Java strumienie

Strumienie

Strumień jest abstrakcyjną reprezentacją dowolnego źródła lub ujścia danych.

Wyróżniamy dwa typy strumieni:

– strumienie wejściowe (odczyt danych)

– strumienie wyjściowe (zapis danych)

Strumień jest zawsze związany z jakimś źródłem lub ujściem danych.

W Javie obsługę strumieni zapewnia pakiet java.io.

Strumienie

W celu stworzenia uniwersalnego modelu przepływu danych wykorzystuje się w Javie model strumieni danych. Strumień (stream) jest sekwencją danych, zwykle bajtów.

Podstawowe typu strumieni to te związane z operacjami wprowadzania danych do programu (operacje wejścia) i z operacjami wyprowadzania danych poza program (operacje wyjścia). W Javie do obsługi operacji wejścia stworzono klasę InputStream, natomiast dla obsługi operacji wyjścia stworzono klasę OutputStream.

Strumienie

Strumień związany jest z typem urządzenia/obszaru, z/do którego sekwencja danych przepływa oraz typem danych. Podstawowe urządzenia/obszary to pamięć operacyjna, dyski (pliki), sieć, drukarka, ekran, itp.

Typy danych to np. byte, String, Object, itp.

Klasy OutputStream oraz InputStream reprezentują strumienie jako sekwencje bajtów, czyli elementów typu byte. Jeśli zachodzi potrzeba formatowania danych strumienia można to wykonać korzystając z klas formatujących dziedziczących z OutputStream i InputStream.

Strumienie bajtowe i znakowe

Dwa najbardziej ogólne strumienie bajtowe to:

– InputStream – do odczytu ciągów bajtów

– OutputStream – do zapisu ciągów bajtów

Podstawowymi strumieniami znakowymi są:

– Reader – do odczytu ciągów znaków

– Writer – do zapisu ciągów znaków

Strumienie bajtowe

InputStream:

- int read(byte[] b)
- int read(byte[] b, int off, int len)
- void close()

OutputStream:

- void write(byte[] b)
- void write(byte[] b, int off, int len)
- void flush()
- void close()

Rozszerzanie funkcjonalności

Zwiększenie funkcjonalności odbywa się przez użycie wyspecjalizowanych strumieni. Korzystają one z innych strumieni, aby wykonywać swoją pracę.

Robi się to przez opakowanie jednego typu strumienia w inny (aby to zadziałało musi istnieć w wybranej klasie strumienia odpowiedni konstruktor).

InputStream	int read(byte[] b)
InputStreamReader	int read(char[] cbuf)
BufferedReader	String readLine()

Rozszerzanie funkcjonalności

```
FileInputStream fin = new FileInputStream("dane.txt");
BufferedInputStream bin = new BufferedInputStream(fin);
```

Można stosować metody read() dla bin i dla fin.

Nieuporządkowane wywołania różnych strumieni, podłączonych do tego samego źródła, mogą zakłócać zależności między strumieniami filtrującymi.

Inaczej:

```
BufferedInputStream bin = new BufferedInputStream(
        new FileInputStream("dane.txt"));
```

Strumienie bajtowe

- OutputStream
  - ByteArrayOutputStream (pamięć)
  - FileOutputStream (plik)
  - ObjectOutputStream (obiekt)
  - PipedOutputStream (potok)
  - FilterOutputStream (filtrowanie)
    - BufferedOutputStream (buforowanie)
    - DataOutputStream (konwersje danych)
    - PrintStream (drukowanie)

Strumienie bajtowe

- InputStream
  - ByteArrayInputStream
  - FileInputStream
  - ObjectInputStream
  - PipedInputStream
  - SequencedInputStream (konkatenacja)
  - StringBufferInputStream
  - FilterInputStream
    - BufferedInputStream
    - DataInputStream (konwersje danych)
    - LineNumberedInputStream (zliczanie wierszy)
    - PushbackInputStream (podgląd)

Strumienie bajtowe

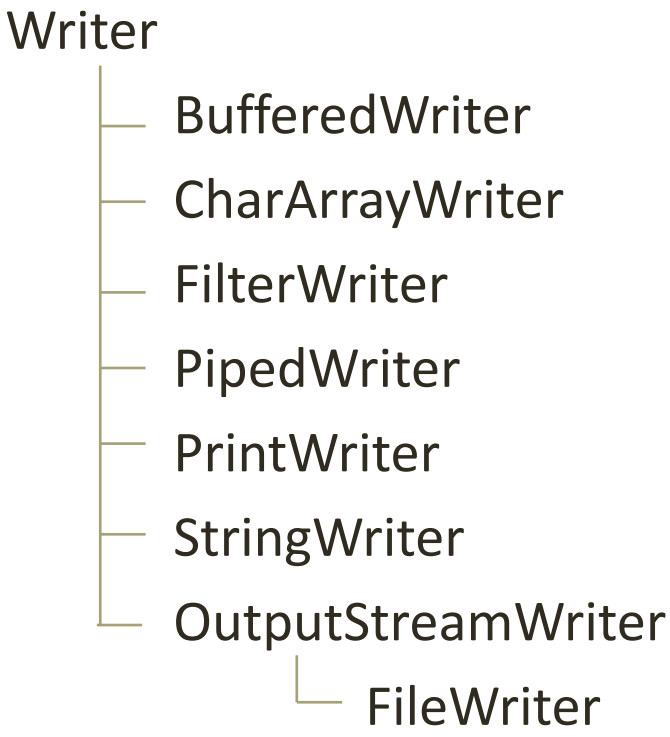
Reader:

- int read()
- int read(char[] cbuf)
- void close()

Writer:

- void write(int c)
- void write(char[] cbuf)
- void flush()
- void close()

Strumienie znakowe



Strumienie znakowe

- Reader
  - BufferedReader
  - CharArrayReader
  - FilterReader
  - PipedReader
  - StringReader
  - InputStreamReader
    - FileReader

Rozszerzanie funkcjonalności – przykład

```
PrintWriter plik = new PrintWriter(
        new BufferedWriter(
                new FileWriter("dane.txt")));
plik.println(….);
-----------------------------------------------------------------
BufferedReader plik = new BufferedReader(
        new FileReader("dane.txt"));
String s = plik.readLine();
```