

- Applets sind "kleinere" Java-Programme, die direkt in (gängigen) Internet-Browsern oder mit dem Applet-Viewer ausgeführt werden;
- benötigen keine `main`-Methode;
- müssen Unterklasse von `Applet` aus dem Paket `java.applet` sein;
- werden mittels eines `<APPLET>`-Tags in HTML-Seiten eingebunden oder mit dem Kommando
`appletviewer file.html`
betrachtet.

```
<HTML>
<HEAD>
<TITLE>An Applet Tester</TITLE>
</HEAD>
<BODY>
Here is the applet:
<br>
<APPLET code = "HelloInternet.class"
         width = "200" height = "200">
</APPLET>
<br>
End of applet...
</BODY>
</HTML>
```

Wichtige Methoden der Applet-Klasse

<code>init()</code>	automatisch ausgeführt nach dem Laden des Applets
<code>start()</code>	nach <code>init</code> , Reload oder Deiconfizierung des Viewers
<code>stop()</code>	beim Verlassen der Internet-Seite (auch bei einem Reload vor <code>start()</code>) oder Iconfizierung des Viewers
<code>destroy()</code>	beim Entfernen des Applets aus dem Viewer
<code>paint(Graphics g)</code>	zeichnet das Applet-Fenster im Graphik-Kontext <code>g</code> neu

```
public void paint (Graphics g) {  
    g.drawString("Hello Internet", 50, 50);  
}
```

Achtung: `Graphics` ist eine Klasse im Paket `java.awt`.

Die Interaktion erfolgt mit dem HTML-Dokument, nicht mit dem Nutzer. Dadurch kann aber der Effekt des Applets angepasst werden, ohne die .class-Datei zu verändern.

```
<APPLET CODE = "...\" WIDTH = "...\"  
                HEIGHT = "...\">  
<PARAM NAME = "...\" VALUE = "...\">  
</APPLET>
```

Das Einlesen erfolgt mit Hilfe der Methode `String getParameter(String)`.

wie Programme: ein Thread ist eine sequentielle Folge von Anweisungen

wie Prozesse: mehrere Threads des gleichen Programms können parallel zueinander laufen

anders als Prozesse:

- Threads des gleichen Programms benutzen gleichen Speicherbereich
(das kann zum Informationsaustausch genutzt werden)
- Zusammenspiel der Threads ist im Programm zu organisieren

- Threads sind Objekte der Klasse `java.lang.Thread`
- jedes Java-Programm ist implizit ein Thread

Daher kann auf alle statischen Methoden der Klasse `java.lang.Thread` zugegriffen werden.

Beispiel in beliebigen Programmen:

```
Thread.sleep(500);
```

- Programm läuft, bis alle Threads beendet sind

<code>void run()</code>	“Hauptprogramm” des Threads; enthält den eigentlichen Anweisungsteil, der nach Aktivierung des Thread ausgeführt wird.
<code>void start()</code>	startet den Thread mit Aufruf von <code>'run()'</code> (<code>'run()'</code> niemals direkt aufrufen!)
<code>Thread Thread.currentThread()</code>	liefert aktuellen Thread
<code>void Thread.sleep(long ms)</code>	lässt aktuellen Thread <code>'ms'</code> Millisekunden pausieren
<code>void join(Thread t)</code>	wartet auf Ende von Thread <code>'t'</code>
<code>void yield()</code>	lässt anderen Threads den Vortritt

Erzeugen von Threads

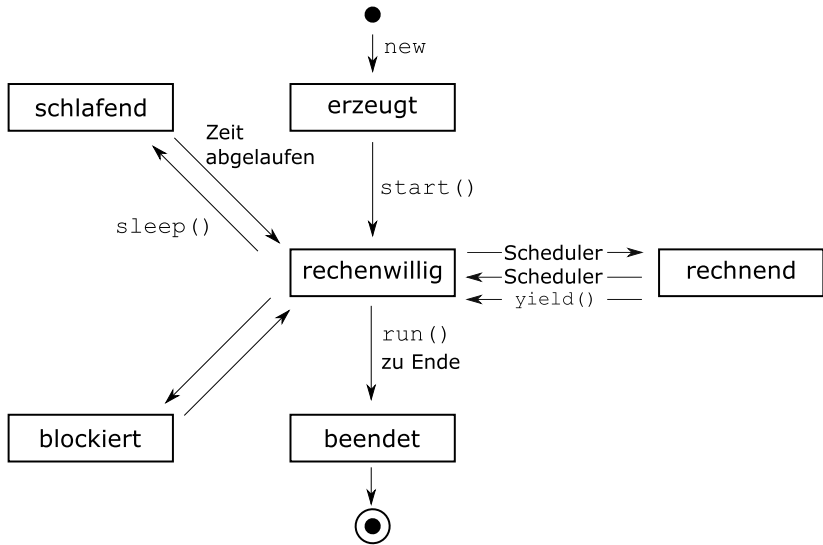
Zwei Wege, um Threads zu erzeugen:

1) Unterklasse von 'Thread' 2) Implementation von 'Runnable'

```
class T
  extends Thread {
  ...
  public void run() { ... }
  ...
}
class XYZ {
  ...
  T t = new T(...);
  t.start();
}
```

```
class R extends X
  implements Runnable {
  ...
  public void run() { ... }
  ...
}
class XYZ
  ...
  Thread t =
    new Thread(new R(...));
  t.start();
}
```


Zustände von Threads



Schlüsselwort 'synchronized' reserviert ein Objekt 'obj' für einen Thread bis der Anweisungsblock (geschützter Bereich) verlassen wird:

```
...  
synchronized (obj) { ... }  
...
```

Oder:

```
synchronized ResultType method(...) { ... }
```

ist Kurzform für

```
ResultType method(...) {  
    synchronized (this) { ... }  
}
```

Andere Threads, die 'obj' reservieren wollen, werden blockiert.

Wenn 'obj' reserviert ist, dann

obj.wait() löst Reservierung und blockiert aktuellen Thread

obj.notify() löst Blockierung des wartenden Threads

Beispiel:

```
synchronized (obj) {
    ...           // obj vorbereiten
    obj.wait();   // auf Ergebnis in obj warten
    ...           // Ergebnis weiterverarbeiten
}
...
synchronized (obj) {
    ...           // rechnen, Ergebnis in obj ablegen
    obj.notify(); // Fertigmeldung
}
```

synchronized-Blöcke vs. wait/notify

