Politechnika Gdańska, ***Inżynieria Biomedyczna***

Język JAVA
podstawy programowania

Jacek Rumiński

Wykład 6, część 2

Plan wykładu:

1. Wprowadzenie do operacji wejścia/wyjścia: strumienie
2. Podstawowe klasy strumieni InputStream/OutputStream
3. Obsługa strumieni znaków
4. Operacje na plikach w systemie plików
5. Specjalne zastosowania strumieni

Podstawowe klasy strumieni znaków

W związku z problemem wynikającym z konwersji znaków Javy (Unicode) na bajty i odwrotnie występujących we wczesnych (JDK 1.0) realizacjach klas obsługi strumieni począwszy od wersji JDK1.1 wprowadzono dodatkowe klasy Reader i Writer. Obie abstrakcyjne klasy są analogicznie skonstruowane (dziedziczenie z klasy Object i deklaracja metod) jak klasy InputStream oraz OutputStream. Klasy dziedziczące po Reader i Writer zapewniają ciekawe możliwości zastosowania.
Odczyt danych odbywa się poprzez zastosowanie metod read() lub readLine() natomiast zapis danych do strumienia poprzez wykorzystanie metod write().

Wybrane klasy dziedziczące po klasie Reader/Writer:

BufferedReader – buforuje otrzymywany tekst (czytaj linię tekstu!),
InputStreamReader – czyta bajty zamieniające je na tekst według podanego systemu kodowania znaków,

FileReader – odczyt danych tekstowych z pliku dla domyślnego systemu kodowania znaków, poprzez podanie ścieżki zależnej systemowo (String) lub abstrakcyjnej (File)

StringReader – obsługa strumienia pochodzącego od obiektu klasy String.

Dla klasy Writer analogicznie, tylko …Writer, np. FileWriter.

Dodatkowa klasa to PrintWriter (analog klasy PrintStream).

Język JAVA – DrukujJedi.java


```
import java.io.*;
class FormatujStrumien extends OutputStreamWriter{
  FormatujStrumien(String kodowanie) throws UnsupportedEncodingException{
            super(System.out,kodowanie);
  }//koniec FormatujStrumien()
}//koniec FormatujStrumien
public class DrukujJedi{
  public static void main(String args[]){
    try{
            FormatujStrumien fs=new FormatujStrumien("Cp852");
            fs.write("ąęśółżźćń");
            fs.close();
            /*  //Albo:
            PrintWriter pw= new PrintWriter(new OutputStreamWriter(System.out,"Cp852"));
            pw.println("ąęśółżźćń");
            pw.close();              */
            //Zamiast System.out może być strumień związany z plikiem,
            //np. new FileOutputStream("jakisPlik.txt");
    } catch (Exception e){ System.out.println("Wyjątek: "+e);}
  }//koniec main()
}// koniec public class DrukujJedi
```

```
import java.io.*;
public class NoweEchoJedi{
  public static void main(String args[]){
    PrintWriter pw=null;
    try{
            pw= new PrintWriter(new OutputStreamWriter(System.out,"Cp852"),true);
            //true na końcu oznacza, że dane są przesyłane od razu (bez flush())
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            String test;
            //Jeśli chcemy dodać znak specjalny, np. cudzysłów musimy użyć znaku \"
            pw.println("Wprowadź dane. Linia z \".\" oznacza koniec wprowadzania.");

            //readLine() - czytaj całą linię
            while(!(test=br.readLine()).equals(".")){
                    pw.println("Wprowadzono: \t"+test);
            }
            pw.close();
    }catch (Exception e){ System.out.println("Wyjatek: "+e);}
  }//koniec main()
}//koniec public class NoweEchoJedi
```

Rozdzielanie tekstu StringTokenizer/StreamTokenizer

Na zakończenie omawiania klas związanych z obsługą strumieni warto zapoznać się z klasą StreamTokenizer, dzieląca strumień tekstowy na leksemy.

Klasa ta daje więc swoistą funkcjonalność wykrywanie elementów strumienia i umieszczania ich w tablicy. Wskazując znak oddzielający leksemy (ordinaryChar()) można dokonać przeformatowania przesłanego tekstu (np. podzielić ścieżkę dostępu, dokonać detekcji liczb w tekście, itp.). Pobranie leksemu z tablicy odbywa się poprzez wywołanie metody nextToken().

Warto zauważyć, że istnieje klasa StringTokenizer o podobnym działaniu (prostsza), której argumentem nie jest jednak strumień, a obiekt klasy String.

Język JAVA – SeparatorJedi.java


```
import java.io.*;
import java.util.StringTokenizer;
public class SeparatorJedi{
  public static void main(String args[]){
    System.out.println("Podaj tekst zawierający znak : .");
    Reader r = new BufferedReader(new InputStreamReader(System.in));
    StreamTokenizer st = new StreamTokenizer(r);
    st.ordinaryChar('.');  //ustaw znak rozdzielający
    try{
            while (st.nextToken() != StreamTokenizer.TT_EOF){ //koniec? CTRL-Z
               if (st.ttype==StreamTokenizer.TT_WORD){  //tekst?
                        System.out.println(new String(st.sval));  //pobierz wartość
              }//if                    }//while
    }catch (IOException ioe){   System.out.println("Błąd operacji I/O");              }
    String dane="153.19.51.66";          System.out.println("Dzielę tekst: "+dane);
    StringTokenizer strT = new StringTokenizer(dane,".");
    while (strT.hasMoreTokens()) {
            System.out.println(strT.nextToken());
    }//koniec while
  }//koniec main()
}//koniec public class SeparatorJedi
```

Plan wykładu:

1. Wprowadzenie do operacji wejścia/wyjścia: strumienie
2. Podstawowe klasy strumieni InputStream/OutputStream
3. Obsługa strumieni znaków
4. Operacje na plikach w systemie plików
5. Specjalne zastosowania strumieni

Operacje na plikach

Dostęp do plików zaprezentowany wcześniej wykorzystywał klasy FileInputStream, FileOutputStream, FileReader i FileWriter.

Konstruktory tych klas umożliwiają utworzenie strumienia poprzez podanie ścieżki do pliku jako argumentu. Ścieżkę można podać stosując dwie metody:

1. poprzez ciąg znaków (obiekt klasy String)
2. poprzez obiekt klasy File, reprezentujący logiczną ścieżkę do plików i katalogów.

Ścieżka dostępu do pliku może być sklasyfikowana ze względu na jej zasięg lub ze względu na środowisko, dla którego jest zdefiniowana (np. Windows, Linux). W pierwszym przypadku dzieli się ścieżki dostępu na absolutne i relatywne. Absolutne to te, które podają adres do pliku względem głównego korzenia systemu plików danego środowiska. Relatywne to te, które adresują plik względem katalogu bieżącego.

Operacje na plikach

Druga klasyfikacja rozróżnia ścieżki dostępu pod względem środowiska, dla którego jest ona zdefiniowana, co w praktyce dzieli ścieżki dostępu na te zdefiniowane dla systemów opartych na UNIX i na te, zdefiniowane dla systemów opartych na MS *Windows* (znak rozdzielający katalogi: własność systemu o nazwie file.separator, lub wartość pola File.separator). Przykłady:

- absolutna ścieżka dostępu:

UNIX: /utl/software/java/projekty

MS Windows: c:\utl\softare\java\projekty

- relatywna ścieżka dostępu:

UNIX: java/projekty

MS Windows: java\projekty.

Operacje na plikach

Tworząc obiekt klasy File dokonywana jest konwersja łańcucha znaków na abstrakcyjną ścieżkę dostępu do pliku (abstrakcyjna ścieżka dostępu do pliku jest tworzona według określonych reguł podanych w dokumentacji API).
Metody klasy File umożliwiają kontrolę podanej ścieżki i plików (np. isFile(), isDirectory(), isHidden, canRead(), itp.) oraz dokonywania konwersji (np. getPath(), getParent(), getName(), toURL(), itp.), jak i wykonywania prostych operacji (list(), mkdir(), itp.).

Uwaga! Należy pamiętać, że zapis tekstowy ścieżki dostępu dla środowiska MS Windows musi zawierać podwójny separator, gdyż pojedynczy znak umieszczony w ciągu znaków oznacz początek kodu ucieczki (czyli znak specjalny), np. „c:\\java\\kurs\\wyklad\\np".

Język JAVA – SystemPlikowJedi.java


```
import java.io.*;  import java.util.*;
public class SystemPlikowJedi{
  public static void main(String args[]){
      //utworzenie obiektu File oznacza adresowanie domniemanego węzła w systemie plików
      File f = new File("DANE_JEDI");
      if(f.exists()){//jeśli dany węzeł istnieje
        if(f.isDirectory()){//jeśli jest to katalog
          System.out.println("Katalog: "+f.getAbsolutePath()+" już istniał.\nData modyfikacji: "+
          new Date(f.lastModified())+". Usuwam go.\n");
          f.delete();//usuń go
        }//if f.isDirectory()
      }//if f.exists()
      if (f.mkdir()) {//jeśli uda się utworzyć katalog
            File g = new File (".");//adresuj bieżący węzeł zawierający nowy katalog
            String s[] = g.list();//generuj listę węzłów (plików i katalogów)
            System.out.println("Zawartość katalogu "+g.getAbsolutePath()+": \n");
            for (int i =0; i<s.length; i++){
                        System.out.println(s[i]);//wyświetl kolejne nazwy          }//for
      } else {          System.out.println("Nie można utworzyć katalogu!");       }//else
            }//koniec main()
}//koniec public class SystemPlikowJedi
```

Plan wykładu:

1. Wprowadzenie do operacji wejścia/wyjścia: strumienie
2. Podstawowe klasy strumieni InputStream/OutputStream
3. Obsługa strumieni znaków
4. Operacje na plikach w systemie plików
5. Specjalne zastosowania strumieni

Specjalne zastosowania strumieni

Poza przedstawionym materiałem strumienie oraz przesyłanie danych obsługiwane jest w Javie w różnych pakietach. Przede wszystkim dotyczy to nowego pakietu java.nio.*
Pakiet ten obsługuje nową bardziej zaawansowane sposoby przesyłania i kontroli bloków bajtów (bufory). Funkcjonalność pakietu java.nio będzie omówiona na kolejnym etapie poznawania Javy.

Ponadto obsługa strumieni związana będzie z pakietami i klasami, które obsługują różne formy komunikacji, np. przesyłanie danych przez sieć (pakiet java.net), przesyłanie danych przez interfejsy komunikacyjne (javax.comm). Wówczas dla danego węzła (źródła lub przeznaczenia) zapewniana będzie funkcjonalność otwarcia strumienia (czyli utworzenia obiektu jednej z klas omówionych wcześniej).

```
import java.net.*;
import java.io.*;
public class PobierzJedi{
  public static void main(String args[]){
    URL url;
    String tekst;
    if (args.length !=1) {
            System.out.println("Wywołanie: PobierzJedi URL; gdzie URL to adres zasobów");
            System.exit(1);
     }//if
     try{
          url = new URL(args[0]);
          //Klasa URL reprezentuje zasób według adresu  zasobu, np. http://www.biomed.gda.pl
           BufferedReader br = new BufferedReader(new InputStreamReader(url.openStream()));
           while( (tekst=br.readLine()) !=null){
                        System.out.println(tekst);
           }//while
      } catch (Exception e) {      e.printStackTrace();      }//catch
  }//koniec main()
}// koniec public class PobierzJedi
```

Specjalne zastosowania strumieni

Kolejną dziedziną wykorzystania strumieni jest wykonywanie operacji na przesyłanych danych. Do operacji takich można zaliczyć kodowanie danych (JPEG, MPEG) czy kompresję danych.

Przykładowo w pakiecie java.util.zip zdefiniowano szereg klas strumieni obsługujących kompresję w formie ZIP i GZIP. Podstawowe klasy strumieni tam przechowywane to:

- CheckedInputStream,
- CheckedOutputStream,
- DeflaterOutputStream,
- GZIPInputStream,
- GZIPOutputStream,
- InflaterInputStream,
- ZipInputStream,
- ZipOutputStream.

Język JAVA – KompresujJedi.java


```
import java.io.*;
import java.util.zip.*;
public class KompresujJedi{
  public static void main(String args[]){
    String s;
    byte b[] = new byte[100];
    for (int i=0; i<100; i++){
            b[i]=(byte) (i/10); //10 różnych wartości
    }//for
    try{
            FileOutputStream o = new FileOutputStream("plik2.txt");
            o.write(b); //zapisanie 100 bajtów, 10*10 różnych wartości
            o.flush();
            o.close();
            FileOutputStream fos = new FileOutputStream("plik2.gz");
            GZIPOutputStream z = new GZIPOutputStream(new BufferedOutputStream(fos));
            z.write(b,0,b.length);//zapis bajtów będących efektem kompresji - 44 bajty
            z.close();
    } catch (Exception e){} //brak obsługi wyjątków
  }// koniec main
}// koniec public class Kompresja
```

Co dalej?

Przedstawiona część wykładu dotycząca obsługi strumieni w Javie to wybrane przeze mnie elementy, które uważam za kluczowe (możliwości są dużo większe). Zainteresowanych odsyłam do zasobów Internetu oraz dedykowanych podręczników.

To już koniec podstawowego kursu z Javy.
Co dalej?
Dla studentów kierunku INŻYNIERIA BIOMEDYCZNA dostępny będzie kurs na następnym poziomie poznawania Javy (wątki, programowanie sieciowe, programowanie rozproszone).
Ponadto dostępne będą dedykowane, pojedyncze prezentacje w zakresie JAK TO ZROBIĆ W JAVIE.

Zapraszam