
Inżynieria oprogramowania

Lecture XXX

# Java™ – część IV: IO

**Bartosz Walter**

<Bartek.Walter@man.poznan.pl>

# Plan

- Klasa java.io.FIle
- Strumienie tekstowe
- Strumienie standardowe
- Properties

# Wprowadzenie

- pakiet java.io.*;
  - import java.io.*;
- wyjątek java.io.IOException – rzucają niemal wszystkie metody
- klasy są zbudowane hierarchicznie wokół klas Reader i Writer (dla plików tekstowych) oraz InputStream i OutputStream (dla danych binarnych)
- obiekty strumieni są często dekorowane innymi obiektami udostępniającymi nowe funkcje

# Klasa java.io.File

- Reprezentuje ona albo nazwę konkretnego pliku, albo nazwę zbioru katalogu (są traktowane w zasadzie tak samo)
- **String[] list()** – metoda która w przypadku katalogów zwraca tablice z nazwami plików znajdujących się w danym katalogu
- **String getName()** – zwraca nazwę pliku
- **String getAbsolutePath()** – zwraca pełną ścieżkę pliku
- **boolean exists()** – czy istnieje
- **boolean canWrite()** – możliwość zapisu
- **boolean canRead()** – możliwość czytania
- **boolean isFile()** – czy plik
- **boolean isDirectory()** – czy katalog
- **boolean renameTo(File newName)** – zmiana nazwy
- **boolean mkdirs()** – tworzenie ścieżki o dowolnej złożonosci
- **boolean delete()** – kasowanie pliku

# Klasa java.io.File - przykład

```
File katalog = new File("c:/katalog1");
File nowyKatalog = new File("c:/katlog2");
//tworzymy katalog
katalog.mkdirs();
//zmieniamy jego nazwę
katalog.renameTo(nowyKatalog);
```

# I/O- informacje podstawowe

- Java wykorzystuje pojęcie strumienia (ang. stream)
- *Strumień* - reprezentacja dowolnego źródła danych, jako obiektu zdolnego do wysyłania i odbierania porcji danych
- W Javie 1.1 zaszły istotne zmiany w stosunku do wcześniejszych wersji (cel: obsługa standardu Unicode)
- Rozróżniamy dwa rodzaje strumieni: tekstowe i binarne
- Do obsługi tekstowych korzystamy z hierarchii klas `Reader` i `Writer`
- Do obsługi binarnych korzystamy z hierarchii klas `InputStream` i `OutputStream`

# Reader i Writer

- Używamy do obsługi strumieni tekstowych
- Powstały w celu obsługi standardu Unicode
- Podział na dwie podstawowe klasy Reader oraz Writer, które posiadają metody `read()` i `write()` pozwalające na operacje czytania i zapisu bajtu lub tablicy bajtów(to samo `InputStream` i `OutputStream`)
- Nie korzystamy z tych metod bezpośrednio. Wykorzystują je klasy dziedziczące, które służą do obsługi konkretnych źródeł.

# Czytanie z plików tekstowych

```
try {
   File plik = new File("test1.txt");
   BufferedReader in = new BufferedReader(
      new FileReader(plik));
   String line;

   while((line = in.readLine()) != null)
      System.out.println(s);

} catch(IOException e1) {
   System.err.println("Błąd przy przetwarzaniu");
}catch(FileNotFoundException e2{
   System.err.println("Brak pliku: "+ plik);
} finally {
   in.close();
}
```

# Zapis do plików tekstowych

```
File plik = new File("test1.txt");
String content ="Pierwsza Linia \nDruga Linia \nKoniec";
String line;
BufferedReader in = new BufferedReader(
    new StringReader(s));
try {
    PrintWriter out = new PrintWriter(
        new BufferedWriter(new FileWriter(plik)));

    while ((line = in.readLine()) != null)
       out.println(s2);

    out.close();
} catch(IOException e1){
  // obsługa błedów
}
```

# Standardowe wejście-wyjście

- Termin odnosi się do koncepcji wywodzących się z Unixa, stosowanej potem w innych systemach operacyjnych
- Dane wejściowe mogą pochodzić ze standardowego wejścia, a dane wyjściowe program może wysyłać na standardowe wyjście i standardowe wyjście błędów
- Wyjście jednego programu może być wejściem drugiego
- System.in, System.out ,System.err są obiektami

# Czytanie ze standardowego wejścia

```
try {
   InputStreamReader ireader =
       new InputStreamReader(System.in);
   BufferedReader in = new BufferedReader(ireader);
   String line;

   // wprowadzenie pustego wiersza kończy pętlę
   while ((line = in.readLine()) != null)
       System.out.println(line);

} catch(IOException e1){
    // obsługa błędów
}
```

# java.util.Properties

- Wygodny system umożliwiający w prosty sposób konfigurację ustawień programu
- Mogą być użyte do zapisywania i wczytywania ustawień przy każdym  uruchomieniu programu
- Poprzez prostą modyfikację wartości par klucz/wartość w pliku możemy łatwo zmienić konfigurację
- Obiekt Properties składa się z par klucz/wartość zapisanych w postaci String
- Properties można wczytać z i zapisać do pliku

# java.util.Properties - przykład

```
Properties props = System.getProperties();
// ale można także new Properties();
Enumeration propNazwy = props.propertyNames();

while (propNazwy.hasMoreElements()){
    String klucz = (String) propNazwy.nextElement();
    String wartosc =
        (String) props.getProperty(propNazwa);
    System.out.println("Property " + klucz
        + ": " + wartosc);
}
```

# Q & A