laboratorium: zadanie nr 10 *18 grudnia 2007 r.*

KURS PROGRAMOWANIA W JAVIE

STRUMIEŃ FILTRUJĄCY DO CZYTANIA TEKSTOWEGO

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie.

Zdefiniuj klasę `ScanReader` opakowującą strumień typu `Reader`, której zadaniem będzie odczytywanie danych ze strumienia tekstowego i automatyczne przekształacanie tych danych do postaci binarnej. Klasa `ScanReader` ma dziedziczyć po `FilterReader` albo po `PushbackReader` i powinna posiadać jeden konstruktor `ScanReader(Reader)`.

Ponieważ obiekt klasy `ScanReader` ma wspomagać automatyczną konwersję danych zapisanych tekstowo do postaci binarnej, więc należy zdefiniować w niej zestaw metod służących do odczytywania ze strumienia liczb całkowitych i rzeczywistych:

```
public byte scanByte () throws NumberFormatException;
public short scanShort () throws NumberFormatException;
public int scanInt () throws NumberFormatException;
public long scanLong () throws NumberFormatException;
public float scanFloat () throws NumberFormatException;
public double scanDouble () throws NumberFormatException;
```

Na przykład, `scanInt()` odczytuje ciąg cyfr dziesiętnych i zamienia je na liczbę typu `int`. Jeśli analizowany ciąg cyfr jest zbyt długi i liczba przez niego reprezentowana nie mieści się w zakresie liczb typu `int`, to należy zasygnalizować błąd a odczytane dane mają być w całości zwrócone do strumienia (wykorzystaj obiekt klasy `StringBuffer`).

Oprócz liczb całkowitych w postaci dziesiętnej należy interpretować też liczby w postaci szesnastkowej, ósemkowej i dwójkowej. Na przykład dla liczb całkowitych typu `int` zdefiniuj następujący zestaw metod:

```
public int scanInt () throws IOException, NumberFormatException; // liczby 10-tne
public int scanIntHex () throws IOException, NumberFormatException;
public int scanIntOct () throws IOException, NumberFormatException;
public int scanIntBin () throws IOException, NumberFormatException;
```

Liczby rzeczywiste można interpretować tylko w postaci dziesiętnej.

Zdefiniuj też funkcje, które będą pobierać jeden znak `char`, czytać łańcuch znakowy `String` (ciąg znaków bez znaków białych), rozpoznawać identyfikator `String` (ciąg liter, cyfr i znaków podkreślenia, rozpoczynający się od litery), pomijać białe znaki i sprawdzać czy strunień nie jest w pozycji końcowej:

```
public char scanChar () throws IOException; // liczby w postaci 10-tnej
public String scanString () throws IOException;
public String scanId () throws IOException;
public int skipWS () throws IOException;
public boolean isEOF () throws IOException;
```

Ewentualne błędy należy sygnalizować zgłaszaniem wyjątków.

Uwaga.

Napisz prosty program konsolowy, który przetestuje działanie twojego strumienia filtrującego zaaplikowanego do kilku różnych strumieni (standardowe wejście, plik tekstowy, strumień skojarzony z łańcuchem znakowym). Filtr należy przetestować również na okoliczność błędów, na przykład w sytuacji, gdy chcemy odczytać liczbę całkowitą a najbliższy znak w strumieniu nie jest cyfrą.