 | 1139 | 1130 | 1131 | 1132 | 1133 | 1134 | 1135 | 1136 | 1137 | 1138 | 1139 | 1140 | 1141 | 1142 | 1143 | 1144 | 1145 | 1146 | 1147 | 1148 | 1149 | 1140 | 1141 | 1142 | 1143 | 1144 | 1145 | 1146 | 1147 | 1148 | 1149 | 1150 | 1151 | 1152 | 1153 | 1154 | 1155 | 1156 | 1157 | 1158 | 1159 | 1150 | 1151 | 1152 | 1153 | 1154 | 1155 | 1156 | 1157 | 1158 | 1159 | 1160 | 1161 | 1162 | 1163 | 1164 | 1165 | 1166 | 1167 | 1168 | 1169 | 1160 | 1161 | 1162 | 1163 | 1164 | 1165 | 1166 | 1167 | 1168 | 1169 | 1170 | 1171 | 1172 | 1173 | 1174 | 1175 | 1176 | 1177 | 1178 | 1179 | 1170 | 1171 | 1172 | 1173 | 1174 | 1175 | 1176 | 1177 | 1178 | 1179 | 1180 | 1181 | 1182 | 1183 | 1184 | 1185 | 1186 | 1187 | 1188 | 1189 | 1180 | 1181 | 1182 | 1183 | 1184 | 1185 | 1186 | 1187 | 1188 | 1189 | 1190 | 1191 | 1192 | 1193 | 1194 | 1195 | 1196 | 1197 | 1198 | 1199 | 1190 | 1191 | 1192 | 1193 | 1194 | 1195 | 1196 | 1197 | 1198 | 1199 | 1200 | 1201 | 1202 | 1203 | 1204 | 1205 | 1206 | 1207 | 1208 | 1209 | 1200 | 1201 | 1202 | 1203 | 1204 | 1205 | 1206 | 1207 | 1208 | 1209 | 1210 | 1211 | 1212 | 1213 | 1214 | 1215 | 1216 | 1217 | 1218 | 1219 | 1210 | 1211 | 1212 | 1213 | 1214 | 1215 | 1216 | 1217 | 1218 | 1219 | 1220 | 1221 | 1222 | 1223 | 1224 | 1225 | 1226 | 1227 | 1228 | 1229 | 1220 | 1221 | 1222 | 1223 | 1224 | 1225 | 1226 | 1227 | 1228 | 1229 | 1230 | 1231 | 1232 | 1233 | 1234 | 1235 | 1236 | 1237 | 1238 | 1239 | 1230 | 1231 | 1232 | 1233 | 1234 | 1235 | 1236 | 1237 | 1238 | 1239 | 1240 | 1241 | 1242 | 1243 | 1244 | 1245 | 1246 | 1247 | 1248 | 1249 | 1240 | 1241 | 1242 | 1243 | 1244 | 1245 | 1246 | 1247 | 1248 | 1249 | 1250 | 1251 | 1252 | 1253 | 1254 | 1255 | 1256 | 1257 | 1258 | 1259 | 1250 | 1251 | 1252 | 1253 | 1254 | 1255 | 1256 | 1257 | 1258 | 1259 | 1260 | 1261 | 1262 | 1263 | 1264 | 1265 | 1266 | 1267 | 1268 | 1269 | 1260 | 1261 | 1262 | 1263 | 1264 | 1265 | 1266 | 1267 | 1268 | 1269 | 1270 | 1271 | 1272 | 1273 | 1274 | 1275 | 1276 | 1277 | 1278 | 1279 | 1270 | 1271 | 1272 | 1273 | 1274 | 1275 | 1276 | 1277 | 1278 | 1279 | 1280 | 1281 | 1282 | 1283 | 1284 | 1285 | 1286 | 1287 | 1288 | 1289 | 1280 | 1281 | 1282 | 1283 | 1284 | 1285 | 1286 | 1287 | 1288 | 1289 | 1290 | 1291 | 1292 | 1293 | 1294 | 1295 | 1296 | 1297 | 1298 | 1299 | 1290 | 1291 | 1292 | 1293 | 1294 | 1295 | 1296 | 1297 | 1298 | 1299 | 1300 | 1301 | 1302 | 1303 | 1304 | 1305 | 1306 | 1307 | 1308 | 1309 | 1300 | 1301 | 1302 | 1303 | 1304 | 1305 | 1306 | 1307 | 1308 | 1309 | 1310 | 1311 | 1312 | 1313 | 1314 | 1315 | 1316 | 1317 | 1318 | 1319 | 1310 | 1311 | 1312 | 1313 | 1314 | 1315 | 1316 | 1317 | 1318 | 1319 | 1320 | 1321 | 1322 | 1323 | 1324 | 1325 | 1326 | 1327 | 1328 | 1329 | 1320 | 1321 | 1322 | 1323 | 1324 | 1325 | 1326 | 1327 | 1328 | 1329 | 1330 | 1331 | 1332 | 1333 | 1334 | 1335 | 1336 | 1337 | 1338 | 1339 | 1330 | 1331 | 1332 | 1333 | 1334 | 1335 | 1336 | 1337 | 1338 | 1339 | 1340 | 1341 | 1342 | 1343 | 1344 | 1345 | 1346 | 1347 | 1348 | 1349 | 1340 | 1341 | 1342 | 1343 | 1344 | 1345 | 1346 | 1347 | 1348 | 1349 | 1350 | 1351 | 1352 | 1353 | 1354 | 1355 | 1356 | 1357 | 1358 | 1359 | 1350 | 1351 | 1352 | 1353 | 1354 | 1355 | 1356 | 1357 | 1358 | 1359 | 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 | 1368 | 1369 | 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 | 1368 | 1369 | 1370 | 1371 | 1372 | 1373 | 1374 | 1375 | 1376 | 1377 | 1378 | 1379 | 1370 | 1371 | 1372 | 1373 | 1374 | 1375 | 1376 | 1377 | 1378 | 1379 | 1380 | 1381 | 1382 | 1383 | 1384 | 1385 | 1386 | 1387 | 1388 | 1389 | 1380 | 1381 | 1382 | 1383 | 1384 | 1385 | 1386 | 1387 | 1388 | 1389 | 1390 | 1391 | 1392 | 1393 | 1394 | 1395 | 1396 | 1397 | 1398 | 1399 | 1390 | 1391 | 1392 | 1393 | 1394 | 1395 | 1396 | 1397 | 1398 | 1399 | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 | 1406 | 1407 | 1408 | 1409 | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 | 1406 | 1407 | 1408 | 1409 | 1410 | 1411 | 1412 | 1413 | 1414 | 1415 | 1416 | 1417 | 1418 | 1419 | 1410 | 1411 | 1412 | 1413 | 1414 | 1415 | 1416 | 1417 | 1418 | 1419 | 1420 | 1421 | 1422 | 1423 | 1424 | 1425 | 1426 | 1427 | 1428 | 1429 | 1420 | 1421 | 1422 | 1423 | 1424 | 1425 | 1426 | 1427 | 1428 | 1429 | 1430 | 1431 | 1432 | 1433 | 1434 | 1435 | 1436 | 1437 | 1438 | 1439 | 1430 | 1431 | 1432 | 1433 | 1434 |

Klassen-Definitionen – Generelle Form

```
class Bicycle {  
    public int gear;  
    public int cadence;  
    public int speed;  
  
    public void changeGear(int newValue)  
    {  
        gear = newValue;  
    }  
  
    public void speedUp(int startSpeed, int startGear)  
    {  
        speed = startSpeed + (startGear * 5);  
    }  
  
    public void applyBrakes(int startSpeed)  
    {  
        speed = startSpeed - (startSpeed * 0.2);  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}  
  
public class MountainBike extends Bicycle {  
    public int seatHeight;  
  
    public MountainBike(int startHeight, int startCadence,  
                        int startSpeed, int startGear)  
    {  
        super(startCadence, startSpeed, startGear);  
        seatHeight = startHeight;  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}
```

Klassen-Definitionen – Generelle Form

```
class Bicycle {  
    public int gear;  
    public int cadence;  
    public int speed;  
  
    public void changeGear(int newValue)  
    {  
        gear = newValue;  
    }  
  
    public void speedUp(int startSpeed, int startGear)  
    {  
        speed = startSpeed + (startGear * 5);  
    }  
  
    public void applyBrakes(int startSpeed)  
    {  
        speed = startSpeed - (startSpeed * 0.2);  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}  
  
public class MountainBike extends Bicycle {  
    public int seatHeight;  
  
    public MountainBike(int startHeight, int startCadence,  
                        int startSpeed, int startGear)  
    {  
        super(startCadence, startSpeed, startGear);  
        seatHeight = startHeight;  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}
```

Klassen-Definitionen – Generelle Form

```
class Bicycle {  
    public int gear;  
    public int cadence;  
    public int speed;  
  
    public void changeGear(int newValue)  
    {  
        gear = newValue;  
    }  
  
    public void speedUp(int startSpeed, int startGear)  
    {  
        speed = startSpeed + (startGear * 5);  
    }  
  
    public void applyBrakes(int startSpeed)  
    {  
        speed = startSpeed - (startSpeed * 0.2);  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}  
  
public class MountainBike extends Bicycle {  
    public int seatHeight;  
  
    public MountainBike(int startHeight, int startCadence,  
                        int startSpeed, int startGear)  
    {  
        super(startCadence, startSpeed, startGear);  
        seatHeight = startHeight;  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}
```

Attribut-Deklarationen – Generelle Form

```
class Bicycle {  
    public int cadence = 0;  
    public int speed = 0;  
    public int gear = 1;  
}  
  
public class MountainBike extends Bicycle {  
    public int seatHeight;  
  
    public MountainBike(int startHeight, int startCadence,  
                        int startSpeed, int startGear)  
    {  
        super(startCadence, startSpeed, startGear);  
        seatHeight = startHeight;  
    }  
  
    public void setHeight(int newValue)  
    {  
        seatHeight = newValue;  
    }  
}
```

Methoden-Definitionen – Generelle Form

```
class Bicycle {
    public i
    public i
    public i
    public B
    gear
    cad
    spe
}
public v
cad
}
public v
gear = newValue;
}

public void speed
speed = speed;
}

public void apply
speed = speed;
}

    Methoden-Deklarationen – Generelle Form:
        modifier typeOfReturnValue methodName ( parameters )
            throwsClauses {
                statements
            }
        • wobei (Access) modifier : Bestimmte Kombinationen von { public,
            protected, private, static, final }
        • wobei typeOfReturnValue : Jeder primitive or Referenz-Typ des
            Rückgabewerts oder void
        • parameters: (kommt gleich)
        • throwsClauses: (kommt bei Exceptions)
    }

    public MountainBike(int startHeight, int startCadence,
        int startSpeed, int startGear)
    {
        super(startCadence, startSpeed, startGear);
        seatHeight = startHeight;
    }

    public void setHeight(int newValue) {
        seatHeight = newValue;
    }
}
```

101

Methoden-Definitionen – Generelle Form

```
class Bicycle {
    public i
    public i
    public i
    public B
    gear
    cad
    spe
}
public v
cad
}
public v
gear = newValue;
}

public void speed
speed = speed;
}

public void apply
speed = speed;
}

    Methoden-Deklarationen – Generelle Form:
        modifier typeOfReturnValue methodName ( parameters )
            throwsClauses {
                statements
            }
        • wobei (Access) modifier : Bestimmte Kombinationen von { public,
            protected, private, static, final }
        • wobei typeOfReturnValue : Jeder primitive or Referenz-Typ des
            Rückgabewerts oder void
        • parameters: (kommt gleich)
        • throwsClauses: (kommt bei Exceptions)
    }

    public MountainBike(int startHeight, int startCadence,
        int startSpeed, int startGear)
    {
        super(startCadence, startSpeed, startGear);
        seatHeight = startHeight;
    }

    public void setHeight(int newValue) {
        seatHeight = newValue;
    }
}
```

101

Methoden-Definitionen – Generelle Form

```
class Bicycle {
    public i
    public i
    public i
    public B
    gear
    cad
    spe
}
public v
cad
}
public v
gear = newValue;
}

public void speed
speed = speed;
}

public void apply
speed = speed;
}

    Methoden-Deklarationen – Generelle Form:
        modifier typeOfReturnValue methodName ( parameters )
            throwsClauses {
                statements
            }
        • wobei (Access) modifier : Bestimmte Kombinationen von { public,
            protected, private, static, final }
        • wobei typeOfReturnValue : Jeder primitive or Referenz-Typ des
            Rückgabewerts oder void
        • parameters: (kommt gleich)
        • throwsClauses: (kommt bei Exceptions)
    }

    public MountainBike(int startHeight, int startCadence,
        int startSpeed, int startGear)
    {
        super(startCadence, startSpeed, startGear);
        seatHeight = startHeight;
    }

    public void setHeight(int newValue) {
        seatHeight = newValue;
    }
}
```

101

Konstruktoren (Constructors)

- können vollständige und konsistente **Initialisierung** der **Objekte** sicherstellen
- (auch generell für Methoden): Mehrere **Varianten** anbieten --> verschiedene Grade an Details für verschiedene Nutzer der Klasse (<--> Abstraktion (API), Information Hiding)
- **Unterklassen-Konstruktoren**: Zugriff auf Oberklassen-Konstruktor mit **super** und Erweiterung nach Bedarf

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

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

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

102

```
class Tandem extends Bicycle {
    int numberofDrivers;

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

103

Konstruktoren (Constructors)

- können vollständige und konsistente **Initialisierung** der **Objekte** sicherstellen
- (auch generell für Methoden): Mehrere **Varianten** anbieten --> verschiedene Grade an Details für verschiedene Nutzer der Klasse (<--> Abstraktion (API), Information Hiding)
- **Unterklassen-Konstruktoren**: Zugriff auf Oberklassen-Konstruktor mit **super** und Erweiterung nach Bedarf

```
class Bicycle {  
    int cadence;  
    int speed;  
    int gear;  
  
    Bicycle(int c, int s, int g) {  
        cadence = c;  
        speed = s;  
        gear = g;  
    }  
  
    Bicycle(int g) {  
        cadence = 0;  
        speed = 0;  
        gear = g;  
    }  
}
```



103

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

Konstruktoren (Constructors)

- können vollständige und konsistente **Initialisierung** der **Objekte** sicherstellen
- (auch generell für Methoden): Mehrere **Varianten** anbieten --> verschiedene Grade an Details für verschiedene Nutzer der Klasse (<--> Abstraktion (API), Information Hiding)
- **Unterklassen-Konstruktoren**: Zugriff auf Oberklassen-Konstruktor mit **super** und Erweiterung nach Bedarf

```
class Bicycle {  
    int cadence;  
    int speed;  
    int gear;  
  
    Bicycle(int c, int s, int g) {  
        cadence = c;  
        speed = s;  
        gear = g;  
    }  
  
    Bicycle(int g) {  
        cadence = 0;  
        speed = 0;  
        gear = g;  
    }  
}
```



↳

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

103

Konstruktoren - Beispiel

```
class Person {  
    // no constructor provided  
    String firstName;  
    String lastName;  
    long taxIdent;  
  
    // Requirement: taxIdent must be unique!  
}
```



Konstruktoren - Beispiel

```
class Person {  
    // no constructor provided  
    String firstName;  
    String lastName;  
    long taxIdent;  
}  
    // Requirement: taxIdent must be unique!
```

```
// Manual initialization, easy to make a mistake (e.g. what about 'taxIdent'?)  
Person p1 = new Person();  
p1.firstName = "Max";  
p1.lastName = "Mustermann";  
p1.taxIdent = 12345;  
  
Person p2 = new Person();  
p2.firstName = "Fabienne";  
p2.lastName = "Fabelhaft";  
p2.taxIdent = 12345; // oops!
```

104

```
// Manual initialization, easy to make a mistake (e.g. what about 'taxIdent'?)  
Person p1 = new Person();  
p1.firstName = "Max";  
p1.lastName = "Mustermann";  
p1.taxIdent = 12345;  
  
Person p2 = new Person();  
p2.firstName = "Fabienne";  
p2.lastName = "Fabelhaft";  
p2.taxIdent = 12345; // oops!
```

104

Konstruktoren - Beispiel

```
class Person {  
    String firstName;  
    String lastName;  
    long taxIdent;                                // Requirement: taxIdent must be unique!  
  
    Person(String fName, String lName, long tIdent) { // now: constructor provided  
        firstName = fName;  
        lastName = lName;  
  
        // Use the given tax identifier `tIdent` only if we can make sure it is unique:  
        if (isUniqueTaxIdentifier(tIdent)) {  
            taxIdent = tIdent;  
        } else {  
            System.err.println("Not unique!");  
        }  
    }  
}
```

Konstruktoren - Beispiel

```
class Person {  
    String firstName;  
    String lastName;  
    long taxIdent;                                // Requirement: taxIdent must be unique!  
  
    Person(String fName, String lName, long tIdent) { // now: constructor provided  
        firstName = fName;  
        lastName = lName;  
  
        // Use the given tax identifier `tIdent` only if we can make sure it is unique:  
        if (isUniqueTaxIdentifier(tIdent)) {  
            taxIdent = tIdent;  
        } else {  
            System.err.println("Not unique!");  
        }  
    }  
}
```

// Complete and consistent.
Person p1 = new Person("Max", "Mustermann", 12345);
◀ ▶ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

105

// Complete and consistent.
Person p1 = new Person("Max", "Mustermann", 12345);
◀ ▶ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

105

Konstruktoren - Beispiel

```
class Person {  
    String firstName;  
    String lastName;  
    long taxIdent;                                // Requirement: taxIdent must be unique!  
  
    Person(String fName, String lName, long tIdent) { // now: constructor provided  
        firstName = fName;                            ↵  
        lastName = lName;  
  
        // Use the given tax identifier `tIdent` only if we can make sure it is unique:  
        if (isUniqueTaxIdentifier(tIdent)) {  
            taxIdent = tIdent;  
        } else {  
            System.err.println("Not unique!");  
        }  
    }  
}
```

// Complete and consistent.
Person p1 = new Person("Max", "Mustermann", 12345);
◀ ▶ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

105

Methodenaufruf und Parameter-Übergabe

Übergabe einer Liste von Parametern an Methoden / Konstruktoren

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

108