

## V. Client-Technologien

- ❑ Einführung
- ❑ Exkurs: Programmiersprachen
- ❑ JavaScript
- ❑ Java Applet
- ❑ Weitere Client-Technologien

# Java Applet

## Einführung [\[Einordnung\]](#)

*“A Java applet is a special kind of **Java program** that a browser enabled with Java technology can download from the internet and run.”*

[\[Oracle\]](#)

## Charakteristika:

- ❑ programmiert in der Multi-Purpose-Programmiersprache Java
- ❑ in HTML-Dokumente eingebettete Software**komponenten**

## Anwendung [\[Oracle Demos\]](#) :

- ❑ leistungsfähige grafische Oberflächen
- ❑ hohe Interaktivität zwischen Anwender und Software
- ❑ Netzwerkkommunikation kann beliebige Protokolle implementieren
- ❑ Präsentationsschicht für komplexe (n-Tier-)Architekturen

# Java Applet

## Einführung (Fortsetzung)

### Ein einfaches Applet:

```
package applet;  
import java.applet.*;  
import java.awt.*;  
  
public class AppletHelloWorld extends Applet{  
    public void init(){  
        add(new Label("Hello World!"));    }  
}
```

# Java Applet

## Einführung (Fortsetzung)

### Ein einfaches Applet:

```
package applet;  
import java.applet.*;  
import java.awt.*;  
  
public class AppletHelloWorld extends Applet{  
    public void init(){  
        add(new Label("Hello World!"));    }  
}
```

### Eine HTML-Seite, die das Applet einbindet:

```
<!DOCTYPE html>  
<html>  
    <head> <title>Applet Sample</title> </head>  
    <body>  
        <p>Here is the output of my program:</p>  
        <object type="application/x-java-applet" width="150" height="25">  
            <param name="code" value="applet.AppletHelloWorld">  
            <param name="codebase" value=".">  
            <param name="archive" value="part-client-...-code-java.jar">  
        </object>  
    </body>  
</html>
```

# Java Applet

## Einführung (Fortsetzung)

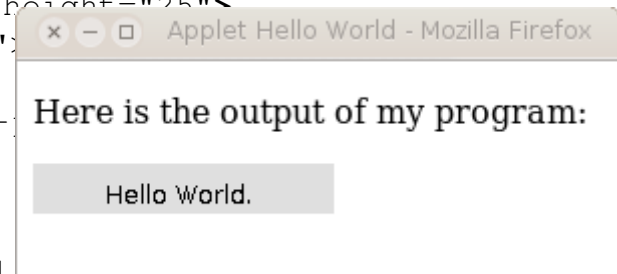
### Ein einfaches Applet:

```
package applet;  
import java.applet.*;  
import java.awt.*;  
  
public class AppletHelloWorld extends Applet{  
    public void init(){  
        add(new Label("Hello World!"));    }  
}
```

### Eine HTML-Seite, die das Applet einbindet:

```
<!DOCTYPE html>  
<html>  
  <head> <title>Applet Sample</title> </head>  
  <body>  
    <p>Here is the output of my program:</p>  
    <object type="application/x-java-applet" width="150" height="25">  
      <param name="code" value="applet.AppletHelloWorld">  
      <param name="codebase" value=".">  
      <param name="archive" value="part-client-...-code-...>  
    </object>  
  </body>  
</html>
```

[\[Applet-Ausführung\]](#)



# Java Applet

## Einführung (Fortsetzung)

### Ein einfaches Applet:

```
package applet;
import java.applet.*;
import java.awt.*;

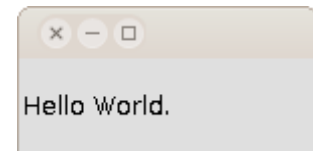
public class AppletHelloWorld extends Applet{
    public void init(){
        add(new Label("Hello World!"));    }
}
```

### Eine vergleichbare Java-Anwendung:

```
package applet;
import java.awt.*;

public class ApplicationHelloWorld extends Frame{
    public ApplicationHelloWorld(){
        add(new Label("Hello World!"));    }

    public static void main(String[] args){
        ApplicationHelloWorld hwa = new ApplicationHelloWorld();
        hwa.setSize(150,75);
        hwa.setVisible(true);    }
}
```



# Java Applet

## Einbindung in HTML-Dokumente

```
<object
  type="application/x-java-applet"
  width="pixels"
  height="pixels"
  [name="applet_instance_name"] >

  <param name="code" value="applet_file">
  [<param name="archive" value="file1.jar, file2.jar">]
  [<param name="codebase" value="codebase_url">]
  [<param name="applet_parameter1" value="value1">]

  . . .
  [alternate_html]
</object>
```

# Java Applet

## Einbindung in HTML-Dokumente

```
<object
  type="application/x-java-applet"
  width="pixels"
  height="pixels"
  [name="applet_instance_name"] >

  <param name="code" value="applet_file">
  [<param name="archive" value="file1.jar, file2.jar">]
  [<param name="codebase" value="codebase_url">]
  [<param name="applet_parameter1" value="value1">]

  ...
  [alternate_html]
</object>
```

### Beispiel:

```
<object type="application/x-java-applet" width="500" height="20">
  <param name="code" value="aisearch/client/Client.class">
  <param name="archive" value="engine/aisearch.jar">
  <param name="imagesource" value="images/picture1.jpg">
  <param name="backgroundcolor" value="0xc0c0c0">
  ...
</object>
```

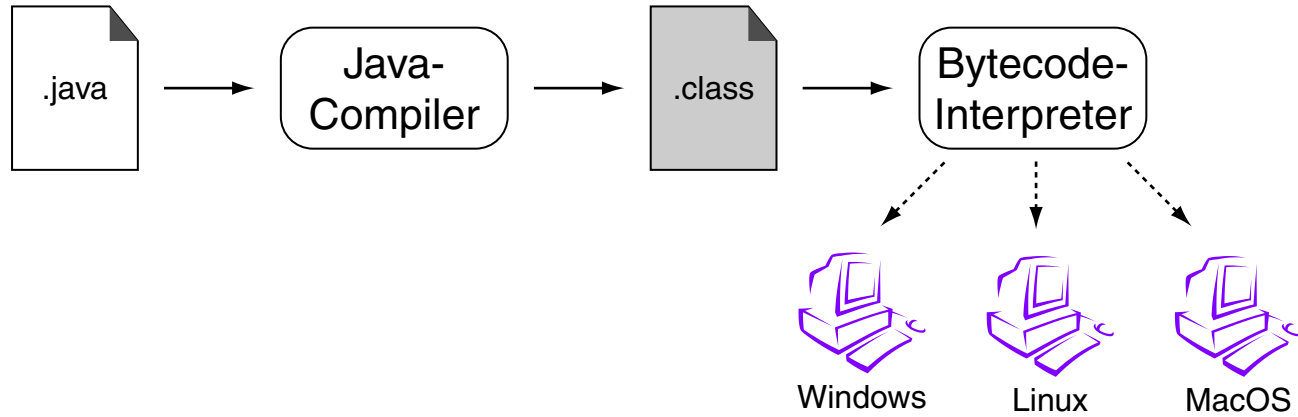


## Bemerkungen:

- ❑ Der Parameter `code` spezifiziert die Applet-Klasse, die vom Browser instantiiert wird.
- ❑ Die Wertzuweisung bei den Parametern `code`, `codebase` und `archive` zeigt, dass – wie in Java üblich – die Paketstruktur der Anwendung zu berücksichtigen ist. Mittels `codebase` wird der Java-Classpath gesetzt.

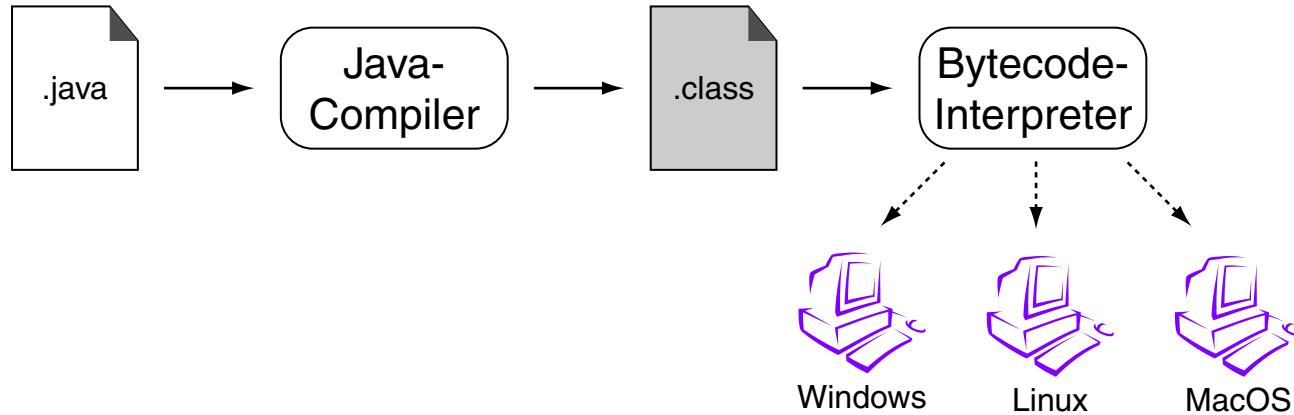
# Java Applet

## Java-Plattform

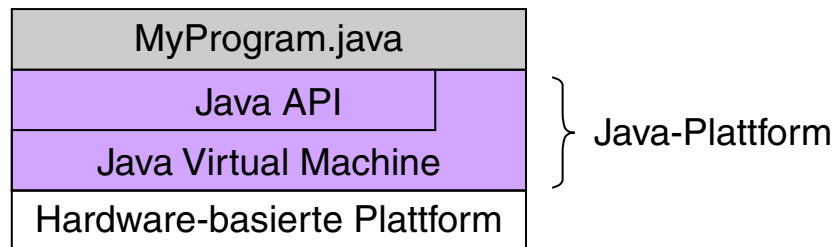


# Java Applet

## Java-Plattform

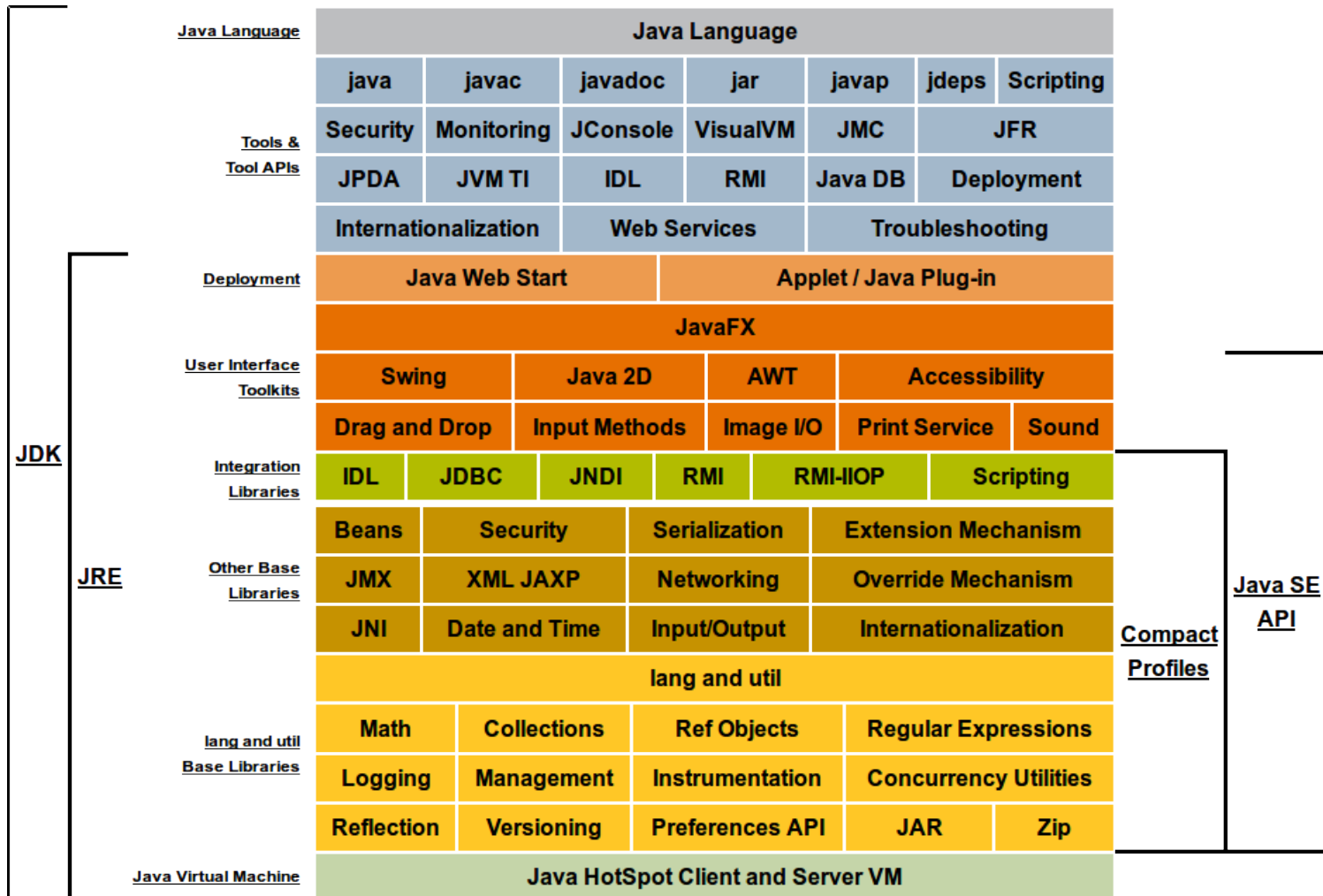


*“A platform is the hardware or software environment in which a program runs. [...] Most platforms can be described as a combination of the operating system and underlying hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.”* [\[Oracle\]](#)



# Java Applet

## Java-Plattform (Fortsetzung)



# Java Applet

## Applet-Lebenszyklus

Applets können

1. initialisiert,
2. gestartet,
3. gestoppt und
4. aus dem Speicher entfernt werden.

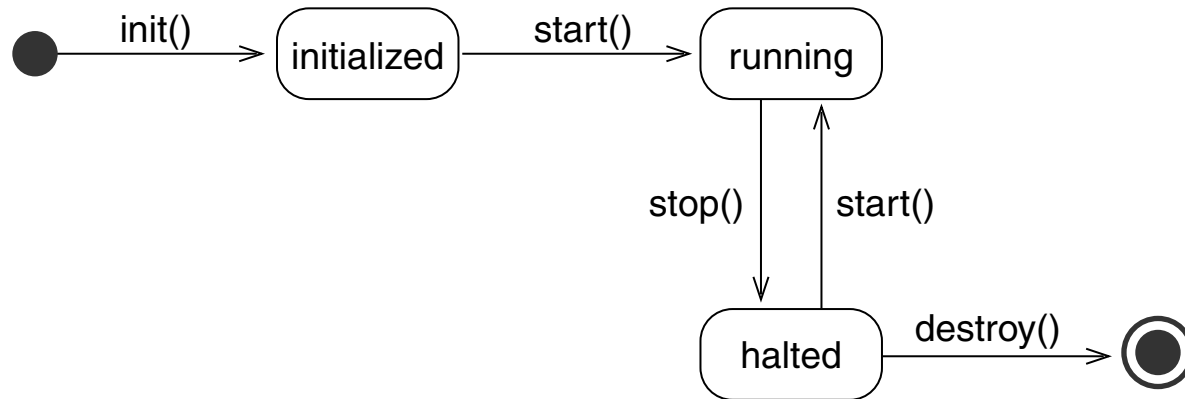
Die entsprechenden Java-Befehle:

```
public class myApplet extends Applet {  
    ...  
    public void init() {...}           // prepare variables, GUI  
    public void start() {...}          // start running  
    public void stop() {...}           // stop running  
    public void destroy() {...}        // cleanup  
    ...  
}
```

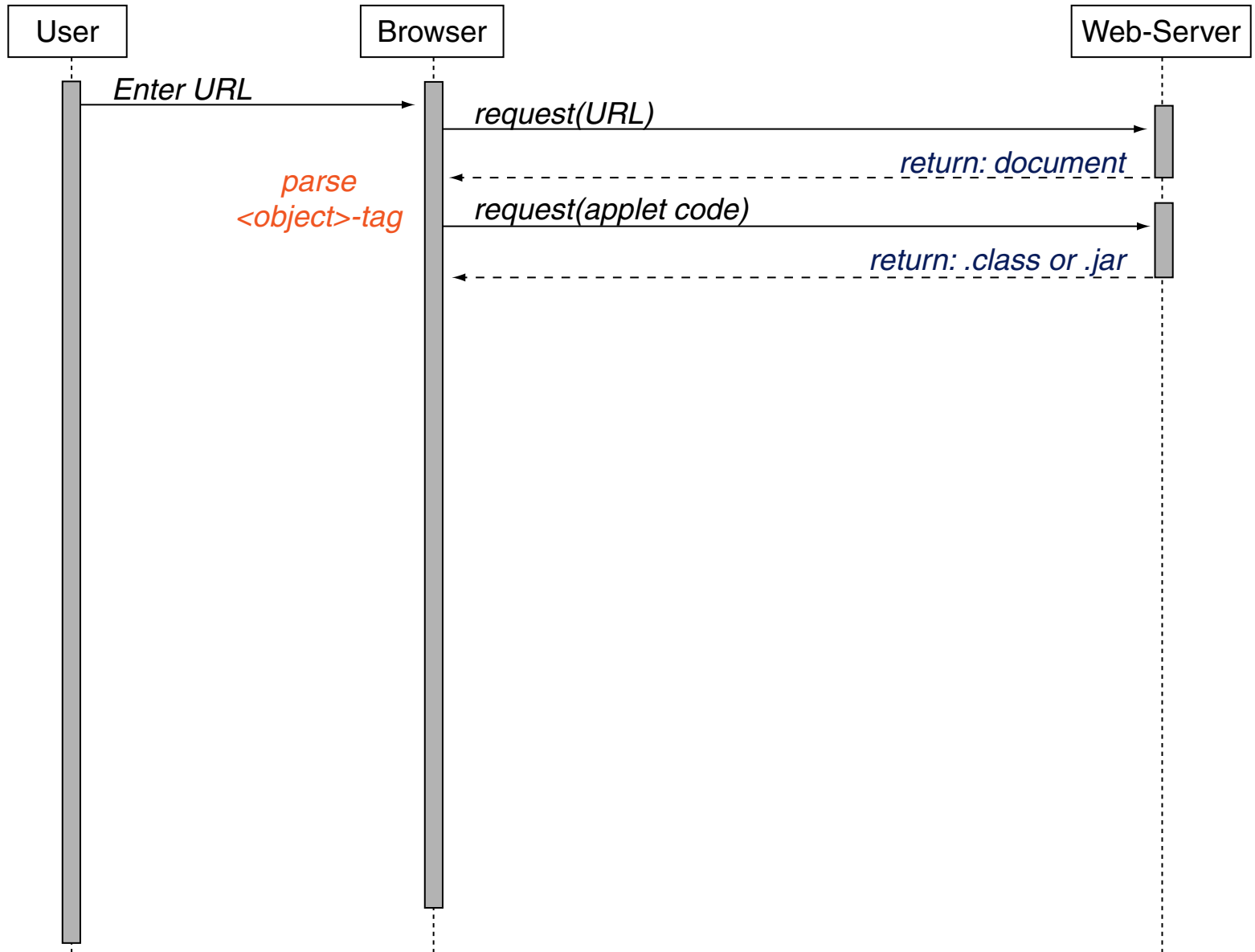
# Java Applet

## Applet-Lebenszyklus (Fortsetzung)

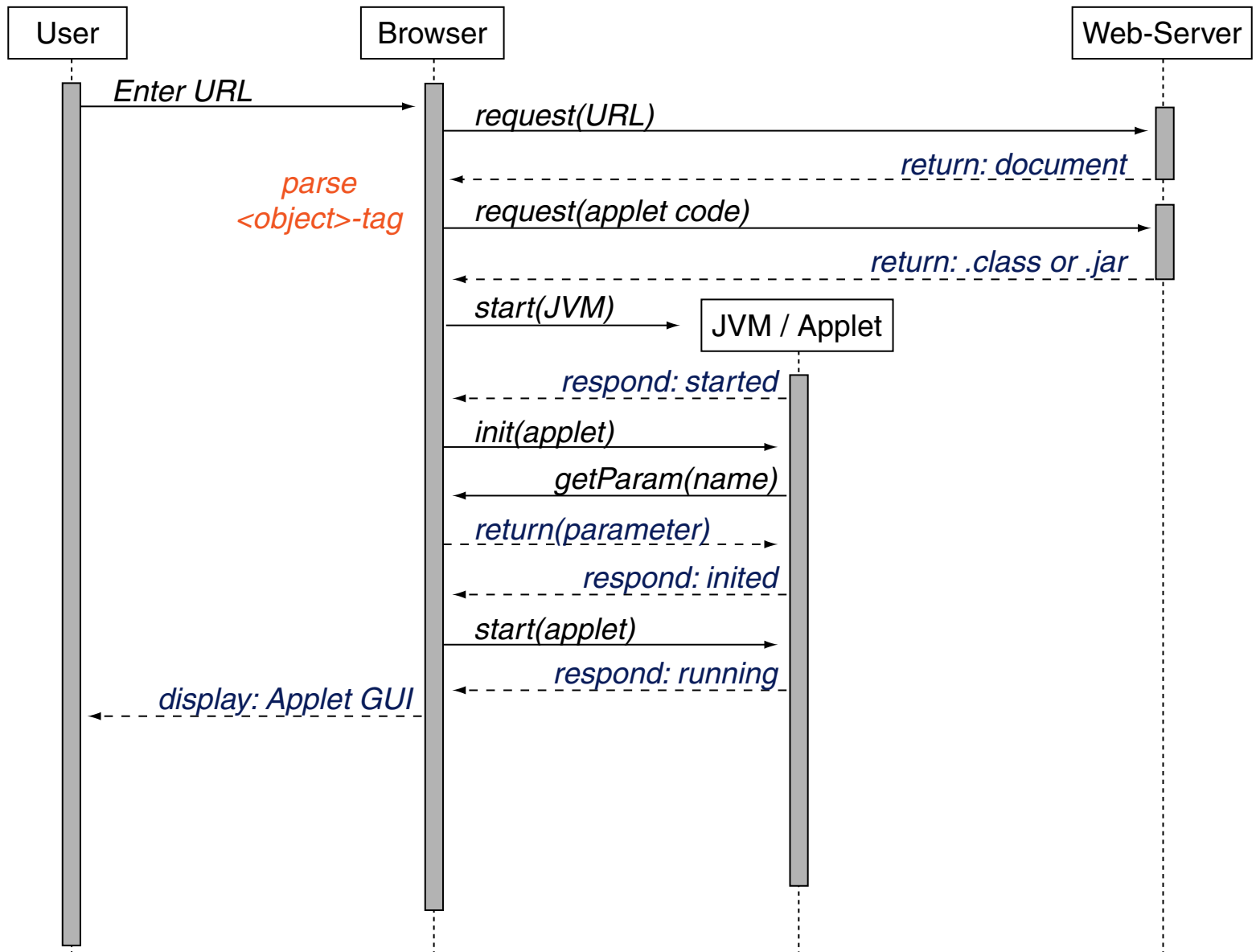
Nach dem Laden instantiiert der Browser die Applet-Klasse, ruft dann die `init()`-Methode und danach die `start()`-Methode auf.



# Java Applet

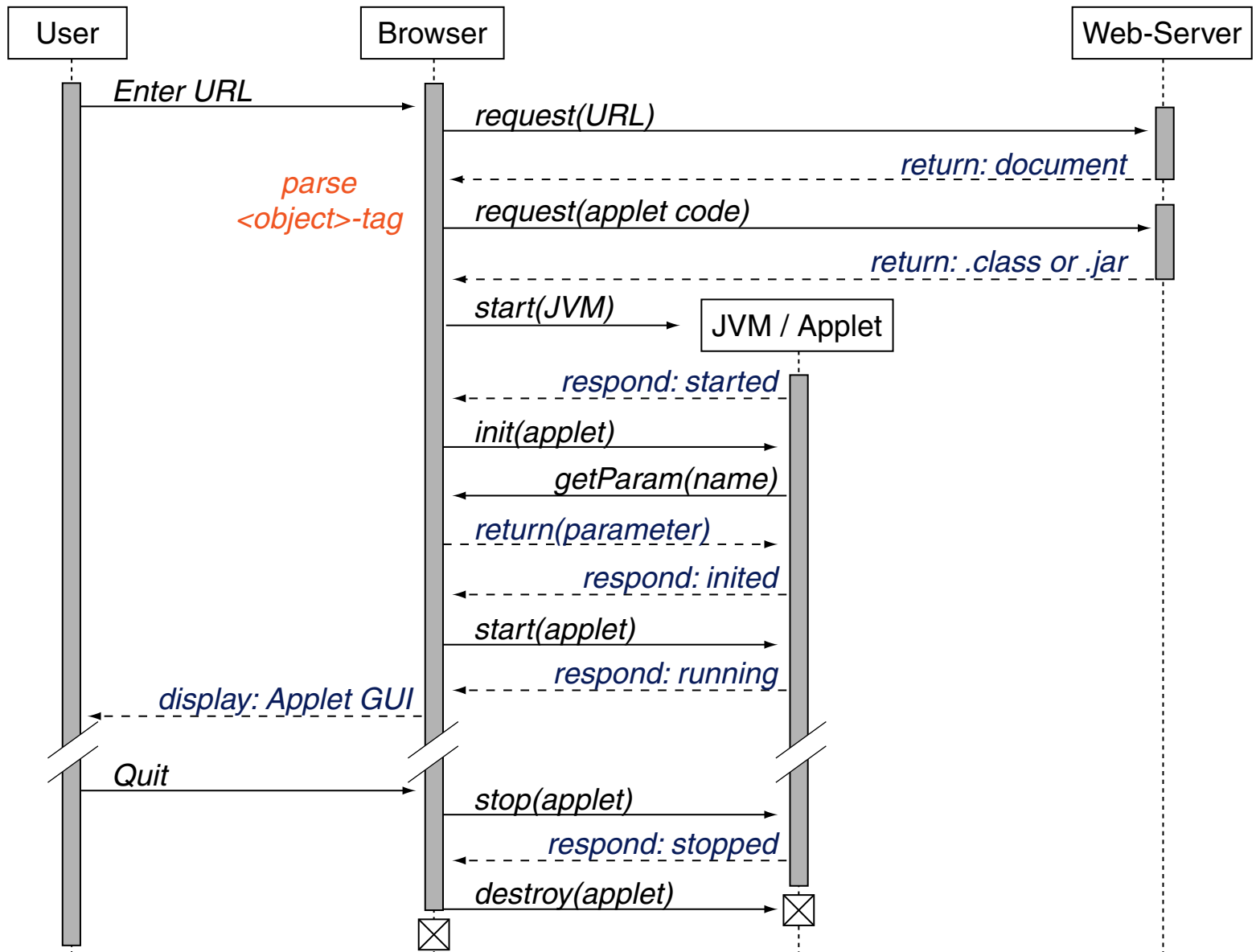


# Java Applet





# Java Applet



# Java Applet

## Beispiel: Anfordern von Web-Dokumenten

Applet Window Control Example - Mozilla Firefox

Alsearch ChatNoir Netspeak

human interaction

behavioural economics tea flavour human interaction

ROBOT PHYSICAL [6] INSTITUTE CMU [5] INTERFACE USER [2]

PROGRAMMING DEGREE [6]

INFORMATION TERM [3] ZON EDITION [2]

BOOKS - [4] MANAGEMENT PROCESS

VIRTUAL LAB [4] SIGCHI CONFERENCE [4]

☒ portr. priv ☒ portr. non-priv ☒ article ☒ shop ☒ link list ☒ help

de.aisearch. by benno stein and sven meyer zu eissen.

/ece/CCCE  
/Human\_Computer\_Interaction.pdf  
(Yahoo)

[35. HCII Home | Human-Computer Interaction Institute](#)

HCII Home | Human-Computer Interaction Institute  
Study how people design, implement and use interactive computer systems, and how ...  
CMU and SCS ceremonies are Sunday, May 17 2009 ...  
<http://www.hcii.cmu.edu/>  
(Yahoo)

[36. Human Interaction Research Institute](#)

Human Interaction Research Institute HIRI has helped nonprofits, communities and funders meet the challenges of ... The Human Interaction Research Institute (HIRI) helps nonprofits, funders and ...  
<http://www.humaninteract.org/>  
(Yahoo)

PROGRAMMING DEGREE

[37. Human Interactions in](#)

[Applet-Ausführung]

# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

Organisiert als Frameset in drei HTML-Dokumenten:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Applet Window Control Example</title>
  </head>

  <frameset cols="150,*" border="0" >
    <frame src="AppletWindowControl.html" name="control"/>
    <frame src="AppletWindowContent.html" name="content"/>

    <noframes>
      <body>
        <p>Your browser cannot display frames.</p>
      </body>
    </noframes>
  </frameset>
</html>
```

# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

### HTML-Dokument AppletWindowControl.html:

```
<!DOCTYPE html>
<html>
  <head> <title>Applet Window Control</title> </head>

  <body>
    <object type="application/x-java-applet" width="120" height="75" >
      <param name="code" value="applet.AppletWindowControl">
      <param name="codebase" value=".">
      <param name="archive"
        value="part-client-technologies-code-java.jar">
      <h1>Please install Java.</h1>
    </object>
  </body>
</html>
```

### HTML-Dokument AppletWindowContent.html:

```
<!DOCTYPE html>
<html>
  <head> <title>Applet Window Content</title> </head>

  <body>
    <p>Please make your choice on the left side.</p>
  </body>
</html>
```

# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

```
package applet;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.MalformedURLException;
import java.net.URL;

import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JPanel;

public class AppletWindowControl extends JApplet implements ActionListener {

    ...

    public void init() {...

        private JButton makeButton(String name) {...
            // Event handler.
            public void actionPerformed(ActionEvent event) {...
        }
    }
}
```

# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

```
package applet;

import java.awt.BorderLayout;...

public class AppletWindowControl extends JApplet implements ActionListener {

    JButton aisearchButton;
    JButton microsoftButton;
    JButton googleButton;

    String targetWindow = "content";
    String aisearchString = "AIsearch";
    String chatnoirString = "ChatNoir";
    String netspeakString = "Netspeak";

    URL aisearchURL = null;
    URL chatnoirURL = null;
    URL netspeakURL = null;

    public void init() {...

    private JButton makeButton(String name) {...
        // Event handler.
    public void actionPerformed(ActionEvent event) {...

}
```

# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

```
public void init() {  
  
    // Create a panel for the buttons.  
    JPanel buttonPanel = new JPanel();  
    buttonPanel.setLayout(new GridLayout(3, 1));  
  
    // Add the buttons to the panel.  
    aisearchButton = makeButton(aisearchString);  
    chatnoirButton = makeButton(chatnoirString);  
    netspeakButton = makeButton(netspeakString);  
    buttonPanel.add(aisearchButton);  
    buttonPanel.add(chatnoirButton);  
    buttonPanel.add(netspeakButton);  
  
    // Add the panel to the applet.  
    this.getContentPane().add(buttonPanel, BorderLayout.CENTER);  
  
    // Create URLs.  
    try {  
        aisearchURL = new URL("http://www.aisearch.de");  
        chatnoirURL = new URL("http://chatnoir.webis.de");  
        netspeakURL = new URL("http://www.netspeak.org");  
    } catch (MalformedURLException mue) {  
        System.out.println(mue.getMessage());  
    }  
}
```

# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

```
private JButton makeButton(String name) {

    JButton button = new JButton(name);
    button.addActionListener(this);
    return button;
}

// Event handler.
public void actionPerformed(ActionEvent event) {

    if (aisearchString.equals(event.getActionCommand())) {
        this.getAppletContext().showDocument(aisearchURL, targetWindow);
    }
    if (chatnoirString.equals(event.getActionCommand())) {
        this.getAppletContext().showDocument(chatnoirURL, targetWindow);
    }
    if (netspeakString.equals(event.getActionCommand())) {
        this.getAppletContext().showDocument(netspeakURL, targetWindow);
    }
}
```



# Java Applet

## Beispiel: Anfordern von Web-Dokumenten (Fortsetzung)

Die Methode `showDocument()` des Interfaces `AppletContext` [\[Javadoc\]](#):

```
public void showDocument(java.net.URL url)
```

```
public void showDocument(java.net.URL url, String targetWindow)
```

Optionen für *targetWindow*:

Option	Beschreibung
<i>WindowName</i>	Anzeige in dem (eventuell neu erzeugten) Fenster <i>WindowName</i> .
"_blank"	Anzeige in neuem, unbenanntem Fenster.
"_self"	Anzeige in dem Frame des Applet-Fensters.
"_parent"	Anzeige im Parent-Frame des Applet-Fensters.
"_top"	Anzeige im Top-Level-Frame des Applet-Fensters.

# Java Applet

## GUI-Programmierung

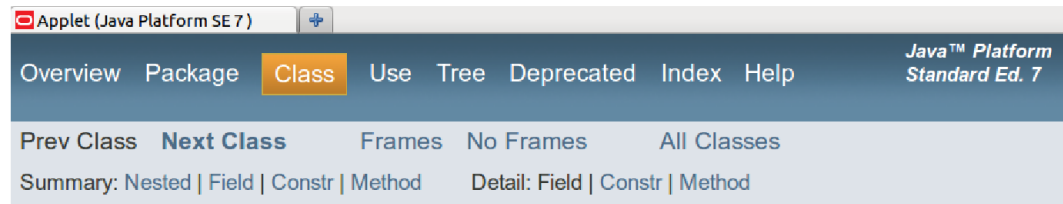
Applet-Programmierung heißt oft Oberflächenprogrammierung. Das JDK stellt verschiedene Klassen zur Realisierung von Benutzer-Interfaces bereit:

Java-Klasse	GUI-Element, Widget
java.awt.Button	Buttons
java.awt.Checkbox	Checkboxes
java.awt.TextField	einzeilige Textfelder
java.awt.TextArea	größere Textbereiche und Editierfelder
java.awt.Label	Labels
java.awt.List	Listen
java.awt.Choice	Pop-up- und Auswahllisten
java.awt.Scrollbar	Schieberegler und Scrollbars
java.awt.Canvas	Zeichenflächen
java.awt.Menu, java.awt.MenuItem, java.awt.CheckboxMenuItem	Menüs
java.awt.Panel, java.awt.Window	Container

# Java Applet

## GUI-Programmierung (Fortsetzung)

Die Vererbungshierarchie der Klasse Applet zeigt den starken grafischen Bezug der Applet-Programmierung [\[Javadoc\]](#) :



`java.applet`

### Class Applet

```
java.lang.Object
    java.awt.Component
        java.awt.Container
            java.awt.Panel
                java.applet.Applet
```

#### All Implemented Interfaces:

`ImageObserver`, `MenuContainer`, `Serializable`, `Accessible`

#### Direct Known Subclasses:

`JApplet`

---

```
public class Applet
    extends Panel
```

An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application.

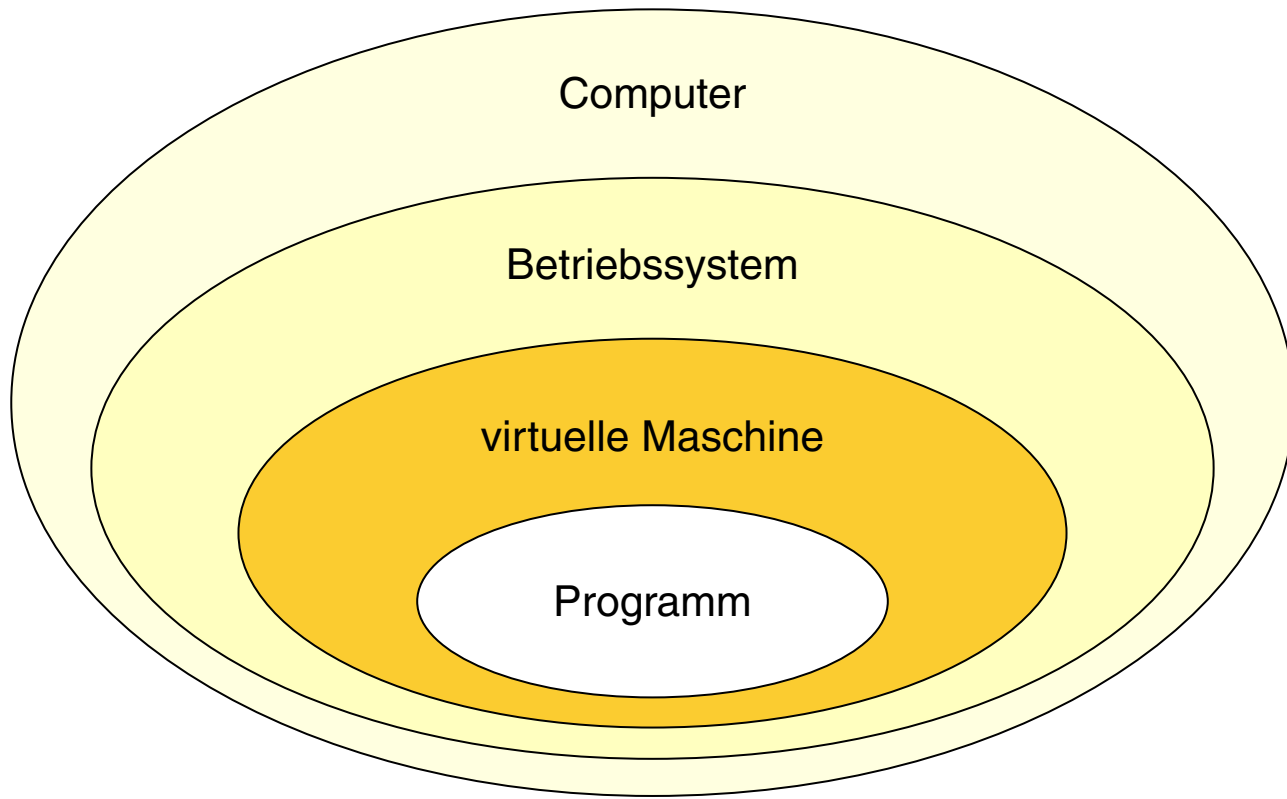
## Bemerkungen:

- ❑ Alle Swing-Komponenten sind verwendbar, wenn ein Applet von der Klasse `javax.swing.JApplet` anstatt von der Klasse `java.applet.Applet` erbt. [Javadoc: [1](#), [2](#)]

# Java Applet

## Sandbox-Prinzip

Die Java VM kapselt die Ausführung von Java-Programmen gegenüber dem Betriebssystem:



# Java Applet

## Sandbox-Prinzip (Fortsetzung)

### Beschränkungen von Applets:

- ❑ Systembibliotheken dürfen nicht geladen, „Native-Methoden“ nicht definiert werden.
- ❑ Netzwerk-Verbindungen zu beliebigen Hosts sind nicht erlaubt. Nur das Anfordern von Web-Dokumenten ist möglich.
- ❑ Auf dem Applet-**ausführenden** Host sind nicht erlaubt:
  - gewöhnliche Lese- und Schreibzugriffe
  - das Starten von Programmen
  - das Abfragen von Systemeigenschaften

# Java Applet

## Sandbox-Prinzip (Fortsetzung)

### Möglichkeiten von Applets:

- ❑ Zum Applet-ausliefernden Host (Web-Server) dürfen Netzwerkverbindungen initiiert werden.
- ❑ HTML-Dokumente dürfen von beliebigen Hosts angefordert werden.
- ❑ Mit `public` deklarierte Methoden anderer Applets derselben HTML-Seite dürfen aufgerufen werden.
- ❑ Auf eine `public` deklarierte Methode *method* eines Applets *applet* ist der Zugriff mit JavaScript möglich: `document.applet.method`
- ❑ Applets können weiterlaufen, auch wenn der Browser die zugehörige HTML-Seite verwirft.
- ❑ Applets, die nicht über das Web mit dem Browser-Plugin, sondern mit dem Java Runtime Environment (JRE) gestartet wurden, haben die Einschränkungen nicht.
- ❑ Der Anwender kann die **Beschränkungen für Applets aufheben**.

## Bemerkungen:

- ❑ Es bleibt dem Applet-ausliefernden Host natürlich vorbehalten, Netzwerkverbindungen von außen zu akzeptieren. Der entscheidende Punkt hier ist, *wo* die Restriktion der Verbindungserstellung auferlegt wird: durch die JVM, die das Applet ausführt, oder durch einen Host im Internet.
- ❑ Der Anwender gibt durch seine Zustimmung zur Aufhebung der Beschränkungen dem Applet die gleichen Rechte, die er auf seinem System besitzt. Die Gefahr dabei ist essentiell dieselbe wie beim Herunterladen ausführbarer Programme aus dem Netz, dem Öffnen von unbekannten E-Mail-Anhängen etc.



# Java Applet

## Signierung

Welche Grundvoraussetzung sollte erfüllt sein, damit ein Anwender einem Applet relativ gefahrlos mehr Rechte einräumen kann?

# Java Applet

## Signierung

Welche Grundvoraussetzung sollte erfüllt sein, damit ein Anwender einem Applet relativ gefahrlos mehr Rechte einräumen kann?

Bei der Übertragung eines Applets (allgemein: Nachricht) kann „viel passieren“: sie kann abgefangen, umgeleitet oder ausgetauscht werden.

Standardszenario aus der Kryptografie:



Wichtige Aspekte in diesem Zusammenhang sind:

- ❑ Vertraulichkeit: Geheimhaltung des Inhalts
- ❑ Integrität: Aufdeckung von Inhaltsveränderungen
- ❑ Authentizität: Garantie der Urheberschaft

# Java Applet

## Signierung (Fortsetzung)

Sei  $P$  die Menge aller Texte (*plain texts*),  $K$  die Menge aller Schlüssel (*keys*),  $C$  die Menge aller verschlüsselten Texte (*cipher texts*) und  $e_k, d_k$  zwei Funktionen:

$$\begin{array}{ll} e_k : P \rightarrow C & \\ d_k : C \rightarrow P & \end{array} \quad \text{mit} \quad d_k(e_k(x)) = x, \quad x \in P, k \in K$$

Protokoll einer symmetrischen Verschlüsselung:

1. Alice und Bob wählen einen gemeinsamen Schlüssel  $k \in K$ .
2. Alice versendet Nachricht  $x$  als  $y = e_k(x)$  zu Bob.
3. Bob entschlüsselt  $y$  und erhält  $x = d_k(y)$ .

# Java Applet

## Signierung (Fortsetzung)

Sei  $P$  die Menge aller Texte (*plain texts*),  $K$  die Menge aller Schlüssel (*keys*),  $C$  die Menge aller verschlüsselten Texte (*cipher texts*) und  $e_k, d_k$  zwei Funktionen:

$$\begin{array}{ll} e_k : P \rightarrow C & \\ d_k : C \rightarrow P & \end{array} \quad \text{mit} \quad d_k(e_k(x)) = x, \quad x \in P, k \in K$$

Idee der **asymmetrischen** Public-Key-Kryptografie: Alice und Bob haben je zwei Schlüssel  $k_1$  (öffentlich) und  $k_2$  (privat) mit  $d_{k_1}(e_{k_2}(x)) = d_{k_2}(e_{k_1}(x)) = x$ .

Protokoll einer asymmetrischen Verschlüsselung:

1. Alice und Bob wählen jeder für sich die Schlüssel  $k_1^{(A)}, k_2^{(A)}$  und  $k_1^{(B)}, k_2^{(B)}$ .
2. Beide veröffentlichen ihren Schlüssel  $k_1$ .
3. Alice versendet Nachricht  $x$  als  $y = e_{k_1}^{(B)}(x)$  zu Bob.
4. Bob entschlüsselt  $y$  und erhält  $x = d_{k_2}^{(B)}(y)$ .

## Bemerkungen:

- ❑ Bekannte Verfahren zur symmetrischen Verschlüsselung sind der Data Encryption Standard, DES, der Advanced Encryption Standard, AES, und der International Data Encryption Standard, IDEA.
- ❑ Ein bekanntes Verfahren zur asymmetrischen Verschlüsselung ist RSA, unter anderem implementiert in PGP und GnuPG.

# Java Applet

## Signierung (Fortsetzung)

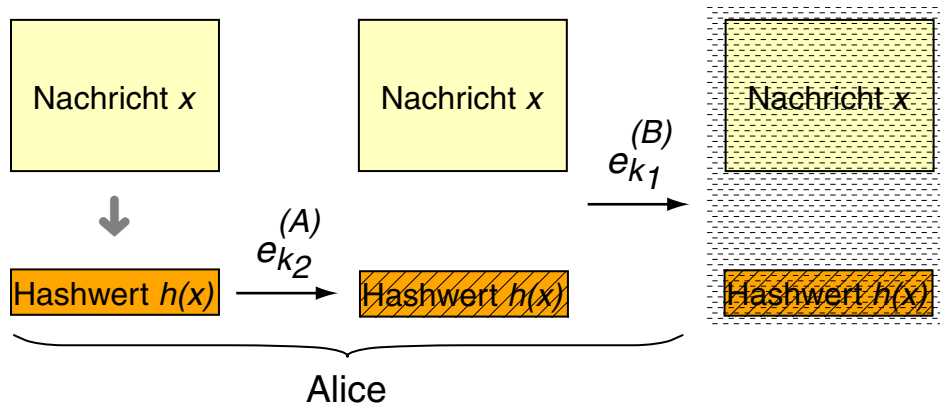
Woher weiß Bob, dass Alice und nicht Eve der Autor von Nachricht  $x$  ist?

# Java Applet

## Signierung (Fortsetzung)

Woher weiß Bob, dass Alice und nicht Eve der Autor von Nachricht  $x$  ist?

Sei  $h : P \rightarrow N$  eine Hashfunktion, die mit extrem hoher Wahrscheinlichkeit eine eindeutige Charakterisierung  $h(x)$  einer Nachricht  $x$  berechnet.



Protokoll zum digitalen Signieren:

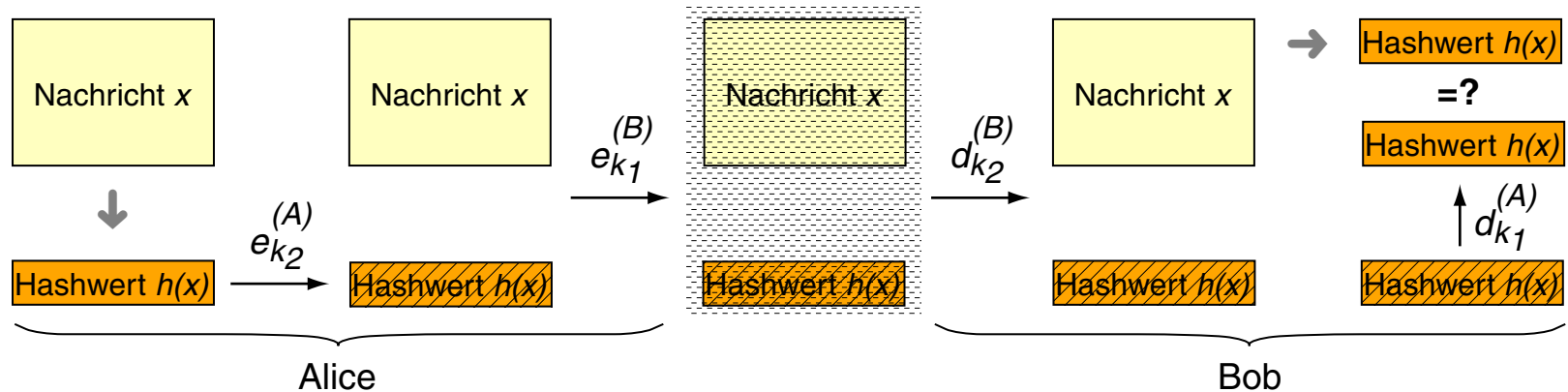
1. Alice berechnet für Nachricht  $x$  den Hashwert  $h(x)$ .
2. Alice verschlüsselt  $h(x)$  als  $y_h = e_{k_2}^{(A)}(h(x))$ .
3. Alice versendet Nachricht  $x + y_h$  als  $e_{k_1}^{(B)}(x + y_h)$  zu Bob.

# Java Applet

## Signierung (Fortsetzung)

Woher weiß Bob, dass Alice und nicht Eve der Autor von Nachricht  $x$  ist?

Sei  $h : P \rightarrow N$  eine Hashfunktion, die mit extrem hoher Wahrscheinlichkeit eine eindeutige Charakterisierung  $h(x)$  einer Nachricht  $x$  berechnet.



Protokoll zum Verifizieren:

1. Bob entschlüsselt mittels  $d_{k_2}^{(B)}$  die Nachricht und erhält  $x + y_h$ .
2. Bob berechnet für Nachricht  $x$  den Hashwert  $h(x)$ .
3. Bob berechnet  $d_{k_1}^{(A)}(y_h)$  und vergleicht den Wert mit  $h(x)$ .



## Bemerkungen:

- ❑ Der Vergleich von digitalen Signaturen mit realen Unterschriften ist gerechtfertigt, da Bob durch Abgleich der Signatur von Alice ihre Identität zu verifizieren sucht. Entscheidender Punkt bei digitalen Signaturen ist die Verhinderung der Trennung von Nachricht und Signatur.
- ❑ Falls Alice und Bob sich nicht kennen, kann digitale Signierung durch Zwischenschaltung eines gemeinsamen Vertrauensgebers, einer sogenannten Zertifizierungsinstanz, geschehen.
- ❑ Tutorial zur Code-Signierung in Java. [\[Oracle\]](#)

# Java Applet

## Quellen zum Nachlernen und Nachschlagen im Web

- ❑ Gosling, McGilton. *The Java Language Environment*.  
[www.oracle.com/technetwork/java/](http://www.oracle.com/technetwork/java/)
- ❑ Oracle. *Learning the Java Language*.  
[docs.oracle.com/javase/tutorial/java/](http://docs.oracle.com/javase/tutorial/java/)
- ❑ Oracle. *Signing Code and Granting It Permissions*.  
[docs.oracle.com/javase/tutorial/security/toolsign/](http://docs.oracle.com/javase/tutorial/security/toolsign/)
- ❑ Oracle. *Applets*.  
[docs.oracle.com/javase/tutorial/deployment/applet/](http://docs.oracle.com/javase/tutorial/deployment/applet/)
- ❑ Oracle. *How to Make Applets. (Swing)*  
[docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html](http://docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html)

# Weitere Client-Technologien

# Weitere Client-Technologien

## Java Web Start [\[Einordnung\]](#)

Idee: Realisierung bestimmter Aspekte reiner Web-basierter Software für beliebige Anwendungssoftware.

### Web-basierte Software:

- ❑ Download und Ausführung auf dem Client per Maus-Click
- ❑ Software zentral und immer aktuell auf dem Server
- ❑ keine Installation und keine Administrationsproblematik auf dem Client

### Java Web Start [\[Oracle\]](#) :

- ❑ Installation und Ausführung auf dem Client per Maus-Click
- ❑ bei Programmstart Kontaktierung des Servers und evtl. Aktualisierung
- ❑ skalierbare Ausführungsrechte: von Sandbox-gesichert bis unbeschränkt

## Bemerkungen:

- ❑ Entsprechende Technologien von Microsoft sind ActiveX und die Common Language Runtime, CLR.
- ❑ Das Microsoft-Konzept des Ladens und Ausführens von ActiveX-Komponenten ist deutlich weniger restriktiv als das Applet-Konzept.