Strumienie wejścia i wyjścia

Strumień danych oznacza ciąg danych, do którego dane mogą być dodawane i z którego dane mogą być pobierane. Strumienie służą do przeprowadzenia operacji wejścia-wyjścia.

Cechy strumieni:

- strumień związany jest ze źródłem lub odbiornikiem danych

- źródło lub odbiornik danych mogą być dowolne: plik, pamięć, URL, gniazdo, potok …

- strumień służy do zapisania-odczytania informacji – dowolnych danych

- działanie strumieni w programie:

- **Program** kojarzy strumień z zewnętrznym źródłem/odbiornikiem
- Otwiera strumień
- Dodaje lub pobiera dane ze strumienia
- Zamyka strumień

- przy zapisie lub odczycie danych do/z strumienia mogą być wykonywane dodatkowe operacje (np. buforowanie, kodowanie-dekodowanie, kompresja-dekompresja)

- w Javie dostarczono klas, które reprezentują strumienie.

Klasy, które reprezentują strumienie to klasy strumieniowe – są **podstawowym środkiem programowania operacji wejścia i wyjścia** w Javie.

Pakiet nio (New input-output) wprowadza nowe środki wejścia-wyjścia takie jak kanały, bufory i selektory.

Inne od strumieni obiekty operacji wejścia-wyjścia:

- klasa File – opisuje pliki i katalogi

- klasy reprezentujące obiekty „sieciowe” takie jak URL czy gniazdo (socket), mogące być źródłem lub odbiornikiem danych w sieci.

Obiekty tych klas nie stanowią strumieni. Do operowania na nich są jednak potrzebne strumienie (lub kanały) i możemy je uzyskać za pomocą odpowiednich konstruktorów lub metod.

(Czyli ogólnie przy operacji wejścia-wyjścia niezbędne są obiekty strumieni oraz również obiekty które są źródłem lub odbiornikiem danych – np. obiekty typu File reprezentujące pliki i katalogi, obiekty sieciowe reprezentujące URL czy gniazdo (socket)).

2) Klasy strumieniowe

Za pomocą strumieni możemy wykonać dwie podstawowe operacje: odczytanie danych i zapis danych. Z tego punktu widzenia możemy mówić o strumieniach wejściowych i wyjściowych. Tutaj są w Javie dostarczone dwie rozłączne hierarchie klas strumieniowych:

- klasy strumieni wejściowych

- klasy strumieni wyjściowych

Zapis/ odczyt dotyczy atomistycznych porcji danych – minimalnie rozróżnialnych w trakcie operacji. Okazuje się że nie są to tylko bajty. W Javie ze względu na przyjęty standard kodowania znaków (Unikod) – rozróżnia się również inny atom danych – znak Unikodu, złożony z dwóch bajtów. Stąd powstają kolejne rozłączne hierarchie klas strumieniowych:

- klasy strumieni bajtowych (atomem operacji jest bajt), najmniejsza jednostka danych to bajt,

- klasy strumieni znakowych (atomem operacji jest znak Unikod – 2 bajty), najmniejsza jednostka danych w tych strumieniach to znak Unikod czyli dwa bajty,

(Przy przetwarzaniu tekstów należy korzystać ze strumieni znakowych).

Stąd wyróżniamy 4 hierarchie klas strumieniowych:

	Wejście:	Wyjście:
Strumienie bajtowe	**InputStream**	**OutputStream**
Strumienie znakowe	**Reader**	**Writer**

Są to klasy abstrakcyjne, więc nie można utworzyć obiektu tych klas. Dostarczają jednak podstaw dla wszytkich innych klas:

- read() – czytanie bajtów/znaków

- write() – zapis bajtów/znaków

- skip(), mark(), reset() – pozycjonowanie strumieni

- close() – zamykanie strumienia

Metody te są zazwyczaj odpowiednio przedefiniowane w klasach dziedziczących (polimorfizm zapewnia oszczędne i właściwe ich użycie).

3) Klasy przedmiotowe. Wiązanie strumieni ze źródłem/odbiornikiem

Źródła czy odbiorniki mogą być różnorodne. Strumień może być związany np. z plikiem, z pamięcią operacyjną, z potokiem, z URLem, z gniazdem (socket)…
Klasy przedmiotowe wprowadzono dla wygody operowania na konkretnych rodzajach źródeł i odbiorników.

Klasy przedmiotowe:

Źródło/odbiornik	Strumienie znakowe	Strumienie bajtowe
Pamięć	**CharArrayReader, CharArrayWriter**	**ByteArrayInputStream, ByteArrayOutputStream**
	StringReader, StringWriter	**StringBufferInputStream**
Potok	**PipedReader, PipedWriter**	**PipedInputStream, PipedOutputStream**
Plik	**FileReader, FileWriter**	**FileInputStream, FileOutputStream**

4) Klasy przetwarzające (przekształcanie danych w trakcie operacji na strumieniach)

Przy wykonywaniu operacji we-wy mogą być dokonane przekształcenia danych. Java oferuje wiele wyspecjalizowanych klas w konkretnych rodzajach automatycznego przetwarzania strumieni. Klasy te implementują określone rodzaje przetwarzania strumieni, niezależnie od rodzaju źródła/odbiornika.

Rodzaj przetwarzania	Strumienie znakowe	Strumienie bajtowe
Buforowanie	BufferedReader, BufferedWriter	BufferedInputStream, BufferedOutputStream
Filtrowanie	FilterReader, FilterWriter	FilterInputStream, FilterOutputStream
Konwersja: bajty-znaki	InputStreamReader, OutputStreamWriter	
Konkatenacja		SequenceInputStream
Serializacja obiektów		ObjectInputStream, ObjectOutputStream
Konwersje danych		DataInputStream, DataOutputStream
Zliczanie wierszy	LineNumberReader	LineNumberInputStream
Podglądanie	PushbackReader	PushbackInputStream
Drukowanie	PrintWriter	PrintStream

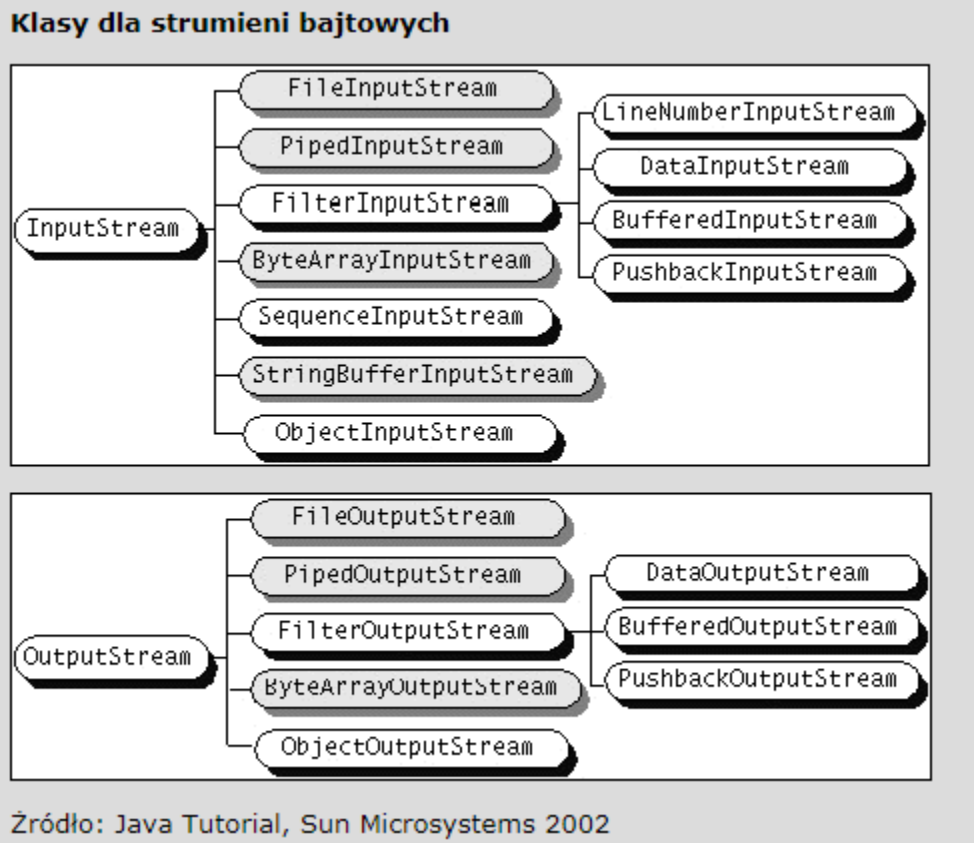
Buforowanie ogranicza liczbę fizycznych odwołań do urządzeń zewnętrznych.

Konstruktory klas przetwarzających mają jako argument referencję do obiektów podstawowych klas abstrakcyjnych hierarchii dziedziczenia (InputStream, OutputStream, Reader, Writer). Dlatego przetwarzanie (automatyczna

transformacja) danych jest logicznie oderwana od fizycznego strumienia, stanowi nakładkę na niego. Czyli innymi słowy klasy przetwarzające nie mają bezpośrednio podanego źródła/odbiornika danych w konstruktorze. Np. przy czytaniu plików źródło podajemy przy konstruowaniu obiektu typu FileReader, a dla celu uzyskania buforowania opakowujemy FileReadera BufferedReaderem.

Zastosowanie klas przetwarzających wymaga:

a) Stworzenie obiektu (strumienia) związanego z fizycznym źródłem/odbiornikiem
b) Stworzenie obiektu odpowiedniej klasy przetwarzającej, nałożonego na fizyczny strumień.



Posumowanie:

- strumienie we/wy to ciąg danych, które można dodawać do strumienia lub pobierać

- strumień jest użyteczny jeśli jest połączony ze źródłem lub odbiornikiem danych

- dzielą się na we/wy oraz bajtowe i znakowe

- strumienie to klasy oraz źródło danych i odbiornik to też klasy

- strumienie przedmiotowe, stworzone aby ułatwić pracę na strumieniu połączonym z konkretnym źródłem/ odbiornikiem danych

- strumienie przetwarzające (opakowujące strumienie we/wy) jako argument w konstruktorze przyjmują typy podstawowych klas abstrakcyjnych odpowiedzialnych za operacje we/wy

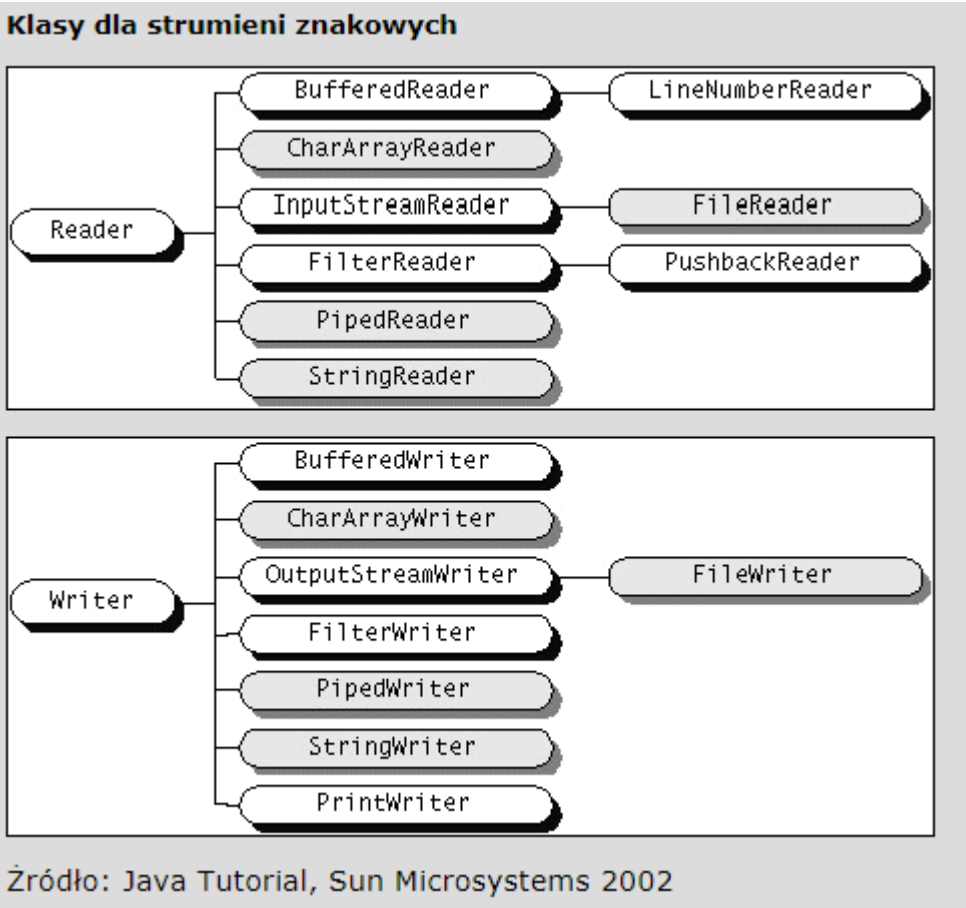
Przy przetwarzaniu tekstów należy korzystać ze strumieni znakowych ze względu na to, iż w trakcie czytania/pisania wykonywane są odpowiednie operacje kodowania/dekodowania ze względu na stronę kodową właściwą dla źródła/odbiornika

Strona kodowa – wariant przypisania poszczególnym kodom binarnym różnych znaków pisarskich w ramach danego systemu kodowania.

http://www.kurshtml.edu.pl/html/strony_kodowe,jezyki.html

Strona kodowa to standard który mówi „maszynie” które kombinacje zer i jedynek odpowiadają jakim literom.

Unicode to specjalna strona kodowa. Jest to standard 16-bitowy i dzięki niemu można zapisać 65536 znaków. Pozwala to umieścić w jednej stronie kodowej większość alfabetów świata – sprawia to że jest to jedyny standard umożliwiający pisanie dokumentów w wielu językach jednocześnie.



Źródło: Java Tutorial, Sun Microsystems 2002

5) Strumienie binarne

Klasy przetwarzające DataInputStream i DataOutputStream służą do odczytu/zapisu danych typów podstawowych w postaci binarnej (oraz łańcuchów znaków). Metody tych klas mają postać:

Typ readTyp()

Void writeTyp (typ arg)

Gdzie: typ odpowiada nazwie któregoś z typów podstawowych np. readDouble().

Dane typu String są czytane/zapisywane z/do strumieni binarnych za pomocą metod writeUTF i readUTF.

Wnioski:

Gdy stosujemy strumieni bajtowych i zapisujemy/odczytujemy znaki Unicode które są powyżej indexu 255 to jest to zapis/odczyt nieprawidłowy. Należy użyć strumieni znakowych, które reprezentują informacje dwu bajtowe.

6) Kodowanie

Java posługuje się znakami w standardzie (formacie) Unicode – są to wielkości 16-bitowe. Środowiska natywne (np. Windows) zapisują teksty jako sekwencje bajtów (z przyjętą stroną kodową). Jak pogodzić bajtowy charakter plików natywnych ze znakami strumieniowymi? Strumienie takie jak InputStreamReader i OutputStreamWriter dokonują konwersji w trakcie czytania/pisania a dokładnie przekształcają bajtowe źródła na znaki Unikodu i odwrotnie.