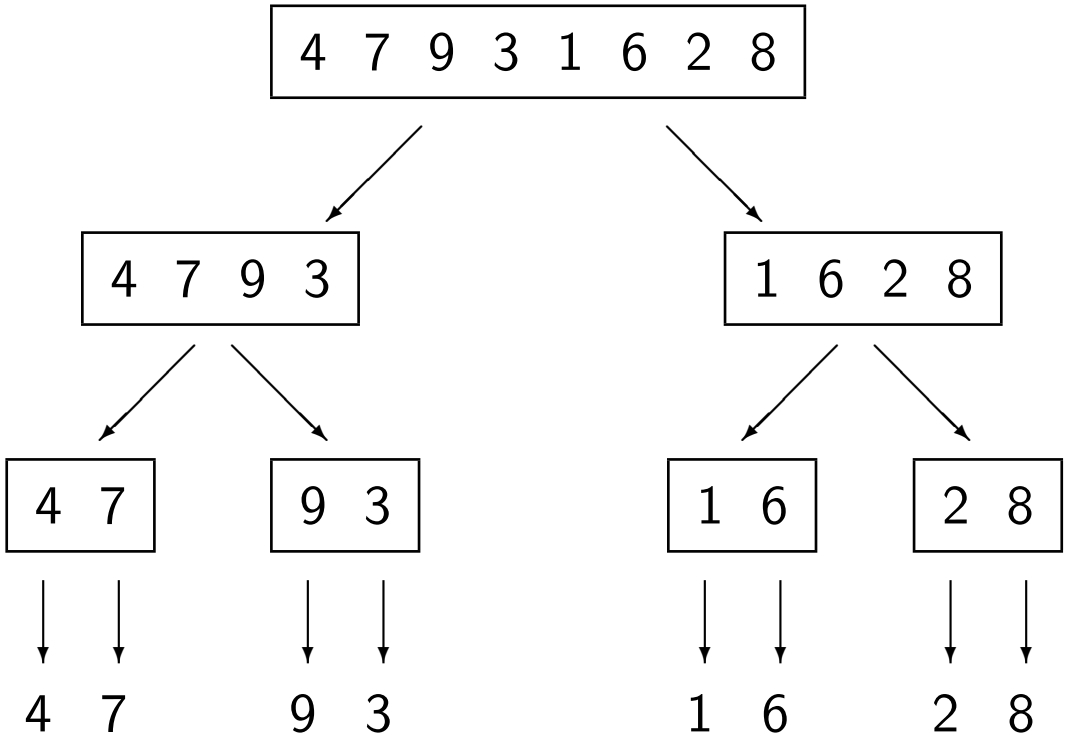
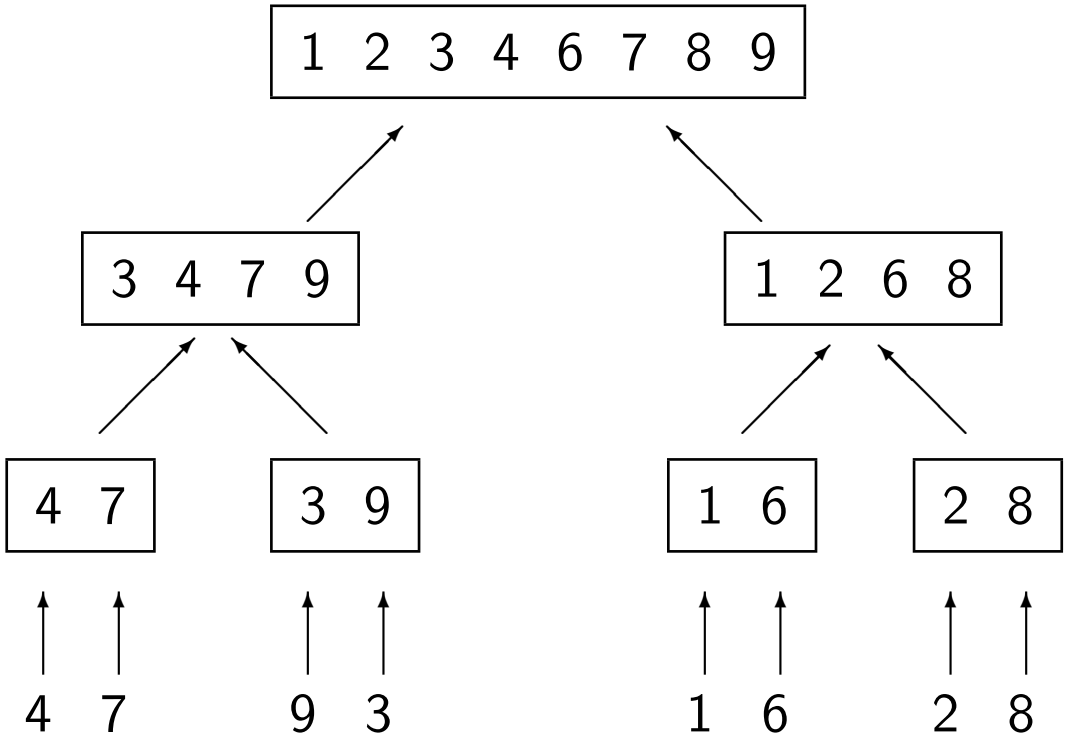
Mergesort – ein Beispiel

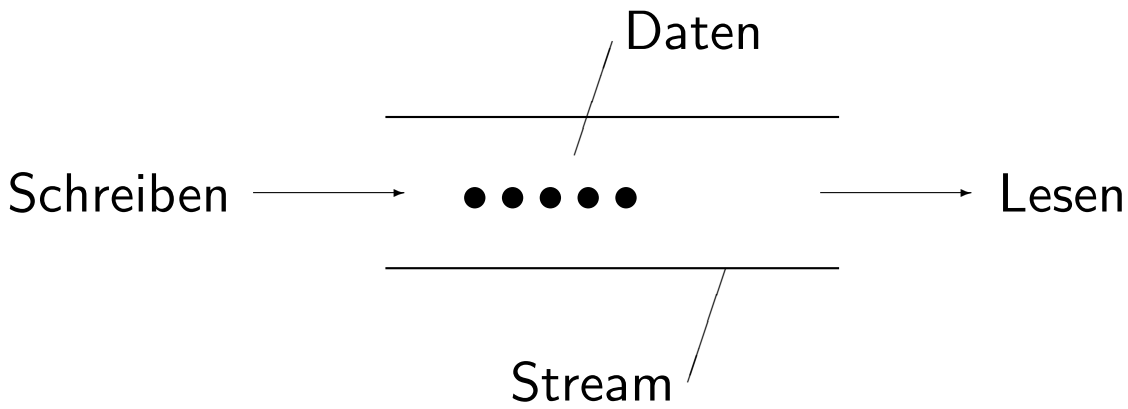


Mergesort – ein Beispiel



Streams

- Stream als unidirektionale Pipeline



- Streams in Java sind *Objekte*,
  und zwar Exemplare von Klassen aus dem Paket `java.io`

Standardstreams in Java

`System.in`	Eingabestream Quelle: Standardeingabe
`System.out`	Ausgabestream Senke: Standardausgabe
`System.err`	Ausgabestream Senke: Standardfehlerausgabe

Diese Streams sind konstante Datenelemente der Klasse `java.lang.System` und Exemplare von `java.io.InputStream` bzw. `java.io.PrintStream`.

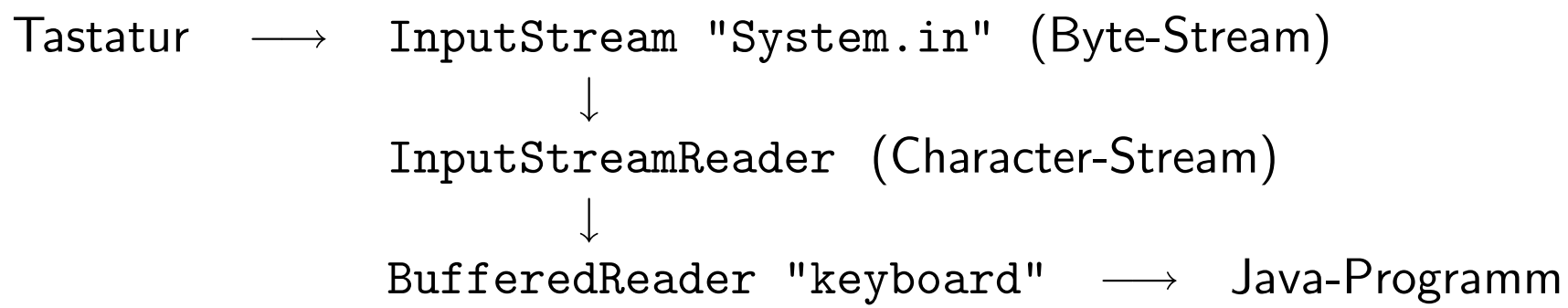
Methode von `InputStream`:	`read()` liest ein Byte aus dem Stream und liefert es als `int`-Wert zurück
Methoden von `PrintStream`:	`print(...)` und `println(...)` schreiben beliebige Datentypen als Strings

Byte- vs. Character-Streams

- Die Standardstreams in Java sind *Byte*-Streams.
- Intern benutzt Java Unicode zur Darstellung der Character, in dem jedes Zeichen durch 16 Bit (zwei Byte) kodiert ist.
- Verbindet man Java-Programme direkt mit Byte-Streams, so wird das höherwertige Byte stets abgeschnitten bzw. Null gesetzt. Daher ist eine fehlerfreie Darstellung nur möglich, wenn auf der zugrundeliegenden Plattform der Zeichensatz iso-latin-1 verwendet wird.
- Zur plattformabhängigen Konvertierung von Byte- in *Character*-Streams oder umgekehrt existieren folgende Adapterklassen:
  - `java.io.InputStreamReader`, deren Konstruktor ein Exemplar von `InputStream` erwartet;
  - `java.io.OutputStreamWriter`, deren Konstruktor ein Exemplar von `OutputStream` erwartet.

Texteingabe von der Tastatur

```
BufferedReader keyboard =
          new BufferedReader(new InputStreamReader(System.in))
```



`java.io.BufferedReader` ist ein Character-Stream, der eine Methode `readLine()` besitzt, die Text zeilenweise (!) aus dem Stream liest.