Wrocław 2018
Cezary Hołub

Wrocławska Wyższa Szkoła Informatyki Stosowanej

Przedmiot	Zaawansowane praktyki programistyczne
Semestr	Lato 2018

Materiały do ćwiczeń - Lab. 1 17.03.2018

Programujemy w Javie, najlepiej używać IDE Eclipse. Ostatnie zadanie jest nieobowiązkowe.
Zadania proszę do wysłać w ciągu 7 dni na adres holub.cezary@gmail.com

ZADANIE 1	Wyświetl liczby od 1 do 100. Jeżeli liczba jest podzielna przez 3, to zamiast liczby wyświetl słowo “Fizz”. Jeżeli liczba jest podzielna przez 5 to zamiast liczby wyświetl słowo “Buzz”. Jeżeli liczba jest podzielna zarówno przez 3 jak i 5, wyświetl “FizzBuzz”

Cel ćwiczenia:

- Napisanie prostego programu z wykorzystaniem instrukcji warunkowych
- Oczekiwany rezultat (fragment):

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
…

Operatory logiczne w Javie:

warunek1 && warunek2	zwróci true gdy zarówno warunek1 = true jak i warunek2 = true
warunek1 || warunek2	zwróci true gdy przynajmniej jeden warunek jest równy true

Uwaga: istnieje kilka sposobów wykonania tego zadania; wyżej ocenione zostanie użycie nie więcej niż

2 instrukcji ‘if’. Pomocna będzie znajomość poniższych metod:

- System.out.println(“tekst”) -- wyświetla tekst i przechodzi do nowej linii
- System.out.print(“tekst”) -- wyświetla tekst i nie przechodzi do nowej linii

ZADANIE 2	Wyświetlenie na konsoli tabliczki mnożenia

Cel ćwiczenia:

- praktyczne wykorzystanie informacji zdobytych w poprzednich ćwiczeniach
- oczekiwany rezultat wyświetlony na konsoli:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	**4**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**
3	**6**	**9**	**12**	**15**	**18**	**21**	**24**	**27**	**30**
4	**8**	**12**	**16**	**20**	**24**	**28**	**32**	**36**	**40**
5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
6	**12**	**18**	**24**	**30**	**36**	**42**	**48**	**54**	**60**
7	**14**	**21**	**28**	**35**	**42**	**49**	**56**	**63**	**70**
8	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**	**80**
9	**18**	**27**	**36**	**45**	**54**	**63**	**72**	**81**	**90**
10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**

- wskazówki
  - instrukcja System.out.print(“a”) wyświetla literę a na konsoli, a instrukcja System.out.println(“a”) wyświetla literę a na konsoli oraz znak nowej linii;
  - pętle można zagnieżdżać w sobie, przykład zagnieżdżonych pętli ‘while’ poniżej:

```
while(warunek1){
    while(warunek2){
       instrukcja;
    }
}
```

ZADANIE 3	Korzystanie z metod klasy *String* i *StringBuffer*

Wymagane wiadomości wstępne:

- Znajomość operacji na tekstach i łańcuchach w Javie

Przebieg ćwiczenia:

Napisz program szyfrujący

1. Zadeklaruj zmienną String która chcesz zaszyfrować, np: “kołek”
2. Zadeklaruj zmienną tekstową z wartością inicjalną: “Zaszyfrowany tekst to: “, do której później dokleisz wynik szyfrowania
3. Użyj szyfru podmieniającego znaki, a następnie szyfruj podmieniając dane litery na inne. Np. tekst “kołek” może zaszyfrować jako “3wp13”, gdzie 3 to k, w to o, p to ł, a 1 to e.
4. Wydrukuj zaszyfrowany tekst

```
package pl.wsis.java;

public class Zadanie3 {

   public static void main(String[] args) {
      String plainText = "kolek";

      //wykorzystaj metody replace() i append() z klas String oraz StringBuffer

      // [UZUPELNIJ KOD]
   }
}
```

ZADANIE 4*	Zaimplementuj metodę sortującą tablicę liczb wykorzystując algorytm sortowania bąbelkowego

Cel ćwiczenia:

- wskazówki
  - deklaracja metody

```
int[] sortuj(int[] tablica)
```

  - opis algorytmu:
    Sortowanie bąbelkowe polega na porównywaniu par elementów leżących obok siebie i -- jeśli jest to potrzebne -- zmienianiu ich kolejności. Czyli w pierwszym przebiegu porównujemy (i ewentualnie zamieniamy):
    - Element pierwszy i drugi
    - Element drugi i trzeci
    - …
    - Element (n-1) i n-ty

    Każdy element jest tak długo przesuwany, aż napotkany zostanie element większy od niego, wtedy w następnych krokach przesuwany jest ten większy element.
    Po pierwszym przebiegu ciąg nie musi być jeszcze uporządkowany, ale na pozycji n znajdzie się największy element ciągu. Zatem w drugim przebiegu można porządkować tylko elementy na pozycjach od 1 do n-1. Po drugim przebiegu, dwa ostatnie elementy są na swoich miejscach, czyli pozostaje posortować elementy na pozycjach od 1 do n-2 itd.
    Jeżeli w danym przebiegu pętli żadna para liczb nie została zamieniona, to ciąg jest już posortowany i można przerwać działanie algorytmu.

  - Przykładowy pseudo-kod:

```
K01   Dla j = n - 1, n - 2, ..., 1: wykonuj K02...K04
K02   p ← false
K03   Dla i = 1, 2, ..., j: jeśli d[i] > d[i + 1], to zamień d[i] z d[i + 1];  p ← true
K04   Jeśli p = false, to zakończ
K05   Zakończ
```

    (uwaga! pseudo-kod zakłada indeksację od 1; w Javie tablice indeksowane są od 0)

  - Przykład działania:

1. Ciąg do posortowania: [7, 5, 10, 1]
2. Pierwszy przebieg pętli:
   a. 7 zamienione z 5: [5, 7, 10, 1]
   b. 7 jest mniejsze niż 10, nie zamieniamy

    c. 10 zamienione z 1: [5, 7, 1, 10]
3. Drugi przebieg pętli:
    a. 5 jest mniejsze niż 7, więc nie zamieniamy
    b. 7 zamienione z 1: [5, 1, 7, 10]
4. Trzeci przebieg pętli:
    a. 5 zamienione z 1: [1, 5, 7, 10]

- Wskazówka: zamiana dwóch elementów tablicy wymaga zmiennej pomocniczej.