

13. Pakete

13.1 Idee, Export und Import

13.2 Pakete und Verzeichnisse

13.3 Sichtbarkeitsattribute

13.4 Beispielpakete aus der Java-Bibliothek

Paket = Sammlung zusammengehöriger Klassen (Bibliothek)

Zweck

- mehr Ordnung in Programme bringen
- bessere Kontrolle der Zugriffsrechte (wer darf auf was zugreifen)
- Vermeidung von Namenskonflikten

Beispiele für Pakete in der Java-Klassenbibliothek

<i>Paket</i>	<i>enthaltene Klassen</i>
java.lang	System, String, Integer, Character, Object, Math, ...
java.io	File, InputStream, OutputStream, Reader, Writer, ...
java.awt	Button, CheckBox, Frame, Color, Cursor, Event, ...
java.util	ArrayList, Hashtable, BitSet, Stack, Vector, Random, ...
...	...

siehe: <http://docs.oracle.com/javase/9/docs/api/>

Anlegen von Paketen

Datei *Circle.java*

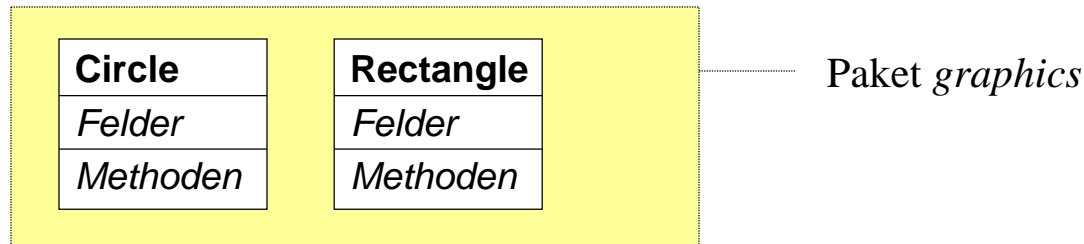
```
package graphics;
class Circle {
    ...
}
```

Datei *Rectangle.java*

```
package graphics;
class Rectangle {
    ...
}
```

← 1. Zeile der Datei

Paket *graphics* enthält die Klassen *Circle* und *Rectangle*

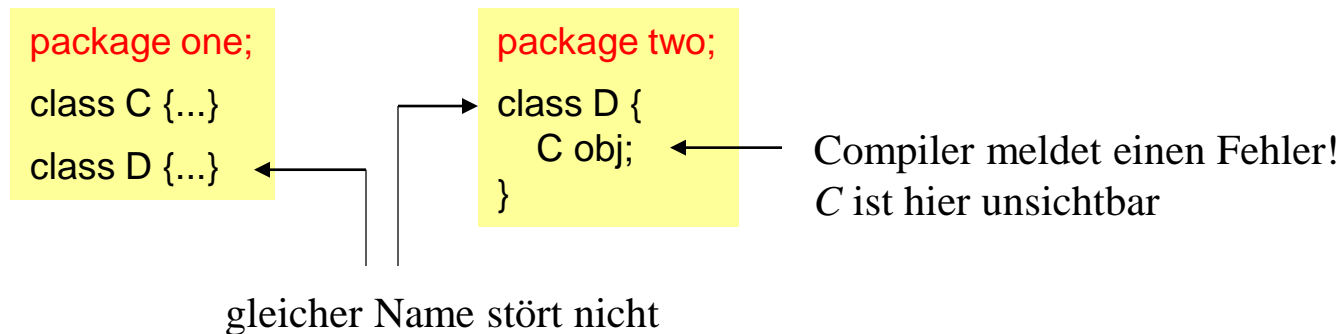


Wenn *package*-Zeile fehlt, gehören die Klassen zu einem namenlosem Standardpaket

Pakete sind Sichtbarkeitsgrenzen



Was in einem Paket deklariert ist, ist in anderen Paketen unsichtbar



Zweck

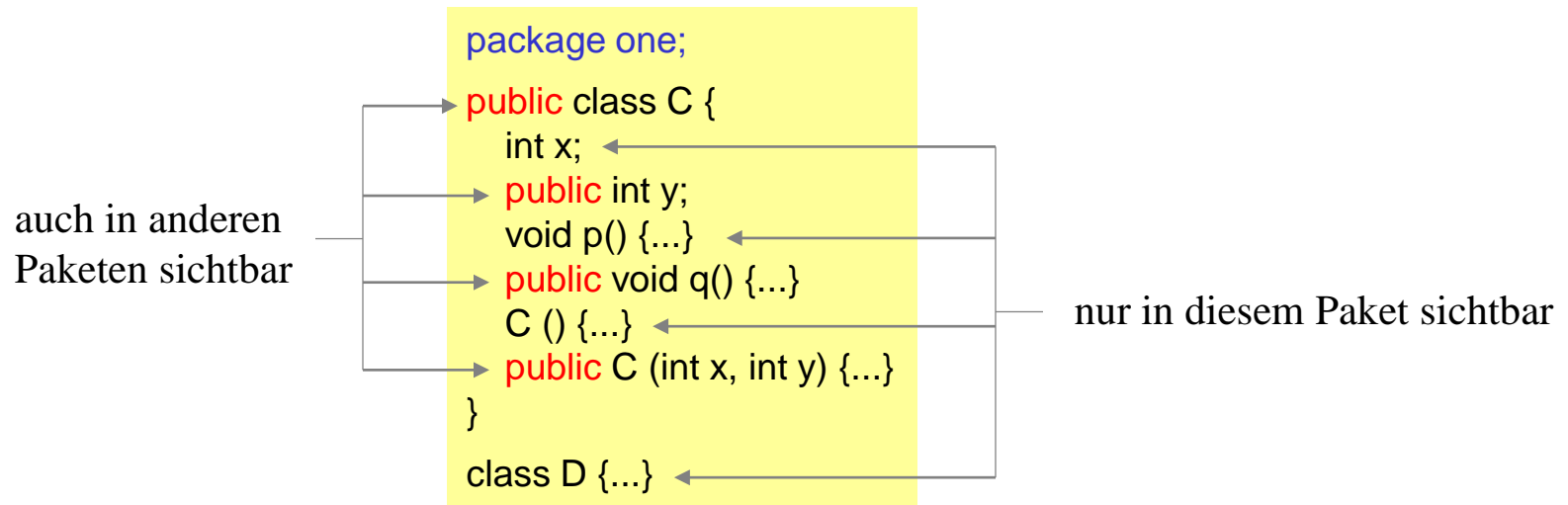
- In verschiedenen Paketen können gleiche Namen verwendet werden
- Programmierer müssen nicht Rücksicht nehmen, welche Namen schon woanders verwendet wurden

Export von Namen



Namen können mit dem Zusatz *public* exportiert werden

(sie sind dann in anderen Paketen sichtbar)



public-Felder und -Methoden werden nur dann exportiert, wenn die Klasse selbst *public* ist.

Lokale Variablen und Parameter können nicht exportiert werden.

Import von Klassennamen



Paket *graphics*

Circle	Rectangle
<i>Felder</i>	<i>Felder</i>
<i>Methoden</i>	<i>Methoden</i>

Exportierte Klassennamen können in anderen Paketen importiert werden

Durch gezielten Import
der Klasse

```
package myPack;  
import graphics.Circle;  
import one.C;  
class MyClass {  
    Circle c;  
    ...  
}
```

Durch Qualifikation
mit dem Paketnamen

```
package myPack;  
class MyClass {  
    graphics.Circle c1;  
    java.awt.Circle c2;  
    ...  
}
```

Durch Import aller public-
Klassen eines Pakets

```
package myPack;  
import graphics.*;  
class MyClass {  
    Circle c;  
    Rectangle r;  
    ...  
}
```

13. Pakete

13.1 Idee, Export und Import

13.2 Pakete und Verzeichnisse

13.3 Sichtbarkeitsattribute

13.4 Beispielpakete aus der Java-Bibliothek

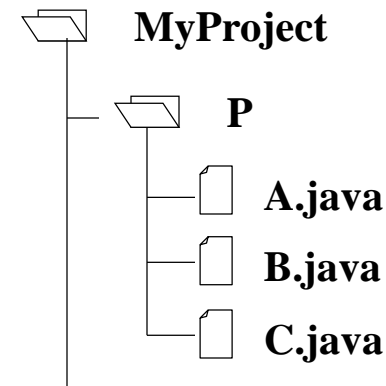
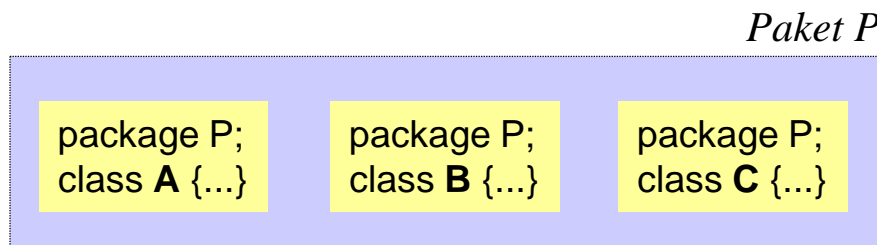
Pakete und Verzeichnisse



Pakete werden auf Verzeichnisse abgebildet, Klassen auf Dateien

Klasse *C* \mapsto Datei *C.java*

Paket *P* \mapsto Verzeichnis *P*



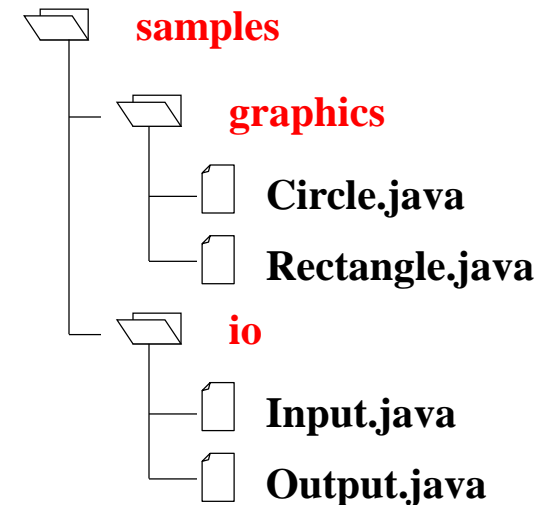
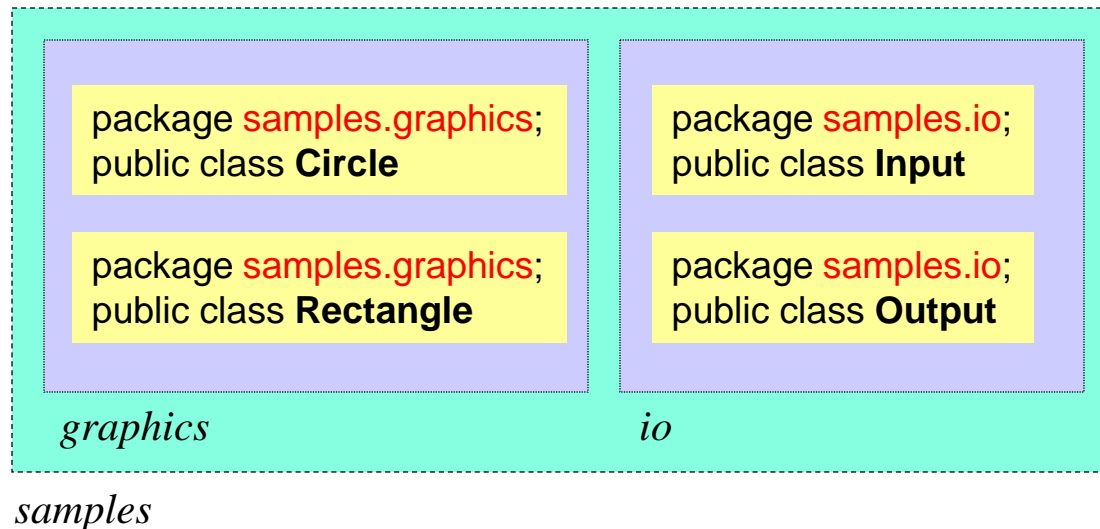
Übersetzung und Ausführung mit dem JDK

```
cd C:\MyProject
javac P/A.java
java P/A
java P.A
```

} beides möglich

Geschachtelte Pakete

Pakete können zu größeren Paketen zusammengefasst werden



Benutzung

```
import samples.graphics.Circle;
import samples.graphics.*;
import samples.*;
```

```
samples.io.Output out;
```

importiert die Klasse *Circle*

importiert alle public-Klassen aus *samples.graphics*

importiert alle public-Klassen aus *samples*

(nicht aus *samples.graphics*)

Qualifikation einer Klasse aus einem geschachtelten Paket

13. Pakete

13.1 Idee, Export und Import

13.2 Pakete und Verzeichnisse

13.3 Sichtbarkeitsattribute

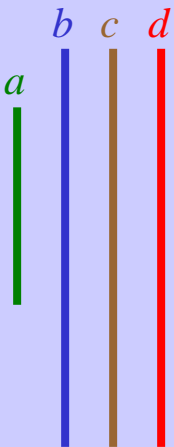
13.4 Beispielpakete aus der Java-Bibliothek

Sichtbarkeitsattribute

für Felder, Methoden, Konstruktoren, Klassen (nicht für lokale Variablen)

- private** int a; nur in der Klasse sichtbar, in der das Element deklariert wurde
- int b; nur im Paket sichtbar, in dem das Element deklariert wurde
- protected** int c; sichtbar:
- in der deklarierenden Klasse
 - im deklarierenden Paket
 - in Unterklassen (selbst wenn diese in anderen Paketen liegen)
- public** int d; auch in anderen Paketen sichtbar, wenn Klasse importiert


```
package one;
public class C {
    private int a;
    int b;
    protected int c;
    public int d;
}
public class D {
    ...
}
```



```
package two;
import one.C;

public class E extends C {
    C x = new C();
    x.a = ...;
    x.b = ...;
    x.c = ...;
    x.d = ...;
}

public class F {
    ...
}
```

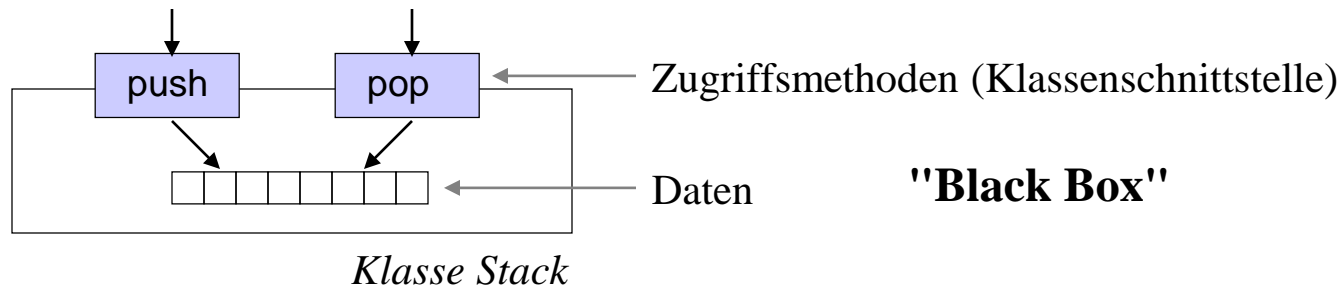


Information Hiding (Geheimnisprinzip)



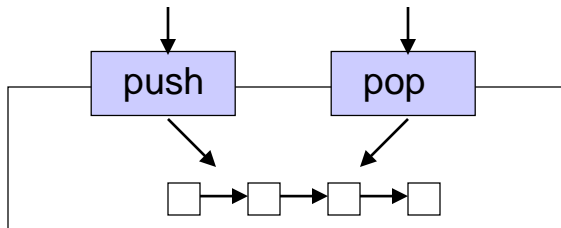
Prinzip

- Verstecke die Implementierung komplexer Datenstrukturen vor Benutzern
- Erlaube den Zugriff auf die Daten nur über Methoden



Warum?

- Verringert die Komplexität (Arbeiten mit den Daten wird einfacher)
- Implementierung der Daten kann geändert werden, ohne dass Benutzer etwas merken
- Schutz vor mutwilliger oder unabsichtlicher Zerstörung



Beispiel: Stack mit Information Hiding



```
public class Stack {  
    protected int[] data;  
    protected int top;  
  
    public Stack (int size) {  
        data = new int[size]; top = -1;  
    }  
  
    public void push (int x) {  
        if (top >= data.length) error("stack overflow");  
        else data[++top] = x;  
    }  
  
    public int pop () {  
        if (top < 0) { error("stack underflow"); return 0; }  
        else return data[top--];  
    }  
  
    private void error (String msg) {  
        Out.println(msg);  
        System.exit(0);  
    }  
}
```

Klassenschnittstelle

Stack
Stack() push(int x) pop(): int

```
public class BetterStack extends Stack {  
    public boolean contains (int x) {  
        for (int i = 0; i <= top; i++) {  
            if (data[i] = x) return true;  
        }  
        return false;  
    }  
}
```

Zugriff nur möglich, weil protected

13. Pakete

13.1 Idee, Export und Import

13.2 Pakete und Verzeichnisse

13.3 Sichtbarkeitsattribute

13.4 Beispielpakete aus der Java-Bibliothek

Beispielpakete



Bibliothek enthält derzeit ca. 330 Pakete mit ca. 6000 Klassen

<https://docs.oracle.com/javase/9/docs/api/>

Einige Beispielpakete und ihre Klassen

java.lang	java.util	java.io	java.net
Object System String StringBuilder Math Thread Integer Float Character ...	Arrays Date Random StringTokenizer ... ArrayList LinkedList Hashtable HashMap HashSet TreeSet ...	InputStream OutputStream FileInputStream FileOutputStream PrintStream File Reader Writer ...	Socket ServerSocket URL URLConnection CookieManager ...