Textové soubory

- Textový soubor je posloupnost znaků členěná na řádky
  - – každý znak je reprezentován jedním bytem, jehož obsah je dán nějakým kódováním znaků
  - – členění na řádky je závislé na platformě a obvykle je dáno jedním nebo dvěma řídicími znaky (*CR LF* , 0x0d 0x0c, “\r\n”)
- Program v jazyku Java pracuje se znaky jako s hodnotami typu *char*, které jsou zakódovany v 16-ti bitovém kódu Unicode
- Při práci s textovými soubory jsou proto třeba transformace mezi 16-ti bitovými hodnotami typu *char* a 8-mi bitovými znaky textového souboru
- Vytváření textového souboru z posloupnosti hodnot typu *char* s použitím implicitního kódování umožňuje třída *FileWriter*
- Čtení textového souboru jako posloupnosti hodnot typu *char* s použitím implicitního kódování umožňuje třída *FileReader*
- Pro OS Windows je implicitní kódování *Cp*1250

Příklad textového souboru

- Následující program zapíše do textové souboru znaky dané řetězcem a pak soubor přečte do pole prvků typu *char* a pole vypíše na obrazovku

```
package alg9;

import java.io.*;
import sugar.Sys;

public class Text1 {
  public static void main(String[] args)
                                  throws Exception  {
    FileWriter out = new FileWriter("text1.txt");
    out.write("čau, nazdar");
    out.close();
    FileReader in = new FileReader("text1.txt");
    char znaky[] = new char[20];
    int pocet = in.read(znaky);
    for (int i=0; i<pocet; i++)
      Sys.p(znaky[i]);
  }
}
```


- Soubor *text1.txt* si můžete prohlédnout textovým editorem

Metody tříd FileWriter a FileReader

- Některé metody třídy *FileWriter* (všechny mohou vyhodit výjimku *IOException*)
  `void write(int c)`
    – zapíše znak *c*
  `void write(char[] cbuf)`
    – zapíše znaky z pole *cbuf*
  `void write(String str)`
    – zapíše znaky tvořící řetězec *str*
- Některé metody třídy *FileReader* (všechny mohou vyhodit výjimku *IOException*)
  `int read()`
    – vrátí přečtený znak; nejsou-li v souboru nepřečtené znaky, vrátí -1
  `int read(char[] cbuf)`
    – přečte znaky do pole *cbuf* a vrátí jejich počet; nejsou-li v souboru nepřečtené znaky, vrátí -1

Výstupní konverze

- Do textového souboru často potřebujeme zapisovat posloupnosti znaků tvořící zápisy čísel
- Řetězce tvořící dekadické zápisy čísel lze získat statickými metodami *valueOf* třídy *String*
- Příklad:

```
public class Text2 {
  public static void main(String[] args)
                                  throws Exception  {
    FileWriter out = new FileWriter("text2.txt");
    int a = 10, b = 20;
    out.write(String.valueOf(a));
    out.write("+");
    out.write(String.valueOf(b));
    out.write("=");
    out.write(String.valueOf(a+b));
    out.close();
  }
}
```


- Program vytvoří textový soubor obsahující text

```
10+20=30
```

Třída PrintWriter

- Třída *PrintWriter* umožňuje vytvářet textový soubor pomocí metod, které:
  - – provádějí výstupní konverze primitivních typů
  - – oddělují řádky způsobem daným platformou
- Příklad:

```
public class Text3 {
  public static void main(String[] args)
                              throws Exception  {
    PrintWriter out = new PrintWriter(
                              new FileWriter("text3.txt"));
    int a = 10, b = 20;
    out.print(a); out.print("+"); out.print(b);
    out.print("="); out.println(a+b);
    out.println("konec programu");
    out.close();
  }
}
```

- Program vytvoří soubor obsahující text

```
10+20=30
konec programu
```

Metody třídy PrintWriter

`void print(T x)`

– metoda je definovaná pro všechny primitivní typy, do souboru zapíše textovou reprezentaci hodnoty *x*

`void print(String str)`

– zapíše znaky tvořící řetězec *str*; je-li *str* rovno *null*, zapíše se *null*

`void print(Object o)`

– zapíše textovou reprezentaci objektu *o*, která se získá voláním metody *o.toString*(); je-li *o* rovno *null*, zapíše se *null*

`void println()`

– zapíše oddělovač řádků

`void println(T x)`

– zapíše textovou reprezentaci hodnoty *x* a zakončí řádek

a další ...

- Metody třídy *PrintWriter* nekončí vyhozením výjimky

Čtení textového souboru po řádcích

- Třída *FileReader* umožňuje číst textový soubor po jednotlivých znacích nebo po skupinách znaků (ukládaných do pole znaků), přičemž oddělování řádků je závislé na platformě (dvojice řídicích znaků *CR LF* pod Widnows nebo jediný znak *LF* pod Unixem)
- Čtení textového souboru po řádcích nezávisle na platformě umožňuje třída *BufferedReader*
- Příklad: program který přečte textový soubor daný parametrem příkazového řádku a vypíše jej na obrazovku


```
public class Text4 {
  public static void main(String[] args)
                                  throws Exception {
    if (args.length==0) {
      Sys.pln("použití: Text4 <vstup>");
      return;
    }
    BufferedReader in = new BufferedReader(
                                  new FileReader(args[0]));
    String radek = in.readLine();
    while (radek!=null) {
      Sys.pln(radek);
      radek = in.readLine();
    }
  }
}
```

Metody třídy BufferedReader

- Všechny metody mohou vyhodit výjimku *IOException*

  `int read()`

  – vrátí přečtený znak; nejsou-li v souboru nepřečtené znaky, vrátí -1

  `int read(char[] cbuf)`

  – přečte znaky do pole *cbuf* a vrátí jejich počet; nejsou-li v souboru nepřečtené znaky, vrátí -1

  `int read(char[] cbuf, int off, int len)`

  – přečte *len* znaků do *cbuf* počínaje indexem *off*, výsledkem je počet přečetných znaků; nejsou-li v souboru nepřečetné znaky, vrátí –1

  `String readLine()`

  – přečte znaky až do konce řádku a vytvoří z nich řetězec (bez oddělovače řádků), který vrátí jako výsledek; nejsou-li v souboru nepřečtené znaky, vrátí *null*

  a další ...

Čtení textu po lexikálních elementech

- Textový soubor se často skládá z lexikálních elementů tvořených posloupnostmi znaků s určitou syntaxí
- Čtení textového souboru po lexikálních elementech umožňuje třída *StreamTokenizer*
- Třída rozlišuje 4 druhy lexikálních elementů (tokens):
  - číslo (posloupnost dekadických číslic příp. začínající znaménkem – a příp. obsahující desetinnou tečku)
  - slovo (posloupnost písmen a číslic začínající písmenem)
  - oddělovač řádku
  - konec souboru

Čtení textu po lexikálních elementech

- Příklad:

```
public class Text5 {
  public static void main(String[] args)
                                throws Exception {
    StreamTokenizer st = new StreamTokenizer(
                                  new FileReader("text5.txt"));
    st.eolIsSignificant(true);
    for (;;) {
      switch (st.nextToken()) {
        case StreamTokenizer.TT_NUMBER:
          Sys.p(st.nval+" "); break;
        case StreamTokenizer.TT_WORD:
          Sys.p(st.sval+" "); break;
        case StreamTokenizer.TT_EOF:
          return;
        case StreamTokenizer.TT_EOL:
          Sys.pln(); break;
      }
    }
  }
}
```