

## Script generated by TTT

Title: groh: profile1 (10.07.2015)

Date: Fri Jul 10 09:15:10 CEST 2015

Duration: 90:17 min

Pages: 62

The screenshot shows the Eclipse IDE interface with the title bar "InterfaceDemoNew/src/original/InterfaceDemoMain.java - Eclipse". The left side features the "Package Explorer" view displaying various Java packages and files. The main central area contains the Java code for `InterfaceDemoMain`. The code includes a `bubbleSort` method that sorts an array of integers in descending order using a bubble sort algorithm. It also includes a `printArray` method for printing the array.

```
package original;
public class InterfaceDemoMain {
    public static void main(String[] args) {
        int[] foo = {1, 9, 3, 5, 7, 2, -3};
        bubbleSort(foo);
        printArray(foo);
    }

    public static void bubbleSort(int[] data){
        for(int i=data.length-1; i>0; i--){
            for(int j=0; j<i; j++){
                if(data[j] > data[j+1]){
                    //swap elements
                    int backup = data[j];
                    data[j] = data[j+1];
                    data[j+1] = backup;
                }
            }
        }
    }

    public static void printArray(int[] data){
    }
}
```

This screenshot is similar to the one above, showing the same Eclipse IDE setup. However, the cursor is positioned on the closing brace of the `bubbleSort` method's inner loop. The code remains the same as in the previous screenshot.

```
package original;
public class InterfaceDemoMain {
    public static void main(String[] args) {
        int[] foo = {1, 9, 3, 5, 7, 2, -3};
        bubbleSort(foo);
        printArray(foo);
    }

    public static void bubbleSort(int[] data){
        for(int i=data.length-1; i>0; i--){
            for(int j=0; j<i; j++){
                if(data[j] > data[j+1]){
                    //swap elements
                    int backup = data[j];
                    data[j] = data[j+1];
                    data[j+1] = backup;
                }
            }
        }
    }

    public static void printArray(int[] data){
    }
}
```

erfaceDemoNew/src/original/InterfaceDemoMain.java - Eclipse

```
8     printArray(foo);
9
10    }
11
12    public static void bubbleSort(int[] data){
13        for(int i=data.length-1; i>0; i--){
14            for(int j=0; j<i; j++){
15                if(data[j] > data[j+1]){
16                    //swap elements
17                    int backup = data[j];
18                    data[j] = data[j+1];
19                    data[j+1] = backup;
20                }
21            }
22        }
23    }
24
25    public static void printArray(int[] data){
26        for(int i=0; i<data.length; i++){
27            System.out.println(data[i]);
28        }
29    }
30
31
32 }
```

Console

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

erfaceDemoNew/src/original/InterfaceDemoMain.java - Eclipse

```
1 package original;
2
3 public class InterfaceDemoMain {
4
5     public static void main(String[] args) {
6         int[] foo = {1, 9, 3, 5, 7, 2, -3};
7         bubbleSort(foo);
8         printArray(foo);
9     }
10
11     public static void bubbleSort(int[] data){
12         for(int i=data.length-1; i>0; i--){
13             for(int j=0; j<i; j++){
14                 if(data[j] > data[j+1]){
15                     //swap elements
16                     int backup = data[j];
17                     data[j] = data[j+1];
18                     data[j+1] = backup;
19                 }
20             }
21         }
22     }
23
24     public static void printArray(int[] data){
25 }
```

Console

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

erfaceDemoNew/src/original/InterfaceDemoMain.java - Eclipse

```
1 package original;
2
3 public class InterfaceDemoMain {
4
5     public static void main(String[] args) {
6         int[] foo = {1, 9, 3, 5, 7, 2, -3};
7         foo[0] = 1;
8         foo[1] = 9;
9
10         bubbleSort(foo);
11         printArray(foo);
12     }
13
14     public static void bubbleSort(int[] data){
15         for(int i=data.length-1; i>0; i--){
16             for(int j=0; j<i; j++){
17                 if(data[j] > data[j+1]){
18                     //swap elements
19                     int backup = data[j];
20                     data[j] = data[j+1];
21                     data[j+1] = backup;
22                 }
23             }
24         }
25     }
26
27 }
```

Console

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

erfaceDemoNew/src/original/InterfaceDemoMain.java - Eclipse

```
1 package original;
2
3 public class InterfaceDemoMain {
4
5     public static void main(String[] args) {
6         int[] foo = {1, 9, 3, 5, 7, 2, -3};
7         bubbleSort(foo);
8         printArray(foo);
9     }
10
11     public static void bubbleSort(int[] data){
12         for(int i=data.length-1; i>0; i--){
13             for(int j=0; j<i; j++){
14                 if(data[j] > data[j+1]){
15                     //swap elements
16                     int backup = data[j];
17                     data[j] = data[j+1];
18                     data[j+1] = backup;
19                 }
20             }
21         }
22     }
23
24     public static void printArray(int[] data){
25 }
```

Console

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

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

- Title Bar:** erfaceDemoNew/src/original/InterfaceDemoMain.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:** Quick Access, Java, Debug.
- Package Explorer:** Shows the project structure with packages like BankAccount, BeeAndFlowers, Bicycledemo, ControlFlowDemo, DritteLebung, Eratosthenes, Exceptions, Fakultet, FloodFill, Histogram, ImageDemo, InterfaceDemo, InterfaceDemoNew, and several sub-packages under InterfaceDemoNew (src, original, reusable).
- Editor:** Displays the code for `InterfaceDemoMain.java`. The code defines a main method that prints an array of integers.
- Console:** Shows the output of the application's execution. The output is:

```
<terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)
1
9
3
5
7
2
-3
-----
-3
1
2
3
5
7
9
```
- Bottom Status Bar:** Writable, Smart Insert, 8 : 48.
- Taskbar:** Shows icons for Start, Task View, Favorites, Internet Explorer, and Eclipse.
- System Tray:** Shows battery level (99%), signal strength, and system status.
- Bottom Right:** Date and time (10.07.2015) and language (DE).

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

- Top 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. The 'src' package contains 'original' and 'reusable' sub-packages, each with its own Java files: InterfaceDemoMain, ICanBeCompared.java, InterfaceDemoMain.java, Student.java, and Vehicle.java.
- Central Editor:** Displays the content of the 'InterfaceDemoMain.java' file. The code implements a static main method that prints an array of integers (1, 9, 3, 5, 7, 2, -3) and sorts it using bubbleSort().
- Console:** Two tabs labeled 'Console' show the output of the application. The first tab shows the sorted array: 1, 9, 3, 5, 7, 2, -3. The second tab shows the command-line interface: <terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41).
- Status Bar:** Shows the current status as 'Writable' and 'Smart Insert', with a timestamp of 09:35 10.07.2015.

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

- Title Bar:** InterfaceDemoNew/src/reusable/ICanBeCompared.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 the project structure with packages like BankAccount, BeesAndFlowers, BicycleDemo, ControlFlowDemo, DritteLeibung, Eratosthenes, Exceptions, Fakultaet, FloodFill, Histogram, ImageDemo, InterfaceDemo, and InterfaceDemoNew.
- Editor:** The main editor window displays the code for the `ICanBeCompared` interface and its implementation in `InterfaceDemoMain`. The code is as follows:

```
1 package reusable;
2
3 public interface ICanBeCompared {
4
5     public int compareTo(ICanBeCompared other);
6     public void printitself();
7 }
8
9 }
```

- Console:** Shows the output of a terminated Java application named `InterfaceDemoMain` with the command `C:\Program Files\Java\jre7\bin\javaw.exe` run on 10.07.2015 at 09:34:41. The output is:  
<terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)  
1  
9  
3  
5
- Bottom Status Bar:** Writable, Smart Insert, 5:25, DE, 99%, signal icons.

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

- Project Explorer View (left):** Shows the project structure with several Java packages and source files.
- Editor View (center):** Displays the code for the `ICanBeCompared.java` interface and its implementation classes (`Student.java` and `Vehicle.java`). The code includes methods `compareTo` and `printItself`.
- Console View (bottom):** Shows the output of a Java application named `InterfaceDemoMain`. The console output is:

```
<terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)
1
9
3
5
```
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and navigation.
- Quick Access Bar:** Located at the top right, showing links for Java and Debug.

InterfaceDemoNew/src/reusable/ICanBeCompared.java - Eclipse

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer InterfaceDemoMain ICanBeCompared.java InterfaceDemoMain Student.java Vehicle.java
src
  original
    ICanBeCompared.java
  reusable
    ICanBeCompared.java
    InterfaceDemoMain.java
    Student.java
    Vehicle.java
JRE System Library [JavaSE-1.7]
javaJebung1
javaJebung1a
javaJebung1b
KlausurJul14
OverloadAndOverride
Polymorphism
Quicksort
SimulatedSelection
1 package reusable;
2
3 public interface ICanBeCompared {
4     public int compareTo(ICanBeCompared other);
5     public void printItself();
6 }
7
8
9
```

Console <terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)

```
1
9
3
5
```

InterfaceDemoNew/src/reusable/Student.java - Eclipse

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer InterfaceDemoMain ICanBeCompared.java InterfaceDemoMain Student.java Vehicle.java
src
  original
    ICanBeCompared.java
    InterfaceDemoMain.java
    Student.java
    Vehicle.java
  reusable
    Student.java
JRE System Library [JavaSE-1.7]
javaJebung1
javaJebung1a
javaJebung1b
KlausurJul14
OverloadAndOverride
Polymorphism
Quicksort
SimulatedSelection
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
public Student(int someMatriculationNumber, int someHeight){
    matriculationNumber = someMatriculationNumber;
    bodyHeight = someHeight;
}
public void printItself(){
    System.out.println(matriculationNumber+bodyHeight);
}
public int compareTo(ICanBeCompared somethingThatCanBeCompared){
    if(somethingThatCanBeCompared instanceof Student){
        Student student = (Student) somethingThatCanBeCompared;
        if((this.matriculationNumber + this.bodyHeight) < (student.matriculationNumber + student.bodyHeight))
            return -1;
    } else if (this.matriculationNumber == student.matriculationNumber){
        return 0;
    } else {
        return 1;
    }
}
return 0; // to be fixed
```

Console <terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)

```
1
9
3
5
```

InterfaceDemoNew/src/reusable/ICanBeCompared.java - Eclipse

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer InterfaceDemoMain ICanBeCompared.java InterfaceDemoMain Student.java Vehicle.java
src
  original
    ICanBeCompared.java
    InterfaceDemoMain.java
    Student.java
    Vehicle.java
  reusable
    ICanBeCompared.java
    InterfaceDemoMain.java
    Student.java
    Vehicle.java
JRE System Library [JavaSE-1.7]
javaJebung1
javaJebung1a
javaJebung1b
KlausurJul14
OverloadAndOverride
Polymorphism
Quicksort
SimulatedSelection
1 package reusable;
2
3 public interface ICanBeCompared {
4     public int compareTo(ICanBeCompared other);
5     public void printItself();
6 }
7
8
9
```

Console <terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)

```
1
9
3
5
```

InterfaceDemoNew/src/reusable/ICanBeCompared.java - Eclipse

```
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access Java Debug
Package Explorer InterfaceDemoMain ICanBeCompared.java InterfaceDemoMain Student.java Vehicle.java
src
  original
    ICanBeCompared.java
    InterfaceDemoMain.java
    Student.java
    Vehicle.java
  reusable
    Student.java
JRE System Library [JavaSE-1.7]
javaJebung1
javaJebung1a
javaJebung1b
KlausurJul14
OverloadAndOverride
Polymorphism
Quicksort
SimulatedSelection
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
public Student(int someMatriculationNumber, int someHeight){
    matriculationNumber = someMatriculationNumber;
    bodyHeight = someHeight;
}
public void printItself(){
    System.out.println(matriculationNumber+bodyHeight);
}
public int compareTo(ICanBeCompared somethingThatCanBeCompared){
    if(somethingThatCanBeCompared instanceof Student){
        Student student = (Student) somethingThatCanBeCompared;
        if((this.matriculationNumber + this.bodyHeight) < (student.matriculationNumber + student.bodyHeight))
            return -1;
    } else if (this.matriculationNumber == student.matriculationNumber){
        return 0;
    } else {
        return 1;
    }
}
return 0; // to be fixed
```

Console <terminated> InterfaceDemoMain [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (10.07.2015 09:34:41)

```
1
9
3
5
```

Eclipse IDE screenshot showing the Java code for Student.java:

```
public Student(int someMatriculationNumber, int someHeight){  
    matriculationNumber = someMatriculationNumber;  
    bodyHeight = someHeight;  
}  
  
public void printItself(){  
    System.out.println(matriculationNumber+bodyHeight);  
}  
  
public int compareTo(ICanBeCompared somethingThatCanBeCompared){  
    if(somethingThatCanBeCompared instanceof Student){  
        Student student = (Student) somethingThatCanBeCompared;  
        if((this.matriculationNumber + this.bodyHeight) < (student.matriculationNumber + student.bodyHeight))  
            return -1;  
        else if (this.matriculationNumber == student.matriculationNumber){  
            return 0;  
        } else {  
            return 1;  
        }  
    }  
    return 0; // to be fixed  
}
```

Eclipse IDE screenshot showing the Java code for Student.java with a cursor at the start of the line "if(somethingThatCanBeCompared instanceof Student){".

```
public Student(int someMatriculationNumber, int someHeight){  
    matriculationNumber = someMatriculationNumber;  
    bodyHeight = someHeight;  
}  
  
public void printItself(){  
    System.out.println(matriculationNumber+bodyHeight);  
}  
  
public int compareTo(ICanBeCompared somethingThatCanBeCompared){  
    if(somethingThatCanBeCompared instanceof Student){  
        Student student = (Student) somethingThatCanBeCompared;  
        if((this.matriculationNumber + this.bodyHeight) < (student.matriculationNumber + student.bodyHeight))  
            return -1;  
        else if (this.matriculationNumber == student.matriculationNumber){  
            return 0;  
        } else {  
            return 1;  
        }  
    }  
    return 0; // to be fixed  
}
```

Eclipse IDE screenshot showing the Java code for Student.java with a cursor at the start of the line "if(somethingThatCanBeCompared instanceof Student){".

```
public Student(int someMatriculationNumber, int someHeight){  
    matriculationNumber = someMatriculationNumber;  
    bodyHeight = someHeight;  
}  
  
public void printItself(){  
    System.out.println(matriculationNumber+bodyHeight);  
}  
  
public int compareTo(ICanBeCompared somethingThatCanBeCompared){  
    if(somethingThatCanBeCompared instanceof Student){  
        Student student = (Student) somethingThatCanBeCompared;  
        if((this.matriculationNumber + this.bodyHeight) < (student.matriculationNumber + student.bodyHeight))  
            return -1;  
        else if (this.matriculationNumber == student.matriculationNumber){  
            return 0;  
        } else {  
            return 1;  
        }  
    }  
    return 0; // to be fixed  
}
```

Eclipse IDE screenshot showing the Java code for Student.java with a cursor at the start of the line "if(somethingThatCanBeCompared instanceof Student){".

```
public Student(int someMatriculationNumber, int someHeight){  
    matriculationNumber = someMatriculationNumber;  
    bodyHeight = someHeight;  
}  
  
public void printItself(){  
    System.out.println(matriculationNumber+bodyHeight);  
}  
  
public int compareTo(ICanBeCompared somethingThatCanBeCompared){  
    if(somethingThatCanBeCompared instanceof Student){  
        Student student = (Student) somethingThatCanBeCompared;  
        if((this.matriculationNumber + this.bodyHeight) < (student.matriculationNumber + student.bodyHeight))  
            return -1;  
        else if (this.matriculationNumber == student.matriculationNumber){  
            return 0;  
        } else {  
            return 1;  
        }  
    }  
    return 0; // to be fixed  
}
```

The screenshot shows the Eclipse IDE interface with two windows side-by-side. Both windows display the same Java code for a sorting algorithm:

```

1 package reusable;
2
3 public class InterfaceDemoMain {
4
5     public static void main(String[] args) {
6         Vehicle[] vehicles = new Vehicle[50];
7         for(int i = 0; i<vehicles.length; i++){
8             vehicles[i] = new Vehicle((int)(50 * Math.random()));
9         }
10        printArray(vehicles);
11        bubbleSort(vehicles);
12        System.out.println("-----");
13        printArray(vehicles);
14
15        Student[] students = {
16            new Student(1234, 184),
17            new Student(1235, 160),
18            new Student(1236, 190)
19        };
20        printArray(students);
21        bubbleSort(students);
22        System.out.println("-----");
23    }
24
25    void printArray(Vehicle[] vehicles) {
26        for(Vehicle v : vehicles) {
27            System.out.print(v);
28        }
29    }
30
31    void bubbleSort(Vehicle[] vehicles) {
32        for(int i = 0; i<vehicles.length-1; i++) {
33            for(int j = 0; j<vehicles.length-i-1; j++) {
34                if(vehicles[j].getAge() > vehicles[j+1].getAge()) {
35                    Vehicle temp = vehicles[j];
36                    vehicles[j] = vehicles[j+1];
37                    vehicles[j+1] = temp;
38                }
39            }
40        }
41    }
42
43    class Vehicle {
44        int age;
45        int height;
46
47        Vehicle(int age, int height) {
48            this.age = age;
49            this.height = height;
50        }
51
52        void print() {
53            System.out.print("Age: " + age + " Height: " + height);
54        }
55    }
56
57    class Student {
58        int id;
59        int grade;
60
61        Student(int id, int grade) {
62            this.id = id;
63            this.grade = grade;
64        }
65
66        void print() {
67            System.out.print("ID: " + id + " Grade: " + grade);
68        }
69    }
70
71    void printArray(Student[] students) {
72        for(Student s : students) {
73            s.print();
74        }
75    }
76
77    void bubbleSort(Student[] students) {
78        for(int i = 0; i<students.length-1; i++) {
79            for(int j = 0; j<students.length-i-1; j++) {
80                if(students[j].getGrade() > students[j+1].getGrade()) {
81                    Student temp = students[j];
82                    students[j] = students[j+1];
83                    students[j+1] = temp;
84                }
85            }
86        }
87    }
88
89    void print() {
90        System.out.println("-----");
91    }
92
93 }

```

The code implements a bubble sort algorithm for both `Vehicle` and `Student` arrays. It includes methods for printing arrays and individual objects.

This screenshot shows the same Java code for the sorting algorithm, but with a different line of code highlighted in blue:

```

1 package reusable;
2
3 public class InterfaceDemoMain {
4
5     public static void main(String[] args) {
6         Vehicle[] vehicles = new Vehicle[50];
7         for(int i = 0; i<vehicles.length; i++){
8             vehicles[i] = new Vehicle((int)(50 * Math.random()));
9         }
10        printArray(vehicles);
11        bubbleSort(vehicles);
12        System.out.println("-----");
13        printArray(vehicles);
14
15        Student[] students = {
16            new Student(1234, 184),
17            new Student(1235, 160),
18            new Student(1236, 190)
19        };
20        printArray(students);
21        bubbleSort(students);
22        System.out.println("-----");
23    }
24
25    void printArray(Vehicle[] vehicles) {
26        for(Vehicle v : vehicles) {
27            System.out.print(v);
28        }
29    }
30
31    void bubbleSort(Vehicle[] vehicles) {
32        for(int i = 0; i<vehicles.length-1; i++) {
33            for(int j = 0; j<vehicles.length-i-1; j++) {
34                if(vehicles[j].getAge() > vehicles[j+1].getAge()) {
35                    Vehicle temp = vehicles[j];
36                    vehicles[j] = vehicles[j+1];
37                    vehicles[j+1] = temp;
38                }
39            }
40        }
41    }
42
43    class Vehicle {
44        int age;
45        int height;
46
47        Vehicle(int age, int height) {
48            this.age = age;
49            this.height = height;
50        }
51
52        void print() {
53            System.out.print("Age: " + age + " Height: " + height);
54        }
55    }
56
57    class Student {
58        int id;
59        int grade;
60
61        Student(int id, int grade) {
62            this.id = id;
63            this.grade = grade;
64        }
65
66        void print() {
67            System.out.print("ID: " + id + " Grade: " + grade);
68        }
69    }
70
71    void printArray(Student[] students) {
72        for(Student s : students) {
73            s.print();
74        }
75    }
76
77    void bubbleSort(Student[] students) {
78        for(int i = 0; i<students.length-1; i++) {
79            for(int j = 0; j<students.length-i-1; j++) {
80                if(students[j].getGrade() > students[j+1].getGrade()) {
81                    Student temp = students[j];
82                    students[j] = students[j+1];
83                    students[j+1] = temp;
84                }
85            }
86        }
87    }
88
89    void print() {
90        System.out.println("-----");
91    }
92
93 }

```

The line `for(int i = 0; i<vehicles.length-1; i++)` is highlighted, likely indicating the current step in the execution or analysis of the code.

The screenshot shows a Microsoft PowerPoint slide titled "Einführung in die Informatik für Hörer anderer Fachrichtungen (WZW)" with the subtitle "IN8003". The slide content includes the name "PD Dr. Georg Groh" and logos for "Angewandte Informatik / Kooperative Systeme AICOS" and "Fakultät für Informatik". The slide is part of a presentation with multiple slides visible in the navigation pane on the left.

## Rekursion

- **Rekursion:**
  - Definiere eine Funktion **durch sich selbst** bzw.
  - formuliere **Lösung** eines Problems durch **Rückgriff** auf die **Lösung selbst**:  
Gebe Lösung für **Basisfälle** an und gib **Regeln** an, wie sich allg. Problem in Richtung der Basisfälle **aufteilen** lässt.
  - d.h. bspw. durch eine **Methode**, die sich **selbst wieder aufruft**

Bsp:

• **Fakultät**

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n-1)! \cdot n & \text{if } n > 0. \end{cases}$$

• **Fibonacci**

$$f(n) = \begin{cases} 1 & \text{if } n = 1, \\ 1 & \text{if } n = 2, \\ f(n-1) + f(n-2) & \text{if } n > 2. \end{cases}$$

## Rekursion

- **Rekursion:**

- Definiere eine Funktion **durch sich selbst** bzw.
- formuliere **Lösung** eines Problems durch **Rückgriff** auf die **Lösung selbst**:  
Gebe Lösung für **Basisfälle** an und gib **Regeln** an, wie sich allg. Problem in Richtung der Basisfälle **aufteilen** lässt.
- d.h. bspw. durch eine **Methode**, die sich **selbst wieder aufruft**

Bsp:

• **Fakultät**

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n - 1)! \cdot n & \text{if } n > 0. \end{cases}$$

• **Fibonacci**

$$f(n) = \begin{cases} 1 & \text{if } n = 1, \\ 1 & \text{if } n = 2, \\ f(n - 1) + f(n - 2) & \text{if } n > 2. \end{cases}$$



147

Fakultät

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n - 1)! \cdot n & \text{if } n > 0. \end{cases}$$



## Rekursion

- **Rekursion:**

- Definiere eine Funktion **durch sich selbst** bzw.
- formuliere **Lösung** eines Problems durch **Rückgriff** auf die **Lösung selbst**:  
Gebe Lösung für **Basisfälle** an und gib **Regeln** an, wie sich allg. Problem in Richtung der Basisfälle **aufteilen** lässt.
- d.h. bspw. durch eine **Methode**, die sich **selbst wieder aufruft**

Fakultät

$$n! = \begin{cases} 1 & \text{if } n = 0, \\ (n - 1)! \cdot n & \text{if } n > 0. \end{cases}$$



```
long factorial(int n) {
    long temp;
    if (n == 0) {
        return 1;
    } else {
        temp = factorial(n-1);
        return n * temp;
    }
}
```

148

## Rekursion

- **Rekursion:**

- Definiere eine Funktion **durch sich selbst** bzw.
- formuliere **Lösung** eines Problems durch **Rückgriff** auf die **Lösung selbst**:  
Gebe Lösung für **Basisfälle** an und gib **Regeln** an, wie sich allg. Problem in Richtung der Basisfälle **aufteilen** lässt.
- d.h. bspw. durch eine **Methode**, die sich **selbst wieder aufruft**

```
long factorial(int n) {
    long temp;
    if (n == 0) {
        return 1;
    } else {
        temp = factorial(n-1);
        return n * temp;
    }
}
```

148

## Call Stack

Für jeden Methodenaufruf (auch Klassenmethoden oder nicht rekursive Methodenaufrufe) werden lokale Variablen + Parameter + Rücksprungadresse („von wem (von wo aus) wurde die Methode aufgerufen?“) auf **Call-Stack** gespeichert

```
public class someClass {
    int a;
    int b;

    int someMethodOne(int paramOne1, int paramOne2) {
        int localOne = 5;
        int result = someMethodTwo(17) + localOne;
        return result;
    }

    int someMethodTwo(int paramTwo) {
        int localTwo = 8;
        return paramTwo * localTwo;
    }
}

SomeClass someObject = new SomeClass();
int bbb = someObject.someMethodOne(12, 34);
```



...	...	...
2345	someObject	<2346>
2346	someObject.a	
2347	someObject.b	
...		
...		
5467	int result = someMethodTwo(17) + localOne;	
...		
5899	int bbb = someObject.someMethodOne(12, 34);	
...		
...		
6000	(Rücksprung-Adresse) <5467>	
6001	(Aufruf-Objekt) <2346>	
6002	paramTwo 17	
6003	localTwo 8	
...		
6004	(Rücksprung-Adresse) <5899>	
6005	(Aufruf-Objekt) <...>	
6006	paramOne1 12	
6007	paramOne2 34	
6008	localOne 5	149
...		

## Call Stack

Für jeden Methodenaufruf (auch Klassenmethoden oder nicht rekursive Methodenaufrufe) werden lokale Variablen + Parameter + Rücksprungadresse („von wem (von wo aus) wurde die Methode aufgerufen?“) auf **Call-Stack** gespeichert

```
public class someClass {
    int a;
    int b;

    int someMethodOne(int paramOne1, int paramOne2) {
        int localOne = 5;
        int result = someMethodTwo(17) + localOne;
        return result;
    }

    int someMethodTwo(int paramTwo) {
        int localTwo = 8;
        return paramTwo * localTwo;
    }

    ...
    SomeClass someObject = new SomeClass();
    int bbb = someObject.someMethodOne(12, 34);
    ...
}
```

...	...	...
2345	someObject	<2346>
2346	someObject.a	
2347	someObject.b	
...		
...	...	...
5467	int result = someMethodTwo(17) + localOne;	
...		
5899	int bbb = someObject.someMethodOne(12, 34);	
...		
...	...	...
6000	(Rücksprung-Adresse)	<5467>
6001	(Aufruf-Objekt)	<2346>
6002	paramTwo	17
6003	localTwo	8
6004	(Rücksprung-Adresse)	<5899>
6005	(Aufruf-Objekt)	<...>
6006	paramOne1	12
6007	paramOne2	34
6008	localOne	5 149
...	...	...

## Call Stack

Für jeden Methodenaufruf (auch Klassenmethoden oder nicht rekursive Methodenaufrufe) werden lokale Variablen + Parameter + Rücksprungadresse („von wem (von wo aus) wurde die Methode aufgerufen?“) auf **Call-Stack** gespeichert

```
public class someClass {
    int a;
    int b;

    int someMethodOne(int paramOne1, int paramOne2) {
        int localOne = 5;
        int result = someMethodTwo(17) + localOne;
        return result;
    }

    int someMethodTwo(int paramTwo) {
        int localTwo = 8;
        return paramTwo * localTwo;
    }

    ...
    SomeClass someObject = new SomeClass();
    int bbb = someObject.someMethodOne(12, 34);
    ...
}
```

Speicherbereich für Objekte: „**Heap**“

```
...
SomeClass someObject = new SomeClass();
int bbb = someObject.someMethodOne(12, 34);
...
```

Einfügende des Call Stacks →

„**Call Stack**“

...	...	...
6000	(Rücksprung-Adresse)	<5467>
6001	(Aufruf-Objekt)	<2346>
6002	paramTwo	17
6003	localTwo	8
6004	(Rücksprung-Adresse)	<5899>
6005	(Aufruf-Objekt)	<...>
6006	paramOne1	12
6007	paramOne2	34
6008	localOne	5 150
...	...	...

## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {
    long temp;
    if (n == 0) {
        return 1;
    } else {
        temp = factorial(n-1);
        return n * temp;
    }
}
```

```
...
int ccc = factorial(3);
...
```



n	3
temp	0
ccc	...
...	...
...	154
...	...

## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {
    long temp;
    if (n == 0) {
        return 1;
    } else {
        temp = factorial(n-1);
        return n * temp;
    }
}
```

```
...
int ccc = factorial(3);
...
```



n	2
temp	0
n	3
temp	0
ccc	...
...	...
...	157
...	...

## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {  
    long temp;  
    if (n == 0) {  
        return 1;  
    } else {  
        temp = factorial(n-1);  
        return n * temp;  
    }  
}
```

```
...  
int ccc = factorial(3);  
...
```



n	1
temp	0
n	2
temp	0
n	3
temp	0
ccc	...
...	...
	159
...	...

## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {  
    long temp;  
    if (n == 0) {  
        return 1;  
    } else {  
        temp = factorial(n-1);  
        return n * temp;  
    }  
}
```

```
...  
int ccc = factorial(3);  
...
```



## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {  
    long temp;  
    if (n == 0) {  
        return 1;  
    } else {  
        temp = factorial(n-1);  
        return n * temp;  
    }  
}
```

```
...  
int ccc = factorial(3);  
...
```



n	0
temp	0
n	1
temp	0
n	2
temp	0
n	3
temp	0
ccc	...
...	...
	161
...	...

## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {  
    long temp;  
    if (n == 0) {  
        return 1;  
    } else {  
        temp = factorial(n-1);  
        return n * temp;  
    }  
}
```

```
...  
int ccc = factorial(3);  
...
```



n	0
temp	0
n	1
temp	0
n	2
temp	0
n	3
temp	0
ccc	...
...	...
	162
...	...

n	3
temp	2
ccc	...
...	...
	167
...	...

## Rekursive Methodenaufrufe – Call Stack

```
long factorial(int n) {
    long temp;
    if (n == 0) {
        return 1;
    } else {
        temp = factorial(n-1);
        return n * temp;
    }
}
```

```
...
int ccc = factorial(3);
...
```



n	3
temp	2
ccc	...
...	...
...	168

## Java Klassenbibliothek

Java Klassenbibliothek: <http://java.sun.com/javase/7/docs/api/>

(Äquivalente gibt es auch für andere Sprachen (bspw. .NET mit C#))

The screenshot shows the Java API documentation for Standard Edition 7. The left sidebar has 'Packages' selected under 'All Classes'. The main content area shows a table of packages with their descriptions. Some packages like java.awt and java.io have detailed descriptions below them.

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems, that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.image	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing beans – components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.

## Java Klassenbibliothek

Java Klassenbibliothek: <http://java.sun.com/javase/7/docs/api/>

(Äquivalente gibt es auch für andere Sprachen (bspw. .NET mit C#))

This screenshot is identical to the one above, showing the Java API documentation for Standard Edition 7. It displays the 'Packages' section of the 'All Classes' table, listing various Java packages and their descriptions.

## Packages

- Java Klassen sind in Hierarchie von **packages** organisiert

```
java.lang.String  
java.net.URLConnection  
java.util.Collection  
javax.xml.parsers.SAXParser  
org.w3c.dom.events.DocumentEvent
```

- Klassen die man nutzen möchte, muss man **importieren**.  
Alle Klassen aus `java.lang` werden automatisch importiert

```
import java.netURLConnection;  
import java.util.Collection;  
import some.other.package.*; // * means all classes  
  
class SomeClass{  
    ...  
}
```

## Packages

- Java Klassen sind in Hierarchie von **packages** organisiert

```
java.lang.String  
java.net.URLConnection  
java.util.Collection  
javax.xml.parsers.SAXParser  
org.w3c.dom.events.DocumentEvent
```

- Klassen die man nutzen möchte, muss man **importieren**.

Alle Klassen aus `java.lang` werden automatisch importiert

```
import java.netURLConnection;  
import java.util.Collection;  
import some.other.package.*; // * means all classes  
  
class SomeClass{  
    ...  
}
```

## Packages

- Java Klassen sind in Hierarchie von **packages** organisiert

```
java.lang.String  
java.net.URLConnection  
java.util.Collection  
javax.xml.parsers.SAXParser  
org.w3c.dom.events.DocumentEvent
```

- Klassen die man nutzen möchte, muss man **importieren**.

Alle Klassen aus `java.lang` werden automatisch importiert

```
import java.netURLConnection;  
import java.util.Collection;  
import some.other.package.*; // * means all classes  
  
class SomeClass{  
    ...  
}
```



173

## Packages

- Java Klassen sind in Hierarchie von **packages** organisiert

```
java.lang.String  
java.net.URLConnection  
java.util.Collection  
javax.xml.parsers.SAXParser  
org.w3c.dom.events.DocumentEvent
```

- Klassen die man nutzen möchte, muss man **importieren**.

Alle Klassen aus `java.lang` werden automatisch importiert

```
import java.netURLConnection;  
import java.util.Collection;  
import some.other.package.*; // * means all classes  
  
class SomeClass{  
    ...  
}
```

- Java Klassen sind in Hierarchie von **packages** organisiert

```
java.lang.String  
java.net.URLConnection  
java.util.Collection  
javax.xml.parsers.SAXParser  
org.w3c.dom.events.DocumentEvent
```

- Klassen die man nutzen möchte, muss man **importieren**.

Alle Klassen aus `java.lang` werden automatisch importiert

```
import java.netURLConnection;  
import java.util.Collection;  
import some.other.package.*; // * means all classes  
  
class SomeClass{  
    ...  
}
```



173



173

## Generics

- Manche Klassen kann man mit anderen Klassen parametrisieren
- Typisches Beispiel: Datenstrukturen
- Vorteil: Algorithmen und Datenstrukturen können generisch implementiert und genutzt werden.

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```



174

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```

174

## Generics

- Manche Klassen kann man mit anderen Klassen parametrisieren
- Typisches Beispiel: Datenstrukturen
- Vorteil: Algorithmen und Datenstrukturen können generisch implementiert und genutzt werden.

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```



174

## Generics

- Manche Klassen kann man mit anderen Klassen parametrisieren
- Typisches Beispiel: Datenstrukturen
- Vorteil: Algorithmen und Datenstrukturen können generisch implementiert und genutzt werden.

ohne Generics:

```
Vector nonGenericVector = new Vector();
nonGenericVector.add( 1234 );
nonGenericVector.add( "Hello world" );
Object typeUnknown = nonGenericVector.get(0);
```

mit Generics:

```
Vector<Bicycle> bicycles = new Vector<Bicycle>();
bicycles.add( new Bicycle() );
bicycles.add( 123 ); // Compile time error!
Bicycle typeKnown = bicycles.get(0);
```



175

## Wrapper Classes

- Für jeden primitiven Typ gibt es eine entsprechende Wrapper-Klasse

```
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Float
java.lang.Double
java.lang.Boolean
...
```

- Zur Bequemlichkeit im Umgang mit diesen Wrapper Klassen gibt es Boxing:

statt:

```
Integer someInteger = new Integer(723);
```

kann man schreiben:

```
Integer someInteger = 723;
```



- Für jeden primitiven Typ gibt es eine entsprechende Wrapper-Klasse

```
java.lang.Short
java.lang.Integer
java.lang.Long
java.lang.Float
java.lang.Double
java.lang.Boolean
...
```

- Zur Bequemlichkeit im Umgang mit diesen Wrapper Klassen gibt es **Boxing**:

statt:

```
Integer someInteger = new Integer(723);
```

kann man schreiben:

```
Integer someInteger = 723;
```



**Reading/Loading an Image (The Java™ Tutorials > 2D Graphics > Working with Images)**

The Java™ Tutorials

Working with Images  
Reading/Loading an Image  
Drawing an Image  
Creating and Drawing to an Image  
Writing/Saving an Image

« Previous • Trail • Next » Home Page > 2D Graphics > Working with Images

### Reading/Loading an Image

When you think of digital images, you probably think of sampled image formats such as the JPEG image format used in digital photography, or GIF images commonly used on web pages. All programs that can use these images must first convert them from their external format into an internal format.

Java 2D™ supports loading these external image formats into its `BufferedImage` format using its Image I/O API which is in the `javax.imageio` package. Image I/O has built-in support for GIF, PNG, JPEG, BMP, and WBMP. Image I/O is also extensible so that developers or administrators can "plug-in" support for additional formats. For example, plug-ins for TIFF and JPEG 2000 are separately available.

To load an image from a specific file use the following code:

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) { }
```

Image I/O recognises the contents of the file as a JPEG format image, and decodes it into a `BufferedImage` which can be directly used by Java 2D.

`LoadImageApp.java` shows how to display this image.

If the code is running in an applet, then it's just as easy to obtain the image from the applet codebase:

```
try {
    URL url = new URL(getCodeBase(),
                      "strawberry.jpg");
    img = ImageIO.read(url);
} catch (IOException e) { }
```

The `getCodeBase` method used in this example returns the URL of the directory containing this applet.

The following example shows how to use the `getCodeBase` method to load the `strawberry.jpg` file.

178

- Bsp.-Problem: Ein Bild aus dem Web laden und in einem Fenster darstellen

- Wie löse ich das Problem?

1. Googlen
2. Java Tutorial zu Rate ziehen
3. Entsprechende Klassen, Methoden, Attribute in API-Doc nachschlagen
4. Programm schreiben, testen
5. Wenns nicht klappt : Gehe zu (1)



175

**URL (Java Platform SE 6)**

http://foo.com/hello world/ and http://foo.com/hello%20world

would be considered not equal to each other.

Note, the `URL` class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use `URI`, and to convert between these two classes using `toURI()` and `URI.toURL()`.

The `URLEncoder` and `URLDecoder` classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

Since: JDK1.0  
See Also: Serialized Form

### Constructor Summary

<code>URL(String spec)</code>	Creates a URL object from the String representation.	?
<code>URL(String protocol, String host, int port, String file)</code>	Creates a URL object from the specified protocol,host,port number, and file.	?
<code>URL(String protocol, String host, int port, String file, URLStreamHandler handler)</code>	Creates a URL object from the specified protocol,host,port number,file, and handler.	?
<code>URL(String protocol, String host, String file)</code>	Creates a URL from the specified protocol name,host name, and file name.	?
<code>URL(URL context, String spec)</code>	Creates a URL by parsing the given spec within a specified context.	?
<code>URL(URL context, String spec, URLStreamHandler handler)</code>	Creates a URL by parsing the given spec with the specified handler within a specified context.	?

### Method Summary

<code>boolean equals(Object obj)</code>	CCompares this URL for equality with another object.
<code>String getAuthority()</code>	Gets the authority part of this URL.
<code>Object getContent()</code>	Gets the contents of this URL.

180

URL (Java Platform SE 6)

<http://docs.oracle.com/javase/6/docs/api/java/net/URL.html>

java 6 api url

http://foo.com/hello\_world/ and http://foo.com/hello%20world

would be considered not equal to each other.

Note, the `URI` class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use `URI`, and to convert between these two classes using `toURI()` and `URI.toURL()`.

The `URLEncoder` and `URLEncoder` classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

Since: JDK1.0

See Also:

- [Serialized Form](#)

## Constructor Summary

<code>URL(String spec)</code>	Creates a URL object from the String representation.	?
<code>URL(String protocol, String host, int port, String file)</code>	Creates a URL object from the specified protocol,host,port number, and file.	
<code>URL(String protocol, String host, int port, String file, URLStreamHandler handler)</code>	Creates a URL object from the specified protocol,host,port number,file, and handler.	
<code>URL(String protocol, String host, String file)</code>	Creates a URL from the specified protocol name,host name, and file name.	
<code>URL(URL context, String spec)</code>	Creates a URL by parsing the given spec within a specified context.	?
<code>URL(URL context, String spec, URLStreamHandler handler)</code>	Creates a URL by parsing the given spec with the specified handler within a specified context.	

## Method Summary

<code>boolean equals(Object obj)</code>	Compares this URL for equality with another object.
<code>String getAuthority()</code>	Gets the authority part of this URL.
<code>Object getContent()</code>	Gets the contents of this URI.

180

Reading/Loading an Image (The Java™ Tutorials > 2D Graphics > Working with Images)

<http://docs.oracle.com/javase/tutorial/2d/images/loadimage.html>

Reader

The Java™ Tutorials

Working with Images  
Reading/Loading an Image  
Drawing an Image  
Creating and Drawing to an Image  
Writing/Saving an Image

« Previous • Trail • Next »

Home Page > 2D Graphics > Working with Images

## Reading/Loading an Image

When you think of digital images, you probably think of sampled image formats such as the JPEG image format used in digital photography, or GIF images commonly used on web pages. All programs that can use these images must first convert them from that external format into an internal format.

Java 2D™ supports loading these external image formats into its `BufferedImage` format using its Image I/O API, which is in the `javax.imageio` package. Image I/O has built-in support for GIF, PNG, JPEG, BMP, and WBMP. Image I/O is also extensible so that developers or administrators can "plug-in" support for additional formats. For example, plug-ins for TIFF and JPEG 2000 are separately available.

To load an image from a specific file use the following code:

```
BufferedImage img = null;
try {
    img = ImageIO.read(new File("strawberry.jpg"));
} catch (IOException e) {
```

Image I/O recognises the contents of the file as a JPEG format image, and decodes it into a `BufferedImage` which can be directly used by Java 2D.

`LoadImageApp.java` shows how to display this image.

If the code is running in an applet, then it's just as easy to obtain the image from the applet codebase :

```
try {
    URL url = new URL(getCodeBase(),
                       "strawberry.jpg");
    img = ImageIO.read(url);
} catch (IOException e) {
```

The `getCodeBase` method used in this example returns the URL of the directory containing this applet.

The following example shows how to use the `getCodeBase` method to load the strawberry.jpg file.

182

Reading/Loading an Image (The Java™ Tutorials > 2D Graphics > Working with Images)

<http://docs.oracle.com/javase/tutorial/2d/images/loadimage.html>

java load picture from url and display

`LoadImageApp.java` shows how to display this image.

If the code is running in an applet, then it's just as easy to obtain the image from the applet codebase :

```
try {
    URL url = new URL(getCodeBase(),
                       "strawberry.jpg");
    img = ImageIO.read(url);
} catch (IOException e) {
```

The `getCodeBase` method used in this example returns the URL of the directory containing this applet.

The following example shows how to use the `getCodeBase` method to load the strawberry.jpg file.



**Note:**  
If you don't see the applet running, you need to install [release 6 \(or later\) of the Java SE Development Kit \(JDK\)](#).

`LoadImageApp.java` contains the complete code for this example and this applet requires the `strawberry.jpg` image file.

In addition to reading from files or URLs, Image I/O can read from other sources, such as an `InputStream`. `ImageIO.read()` is the most straightforward convenience API for most applications, but the `javax.imageio.ImageIO` class provides many more static methods for more advanced usages of the Image I/O API. The collection of methods on this class represent just a subset of the rich set of APIs for discovering information about the images and for controlling the image decoding (reading) process.

We will explore some of the other capabilities of Image I/O later in the [Writing/Saving an Image](#) section.

183

What's an easy way to display an Image in Java/Scala? - Stack Overflow

<http://stackoverflow.com/questions/9027257/whats-an-easy-way-to-display-an-image-in-scala>

50.4k 4 20 54

256 1 5 91% accept rate

scala x 9773

asked 5 months ago  
viewed 222 times  
active 5 months ago

Work From Home.

CAREERS 2.0

Software Developer  
Ravensburger Digital GmbH  
Munich, Germany

Mobile Developer for Travel Startup  
Triposo  
Berlin, Germany

Software Developer with focus on Application Development  
OMICRON electronics GmbH  
Nuremberg, Germany

2 Answers

9

Of course, if you have an `URL` for the image (or can form one from a `File` path), it can also be displayed as a tool-tip using HTML.



See [Show full Image when hover over thumbnail](#) for source.

link | improve this answer

answered Jan 27 at 20:06 by Andrew Thompson

Was this post useful to you? Yes No

50.4k 4 20 54

Linked

Show full Image when hover over thumbnail as PopupOverlay

Related

How can I display a bitmap image in 186

**JOptionPane (Java Platform SE 6)**

<http://docs.oracle.com/javase/6/docs/api/javax/swing/JOptionPane.html>

**Overview Package Class Use Tree Deprecated Index Help**

**Frames No Frames All Classes**

**SUMMARY: NESTED | FIELD | CONSTR | METHOD**

**DETAIL: FIELD | CONSTR | METHOD**

**java.swing**

**Class JOptionPane**

**java.lang.Object**  
 └── **java.awt.Component**  
   └── **java.awt.Container**  
 └── **javax.swing.JComponent**  
   └── **javax.swing.JOptionPane**

**All Implemented Interfaces:**  
**ImageObserver, MenuContainer, Serializable, Accessible**

**public class JOptionPane**  
**extends JComponent**  
**implements Accessible**

JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something. For information about using JOptionPane, see [How to Make Dialogs](#), a section in *The Java Tutorial*.

While the JOptionPane class may appear complex because of the large number of methods, almost all uses of this class are one-line calls to one of the static showXxxDialog methods shown below:

Method Name	Description
showConfirmDialog	Asks a confirming question, like yes/no/cancel.
showInputDialog	Prompt for some input.
showMessageDialog	Tell the user about something that has happened.
showOptionDialog	The Grand Unification of the above three.

Each of these methods also comes in a showInternalXXX flavor, which uses an internal frame to hold the dialog box (see [JInternalFrame](#)). Multiple convenience methods have also been defined -- overloaded versions of the basic methods that use different parameter lists.

All dialogs are modal. Each showXxxDialog method blocks the caller until the user's interaction is complete.

The basic appearance of one of these dialog boxes is generally similar to the picture at the right, although the various look-and-feels are ultimately responsible for the final result. In particular, the look-and-feels will adjust the layout to accommodate the option pane's ComponentOrientation property.

187

**JOptionPane (Java Platform SE 6)**

[http://docs.oracle.com/javase/6/docs/api/javax/swing/JOptionPane.html#showMessageDialog\(jav](http://docs.oracle.com/javase/6/docs/api/javax/swing/JOptionPane.html#showMessageDialog(jav)

**showMessageDialog**

```
public static void showMessageDialog(Component parentComponent,
                                     Object message)
                                     throws HeadlessException
```

Brings up an information-message dialog titled "Message".

**Parameters:**

- parentComponent - determines the Frame in which the dialog is displayed; if null, or if the parentComponent has no Frame, a default Frame is used
- message - the object to display

**Throws:**

- HeadlessException - if GraphicsEnvironment.isHeadless returns true

**See Also:**  
[GraphicsEnvironment.isHeadless\(\)](#)

---

**showMessageDialog**

```
public static void showMessageDialog(Component parentComponent,
                                     Object message,
                                     String title,
                                     int messageType)
                                     throws HeadlessException
```

Brings up a dialog that displays a message using a default icon determined by the messageType parameter.

**Parameters:**

- parentComponent - determines the Frame in which the dialog is displayed; if null, or if the parentComponent has no Frame, a default Frame is used
- message - the object to display
- title - the title string for the dialog
- messageType - the type of message to be displayed: ERROR\_MESSAGE, INFORMATION\_MESSAGE, WARNING\_MESSAGE, QUESTION\_MESSAGE, or PLAIN\_MESSAGE

**Throws:**

- HeadlessException - if GraphicsEnvironment.isHeadless returns true

**See Also:**  
[GraphicsEnvironment.isHeadless\(\)](#)

---

**showMessageDialog**

188

**JOptionPane (Java Platform SE 6)**

[http://docs.oracle.com/javase/6/docs/api/javax/swing/JOptionPane.html#showMessageDialog\(jav](http://docs.oracle.com/javase/6/docs/api/javax/swing/JOptionPane.html#showMessageDialog(jav)

convenience methods have also been defined -- overloaded versions of the basic methods that use different parameter lists.

All dialogs are modal. Each showXxxDialog method blocks the caller until the user's interaction is complete.

The basic appearance of one of these dialog boxes is generally similar to the picture at the right, although the various look-and-feels are ultimately responsible for the final result. In particular, the look-and-feels will adjust the layout to accommodate the option pane's ComponentOrientation property.

**Parameters:**  
 The parameters to these methods follow consistent patterns:

**parentComponent**  
 Defines the component that is to be the parent of this dialog box. It is used in two ways: the frame that contains it is used as the frame parent for the dialog box, and its screen coordinates are used in the placement of the dialog box. In general, the dialog box is placed just below the component. This parameter may be null, in which case a default frame is used as the parent, and the dialog will be centered on the screen (depending on the L&F).

**message**  
 A descriptive message to be placed in the dialog box. In the most common usage, message is just a string or String constant. However, the type of this parameter is actually Object. Its interpretation depends on its type:

**Object[]**  
 An array of objects is interpreted as a series of messages (one per object) arranged in a vertical stack. The interpretation is recursive -- each object in the array is interpreted according to its type.

**Component**  
 The component is displayed in the dialog.

**Icon**  
 The icon is wrapped in a JLabel and displayed in the dialog.

**others**  
 The object is converted to a string by calling its toString method. The result is wrapped in a JLabel and displayed.

**messageType**  
 Defines the style of the message. The Look and Feel manager may lay out the dialog differently depending on this value, and will often provide a default icon. The possible values are:

- ERROR\_MESSAGE
- INFORMATION\_MESSAGE
- WARNING\_MESSAGE
- QUESTION\_MESSAGE
- PLAIN\_MESSAGE

189

**Java - ImageDemo/src/ImageDemo.java - Eclipse - /Users/alex/rep/svn/wzw\_ss2012/teil\_java/workspace**

**ImageDemo.java**

```
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class ImageDemo {

    public static void main(String[] args) {
        try {
            URL url = new URL("http://www.google.com/intl/en_ALL/images/logos/images_logo_lg.gif");
            BufferedImage image = ImageIO.read(url);
            JOptionPane.showMessageDialog(null, new JLabel(new ImageIcon(image)));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

192

ImageIcon (Java Platform SE 6)

<http://docs.oracle.com/javase/6/docs/api/java/swing/ImageIcon.html>

protected `ImageIcon.AccessibleImageIcon`  
This class implements accessibility support for the `ImageIcon` class.

### Field Summary

protected <code>Component component</code>
protected <code>MediaTracker tracker</code>

### Constructor Summary

<code>ImageIcon()</code> Creates an uninitialized image icon.
<code>ImageIcon(byte[] imageData)</code> Creates an <code>ImageIcon</code> from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3) PNG.
<code>ImageIcon(byte[] imageData, String description)</code> Creates an <code>ImageIcon</code> from an array of bytes which were read from an image file containing a supported image format, such as GIF, JPEG, or (as of 1.3) PNG.
<code>ImageIcon(Image image)</code> Creates an <code>ImageIcon</code> from an image object.
<code>ImageIcon(Image image, String description)</code> Creates an <code>ImageIcon</code> from the image.
<code>ImageIcon(String filename)</code> Creates an <code>ImageIcon</code> from the specified file.
<code>ImageIcon(String filename, String description)</code> Creates an <code>ImageIcon</code> from the specified file.
<code>ImageIcon(URL location)</code> Creates an <code>ImageIcon</code> from the specified URL.
<code>ImageIcon(URL location, String description)</code> Creates an <code>ImageIcon</code> from the specified URL.

### Method Summary

<code>AccessibleContext getAccessibleContext()</code>
---

What's an easy way to display an Image in Java/Scala? - Stack Overflow

<http://stackoverflow.com/questions/9027257/whats-an-easy-way-to-display-an-image-in>

scala x 9773  
asked 5 months ago  
viewed 222 times  
active 5 months ago

2 Answers

`JOptionPane.showMessageDialog(parent, new JLabel(new ImageIcon(theImage)));`

9  
Of course, if you have an URL for the image (or can form one from a File path), it can also be displayed as a tool-tip using HTML.

Message

See Show full Image when hover over thumbnail for source.

link | improve this answer | edited Jan 28 at 2:18 | answered Jan 27 at 0:06 by Andrew Thompson | Andrew Thompson 50.4k ● 4 ● 20 ● 54

Was this post useful to you? Yes No

CAREERS 2.0

Software Developer  
Ravensburger Digital GmbH  
Munich, Germany

Mobile Developer for Travel Startup  
Triposo  
Berlin, Germany

Software Developer with focus on Application Development  
OMICRON electronics GmbH  
Nuremberg, Germany

Linked

Show full image when hover over thumbnail as Popup/Overlay

Related

How can I display a bitmap image in a Java application? 186

Java - ImageDemo/src/ImageDemo.java - Eclipse - /Users/alex/rep/svn/wzw\_ss2012/teil\_java/workspace

ImageDemo.java

```
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

public class ImageDemo {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://www.google.com/intl/en_ALL/images/logos/images_logo_lg.gif");
            BufferedImage image = ImageIO.read(url);
            JOptionPane.showMessageDialog(null, new JLabel(new ImageIcon(image)));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Problems | @ Javadoc | Declaration | Console | <terminated> ImageDemo [Java Application] /System/Library/Java/javaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java [Jul 5, 2012 2:09:06 F]