

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTEL尼亚

FRAGMENTY KSIĄŻEK ONLINE

Java. Ćwiczenia praktyczne

Autor: Marcin Lis
ISBN: 83-7197-723-9
Format: B5, stron: 166



Chyba każdy, kto interesuje się Internetem, słyszał o Javie. Ten, w końcu stosunkowo młody, język programowania wyjątkowo szybko zdobył sobie bardzo dużą popularność i akceptację ze strony programistów na całym świecie. I choć jego przyszłość wcale nie jest jeszcze przesądzona, wiele wskazuje na to, że Java stanie się jednak dominującą technologią już w niedalekiej przyszłości. Skoro tak, warto poznać ją już teraz. Wiele osób kojarzy Javę tylko z appletami zawartymi na stronach WWW. To jednak tylko część zastosowań. Tak naprawdę to doskonały, obiektowy język programowania, nadający się do różnorodnych zastosowań.

Autor zapoznaje czytelnika z podstawami języka przez serię kilkudziesięciu ćwiczeń. Kurs ten jednak pozwoli zrozumieć istotę języka, a zagadnienia zaawansowane staną się bardziej czytelne, gdy przyszły programista sięgnie po kolejne książki.



Spis treści

Wstęp.....	7
Przyszłość Javy	8
O książce.....	8
Narzędzia	9
Przykłady	9
 Część I Język programowania.....	 11
Rozdział 1. Krótkie wprowadzanie.....	13
Zaczynamy.....	13
B-kod i maszyna wirtualna	14
Java a C++	15
Obiektowy język programowania.....	15
Struktura programu.....	16
 Rozdział 2. Zmienne, operatory i instrukcje	 17
Zmienne	17
Typy podstawowe	17
Deklarowanie zmiennych typów podstawowych	18
Typy odnośnikowe.....	20
Deklarowanie zmiennych typów odnośnikowych	21
Operatory	24
Operatory arytmetyczne.....	24
Operatory bitowe	28
Priorytet operatorów	31
Instrukcje	31
Instrukcja warunkowa if...else	31
Instrukcja switch	35

Pętla for.....	37
Pętla while	40
Rozdział 3. Obiekty i klasy.....	43
Metody.....	44
Konstruktory	50
Specyfikatory dostępu	52
Dziedziczenie.....	55
Rozdział 4. Wyjątki.....	57
Błędy w programach.....	57
Instrukcja try...catch	60
Instrukcja throws	62
Hierarchia wyjątków	63
Część II Aplety.....	65
Rozdział 5. Rysowanie.....	67
Aplikacja a Aplet.....	67
Pierwszy aplet.....	67
Jak to działa?	69
Czcionki.....	70
Rysowanie grafiki.....	73
Kolory	78
Wyświetlanie plików graficznych	80
Rozdział 6. Dźwięki.....	85
Rozdział 7. Animacje.....	89
Pływający napis	89
Pływający napis z buforowaniem.....	93
Zegar cyfrowy.....	94
Animacja poklatkowa	96
Zegar analogowy	98
Rozdział 8. Interakcja z użytkownikiem	103
Obsługa myszy	103
Rysowanie figur (I).....	106
Rysowanie Figur (II)	111
Rysowanie Figur (III)	112

Część III Aplikacje.....	117
Rozdział 9. Okna i menu.....	119
Rozdział 10. Grafika i komponenty.....	133
Klasa Button	134
Klasa TextField.....	136
Klasa TextArea	139
Klasa Label	142
Klasa Checkbox	144
Klasa List.....	146
Rozdział 11. Operacje wejścia-wyjścia.....	151
Wczytywanie danych z klawiatury.....	151
Operacje na plikach	156
Zakończenie	165

Rozdział 2.

Zmienne, operatory i instrukcje

Zmienne

Zmienna jest to miejsce, w którym możemy przechowywać jakieś dane np. liczby czy ciągi znaków. Każda zmienna musi mieć swoją nazwę, która ją jednoznacznie identyfikuje. Każda zmienna ma także swój typ, który informuje o tym, jakiego rodzaju dane można w niej przechowywać. Np. zmienna typu `int` przechowuje liczby całkowite, a zmienna typu `float` liczby zmiennoprzecinkowe. Typy w Javie dzielą się na dwa rodzaje: *typy podstawowe* (ang. *primitive types*) oraz *typy odnośnikowe* (ang. *reference types*).

Typy podstawowe

Typy podstawowe dzielą się na *typy arytmetyczne* oraz *typ boolean*. Zmienna typu `boolean` może przyjmować tylko dwie wartości mianowicie `true` i `false`. `True` oznacza logiczną prawdę, natomiast `false` logiczny fałsz. Typy arytmetyczne dzielą się z kolei na typy całkowitoliczbowe (ang. *integral type*) oraz typy zmiennoprzecinkowe (ang. *floating-point type*). Rodzina typów całkowitoliczbowych składa się z pięciu typów:

- ❖ `byte`,
- ❖ `short`,
- ❖ `int`,
- ❖ `long`,
- ❖ `char`.

W przeciwieństwie do C++ dokładnie określono sposób reprezentacji tych danych. Niezależnie więc od tego, na jakim systemie pracujemy (16-, 32- czy 64-bitowym), dokładnie wiadomo na ilu bitach zapisana jest zmienna danego typu. Wiadomo też dokładnie z jakiego zakresu wartości może ona przyjmować. Nie ma więc dowolności, która w przypadku C mogła prowadzić do sporych trudności przy przenoszeniu programów pomiędzy różnymi platformami. Zakresy poszczególnych typów danych oraz ilość bitów niezbędna do zapisania zmiennych danego typu prezentuje tabela 2.1.

Tabela 2.1. Zakresy typów arytmetycznych w Javie

Typ	Liczba bitów	Liczba bajtów	Zakres
byte	8	1	od -128 do 127
short	16	2	od -32 768 do 32 767
int	32	4	od -2 147 483 648 do 2 147 483 647
long	64	8	od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807

Typ `char` służy do reprezentacji znaków, przy czym w Javie jest on 16-bitowy i zawiera znaki *Unicode*. (*Unicode* to standard pozwalający na zapisanie znaków występujących większości języków świata).

Typy zmiennoprzecinkowe występują tylko w dwóch odmianach:

- ❖ `float` (pojedynczej precyzji),
- ❖ `double` (podwójnej precyzji).

Zakres oraz liczbę bitów i bajtów potrzebnych do zapisu tych zmiennych prezentuje tabela 2.2.

Tabela 2.2. Zakresy dla typów zmiennoprzecinkowych w Javie

Typ	Liczba bitów	Liczba bajtów	Zakres
<code>float</code>	32	4	od -3,4e38 do 3,4e38
<code>double</code>	64	8	od -1,8e308 do 1,8e308

Format danych `float` i `double` jest zgodny z specyfikacją standardu ANSI/IEEE 754. Zapis 3,4e48 oznacza 3,4 razy 10 do potęgi 38.

Deklarowanie zmiennych typów podstawowych

Aby móc użyć jakiegś zmiennej w programie, wpierw trzeba ją zadeklarować, tzn. podać jej typ oraz nazwę. Ogólna deklaracja wygląda w sposób następujący:

```
typ_zmiennej nazwa_zmiennej;
```

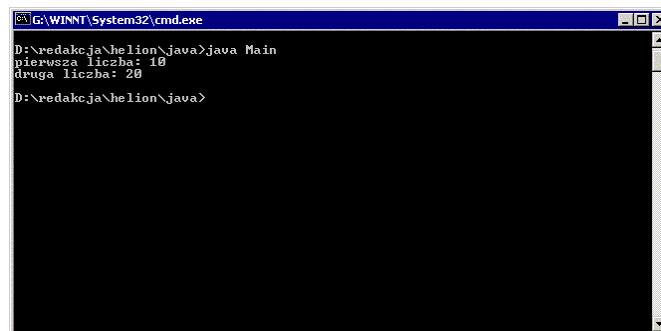
Po takim zadeklarowaniu zmienna jest już gotowa do użycia, tzn. możemy jej przypisywać różne wartości, bądź też wykonywać na niej różne operacje np. dodawanie.

Ćwiczenie 2.1.

Zadeklaruj dwie zmienne całkowite, przypisz im dowolne wartości. Wyświetl wyniki na ekranie (rysunek 2.1).

```
public
class Main
{
    public static void main (String args[])
    {
        int pierwszaLiczba;
        int drugaLiczba;
        pierwszaLiczba = 10;
        drugaLiczba = 20;
        System.out.println ("pierwsza liczba: " + pierwszaLiczba);
        System.out.println ("druga liczba: " + drugaLiczba);
    }
}
```

Rysunek 2.1.
*Wynik działania
programu
z ćwiczenia 2.1*



Instrukcja `System.out.println()` pozwala wyprowadzenie ciągu znaków na ekran. Wartość zmiennej można przypisać już w trakcie deklaracji, pisząc:

```
typ_zmiennej nazwa_zmiennej = wartość;
```

Można również zadeklarować wiele zmiennych danego typu oddzielając ich nazwy przecinkami. Część z nich może też być od razu zainicjalizowana:

```
typ_zmiennej nazwa1, nazwa2, nazwa3;
typ_zmiennej nazwa1 = wartość1, nazwa2, nazwa3 = wartość2;
```

Zmiennej w Javie, podobnie jak w C, czy C++, ale inaczej niż w Pascalu, można deklarować w dowolnym miejscu funkcji czy metody.

Ćwiczenie 2.2.

Zadeklaruj i jednocześnie zainicjalizuj dwie zmienne typu całkowitego. Wynik wyświetl na ekranie.

```
public
class Main
{
    public static void main (String args[])
```

```
{
    int pierwszaLiczba = 10;
    int drugaLiczba = 20;
    System.out.println("pierwsza liczba: " + pierwszaLiczba);
    System.out.println("druga liczba: " + drugaLiczba);
}
```

Ćwiczenie 2.3.

Zadeklaruj kilka zmiennych typu całkowitego w jednym wierszu. Kilka z nich zainicjalizuj.

```
public
class Main
{
    public static void main (String args[])
    {
        int pierwszaLiczba = 10, drugaLiczba = 20, i, j, k;
        System.out.println("pierwsza liczba: " + pierwszaLiczba);
        System.out.println("druga liczba: " + drugaLiczba);
        System.out.println("zmienna i: " + i);
        System.out.println("zmienna j: " + j);
        System.out.println("zmienna k: " + k);
    }
}
```

Przy nazywaniu zmiennych obowiązują pewne zasady. Otóż nazwa taka może składać się z dużych i małych liter oraz cyfr, ale nie może się zaczynać od cyfry. Nie należy również stosować polskich znaków diakrytycznych. Nazwa zmiennej powinna także odzwierciedlać funkcję pełnioną w programie. Np. jeżeli określa ona liczbę punktów w jakimś zbiorze, to najlepiej ją nazwać `liczbaPunktow` lub nawet `liczbaPunktowWZbiorze`. Mimo że tak długa nazwa może wydawać się dziwna, jednak bardzo poprawia czytelność programu oraz ułatwia jego analizę. Naprawdę warto ten sposób stosować. Przyjmuje się też, co również jest bardzo wygodne, że nazwę zmiennej rozpoczynamy małą literą, a poszczególne człony tej nazwy (wyrazy, które się na nią składają) piszemy literą wielką. Dokładnie tak jak w powyższych przykładach.

Typy odnośnikowe

Typy odnośnikowe (ang. *reference types*) dzielą się na trzy rodzaje:

- ❖ typy klasowe (*class types*),
- ❖ typy interfejsowe (*interface types*),
- ❖ typy tablicowe (*array types*).

Interfejsami nie będziemy się zajmować. Zacznijmy od typów tablicowych. Tablice są to wektory elementów danego typu i służą do uporządkowanego przechowywania wartości tego typu. Mogą być jedno bądź wielowymiarowe. Dostęp do danego elementu tablicy jest realizowany poprzez podanie jego indeksu, czyli miejsca w tablicy w którym

się on znajduje. Dla tablicy jednowymiarowej będzie to po prostu kolejny numer elementu, dla tablicy dwuwymiarowej trzeba już podać numer wiersza i kolumny itd. Jeśli chcemy zatem przechować w programie 10 liczb całkowitych, najwygodniej będzie użyć w tym celu 10-elementową tablicę typu `int`.

Typy klasowe pozwalają na deklarowanie zmiennych obiektowych. Zajmiemy się nimi bliżej w rozdziale 3.

Deklarowanie zmiennych typów odnośnikowych

Zmienne typów odnośnikowych deklarujemy podobnie jak w przypadku zmiennych typów podstawowych tzn. pisząc:

```
typ_zmiennej nazwa_zmiennej;
```

lub

```
typ_zmiennej nazwa_zmiennej_1, nazwa_zmiennej_2, nazwa_zmiennej_3;
```

Pisząc w ten sposób zadeklarowaliśmy jednak jedynie tzw. *odniesienie* (ang. *reference*) do zmiennej obiektowej, a nie samą zmienną! Takiemu odniesieniu przypisana jest domyślnie wartość pusta (`null`). Czyli praktycznie nie możemy wykonywać na niej żadnej operacji. Dopiero po utworzeniu odpowiedniego obiektu w pamięci możemy powiązać go z tak zadeklarowaną zmienną. Jeśli zatem napiszemy np.

```
int a;
```

mamy gotową do użycia zmienną typu całkowitego. Możemy jej przypisać np. wartość 10. Żeby jednak móc skorzystać z tablicy musimy zadeklarować zmienną odnośnikową typu tablicowego, utworzyć obiekt tablicy i powiązać go ze zmienną. Dopiero wtedy będziemy mogli swobodnie odwoływać się do kolejnych elementów. Pisząc zatem:

```
int tablica[];
```

zadeklarujemy odniesienie do tablicy, która będzie zawierała elementy typu `int`, czyli 32-bitowe liczby całkowite. Samej tablicy jednak jeszcze wcale nie ma. Przekonamy się o tym wykonując kolejne ćwiczenia.

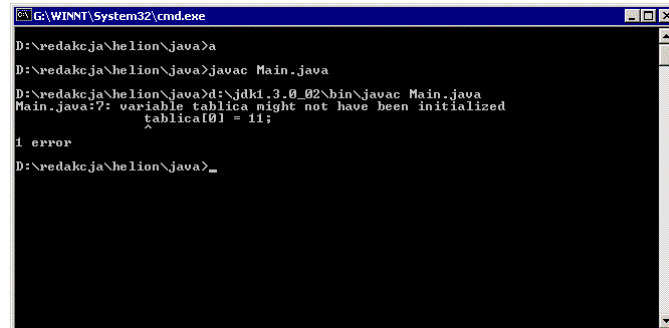
Ćwiczenie 2.4.

Zadeklaruj tablicę elementów typu całkowitego. Przypisz zerowemu elementowi tablicy dowolną wartość. Spróbuj wyświetlić zawartość tego elementu na ekranie.

```
public
class Main
{
    public static void main (String args[])
    {
        int tablica[];
        tablica[0] = 11;
        System.out.println ("Zerowy element tablicy to: " + tablica[0]);
    }
}
```

Rysunek 2.2.

*Błąd kompilacji.
Nie zainicjowaliśmy
zmiennej tablica*



```
G:\WINNT\System32\cmd.exe
D:\redakcja\helion\java>a
D:\redakcja\helion\java>javac Main.java
D:\redakcja\helion\java>d:\jdk1.3.0_02\bin\javac Main.java
Main.java:7: variable tablica might not have been initialized
        tablica[0] = 11;
        ^
1 error
D:\redakcja\helion\java>_
```

Już przy próbie kompilacji kompilator uprzejmie poinformuje nas, że chcemy odwołać się do zmiennej, która prawdopodobnie nie została zainicjalizowana wypisując na ekran `Variable tablica may not have been initialized` (rysunek 2.2). Widzimy też wyraźnie, że w razie wystąpienia błędu na etapie kompilacji, otrzymujemy kilka ważnych i pomocnych informacji. Przede wszystkim jest to nazwa pliku, w którym wystąpił błąd (jest to ważne, gdyż program może składać się bardzo wielu klas, a każda z nich jest definiowana w oddzielnym pliku), numer wiersza w tym pliku oraz konkretne miejsce wystąpienia błędu. Na samym końcu kompilator podaje też całkowitą liczbę błędów.