Informatik B

Vorlesung 20

Applets

Rückblick

- GUI-Entwicklung
  - Container
  - Steuerelemente
  - Layout
  - Events
  - Listener
  - Event-Objekte

Applets

- Applets sind GUIs, die im Webbrowser oder AppletViewer lauffähig sind
- Dadurch können dynamische Webseiten erstellt werden
- Die Erstellung der GUI läuft genau wie bei einer Applikation ab
- Ein Applet muss von der Klasse **`java.applet.Applet`** (oder **`javax.swing.JApplet`**) abstammen
- Es sollte als Top-Level-Container kein **`Frame`** verwendet werden
- Wird bei der Erstellung eines Applets ein **`Frame`** geöffnet, wird dies in einem separaten Fenster geöffnet (ist also nicht in die Webseite eingebunden)

Applets

- Steuerelemente können direkt zum Applet hinzugefügt werden
- Eine Applikation muss eine **`main`**-Methode besitzen, damit sie gestartet werden kann
- Bei einem reinen Applet ist dies nicht notwendig
- Es ist lediglich notwendig, dass ein Default Constructor existiert
- Es gibt aber auch hybride Software, die sowohl als Applikation als auch als Applet lauffähig ist
- Diese Programme erweitern einfach die Klasse (**`J)Applet`** und implementieren zusätzlich die **`main()-`** Methode

Applets

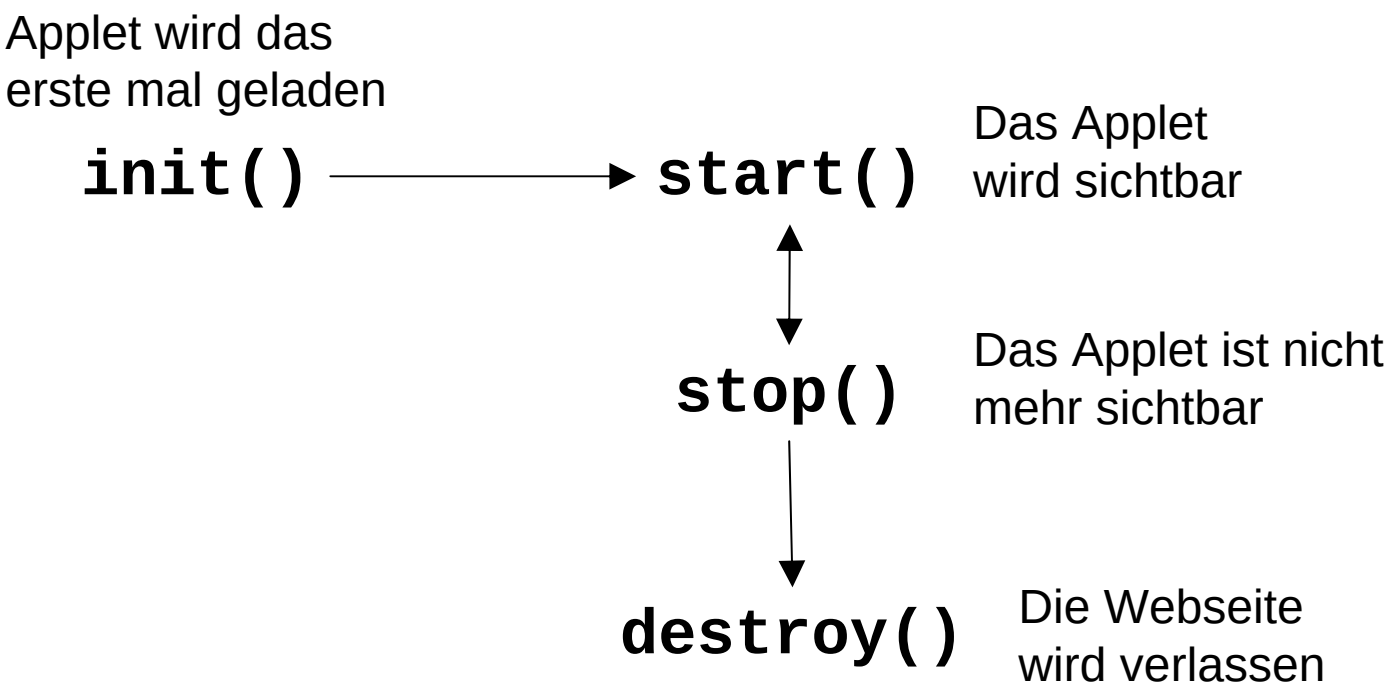
- In eine Webseite kann ein Applet mit folgenden HTML-Tags eingebunden werden:

```
<html>
  <body>
    <applet code="HelloWorldApplet.class"
            width="200" height="100">
    </applet>
  </body>
</html>
```

Zyklen eines Applets

- Beim Start eines Applets werden unterschiedliche Methoden vom Browser automatisch aufgerufen
  - Als erstes wird einmalig die Methode **`init()`** aufgerufen, hier sollten Initialisierungen erfolgen
  - Darauf folgt ein Wechsel der Methoden **`start()`** und **`stop()`**
  - **`start()`** und **`stop()`** werden in der Regel aufgerufen, wenn ein Applet im Browser sichtbar ist und wieder von der Seite verschwindet, etwa wenn der Anwender über die Schieberegler einen anderen Bereich auswählt, in dem das Applet nicht liegt
  - Dies ist aber Webbrowserabhängig!!
  - Beim Verlassen der Seite wird abschließend **`destroy()`** aufgerufen und es können Ressourcen freigegeben werden

Zyklen eines Applets



- Beispiel: **`applet1`**

GUI-Elemente platzieren

- Um die GUI eines Applets zu erstellen, kann ein Applet einen **`LayoutManager`** erhalten, und zum Applet können beliebige **`Container-`** und **`Component`**-Objekte direkt hinzugefügt werden
- Es ist jedoch hilfreich Steuerelemente nicht direkt zum Applet hinzuzufügen, sondern einen separaten **`Container`** zu verwenden und in diesem die GUI-Anwendung zu erstellen
- So ist es später sehr einfach möglich die Anwendung sowohl als Applet, als auch als Applikation zu verwenden
- Beispiel: **`applet2`**

Parameter ans Applet übergeben

- Dem Applet können Parameter im Applet-Tag übergeben werden
- Dazu wird im **<applet>**-Tag ein **<param>**-Tag eingebettet

```
<applet code="Applet.class“
        width="200“ height="100">
   <param name=“var" value=“wert">
</applet>
```

- Mit der Methode **String getParameter(String name)** kann ein Parameter abgefragt werden
- Falls der Parameter nicht gesetzt wurde, wird **null** zurückgeliefert
- Beispiel: **applet3**

`AppletContext`

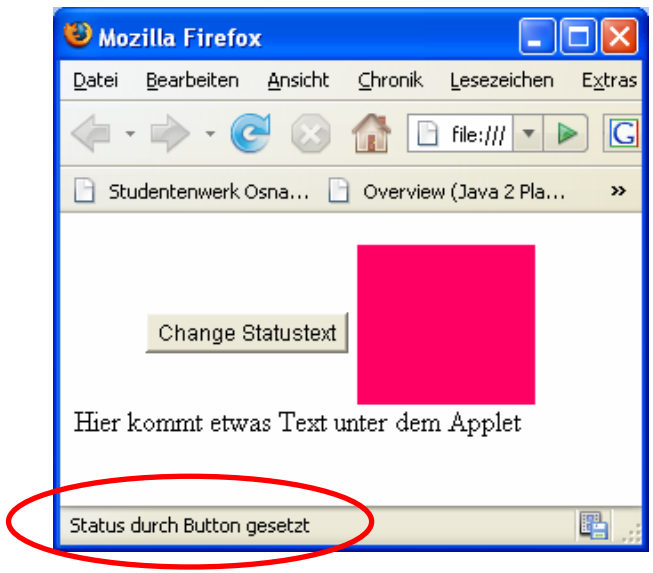
- Der **`AppletContext`** ist ein Objekt, welches das Programm repräsentiert, in dem das Applet ausgeführt wird (z.B. Webbrowser)
- Ein Applet kann ein **`AppletContext`**-Objekt mit der Methode **`getAppletContext()`** liefern
- Mit diesem Objekt kann
  - Eine neue Webseite geladen werden
  - Die Statusangabe geändert werden
  - Ein anderes Applet auf der Webseite angesprochen werden

AppletContext

- Um eine Webseite aus einem Applet heraus zu laden wird die Methode **`void showDocument(URL url)`** verwendet
- Soll die Webseite in einem bestimmten Frame geöffnet werden, kann der Framename angegeben werden:
  `void showDocument(URL url, String target)`
- Beispiel: **`applet4`**

AppletContext

- Die Statuszeile des Webbrowsers kann mit der Methode **`void showStatus(String s)`** verändert werden



- Beispiel: **`applet5`**

AppletContext

- Sind mehrere Applets auf einer Webseite eingebunden, können diese über den **`AppletContext`** miteinander kommunizieren
- Dazu gibt es die Methoden
  - **`Applet getApplet(String name)`**
  - **`Enumeration<Applet> getApplets()`**
- Die auf der Webseite eingebundenen Applets müssen mit einem Namen versehen werden:

```
<applet code="Applet.class“ name=“AppletName“
        width="200„ height="100">
</applet>
```

- Beispiel: **`applet6, applet7`**

JAR-Files

- Besteht ein Applet aus mehreren Dateien, werden diese einzeln über das Netz geladen
- Besser ist es die benötigten Dateien komprimiert in einer JAR-Datei zu übertragen
- Eine JAR-Datei kann leicht mittels Eclipse erzeugt werden
- Auf der Konsole ist dies mit folgendem Befehl möglich:
  `jar cf demo.jar *.class`
- Im Applet-Tag kann mit dem Parameter **`archive`** das JAR-File angegeben werden:

```
<applet code="applet8.MyApplet.class"
        archive="applet8.jar“
        width="800“ height="600“/>
```

Bild aus einem JAR-File laden

- Das Laden zusätzlicher Ressourcen aus einem JAR-File gestaltet sich bei Applets etwas schwierig
- Exkurs:
  - Stellen wir uns eine Applikation vor, die ein Bild laden und darstellen soll
  - Die Klasse **`ImageIO`** kann ein Bild laden
  - Dazu wird die statische Methode **`read(String filename)`** aufgerufen
  - Ergebnis ist ein **`BufferedImage`**-Objekt

Bild aus einem JAR-File laden

- Das Bild soll z.B. auf einem **`Canvas`** dargestellt werden (andere **`Component`**-Objekte sind ebenso möglich)
- Dazu wird eine Unterklasse von **`Canvas`** erstellt und die Methode **`paint(Graphics g)`** überschrieben
- Die Methode **`paint`** wird immer dann aufgerufen, wenn sich das Objekt neu zeichnen soll (Anweisung vom Betriebssystem)
- **`g`** ist der grafische Kontext der **`Component`**
- Das Bild kann dann mit **`g.drawImage(image,0,0,this);`** gezeichnet werden
- Beispiel: **`bildzeichnen1`**

Bild aus einem JAR-File laden

- Implementiert man diese Anwendung als Applet kommt es jedoch zu Problemen beim Laden des Bildes, insofern das Bild in einem JAR-File eingebunden ist
- Beispiel: **`applet8`**
- Das Problem liegt darin begründet, dass mit einer simplen Pfadangabe versucht wird das Bild von dem lokalen Rechner zu landen
- Dies ist nicht erlaubt

Bild aus einem JAR-File laden

- Der Zugriff muss über den Classloader erfolgen
- Dazu muss man sich ein **`Class`**-Objekt einer Klasse aus dem JAR-File besorgen
- Um eine Ressource zu erhalten können folgende Methoden verwendet werden:
  - **`getResource(String name)`**: Liefert ein URL-Objekt zurück
  - **`getResourceAsStream(String name)`**: Liefert ein **`InputStream`**-Objekt zurück
- Der Ort der Ressource muss relativ zum Ort der Klasse angegeben werden
- Beispiel: **`applet9`**

Sicherheit von Applets

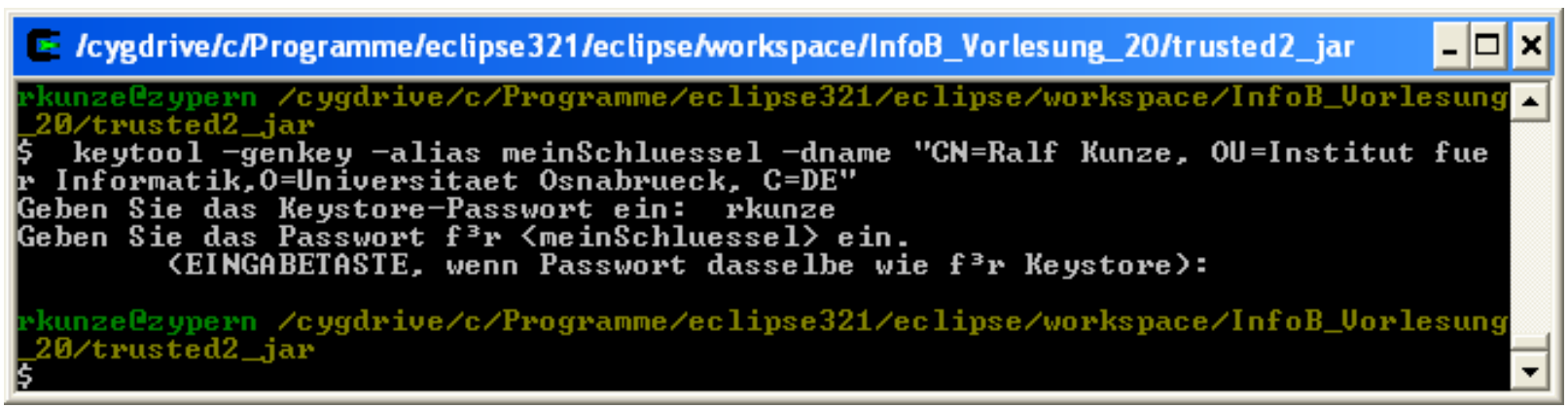
- Ein Applet hat nur eingeschränkte Rechte
- Es kann deshalb:
  - nur Verbindungen zu dem Rechner aufbauen, von dem es geladen wurde
  - keine lokalen Dateien lesen oder schreiben
  - keine externen Programme starten
  - keine native Routinen aufrufen
  - nur beschränkt Systeminformationen auslesen
- Beispiel: **`untrusted1`**

Sicherheit von Applets

- Falls ein Applet mehr Rechte benötigt, kann es signiert werden
- Dazu kann eine Signatur von einem vertrauenswürdigen TrustCenter beschafft werden
- Durch die Signatur kann die Herkunft eines Applets geprüft werden
- Lädt ein Anwender ein solch signiertes Applet, kann er sich die Informationen der Signatur anschauen und entscheiden, ob er einer Freigabe zustimmt oder nicht

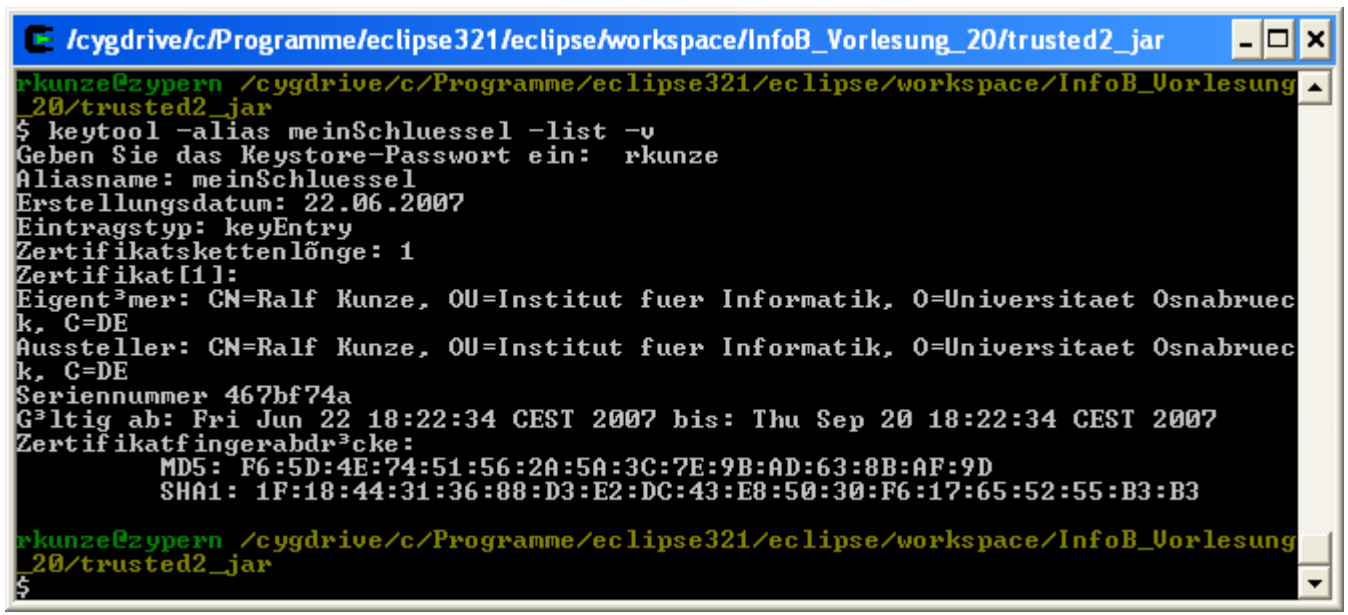
Signierung eines Applets

- Der Zugriff auf eine lokale Datei ist ohne Signatur nicht erlaubt
- Will man ein Applet signieren, muss zunächst ein JAR-File erstellt werden
- Im zweiten Schritt muss ein Schlüssel erzeugt werden
- Dazu dient das Kommandozeilenprogramm **`keytool`**:
  `keytool -genkey -alias meinSchluessel -dname "CN=Ralf Kunze, OU=Institut fuer Informatik,O=Universitaet Osnabrueck, C=DE"`



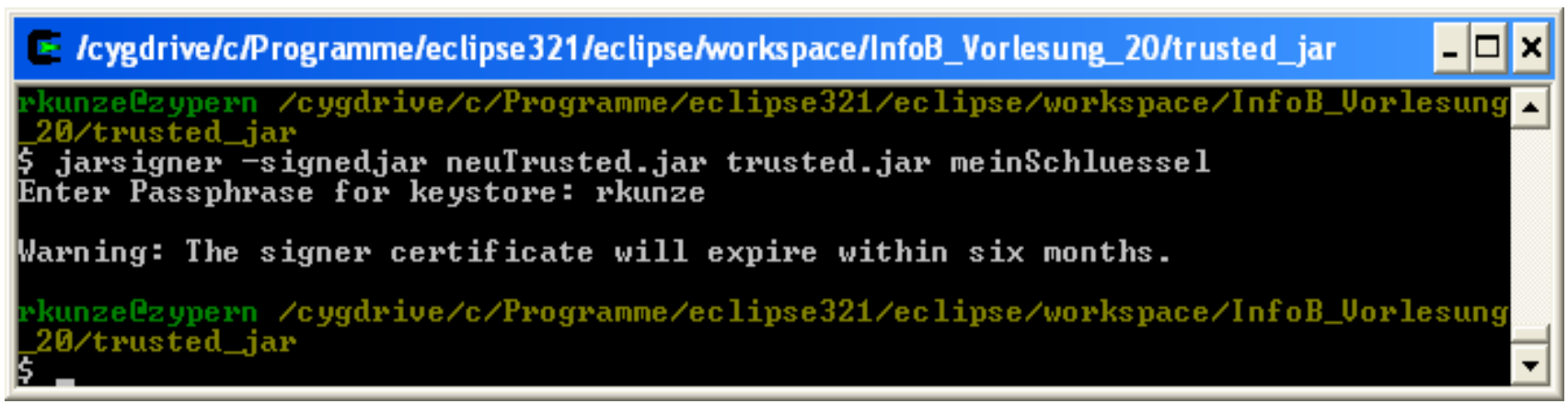
Signierung eines Applets

- Dadurch wird ein Schlüssel angelegt und in einer Datenbank abgelegt
- Vorhandene Schlüssel können mit **`keytool -alias schluesselName -list -v`** angeschaut werden



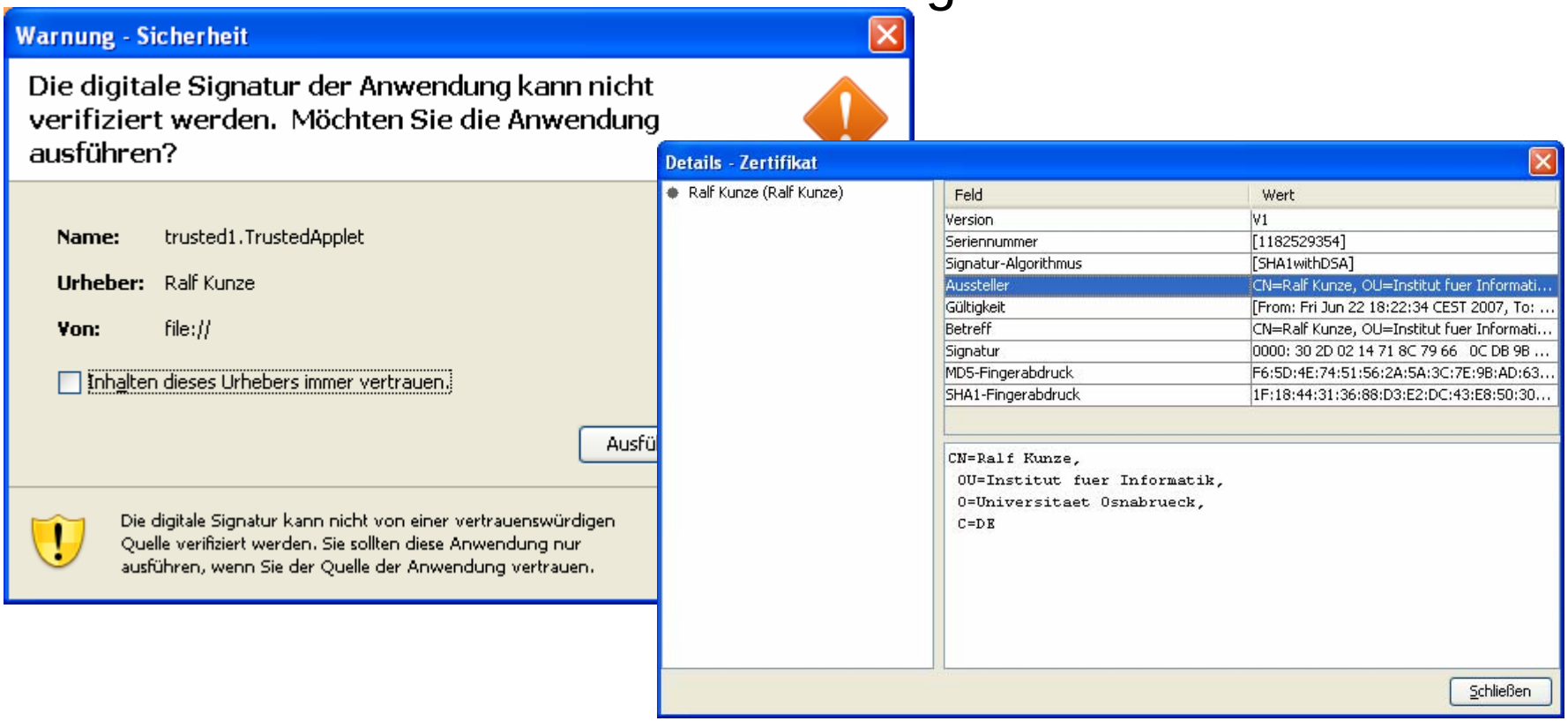
Signierung eines Applets

- Ein JAR-File kann nun mit dem Kommando **`jarsigner`** signiert werden:
  `jarsigner -signedjar neuTrusted.jar trusted.jar meinSchluessel`
- Es wird das signierte JAR-File **`neuTrusted.jar`** erstellt



Signierung eines Applets

- Wird das im JAR-File enthaltene Applet geladen, erfolgt zunächst eine Sicherheitswarnung



Signierung eines Applets

- Akzeptiert der Benutzer diese Warnung, hat das Applet erweiterte Rechte
- Unter anderem besitzt es nun
  - lesenden und schreibenden Zugriff auf das lokale Dateisystem
  - Verbindungen zu dritten können aufgebaut werden
- Beispiel: **`trusted1, trusted2, thirdpartyaccess1`**

Zusammenfassung

- Applets
  - Zyklen eines Applets (**`init`**, **`start`**, **`stop`**, **`destroy`**)
  - **`AppletContext`**
  - JAR-Files
  - Signierung

Ausblick

- MVC (Model View Controller)
- Swing