Kurs programowania

Wykład 10

Wojciech Macyna

Biblioteka wejścia/wyjścia (Input/Output)

- `import java.io.*;`
- Umożliwia przechowywania 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
  - Strumienie bajtowe (Byte Streams)
  - Strumienie znakowe (Character Streams)

Biblioteka wejścia/wyjścia (Input/Output)

Podklasy `InputStream` i `OutputStream`: `FileInputStream` i `FileOutputStream`. Obsługa formatu binarnego.

Podklasy `Reader` i `Writer`: `FileReader` i `FileWriter`. Obsługa formatu tekstowego.

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

Przykład

`CountBytes.java`

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.

Przykłady

`ByteStreamTest.java`, `CharStreamTest.java`

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.

Przykład

`StringWriteTest.java`

Biblioteka wejścia/wyjścia (Input/Output)

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

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

Łącze 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 przesyłać dane od strumienia wyjściowego do wejściowego.

Przykłady

`Pipe.java`, `ReadWriteFilter.java`

Klasa `File`

- 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);
```

Strumienie w C++

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

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

Podklasy `iostream`: `fstream`, `stringstream`.

Przykłady

`Stream.cpp`, `StreamBinary.cpp`, `StreamOut.cpp`

Gniazdka sieciowe

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 odpwiedniego serwisu nazewniczego, adresu IP w postaci tekstowej lub numerycznej.

java.net.ServerSocket

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.

java.net.Socket

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

Jeden klient, jeden serwer

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

Wielu klientow, jeden serwer

- `MultiServerThread.java` – przykład serwera.
- `MultiThread.java` – przykład wątku serwera.
- `MultiClient.java` – przykład klienta.