

Kurs programowania

Wykład 10

Wojciech Macyna

05 maja 2016

Biblioteka wejścia/wyjścia (Input/Output)

- `import java.io.*;`
 - Umożliwia przechowywanie danych na nośniku zewnętrznym
 - Dane mogą być wysyłane do plików, na drukarkę, do bufora pamięci, na wyjście standardowe, ...
 - Dane mogą być czytane z plików, bufora pamięci, standardowego wejścia, ...
-
- Wejście/Wyjście w Javie deklarujemy za pomocą tak zwanych strumieni (Streams)
 - Strumieniem nazywamy uporządkowany ciąg danych (bez określonej długości)
 - Klasy w Javie dzielimy na dwie główne kategorie
 - Strumień bajtowe (Byte Streams)
 - Strumień znakowe (Character Streams)

Biblioteka wejścia/wyjścia (Input/Output)

Podklasy `InputStream` i `OutputStream`: `FileInputStream` i `FileOutputStream`

Podklasy `Reader` i `Writer`: `FileReader` i `FileWriter`

Pozwalają odczytywać i zapisywać pliki dyskowe. Jako parametr konstruktora przekazujemy nazwę pliku dyskowego lub wskazujący go obiekt `File`. Tworząc obiekt wyjściowy, jako drugi argument konstruktora, można przekazać wartość logiczną określającą czy zamiast zamazywać istniejący plik dopisywać kolejne dane na jego końcu.

Biblioteka wejścia/wyjścia (Input/Output)

Podklasy `InputStream` i `OutputStream`: `ByteArrayInputStream` i `ByteArrayOutputStream`

Podklasy `Reader` i `Writer`: `CharArrayReader` i `CharArrayWriter`

Bufor w pamięci oparty na tablicy odpowiednio bajtów lub znaków. Tworząc obiekt wejściowy, przekazujemy konstruktorowi tablicę, na której ma być oparty. Tworząc obiekt wyjściowy, przekazujemy konstruktorowi początkowy rozmiar bufora.

Biblioteka wejścia/wyjścia (Input/Output)

Podklasy `InputStream` i `OutputStream`: `StringBufferInputStream`
(nie ma odpowiednika do zapisu)

Podklasy `Reader` i `Writer`: `StringReader` i `StringWriter`

Bufor w pamięci oparty na klasie `String` (implementacja posługuje się obiektem `StringBuffer`). Tworząc obiekt wejściowy, przekazujemy konstruktorowi napis, na którym ma być oparty. Tworząc obiekt wyjściowy przekazujemy konstruktorowi początkowy rozmiar bufora. Zaleca się używanie klas z hierarchii `Reader/Writer`. `StringBufferInputStream` jest oznaczony jako deprecated.

Biblioteka wejścia/wyjścia (Input/Output)

Podklasy `InputStream` i `OutputStream`: `PipedInputStream` i `PipedOutputStream`

Podklasy `Reader` i `Writer`: `PipedReader` i `PipedWriter`

Łączy do komunikacji między procesami. Przy pomocy konstruktora bezparametrowego należy najpierw utworzyć obiekt jednego rodzaju (wejściowy lub wyjściowy), a następnie przekazać go jako parametr konstruktora obiektu drugiego rodzaju (odpowiednio wyjściowego lub wejściowego). Strumienie zostaną połączone łączem, które będzie przysyłać dane od strumienia wyjściowego do wejściowego.

Przykład zliczający liczbę bajtów w pliku

CountBytes.java

```
1  import java.io.*;
2
3  class CountBytes {
4      public static void main(String[] args)
5          throws FileNotFoundException, IOException
6      {
7          FileInputStream in;
8          in = new FileInputStream("InFile.txt");
9          int total = 0;
10
11         while (in.read() != -1) total++;
12
13         System.out.println(total + " bytes");
14     }
15 }
```

- Przechowuje informacje o pliku i katalogach
- Przykładowe metody klasy: `mkdir()` - założenie katalogu; `delete()` - kasowanie pliku; `exists()` - sprawdzenie, czy plik istnieje.
- Wyjątki: `EOFException` - koniec pliku; `FileNotFoundException` - plik nie mógł zostać otwarty; `InterruptedIOException` - I/O przerwany; `IOException` - ogólny wyjątek I/O.
- Tworzenie pliku

```
1 File inFile = new File("FileIn.txt");  
2 in = new FileInputStream(inFile);
```


Przykład odczytu i zapisu bajtów

ReadWriteFilter.java

```
1 import java.io.*;
2 public class ReadWriteFilter {
3     public static void main(String args[]) throws IOException {
4         FileOutputStream fos = new FileOutputStream("mydata");
5         DataOutputStream dos = new DataOutputStream(fos);
6         dos.writeInt(23);
7         dos.writeDouble(145.10);
8         dos.writeInt('B'+1);
9         dos.writeBoolean(true);
10        dos.writeChar('Y');
11        dos.close();
12        fos.close();
13        FileInputStream fis = new FileInputStream("mydata");
14        DataInputStream dis = new DataInputStream(fis);
15        System.out.println(dis.readInt());
16        System.out.println(dis.readDouble());
17        System.out.println(dis.readInt());
18        System.out.println(dis.readBoolean());
19        System.out.println(dis.readChar());
20        dis.close();
21        fis.close();
22    }
23 }
```

Klasa `std::istream` i jej podklasy `istream`, `ifstream`, `istringstream`.

Klasa `std::ostream` i jej podklasy `ostream`, `ofstream`, `ostringstream`.

Podklasy `istream`: `fstream`, `stringstream`.

Adres gniazda sieciowego

Adres gniazda składa się z adresu komputera (hosta) i numeru portu komunikacji sieciowej.

Adres komputera może być w postaci nazwy tekstowej zgodnej z wymogami odpowiedniego serwisu nazewniczego, adresu IP w postaci tekstowej lub numerycznej.

Konstruktory

- `ServerSocket(int port)` throws `IOException` – utworzenie gniazdka serwera nasłuchującego na porcie `port`.

Metody

- `Socket accept()` throws `IOException` – utworzenie gniazdka właściwego połączenia z klientem.
- `void close()` throws `IOException` – zamknięcie gniazdka serwera.

Konstruktory

- `Socket(String host, int port)` throws `UnknownHostException`, `IOException` – utworzenie gniazdka klienta łączącego się z gniazdkiem serwera `host` na porcie `port`.

Metody

- `void close()` throws `IOException` – zamknięcie gniazdka serwera.
- `InputStream getInputStream()` throws `IOException` – pobranie strumienia wejściowego.
- `OutputStream getOutputStream()` throws `IOException` – pobranie strumienia wyjściowego.

Przykład użycia

- `SocketServer.java` – przykład serwera.
- `SocketClient.java` – przykład klienta.