

Einführung in die Informatik

Klassen

Konstruktoren, Methoden, Implementierung, Dateien

Wolfram Burgard

Erzeugen von Objekten

- Jede Klasse hat wenigstens eine Methode zum Erzeugen von Objekten.
- Solche Methoden heißen **Konstruktoren**.
- Der **Name eines Konstruktors** stimmt stets mit dem der Klasse überein.
- Wie andere Methoden auch, können **Konstruktoren Argumente** haben.
- Da wir mit Konstruktoren neue Objekte erzeugen wollen, können wir sie an kein Objekt senden.
- In Java verwenden wir das **Schlüsselwort new**, um einen **Konstruktor aufzurufen**:

```
new String("hello world")
```

- Dies **erzeugt** ein `String`-Objekt und sendet ihm die Nachricht

```
String("hello world").
```

Die Operation new

- Das **Schlüsselwort** `new` bezeichnet eine **Operation**, die einen Wert zurückgibt.
- Der **Return-Wert** einer `new`-Operation ist die **Referenz auf das neu erzeugte Objekt**.
- Wir nennen `new` einen **Operator**.

Sichern neu erzeugter Objekte

- Der `new-Operator` liefert uns eine Referenz auf ein neues Objekt.
- Um dieses Objekt im Programm verwenden zu können, müssen wir die `Referenz in einer Referenzvariablen sichern`.

Beispiel:

```
String s, t, upper, lower;  
s = new String("Hello");  
t = new String(); // identisch mit ""  
  
upper = s.toUpperCase();  
lower = s.toLowerCase();  
  
System.out.println(s);
```

Dateien

- Bisher gingen alle Ausgaben nach **Standard output**, d.h. auf den **Monitor**.
- Der Vorteil von **Dateien** ist die **Persistenz**, d.h. die **Information bleibt dauerhaft erhalten**.

Grundlegende Eigenschaften von Dateien:

Dateiname: Üblicherweise setzen sich Dateinamen aus Zeichenketten zusammen.

Inhalt (Daten) : Die Daten können beliebige Informationen sein: Brief, Einkaufsliste, Adressen, . . .

Grundlegende Datei-Operationen

- Erzeugen einer Datei
- In eine Datei schreiben.
- Aus einer Datei lesen.
- Eine Datei löschen.
- Den Dateinamen ändern.
- Die Datei überschreiben, d.h. nur den Inhalt verändern.

Die File-Klasse

- Java stellt eine vordefinierte Klasse `File` zur Verfügung
- Der **Konstruktor** für `File` nimmt als Argument den **Dateinamen**.

Beispiel:

```
File f1, f2;  
f1 = new File("letterToJoanna");  
f2 = new File("letterToMatthew");
```

Hinweis:

Wenn ein File-Objekt erzeugt wird, bedeutet das nicht, dass gleichzeitig auch die Datei erzeugt wird.

Dateien Umbenennen und Löschen

- Existierende Dateien können in Java mit `renameTo` umbenannt werden.
- Mit der Methode `delete` können vorhandene Dateien gelöscht werden.

Prototypen:

Methodenname	Return-Wert	Argumente	Aktion
<code>delete</code>	<code>void</code>	keine	löscht die Datei
<code>renameTo</code>	<code>void</code>	<code>File</code> -Objekt-Referenz	nennt die Datei um

Ausgabe in Dateien

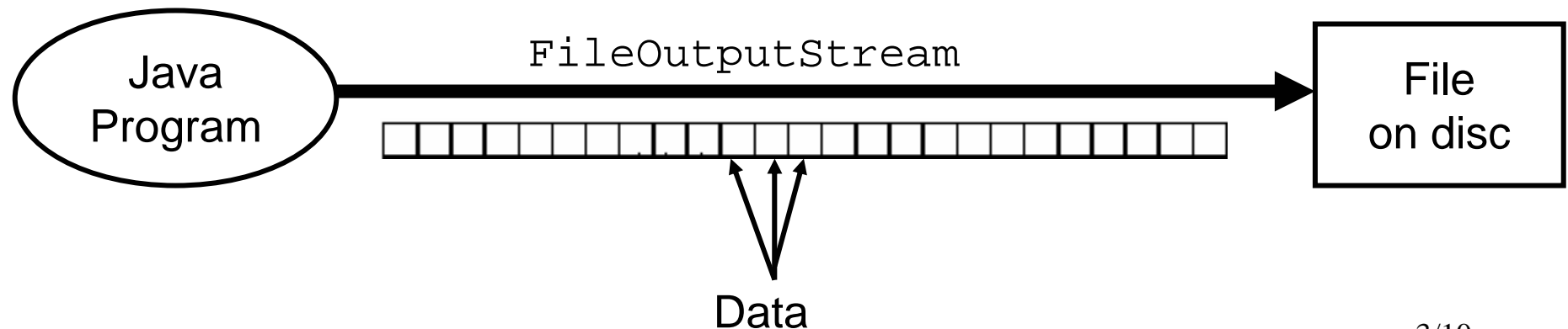
- In Java verwenden wir so genannte **(Ausgabe-) Ströme** bzw. **(Output-)Streams**, um Dateien mit Inhalt zu füllen.
- Die Klasse `FileOutputStream` stellt einen solchen Strom zur Verfügung.
- Der **Konstruktor** von `FileOutputStream` akzeptiert als Argument eine **Referenz auf ein File-Objekt**.
- Die Datei mit dem durch das Argument gegebenen Namen wird geöffnet.
- Ist die Datei nicht vorhanden, so wird sie erzeugt.
- Ist die Datei vorhanden, wird ihr Inhalt gelöscht.

Beispiel:

```
File f = new File("Americas.Most.Wanted");  
FileOutputStream fs = new FileOutputStream(f);
```

Wirkung von `FileOutputStream`

- `FileOutputStream` modelliert die Ausgabe als eine Sequenz von kleinen, uninterpretierten Einheiten bzw. Bytes.
- Sie stellt keine Möglichkeit zur Verfügung, die Daten zu gruppieren.
- Methoden wie `println` zum Ausgeben von Zeilen werden nicht zur Verfügung gestellt.



PrintStream-Objekte

- Um **Ausgaben auf dem Monitor** zu erzeugen, haben wir bisher die Nachrichten `println` oder `print` an das `System.out`-Objekt geschickt.
- Dabei ist `System.out` eine **Referenz auf eine Instanz der Klasse `PrintStream`**.
- Um in eine **Datei** zu **schreiben, erzeugen** wir ein **`PrintStream`-Objekt**, welches die **Datei repräsentiert**.
- **Darauf wenden wir** dann die Methoden `println` oder `print` **an**.

Erzeugen von `PrintStream`-Objekten

Der **Konstruktor von `PrintStream`** akzeptiert eine Referenz auf einen `FileOutputStream`

```
File          diskFile = new File("data.out");
FileOutputStream diskFileStream =
              new FileOutputStream(diskFile);
PrintStream    target =
              new PrintStream(diskFileStream);

target.println("Hello Disk File");
```

Dieser Code erzeugt eine Datei `data.out` mit dem Inhalt

```
Hello Disk File.
```

Eine evtl. existierende Datei mit gleichem Namen wird gelöscht und ihr Inhalt wird überschrieben.

Notwendige Schritte, um in eine Datei zu schreiben

1. Erzeugen eines `File`-Objektes
2. Erzeugen eines `FileOutputStream`-Objektes unter Verwendung des soeben erzeugten `File`-Objektes.
3. Erzeugen eines `PrintStream`-Objektes mithilfe der Referenz auf das `FileOutputStream`-Objekt.
4. Verwenden von `print` oder `println`, um Texte in die Datei auszugeben.

Kompakte Erzeugung von `PrintStream`-Objekten für Dateien

Die Konstruktion der `PrintStream`-Objekte kann auch ohne die `diskFileStream`-Variable durch **Schachteln von Aufrufen** erreicht werden:

```
import java.io.*;

class ProgramFileCompact {
    public static void main(String[] arg) throws IOException{
        String    fileName = new String("data1.out");
        PrintStream target = new PrintStream(new
            FileOutputStream(new File(fileName)));
        target.print("Hello disk file ");
        target.println(fileName);
    }
}
```

Beispiel: Backup der Ausgabe in einer Datei

```
import java.io.*;
class Program1Backup {
    public static void main(String arg[]) throws IOException {
        PrintStream backup;
        FileOutputStream backupFileStream;
        File backupFile;

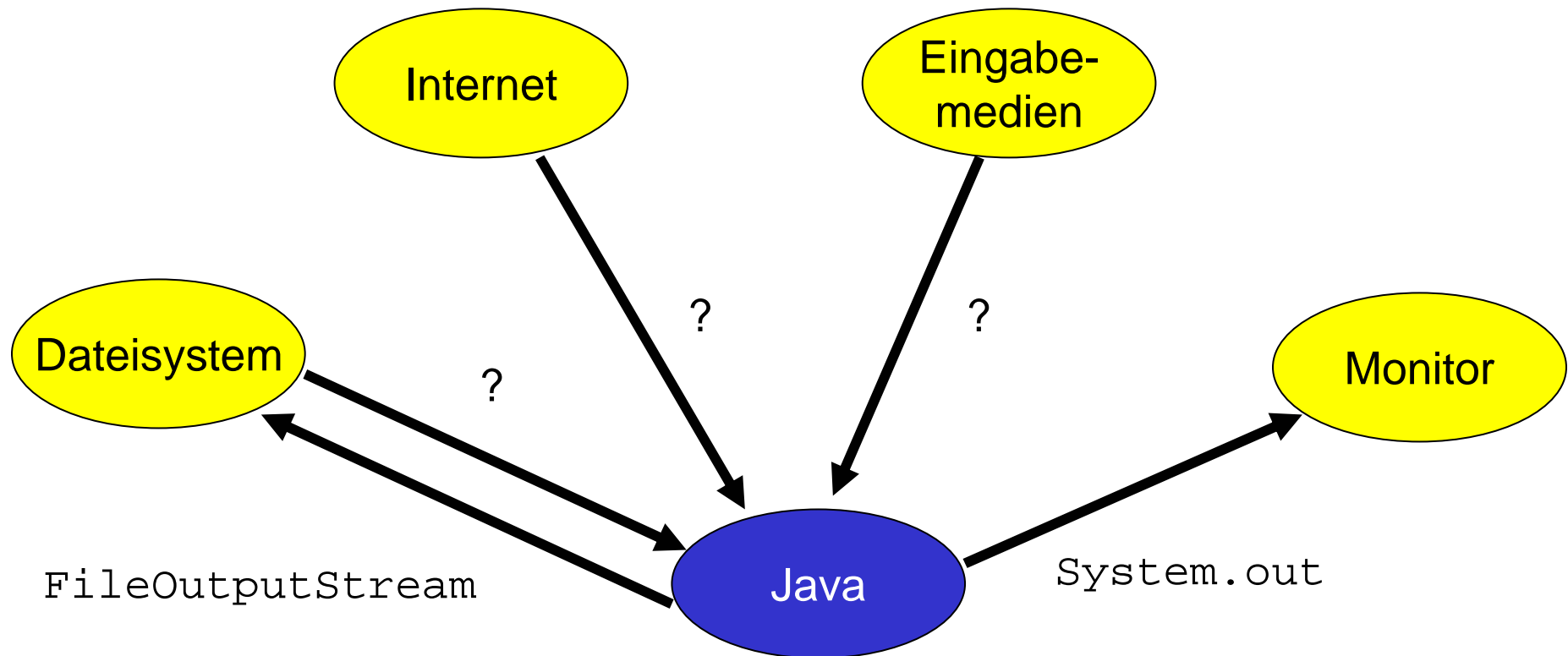
        backupFile = new File("backup");
        backupFileStream = new FileOutputStream(backupFile);
        backup = new PrintStream(backupFileStream);

        System.out.println("This is my first Java program");
        backup.println("This is my first Java program");
        System.out.println("... but it won't be my last.");
        backup.println("... but it won't be my last.");
    }
}
```

Mögliche Fehler

- Das **Erzeugen einer Datei** stellt eine **Interaktion mit externen Komponenten** dar (z.B. Betriebssystem, Hardware etc.)
- Dabei können **Fehler** auftreten, die **nicht durch das Programm selbst verschuldet** sind.
- Beispielsweise kann die **Festplatte voll** sein oder sie kann einen **Schreibfehler** haben. Weiter kann das **Verzeichnis**, in dem das Programm ausgeführt wird, **schreibgeschützt** sein.
- In solchen Fällen wird das einen Fehler produzieren.
- **Java erwartet, dass der Programmierer mögliche Fehler explizit erwähnt.**
- Dazu wird die Phrase `throws Exception` verwendet.

Mögliche Ein- und Ausgabequellen in Java



Eingabe: Ein typisches Verfahren

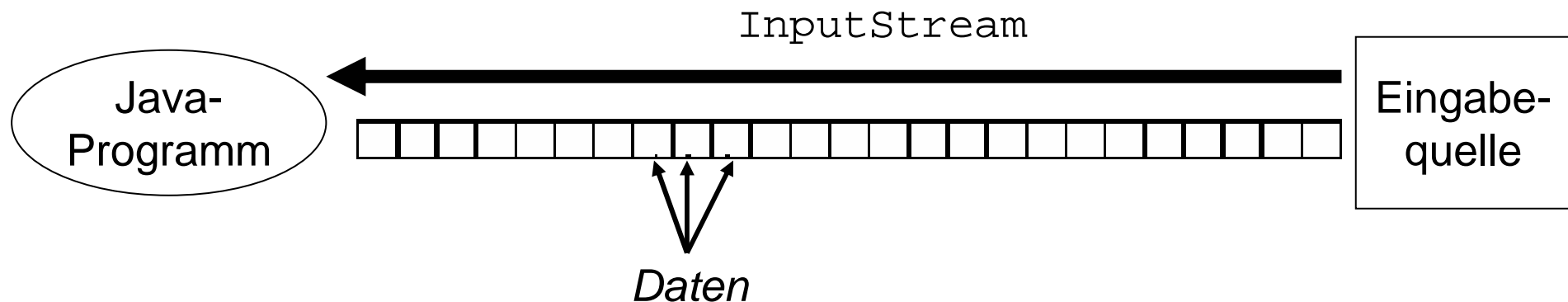
Um Eingaben von einem Eingabestrom verarbeiten zu können, müssen folgende Schritte durchgeführt werden.

1. Erzeugen Sie ein `InputStream`-Objekt, ein `FileInputStream`-Objekt oder verwenden Sie das `System.in`-Objekt.
2. Verwenden Sie dieses Eingabestrom-Objekt, um einen `InputStreamReader`-Objekt zu erzeugen.
3. Erzeugen Sie ein `BufferedReader`-Objekt mithilfe des `InputStreamReader`-Objektes.

Dabei wird `FileInputStream` für das **Einlesen aus Dateien**, `InputStream` für das **Einlesen aus dem Internet** und `System.in` für die **Eingabe von der Tastatur** und verwendet.

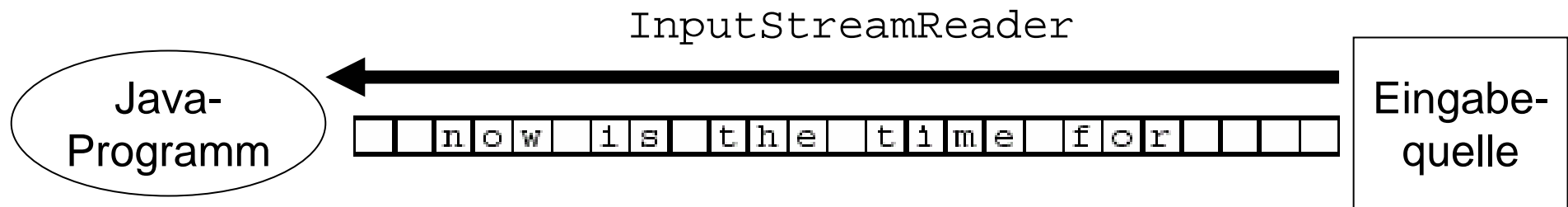
Wirkung eines InputStream-Objektes

InputStream-Objekte, FileInputStream-Objekte oder das System.in-Objekt modellieren die Eingabe als eine **kontinuierliche, zusammenhängende Sequenz kleiner Einheiten**, d.h. als eine **Folge von Bytes**:



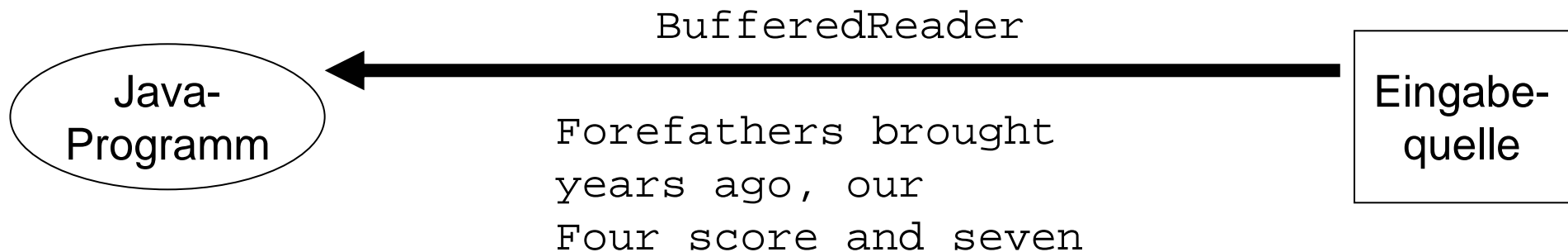
Wirkung eines `InputStreamReader`-Objektes

`InputStreamReader`-Objekte hingegen modellieren die Eingabe als eine **Folge von Zeichen**, sodass daraus **Zeichenketten** zusammengesetzt werden können:



BufferedReader

`BufferedReader`-Objekte schließlich modellieren die Eingabe als eine **Folge von Zeilen**, die einzeln durch **String-Objekte** repräsentiert werden können:



Eingabe vom Keyboard

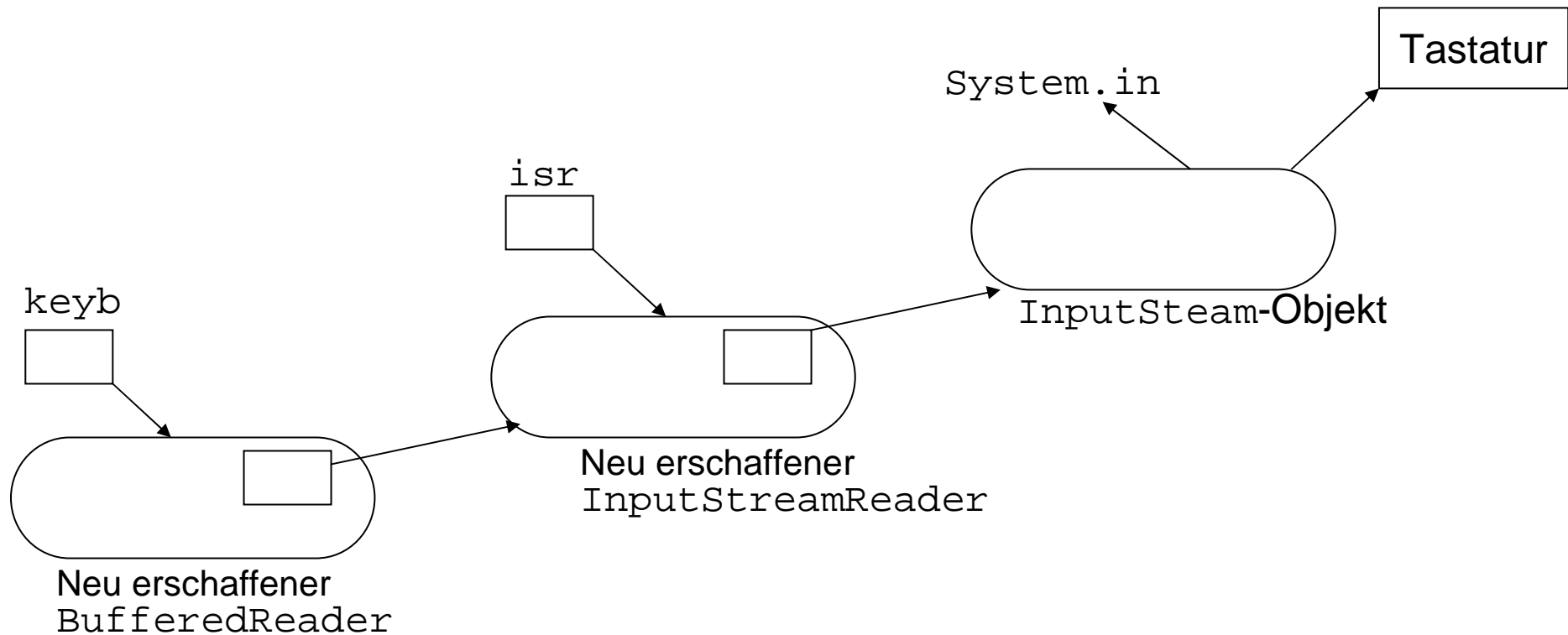
- Java stellt ein vordefiniertes `InputStream`-Objekt zur Verfügung, das die Eingabe von der Tastatur repräsentiert. `System.in` ist eine Referenz auf dieses Objekt.
- Allerdings kann man von `System.in` nicht direkt lesen.
- Vorgehen:

```
InputStreamReader isr;  
BufferedReader keyb;  
isr = new InputStreamReader(System.in)  
keyb = new BufferedReader(isr);
```

Das Einlesen geschieht dann mit:

```
keyb.readLine()
```

Schema für die Eingabe von der Tastatur mit Buffer



Beispiel: Einlesen einer Zeile von der Tastatur

Naives Verfahren zur Ausgabe des Plurals eines Wortes:

```
import java.io.*;
class Program4 {
    public static void main(String arg[]) throws IOException {
        InputStreamReader isr;
        BufferedReader keyboard;
        String inputLine;

        isr = new InputStreamReader(System.in);
        keyboard = new BufferedReader(isr);
        inputLine = keyboard.readLine();

        System.out.print(inputLine);
        System.out.println("s");
    }
}
```


Interaktive Programme

- Um den Benutzer auf eine notwendige Eingabe hinzuweisen, können wir einen so genannten **Prompt** ausgeben.
- `PrintStream` verwendet einen **Buffer**, um **Ausgabeaufträge zu sammeln**. Die Ausgabe erfolgt erst, wenn der Buffer voll oder das Programm beendet ist.
- Da dies eventuell erst nach der Eingabe sein kann, stellt die `PrintStream`-Klasse eine Methode **flush** zur Verfügung. Diese **erzwingt die Ausgabe des Buffers**.
- Vorgehen daher:

```
System.out.println(  
    "Type in a word to be pluralized, please ");  
System.out.flush();  
inputLine = keyboard.readLine();
```

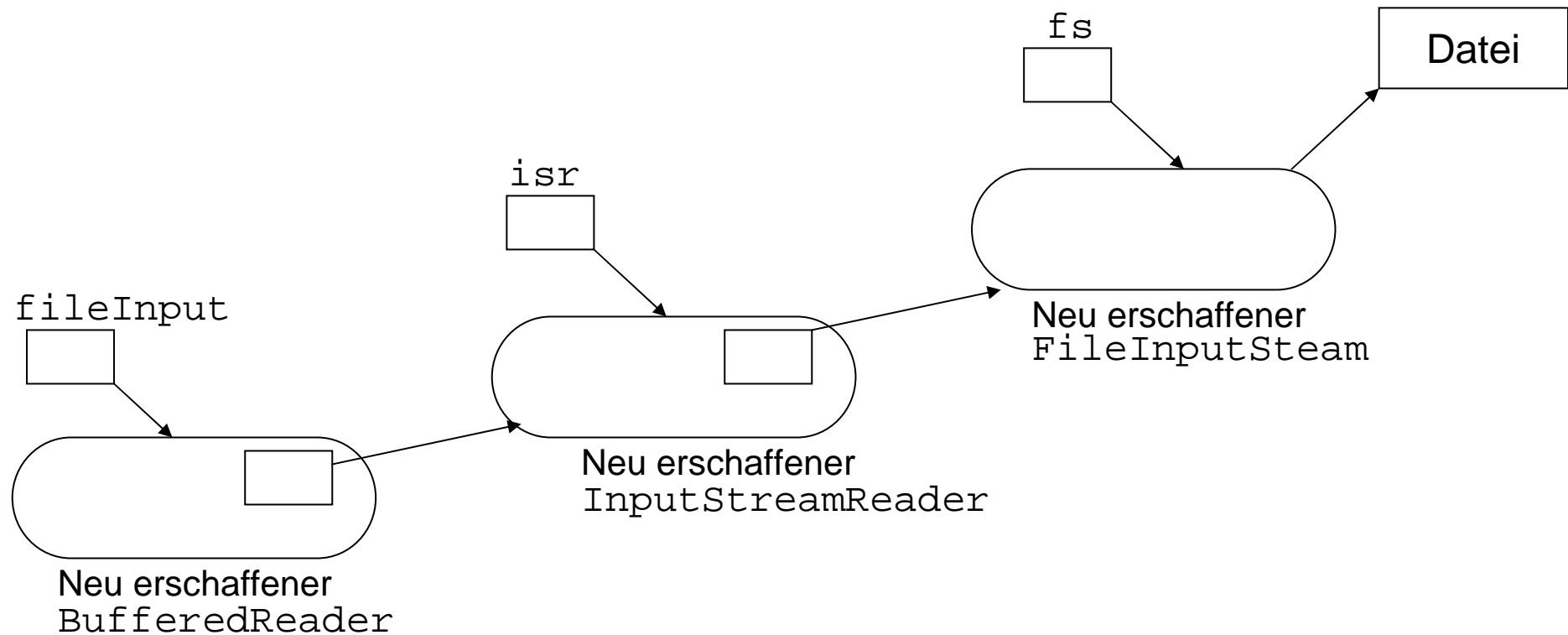
Input aus Dateien

Das **Lesen aus einer Datei** unterscheidet sich vom Lesen von der Tastatur nur dadurch, dass wir ein `FileInputStream`-Objekt und nicht das `System.in`-Objekt verwenden:

```
// Vom Dateinamen zum FileInputStream
File f = new File("Americas.Most.Wanted");
FileInputStream fs = new FileInputStream(f);

// Vom FileInputStream zum BufferedReader
InputStreamReader isr;
BufferedReader fileInput;
isr = new InputStreamReader(fs);
fileInput = new BufferedReader(isr);
```

Einlesen aus Dateien mit Buffer



Einlesen einer Zeile aus einer Datei

```
import java.io.*;
class Program5 {
    public static void main(String arg[]) throws IOException {
        String inputLine;
        // Vom Dateinamen zum FileInputStream
        File f = new File("Americas.Most.Wanted");
        FileInputStream fs = new FileInputStream(f);

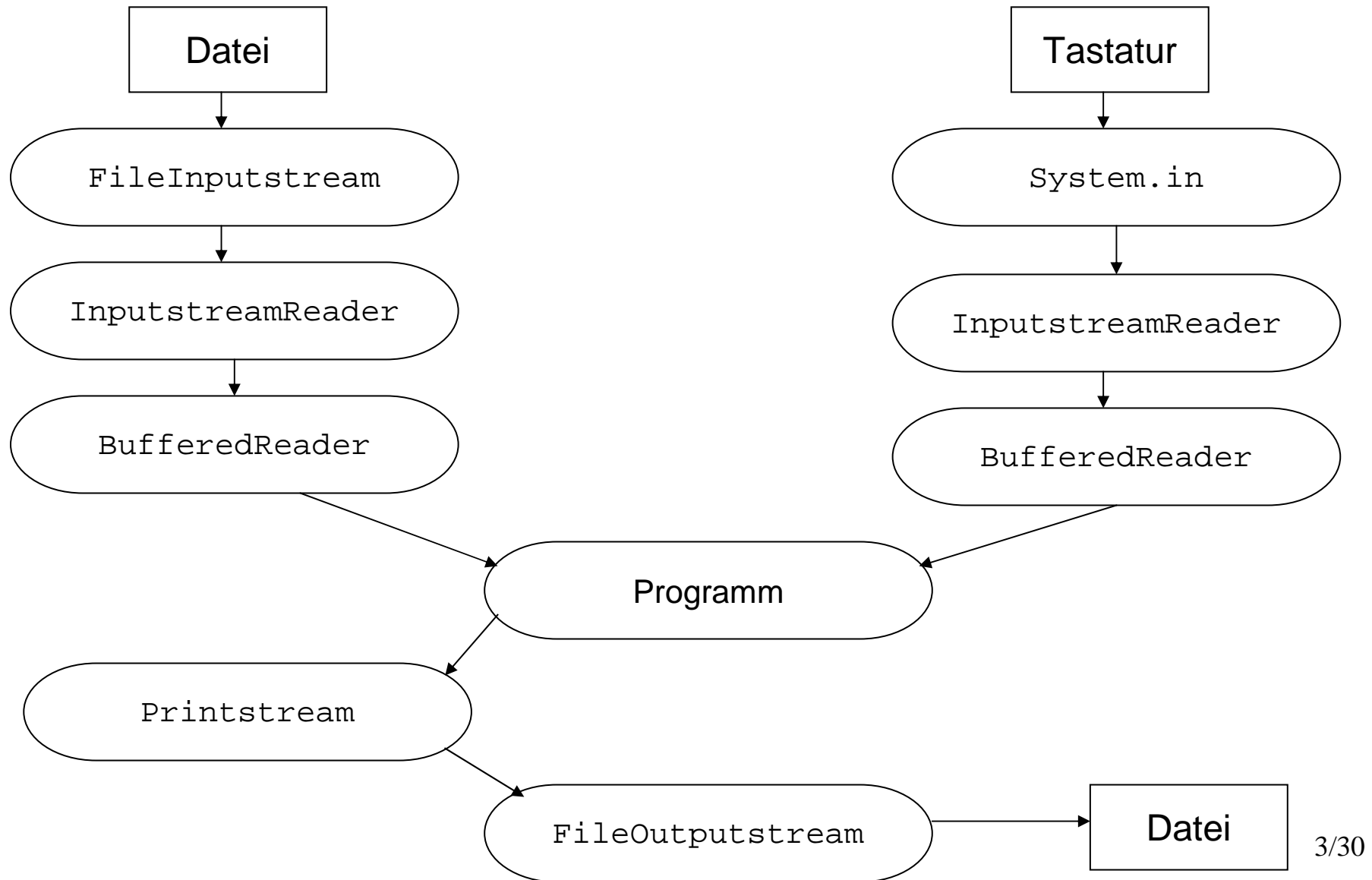
        // Vom FileInputStream zum BufferedReader
        InputStreamReader isr;
        BufferedReader fileInput;
        isr = new InputStreamReader(fs);
        fileInput = new BufferedReader(isr);

        inputLine = fileInput.readLine();
        System.out.println(inputLine);
    }
}
```

Gleichzeitige Verwendung mehrerer Streams: Kopieren einer Datei

1. Frage nach Quelldatei (und Zieldatei).
2. Lies Quelldatei.
3. Schreibe Zieldatei.

Schematische Darstellung



Daten aus dem Internet einlesen

Computer-Netzwerk: Gruppe von Computern, die untereinander direkt Informationen austauschen können (z.B. durch eine geeignete Verkabelung).

Internet: Gruppe von Computer-Netzwerken, die es Rechnern aus einem Netz erlaubt, mit Computern aus dem anderen Netz zu kommunizieren.

Internet-Adresse: Eindeutige Adresse, mit deren Hilfe jeder Rechner im Netz eindeutig identifiziert werden kann. Beispiele:

`www.informatik.uni-freiburg.de`

`www.uni-freiburg.de`

`www.whitehouse.gov`

Netzwerk-Ressource: Einheit von Informationen wie z.B. Text, Bilder, Sounds etc.

URL: (Abk. für Universal Resource Locator)
Eindeutige Adresse von Netzwerk-Ressourcen.

Komponenten einer URL

Bestandteil	Beispiel	Zweck
Protokoll	http	Legt die Software fest, die für den Zugriff auf die Daten benötigt wird
Internet-Adresse	www.yahoo.com	Identifiziert den Computer , auf dem die Ressource liegt
Dateiname	index.html	Definiert die Datei mit der Ressource

Zusammengesetzt werden diese Komponenten wie folgt:

`protocol://internet address/file name`

Beispiel:

`http://www.yahoo.com/index.html`

Netzwerk-Input

- Um Daten aus dem Netzwerk einzulesen, verwenden wir ein `InputStream`-Objekt.
- Die `Java-Klassenbibliothek` stellt eine Klasse `URL` zur Verfügung, um URL's zu modellieren.
- Die `URL`-Klasse stellt einen Konstruktor mit einem String-Argument zur Verfügung:

```
URL u = new URL("http://www.yahoo.com/");
```

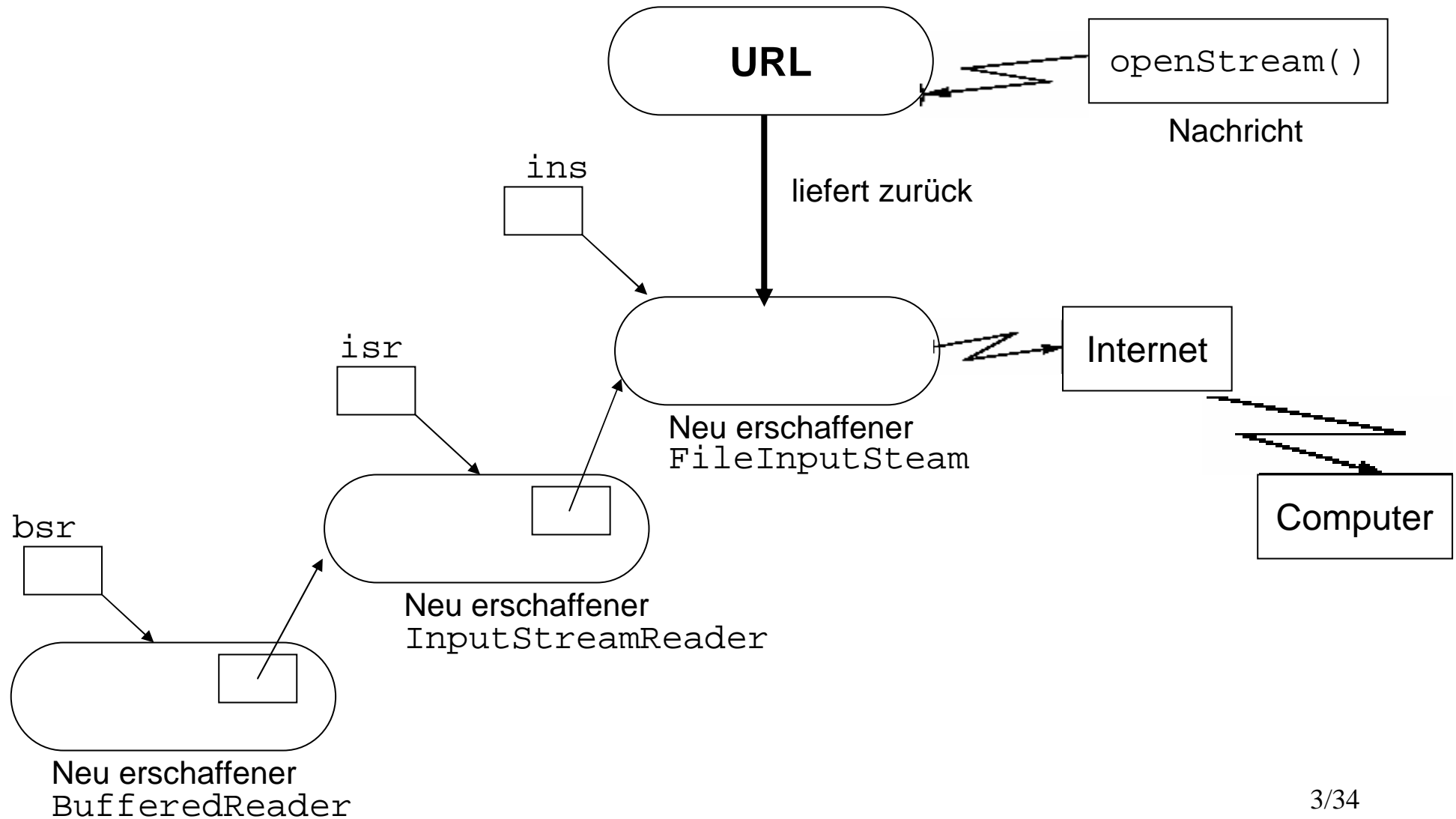
- Weiterhin stellt sie eine Methode `openStream` bereit, die keine Argumente hat und ein `InputStream`-Objekt zurückgibt:

```
InputStream ins = u.openStream();
```

- Sobald wir einen `InputStream` haben, können wir wie üblich fortfahren:

```
InputStreamReader isr = new InputStreamReader(ins);  
BufferedReader remote = new BufferedReader(isr);  
... remote.readLine() ...
```

Einlesen aus dem Internet mit Buffer



Beispiel: Einlesen der ersten fünf Zeilen von `www.informatik.uni-freiburg.de`

```
import java.net.*;
import java.io.*;

class WebPageRead {
    public static void main(String[] arg) throws Exception {
        URL u = new URL("http://www.informatik.uni-freiburg.de/");
        InputStream ins = u.openStream();
        InputStreamReader isr = new InputStreamReader(ins);
        BufferedReader webPage = new BufferedReader(isr);

        System.out.println(webPage.readLine());
        System.out.println(webPage.readLine());
        System.out.println(webPage.readLine());
        System.out.println(webPage.readLine());
        System.out.println(webPage.readLine());
    }
}
```

Ergebnis der Ausführung

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="author" content="www@informatik.uni-freiburg.de">
```

Die Titelseite der Informatik in Freiburg

The screenshot shows a Mozilla browser window displaying the website of the Institute of Informatics at the University of Freiburg. The browser's address bar shows the URL <http://www.informatik.uni-freiburg.de/>. The page content includes a photograph of a modern building, the text 'Institut für Informatik', and a navigation menu with links for 'Abteilungen', 'Veranstaltungen', 'Informationen', and 'Projekte'. Below this, there is a section titled 'Abteilungen' with a grid of departmental links and their respective professors.

Abteilungen <small>[top]</small>		
Algorithmen und Datenstrukturen Prof. Dr. Thomas Ottmann	Rechnerarchitektur Prof. Dr. Bernd Becker	Softwaretechnik Prof. Dr. David Basin
Programmiersprachen Prof. Dr. Peter Thiemann	Grundlagen der Künstlichen Intelligenz Prof. Dr. Bernhard Nebel	Maschinelles Lernen und Natürlichsprachliche Systeme Prof. Dr. Luc De Raedt
Mustererkennung und Bildverarbeitung Prof. Dr. Hans Burkhardt	Datenbanken und Informationssysteme Prof. Dr. Georg Lausen	Paralleles und Verteiltes Rechnen Prof. Dr. Susanne Albers
Autonome Intelligente Systeme Prof. Dr. Wolfram Burgard	Rechnernetze und Telematik Prof. Dr. Stefan Leue	Kommunikationssysteme Prof. Dr. Gerhard Schneider

Der Quellcode der Titelseite

```
Source of: http://www.informatik.uni-freiburg.de/ - Mozilla
File Edit View Help
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <meta name="author" content="www@informatik.uni-freiburg.de">
  <meta name="keywords" content="Institut f&uuml;r Informatik Freiburg">
  <meta name="generator" content="WML">
  <meta name="description" content="Homepage des Instituts für Informatik der Fakultät für
  <meta name="GENERATOR" content="Mozilla/4.61 [en] (X11; I; Linux 2.2.10 i686) [Netscape]
  <title> Institut f&uuml;r Informatik
</title>
</head>

  <body text="#000000" bgcolor="#ffffff">
  <a name="top"></a>
  <table border="0" width="100%">
    <tbody>
      <tr>
        <td valign="Top">

  <td ALIGN=RIGHT VALIGN=BOTTOM ROWSPAN="2"><a href="http://www.uni-freiburg.de/"><img SRC="i
  <br><font size=-1><a href="http://www.faw.uni-freiburg.de/">Fakult&auml;t
  f&uuml;r Angewandte Wissenschaften </a>, <a href="http://www.uni-freiburg.de/">Universit&au
  Freiburg&nbsp;</a></font></td>
</tr>

  <tr>
  <td VALIGN=BOTTOM><img SRC="images/transplx1.gif" ALT="" BORDER=0 ><font face="Helvetica, Ar
  Institut f&uuml;r Informatik&nbsp;</font></font></font></td>
</tr>
</tbody>
</table>

  <hr noshade size="1">
  <table CELLSPACING=0 CELLPADDING=0 WIDTH="100%" >
  <tr>
  <td ALIGN=LEFT><img SRC="images/transplx1.gif" ALT="" height=19 width=37></td>
```

Zusammenfassung

- Neue Objekte einer Klasse können mit dem **new-Operator** erzeugt werden.
- Zusammen mit dem `new`-Operator verwenden wir den **Konstruktor**, der den gleichen Bezeichner hat wie die Klasse selbst.
- Häufig müssen **mehrere Objekte** erzeugt werden, um ein bestimmtes Verhalten zu erreichen.
- Um beispielsweise Zeilen aus dem Internet einzulesen, benötigen wir ein `BufferedReader`-Objekt.
- Dies erfordert das Erzeugen eines `InputStreamReader`-Objektes
- Das `InputStreamReader`-Objekt hingegen benötigt ein entsprechendes `InputStream`-Objekt.

Add-on: Programme als Applets im Web-Browser

- Ein **Applet** ist ein Programm, welches in eine Web-Page integriert ist.
- Java stellt eine **Abstract-Window-Toolkit**-Klasse (**AWT**) zur Verfügung, um grafische Benutzeroberflächen zu programmieren.

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

public class FirstApplet extends Applet {
    public void paint(Graphics g) {
        Color c = new Color(20,120,160);
        g.setColor(c);
        g.fillOval(20,20,60,30);
    }
}
```


Eine einfache Web-Page mit Applet

```
<HTML>
<HEAD>
  <TITLE>Hw1</TITLE>
</HEAD>
<BODY>
  <HR>
  <APPLET CODE="FirstApplet.class" WIDTH=300 HEIGHT=60></APPLET>
  <HR>
  <A HREF="FirstApplet.java">The source.</A>
</BODY>
</HTML>
```

Das Ergebnis

