Materiał pomocniczy do kursu „Podstawy programowania”

Autor: Grzegorz Góralski
ggoralski.com

Łańcuchy znaków, wyrażenia regularne, pliki tekstowe

Manipulacja łańcuchami znaków, podstawy wyrażeń regularnych, zapis i odczyt danych w pliku tekstowym

String

- Klasa `String` oferuje wiele użytecznych metod, np:
  - testy:

```
String tekst = "Chromosom";
System.out.println("Tekst                              : "+tekst);
// testy
System.out.println("equals(\"chromosom\")               : "+tekst.equals("chromosom"));
System.out.println("equalsIgnoreCase(\"chromosom\") :
"+tekst.equalsIgnoreCase("chromosom"));
System.out.println("contains(\"omo\")                  : "+tekst.contains("omo"));
System.out.println("startsWith(\"Chro\")               : "+tekst.startsWith("Chro"));
System.out.println("endsWith(\"Chro\")                 : "+tekst.endsWith("Chro"));
System.out.println("length()                           : "+tekst.length());
```

String

- Klasa `String` oferuje wiele użytecznych metod, np:
  - modyfikacje:

```
// modyfikacje
System.out.println("toLowerCase()                    : "+tekst.toLowerCase());
System.out.println("toUpperCase()                    : "+tekst.toUpperCase());
System.out.println("replace(\"osom\", \"atyda\")        : "+tekst.replace("osom", "atyda"));
System.out.println("replaceFirst(\"o\", \"i\")          : "+tekst.replaceFirst("o", "i"));
System.out.println("replaceAll(\"o\", \"i\")            : "+tekst.replaceAll("o", "i"));
System.out.println("concat(\"owy\")                   : "+tekst.concat("owy"));
```

String

- Klasa `String` oferuje wiele użytecznych metod, np:
  - modyfikacje:

```
// wycinanie i dzielenie
System.out.println("substring(4)                    : "+tekst.substring(4));
System.out.println("substring(3,6)                  : "+tekst.substring(3,6));
System.out.println("split(\"o\")                      : "+tekst.split("o"));
String[] teksty = tekst.split("o");
for (String t : teksty)   System.out.print(t+"#");
System.out.println();
System.out.println("split(\"o\",2)                    : "+tekst.split("o",2));
String[] teksty2 = tekst.split("o",2);
for (String t : teksty2) System.out.print(t+"#");
System.out.println();
tekst = " "+tekst+" ";
System.out.println("tekst = \" \"+tekst+\" \"          : "+tekst);
System.out.println("trim()                         : "+tekst.trim());
```

Wyrażenia regularne (*regular expressions, regex, regexp*)

- Wyrażenia regularne umożliwiają opis łańcuchy znaków.
- Dzięki nim można opisać pewne zbiory łańcuchów znaków, czasem dość skomplikowane w sposób ogólny, np:
  - PESEL (~jako dokładnie 11 liczb)
  - adresy www (~jako zbiór liter i liczb przedzielonych kropkami)
  - tagi HTML (~ wyrażenia pomiędzy znakami < i >)
  - itd, itp...

Wyrażenia regularne - zastosowania

- Wyrażenia regularne mogą się przydać przy pracy z łańcuchami znaków w wielu sytuacjach, np:
  - Wyszukiwanie określonych sekwencji znaków (dokładnie lub z pewną dowolnością) - np. słów, cytatów, liczb, określonych sekwencji DNA
  - Uzyskiwanie informacji - np. uzyskiwanie konkretnych informacji z odpowiednio sformatowanego tekstu (formularzy, tabel itp)
  - Zamianie łańcuchów znaków
  - Sprawdzanie poprawności wprowadzanych informacji w aplikacjach, stronach internetowych itp
  - .....

Wyrażenia regularne - jak to wygląda

- W wyrażeniach regularnych, jak w każdym języku, określone znaki. lub ich zestawy posiadają konkretne znaczenie, które musimy znać (lub wiedzieć gdzie je znaleźć ;-)) aby je prawidłowo wykorzystać.
- Przykłady:
  - \A - początek łańcucha
  - \d lub [0-9] - cyfra
  - \d+ - jedna lub więcej cyfr
  - ^ - początek linii
  - . -
- Umieszczając wyrażenie regularne w łańcuchu należy pamiętać, że przed znakiem \ (który ma specjalne znaczenie) należy umieścić jeszcze jeden znak \. np: `"\\D+\\d?"` co oznacza: „*Jedna lub więcej nie cyfr a potem zero lub jedna cyfra*”

Wyrażenia regularne (wybór)

- Znaki

wyrażenie	znaczenie
np. A	dany znak (tu konkretnie A)
Ala	dany łańcuch znaków (tu Ala)
\t	tabulacja
\n	nowy wiersz
\r	powrót karetki (powrót do początku wiersza)

Wyrażenia regularne (wybór)

- Klasy znaków

wyrażenie	znaczenie
.	dowolny znak
\s	„biały znak” - gł. spacja, tabulacja, nowy wiersz, powrót karetki
\S lub [^\s]	dowolny znak z wyjątkiem białego znaku
\d lub [0-9]	cyfra
\D lub [^0-9]	dowolny znak z wyjątkiem cyfry
\w lub [a-zA-Z_0-9]	dowolny znak tworzący słowa (litery, cyfry i _)
\W lub [^\w]	dowolny znak nie tworzący słowa (nie litery, cyfry i _)
[xyz] lub (x|y|z)	jeden z podanych (łańcuchów) znaków (tu: x, y lub z)
[^xyz]	dowolny znak , oprócz podanych (tu oprócz: x, y, z)
[a-z]	dowolny znak z zakresu (tu dowolna mała litera)
[a-z0-9]	dowolny znak z zakresów (tu dowolna mała litera lub cyfra)

Uwaga: w powyższych sytuacjach znak ^ oznacza zaprzeczenie a nie początek łańcucha

Wyrażenia regularne (wybór)

- Znaki kotwiczące dopasowanie

wyrażenie	znaczenie
^	początek wiersza
$	koniec wiersza
\b	granica słowa
\B	nie granica słowa

- Znaki określające ilość dopasowań

wyrażenie	znaczenie
?	0 lub 1 raz
+	jeden lub więcej razy
*	0 lub więcej razy
{n}	n - razy
{n,}	n lub więcej razy
{n,m}	między n a m razy

Wyrażenia regularne -przykłady

```
// Jedna nie cyfra
System.out.println("matches(\"\\\\D\")          : "+tekst.matches("\\D"));
// Jedna lub więcej nie cyfr
System.out.println("matches(\"\\\\D+\")         : "+tekst.matches("\\D+"));
// Jedna lub więcej cyfr
System.out.println("matches(\"\\\\d+\")         : "+tekst.matches("\\d+"));
// Jedna lub więcej nie cyfr a potem zero lub więcej cyfr
System.out.println("matches(\"\\\\D+\\\\d*\")    : "+tekst.matches("\\D+\\d*"));
// Jedna lub więcej nie cyfr a potem jedna lub więcej cyfr
System.out.println("matches(\"\\\\D+\\\\d+\")    : "+tekst.matches("\\D+\\d+"));
// Jedna lub więcej nie cyfr a potem zero lub jedna cyfra
System.out.println("matches(\"\\\\D+\\\\d?\")    : "+tekst.matches("\\D+\\d?"));
```

Wyrażenia regularne -przykłady

```
String tekst2 = "Chromosom2";
System.out.println("tekst2                    : "+tekst2);
// Jedna lub więcej nie cyfr a potem jedna lub więcej cyfr
System.out.println("matches(\"\\\\D+\\\\d+\")    : "+tekst2.matches("\\D+\\d+"));
// Jedna lub więcej nie cyfr a potem zero lub jedna cyfra
System.out.println("matches(\"\\\\D+\\\\d?\")    : "+tekst.matches("\\D+\\d?"));
// Jeden lub więcej znaków które mogą być częścią słowa (litery, cyfry i znak _)
System.out.println("matches(\"\\\\w+\")          : "+tekst2.matches("\\w+"));
// Jeden lub więcej znaków które nie mogą być częścią słowa (nie litery, cyfry i znak _)
System.out.println("matches(\"\\\\W+\")          : "+tekst2.matches("\\W+"));
// Jeden lub więcej białych znaków (spacje, tabulatory itd.)
System.out.println("matches(\"\\\\s+\")          : "+tekst2.matches("\\s+"));
// Jeden lub więcej nie białych znaków (nie spacje, tabulatory itd.)
System.out.println("matches(\"\\\\S+\")          : "+tekst2.matches("\\S+"));
// Kropka oznacza jakikolwiek znak
System.out.println("matches(\"Chr.m.s.m2\")  : "+tekst2.matches("Chr.m.s.m2"));
// .+ oznacza jeden lub więcej jakikolwiek znaków
System.out.println("matches(\"Chr.+m2\")     : "+tekst2.matches("Chr.+m2"));
```

Wyrażenia regularne -przykłady

```
// {2,9} oznacza od dwu do dziewięciu elementów wskazanych wcześniej - tu dowolnych
znaków
System.out.println("matches(\"Chr.{2,9}m2\") : "+tekst2.matches("Chr.{2,9}m2"));
// {2,9} oznacza conajmniej dwa elementy wskazane wcześniej - tu dowolnych znaków
System.out.println("matches(\"Chr.{2,}m2\")  : "+tekst2.matches("Chr.{2,}m2"));
// {2,9} oznacza dwa elementy wskazane wcześniej - tu dowolnych znaków
System.out.println("matches(\"Chr.{2}m2\")   : "+tekst2.matches("Chr.{2}m2"));
```

Wyrażenia regularne -przykłady

```
String tekst3 = "chromosom to nie jest ciało które tworzy chrom";
System.out.println(tekst3);
// ^ oznacza początek linii
System.out.println("replaceAll(\"^chrom\", \"CHROM\")      : "+
                   tekst3.replaceAll("^chrom", "CHROM"));
// $ oznacza koniec linii
System.out.println("replaceAll(\"chrom$\", \"CHROM\")      : "+
                   tekst3.replaceAll("chrom$", "CHROM"));
// Dzielimy łańcuch po literce "o" po której następuje jeden znak słowa
System.out.println("split(\"o\\\\w\\\")                    : "+
                   tekst3.split("o\\w"));
String[] teksty = tekst3.split("o\\w");
for (String t : teksty)   System.out.print(t+"#");
System.out.println();
// Dzielimy łańcuch po literce "o" przed którą występuje litera "r" lub "s"
System.out.println("split(\"(r|s)o\")                     : "+
                   tekst3.split("(r|s)o")); teksty = tekst3.split("(r|s)o");
for (String t : teksty)   System.out.print(t+"#");
System.out.println();
```

Wyrażenia regularne -przykłady

```
System.out.println("indexOf(\"o\")      : "+tekst3.indexOf("o"));
int i=0;
ArrayList miejsca = new ArrayList();
while (i>=0) {
    // szukamy nastepnego łańcucha (tu literki "o")
    i=tekst3.indexOf("o",i);
    if (i>=0) {
        miejsca.add(i);
        // przesuwamy indeks o 1, aby nie odszukać ponownie tej samej litery
        i++;
    }

}
System.out.println(miejsca);
```

A teraz na chwilę wrócimy do ArrayList

```
ArrayList<String> lista = new ArrayList<String>(Arrays.asList(
        "Ala",
        "Ola",
        "Ula")
);
System.out.println(lista);


Integer a = 1;
System.out.println(a);
a=a+1;
System.out.println(a);
ArrayList<Integer> lista2 = new ArrayList<Integer>(Arrays.asList(a,2,3));
System.out.println(lista2);
```

Zadanie

- Napisz program który:
  - Zapisze w pliku tekstowym kilka serii danych składających się z szeregu liczb całkowitych
  - Odczyta z pliku dane, rozdzieli je na pojedyncze pomiary i umieści w tablicach `double[]` - każda seria w osobnej tablicy
  - tablice z danymi umieści w ArrayList
  - Wypisze wszystkie dane na ekranie - serie w kolejnych liniach

Zadanie

- Zapis danych w pliku

```
private static void zapiszPlik(String plik) {

        PrintWriter out;
        try {
            out = new PrintWriter(plik);
            out.println("3:4:5:4:5:3:4:4:5:6:4:3:4");
            out.println("4:5:7:5:5:4:4:3:7:6:8:3:4");
            out.println("3:3:4:3:2:2:3:4:4:3:4:3:5");
            out.println("6:6:5:8:9:7:5:4:10:9:6:7:6");
            out.close();
        } catch (FileNotFoundException e) {
            System.out.println("Brak Pliku!");
            e.printStackTrace();
        }
    }
```

Zadanie

- odczyt danych z pliku, umieszczenie w tablicach (osobna metoda) i umieszczenie tablic w ArrayList

```
private static ArrayList<double[]> odczytajDaneZPliku(String plik) {

        File plikDane = new File(plik);
        Scanner skaner;
        ArrayList<double[]> dane = new ArrayList<double[]>();
        try {
            skaner = new Scanner(plikDane);
            while (skaner.hasNextLine()) {
                double[] dTab = podajDane(skaner.nextLine());
                dane.add(dTab);
            }
        } catch (FileNotFoundException e) {
            System.out.println("Brak Pliku do odczytania!");
            e.printStackTrace();
        }
        return dane;
    }
```

Zadanie

- pobieranie danych z łańcucha znaków i umieszczenie ich w tablicy `double[]`

```
private static double[] podajDane(String linia){

    String [] tab = linia.split(":");
    double [] data = new double[tab.length];
    for (int i=0 ; i<tab.length; i++) {
        data[i] = Double.parseDouble(tab[i]);
    }
    return data;
}
```

Zadanie

- wypisanie danych

```
private static void wypiszDane(ArrayList<double[]> dane) {
        System.out.println("Odczytane dane: ");
        for (double[] tab : dane) {
            for (double d : tab) {
                System.out.print(d+" ");
            }
            System.out.println();
        }
    }
```

Zadanie

- metoda główna

```
public static void main(String[] args) {
    String plik = "dane.txt";
    zapiszPlik(plik);
    ArrayList<double[]> dane = odczytajDaneZPliku(plik);
    wypiszDane(dane);
}
```