Informatik: Packages, Archive und Anwendungen

Ziele

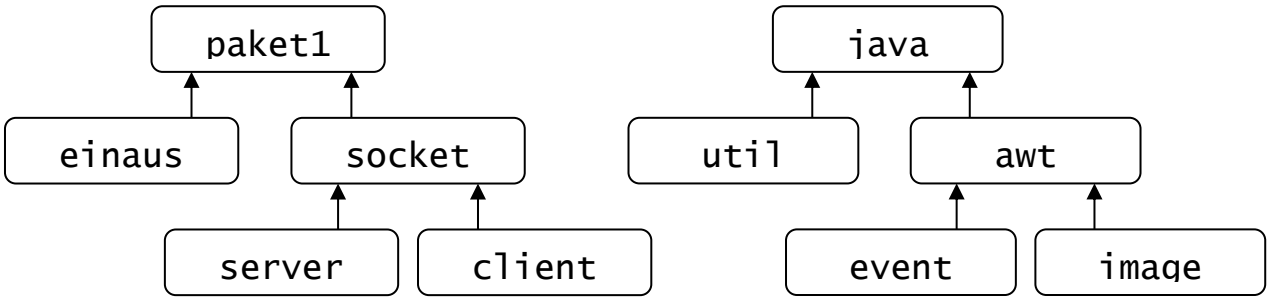
- Packages zur Organisation umfangreicher Programmsysteme verwenden können.
- Verstehen wie Packages im Dateisystem und in Eclipse verwaltet werden.
- Die Benennung von Packages anwenden können.
- Verstehen wie Packages in ein Programm eingebunden werden können.
- Packages in Archivdateien ablegen können.
- Programme in Archivdateien ablegen können.
- Ausführbare Archivdateien erstellen können.

Problem: Eine Klasse ➔ eine Datei

- Mit wachsender Programmgröße sammeln sich viele Klassendeklarationen.
- Orientierung wird schwierig.
- Namenskonflikte zwischen Klassen die denselben Namen haben aber Unterschiedliches leisten.
- Verwaltung von Klassen die zusammen gehören wird schwierig.

Lösung: Packages (dtsch. Pakete)

- Sammlung logisch zusammengehöriger Klassen.
- Klassen mit gleichen Namen, aber aus unterschiedlichen Paketen kollidieren nicht.
- Pakete werden mit Kleinbuchstaben bezeichnet.
- Pakete können geschachtelt werden.
- Jedes Paket gehört zu höchstens einem übergeordneten Paket.



Namen der Pakete:

```
paket1                          java
paket1.einaus                   java.util
paket1.socket                   java.awt
paket1.socket.server            java.awt.event
paket1.socket.client            java.awt.image
```

Dynamisches Laden der benötigten Klasse

- Beim Programmstart werden nicht sofort alle Klassen des Programms geladen (Laden nach Bedarf)

➔ Programme starten damit schneller.

- VM muss Klassen schnell finden können (darf nicht gesamte Dateisystem nach Bytecode durchsuchen)

➔ Paketname legt den Pfad zur Bytecodedatei im Dateisystem fest

➔ Jedes Paket ist ein 

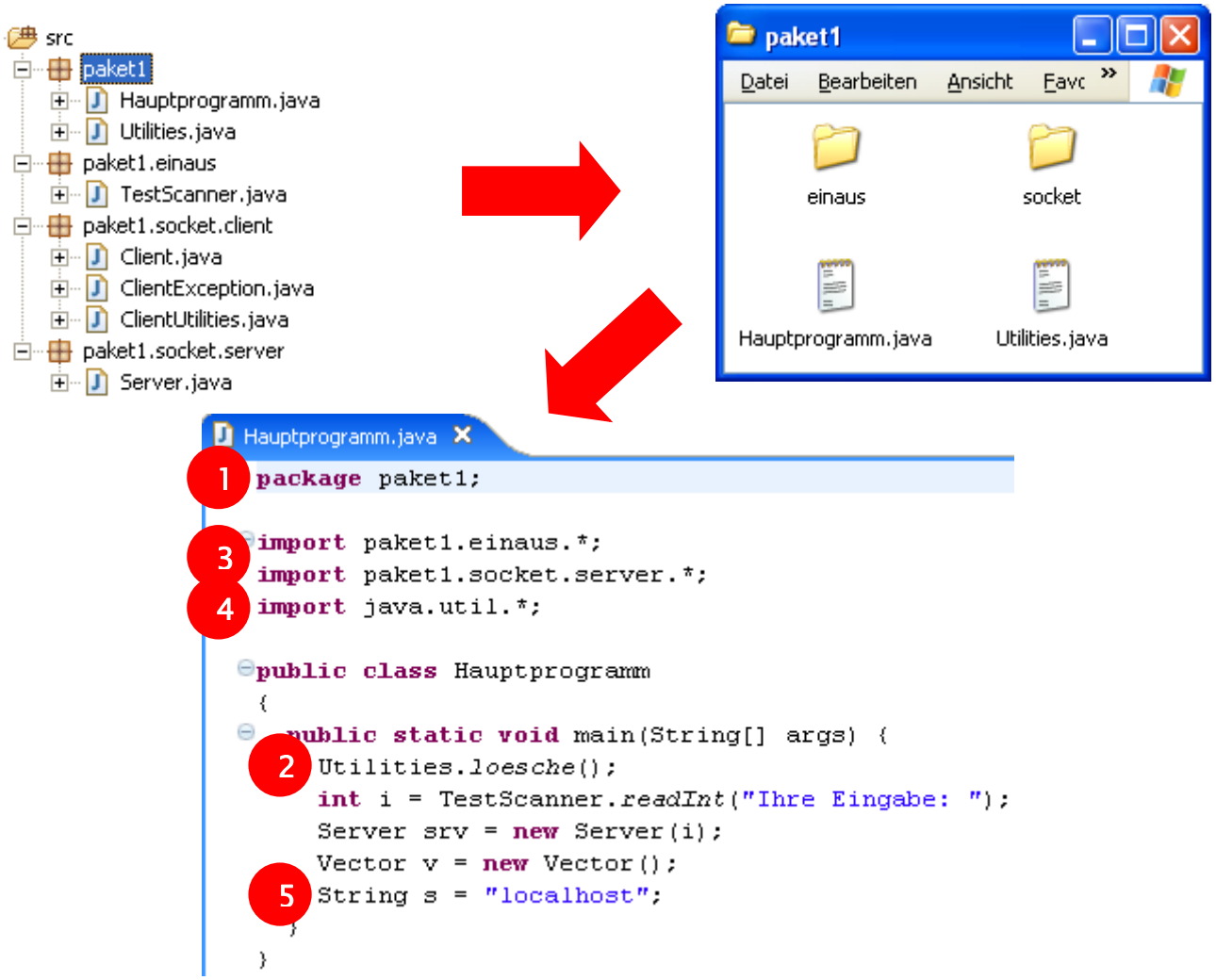
Klasse	*Pfad*
`paket1.socket.Test`	`paket1\socket\Test.class`

Einheitliches Benennungsschema für Pakete

- Um den weltweiten Austausch von Bytecode zu gewährleisten.
- Orientiert sich an Internet-Domainnamen.

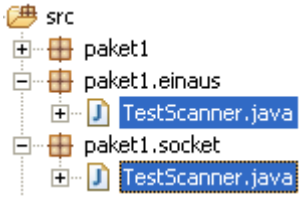
Domainname des Erstellers	*Paketnamen*
`www.sun.com`	**`com.sun.`**`mail.smtp` **`com.sun.`**`mail.pop3` **`com.sun.`**`mail.imap`
`www.tfobz.net`	**`net.tfobz.`**`paket1` **`net.tfobz.`**`paket1.einaus` **`net.tfobz.`**`paket1.socket` **`net.tfobz.`**`paket1.socket.server` **`net.tfobz.`**`paket1.socket.client`
Aber:	
`java` Java-Standardbibliothek	**`java.`**`util` **`java.`**`awt` …
Inhalt ohne **`import`** verwendbar	**`java.`**`lang`
`javax` (eXtended) Erweiterte Java-Bibliothek	**`javax.`**`swing` **`javax.`**`sound` **`javax.`**`sql` …

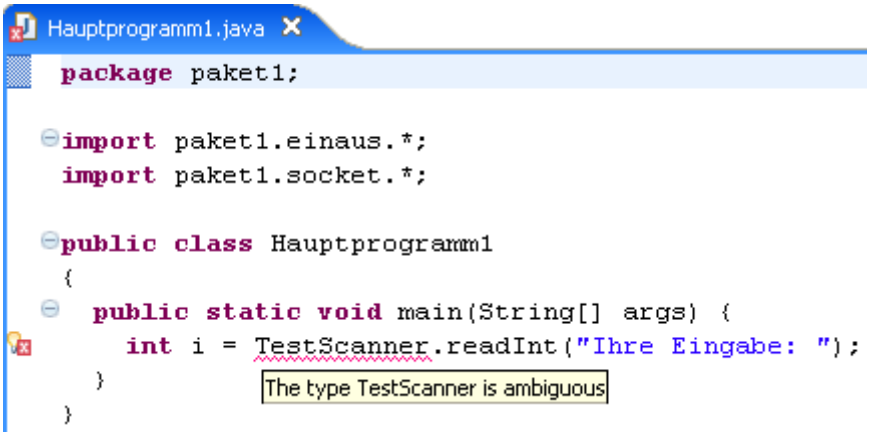
Erstellen von eigenen Paketen



1. legt die Packagezugehörigkeit der Quelltextdatei fest (**package** `paket1.socket.server;`).
2. Klasse `Utilities` ohne weiteren Import verwendbar, da sich diese im selben Paket befindet.
3. Paket `paket1.einaus` und `paket1.socket.server` müssen importiert werden, damit Klassen `TestScanner` und `Server` verwendbar werden.
4. Standardbibliothek `java.util` muss importiert werden, damit Klasse `Vector` verwendbar wird.
5. Klasse `String` ohne Import verwendbar, da in `java.lang` enthalten.

Namenskollisionen





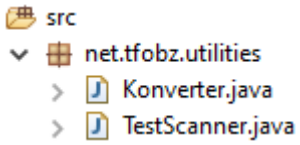
Pakete in Archivdateien (jar-Files) ablegen

ZIP-komprimierte Ordnerstrukturen

- ☺ Klassen sind leichter zusammenzuhalten, einzelne Bestandteile gehen nicht verloren.
- ☺ Kompression sorgt für effiziente Übertragung (z. B. über Internetverbindung).
- ☺ Können digital signiert werden und nicht nur Java-Dateien enthalten.
- ☹ Höherer Aufwand beim Zugriff auf deren Inhalte.

JDK stellt eigenes Werkzeug (`jar.exe`) zur Bearbeitung von jar-Dateien zur Verfügung.

Erstellen eines jar-Archives durch Eclipse

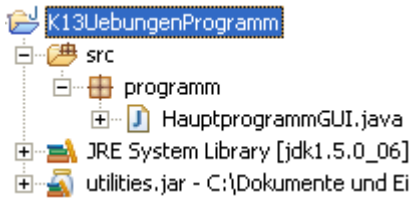


➔ `File.` Export... `.` JAR file

- Welche Klassen?
- Wohin: `utilities.jar`

➔ utilities.jar

Verwenden eines jar-Archives in einem Programm



➔ `File. Properties.`
`Java Build Path.` Libraries `.`
`Add External JARs: utilities.jar`

HauptprogrammGUI.java

```
package programm;

import net.tfobz.utilities.*;
import javax.swing.*;
import java.awt.event.*;

public class HauptprogrammGUI extends JFrame
{

  public static void main(String[] args) {
    HauptprogrammGUI h = new HauptprogrammGUI();
    Konverter k = new Konverter();
  }

  public HauptprogrammGUI() {
    super("JAR-Test");
```

Erstellen eines einfachen, ausführbaren jar-Archives

`File.` Export... `.` JAR file

- Wohin: `programm.jar`
- Main class: `programm.HauptprogrammGUI`

➔ programm.jar

Problem:

In ausführbarem `jar`-Archiv (`programm.jar`) kann kein `jar`-Archiv mit Java-Klassen (`utilities.jar`) enthalten sein.

1. Lösung:

`jar`-Archiv mit Java-Klassen auspacken und in ausführbarem `jar`-Archiv integrieren.

2. Lösung:

Ausführbares `jar`-Archiv und `jar`-Archiv mit Java-Klassen separat dem Kunden übergeben und im ausführbarem `jar`-Archiv folgende *Manifest-Datei* integrieren: