Java od Zera

Paweł Ćwik

Java od Zera

Paweł Ćwik

Ta książka jest do kupienia na http://leanpub.com/javaodzera

Wersja opublikowana 2022-11-13

Spis treści

Java?

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Kanał Discord

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Proces kompilacji kodu

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Instalacja niezbędnych narzędzi i pierwszy projekt

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

JDK

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

IDE

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Utworzenie pierwszego projektu

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Pierwszy program

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Błędy kompilacji

Zawartość nie jest dostępna w przykładowej książce. Książkę można zakupic na platformie Leanpub pod adresem http://leanpub.com/javaodzera.

Zmienne

Bloki budulcowe programu

Powtórzmy to co zrobiliśmy razem do tej pory i przy okazji zobaczymy z jakich bloków budulcowych składa się nasz mały program.

Pierwszy program

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3          System.out.println("Witaj");
4      }
5  }
```

Klasy

Całość tego kodu, od słowa `class` po ostatni, klamrowy nawias zamykający, jest to tak zwana **klasa**.

Struktura klasy

```
1  class HelloWorld {
2
3  }
```

Słowo *class* oznacza, że w tym pliku definiujemy klasę, jest to tak zwane *słowo kluczowe*. Oznacza to tyle, że jest ono zarezerwowane i nie można tak nazywać innych klas, funkcji, zmiennych i innych rzeczy używanych przez programistę w ramach tworzenia programu.

Kolejnym punktem programu jest `HelloWorld`. Jest to nazwa własna klasy. Wszelkie nazwy własne w programowaniu, niezależnie od języka, przyjęło się nadawać w języku angielskim i tak też będziemy robić.

Sama nazwa powinna opisywać co dana klasa opisuje, czyli na przykład `Customer` dla klasy z danymi o kliencie, `Car` dla klasy z informacjami o samochodzie, `HelloWorld` dla klasy wypisującej *Hello World* na ekranie.

Ważną rzeczą jest pisanie za pomocą notacji zwanej *CamelCase*, czyli jeśli nazwa klasy składa się z więcej niż jednego słowa powinniśmy połączyć je w jeden ciąg znaków, gdzie każde słowo rozpoczyna się wielką literą. Na przykład *HelloWorld* albo *DiscountCode*.

Kolejną istotną rzeczą jest to, że w jednym pliku powinna znajdować się jedna klasa, a sam plik powinien nazywać się tak jak sama klasa, czyli klasa `HelloWorld` powinna znajdować się w pliku *HelloWorld.java*.

Metody / funkcje

Wchodząc w głąb naszej klasy napotkamy kolejny blok budulcowy, a mianowicie metodę `main`

Metoda main

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3          System.out.println("Witaj");
4      }
5  }
```

Od razu poruszymy kwestię nomenklatury. W świecie Javy można używać naprzemiennie słowa metoda i funkcja. Technicznie istnieją pewne różnice, ale na tym etapie zupełnie nas one nie interesują.

`public`, `static` i `void` to kolejny zestaw słów zastrzeżonych, do których dotrzemy w dalszej części tej książki. Natomiast `main` to nazwa metody.

Podobnie jak w przypadku klas piszemy ją z angielska i ma jak najlepiej opisywać co klasa robi. Istotna różnica jest taka, że nazwy metod zaczynamy małą literą, więc `ToJestNazwaKlasy`, a `toJestNazwaMetody`.

Jeszcze uwaga co do nazywania czy to klas, czy metod, czy zmiennych. Wbrew pozorom nie jest to rzecz łatwa i nie przejmuj się jeśli wymyślenie nazwy dla klasy czy funkcji zajmie Ci dużo czasu i nigdy nie będzie zadowalające. Wprawa przychodzi z praktyką.

Następnie, pomiędzy parą nawiasów okrągłych `( )` mamy parametry. Są to dane wejściowe, jakie na początku działania otrzyma nasza funkcja, by potem na nich operować. Może ich być dowolna ilość, posiadają one typ (w tym wypadku `String[]`, a o samych typach wkrótce) oraz nazwę własną (`args`).

Jeśli parametrów byłoby więcej niż jeden, to wówcza owe pary `typ nazwa` oddzielane są przecinkiem.

To, co znajduję się pomiędzy parą nawiasów klamrowych `{ }` jest to tak zwane ciało metody. W tym wypadku jest to `System.out.println("Witaj");` czyli kawałek kodu, który na okno terminala wypisze *Witaj Pawle*.

Co niezmiernie istotne `main` jest to specjalna funkcja, od które rozpoczyna się wykonywanie programu i w programie zawsze musi być funkcja `public static void main(String[] args)`.

Deklaracja pierwszej zmiennej

Rozbudujmy nasz pierwszy program do takiej oto formy:

Nowa wersja pierwszego programu

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

        System.out.println("Imię: Paweł");
        System.out.println("Wiek: 35");
        System.out.println("Żonaty: tak");

    }
}
```

Efektem działania tego programu będzie pojawienie się na konsoli takiego oto tekstu:

```
Imię: Paweł
Wiek: 35
Żonaty: tak
```

Oczywiście takie programy, które zarabiają na siebie i często na całą firmę, nigdy nie składają się tylko i wyłącznie z wypisywania statycznego tekstu na ekran. Poza wspomnianymi powyżej dwoma blokami budulcowymi mamy i trzeci, najważniejszy, bo istniejący w każdym języku programowania i pozwalający na tworzenie zaawansowanych programów. Mowa mianowicie o zmiennych.

Jest to blok budulcowy naszego programu, który podczas jego działania zmienia, nomen omen, wartość, jaka się pod nim kryje.

Przykładowa deklaracja zmiennej wygląda w taki oto sposób

```
String name = "Paweł";
```

Pierwszym elementem deklaracji nowej zmiennej jest jej typ, czyli jaka wartość będzie w danej zmiennej przechowywana. Może to być tekst, może to być liczba, może to być wartość logiczna, czyli

prawda albo fałsz. W tym wypadku słowo `String` mówi nam, że mamy do czynienia ze zmienną, która będzie przechowywać typ tekstowy.

Kolejnym elementem jest `name`, jest to nazwa zmiennej. Podobnie jak w przypadku wszystkich innych nazw własnych w programie piszemy ją z angielska oraz, tak jak w przypadku funkcji, piszemy *camelCase*em, czyli pierwsze słowo małą literą.

Znak = jest w programowaniu, nie tylko w Javie, oznacza przypisanie wartości. W tym wypadku do zmiennej o nazwie *name* przypisujemy wartość tekstową *Paweł*. W Javie wartości tesktowe umieszczamy w cudzysłowach *"*.

Po dodaniu zmiennej nasz program wygląda tak

Program z deklaracją zmiennej

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3
4          String name = "Paweł";
5
6          System.out.println("Imię: Paweł");
7          System.out.println("Wiek: 35");
8          System.out.println("Żonaty: tak");
9
10     }
11 }
```

IntelliJ IDE może kolorować nazwę zmiennej na szaro. Oznacza to, że dana zmienna jest nieużywana w programie. Podobnie oznaczane są nieużywane funkcje.

Użycie zmiennej w programie

Oczywiście sama deklaracja zmiennej za wiele nam nie pomoże, trzeba jej użyć gdzieś w programie. Zrobimy to w taki sposób:

Użycie zmiennej przy wypisywaniu tekstu

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3
4          String name = "Paweł";
5
6          System.out.println("Imię: " + name);
7          System.out.println("Wiek: 35");
8          System.out.println("Żonaty: tak");
9
10     }
11 }
```

Efekt działania programu jest niezmienny, na konsoli zostanie wypisane

```
1  Imię: Paweł
2  Wiek: 35
3  Żonaty: tak
```

Jednak tym razem imię jest brane ze zmiennej, a nie podane jako stała wartość.

W przypadku `"Imię: " + name` symbol `+` oznacza łączenie wartości tekstowych. W trakcie działania programu, gdy dotrze on do wspominanej wyżej linii to zamienia zmienną `name` na wartość jaka się pod nią kryje, czyli w tym wypadku *Paweł*. Powstaje więc kod `"Imię :" + "Paweł"`, który jest następnie wykonywany i powstaje `"Imię: Paweł"`, co wraz z użyciem `System.out.println` powoduje wypisanie na konsoli tekstu *Imię: Paweł*.

Dodajmy do programu kolejne zmienne dla określenia danych osobowych.

Więcej zmiennych

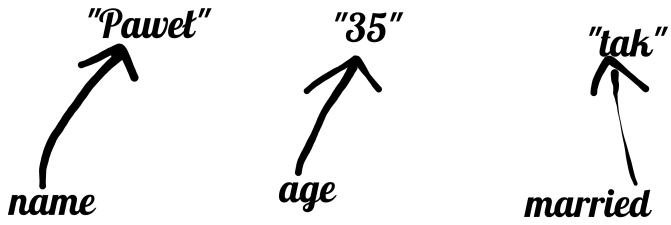
```
public class HelloWorld {
    public static void main(String[] args) {

        String name = "Paweł";
        String age = "35";
        String married = "tak";

        System.out.println("Imię: " + name);
        System.out.println("Wiek: " + age);
        System.out.println("Żonaty: " + married);

    }
}
```

Efekt działania programu pozostaje bez zmian, natomiast mamy już trzy zmienne dla określenia różnych danych o naszej osobie.

Fool around Pozmieniaj wartości zmiennych na różne inne i zobacz jak zmieni się działanie programu.

Działanie zmiennych

W praktyce każdy program ma wydzielony fragment pamięci, w którym przechowuje aktualnie dostępne zmienne i ich wartości.



Przypisanie zmiennych

I tak gdy aplikacja dociera do 8, 9, 10 linii sprawdza w owym wydzielonym miejscu, jaka to wartość kryje się pod daną nazwą zmiennej, następnie podstawia tę wartość w miejsce nazwy zmiennej i wykonuje wskazaną operację. W wypadku tego prostego programu jest to łączenie tekstu, a następnie wyświetlenie go na ekranie.

Typy zmiennych

Poznaliśmy już typ tekstowy *String*, jednak w Javie istnieje ich więcej, te najpopularniejsze to:

`String` - typ tekstowy, przykładowe wartości `"Paweł"`, `"abv 123"`, `"34"`
`int` - typ oznaczający liczby całkowitych, przykładowe wartości to `1`, `34`, `-5`
`boolean` - typ logiczny, który może posiadać tylko jedną z dwóch wartości: `true` i `false`.
`char` - typ oznaczający pojedynczy znak, przykładowe wartości to `a`, `+`, `1`

`double` - typ oznaczający liczby zmiennoprzecinkowe, na przykład `12.12`, `43.05`

Wszystkie powyższe typy (wraz z kilkoma mniej popularnymi, którymi obecnie nie musimy się przejmować) są to tak zwane typy prymitywne. Bazowe typy na podstawie których można tworzyć bardziej zaawansowane typy.

Uzbrojeni w tę, wiedzę możemy dostosować nasz program, by typ zmiennej faktycznie oddawał wartość, jaka ma być tam przechowywana.

Dostosowanie typu zmiennych

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3
4          String name = "Paweł";
5          int age = 35;
6          boolean married = true;
7
8          System.out.println("Imię: " + name);
9          System.out.println("Wiek: " + age);
10         System.out.println("Żonaty: " + married);
11
12     }
13 }
```

Wynik działania programu nieco się różni się od dotychczasowego

```
1  Imię: Paweł
2  Wiek: 35
3  Żonaty: true
```

Teraz zamiast *tak* w ostatnim polu mamy *true*, ponieważ wartośc typu boolean konwertowane wprost do wyświetlanego tekstu mają postać *true* albo *false*. Oczywiście za jakiś czas dowiesz się jak

zrobić tak, żeby dostosować wyświetlany tekst do tego, jaka jest wartość zmiennej.

Można by się zastanawiać, po co nam te typy skoro wszystko mogłoby być tekstem?

Głównie chodzi o to, by ograniczyć możliwe wartości jakie może przybierać zmienna oraz o fakt, ze różne prymitywy pozwalają na różne operacje. Na przykład na liczbach możemy używać operacji arytmetycznych, typ logiczny pozwala na korzystanie z wyrażeń warunkowych, a *String* ma cały zestaw operacji na tekście, gdzie łączenie go za pomocą *+* to tylko początek.

W deklaracji głównej funkcji programu `public static void main(String[] args)` występuje para nawiasów kwadratowych `[]` po typie `String`. Oznaczają one, że mamy do czynienia z tak zwaną tablicą danego typu, co przekłada się na fakt, że owa zmienna może przechowywać więcej niż jedną wartość danego typu. O tablicach w dalszej części.

Duplikacja zmiennych

Ostatnia rzeczą, jaką musisz wiedzieć na tym etapie to fakt, że nazwy zmiennych nie mogą się powtarzać.

Taki kod spowoduje więc błąd kompilacji

Zduplikowana nazwa zmiennej

```
1  public class HelloWorld {
2      public static void main(String[] args) {
3
4          String name = "Paweł";
5          int age = 35;
6          boolean married = true;
7
8          System.out.println("Imię: " + name);
```

```
9           System.out.println("Wiek: " + age);
10          System.out.println("Żonaty: " + married);
11
12          String name = "Tomek";
13
14      }
15  }
```

podobnie jak i taki

Zduplikowana nazwa zmiennej

```
1   public class HelloWorld {
2       public static void main(String[] args) {
3
4           String name = "Paweł";
5           int age = 35;
6           boolean married = true;
7
8           System.out.println("Imię: " + name);
9           System.out.println("Wiek: " + age);
10          System.out.println("Żonaty: " + married);
11
12          int name = 2;
13
14      }
15  }
```

Typ nie ma tu znaczenia, nazwa musi być unikalna. Konkretniej musi być unikalna w danym *scope*, czyli zakresie, który jest określany przez parę nawiasów klamrowych. W naszym przypadku jest to cała funkcja `main`. W ten temat zagłębimy się bardziej nieco później.

Praca ze zmiennymi

Bardzo ważną rzeczą jest fakt, że zmienne mogą ze sobą współpracować w naszym programie.

Na przykład

```
1  public class Calc {
2      public static void main(String[] args) {
3
4          int x = 5;
5          int y = 10;
6
7          int sum = x + y;
8
9          System.out.println(sum);
10
11     }
12 }
```

Tworzymy dwie zmienne o konkretnych wartościach `x` `y`, a następnie tworzymy trzecią `sum`, której to wartość będzie sumą dwóch poprzednich.

Program w efekcie swojego działania wypisze na okno konsoli `15`.

Przechodząc krok po kroku przez jego działania to o to co się dzieję:

1. Tworzona jest zmienna `x` o wartości *5*.
2. Tworzona jest zmienna `y` o wartości *10*.
3. Tworzona jest zmienna `sum`.
4. Jej wartość to `x + y`, więc program podstawia wartości w odpowiednie miejsca i dostaje `5 + 10`.
5. Program wykonuje operację dodawania czego efektem jest `15`.
6. To zmiennej `sum` przypisywana jest warość `15`.

Zadanie praktyczne

Treść

W ramach zadania praktycznego rozbuduj istniejący program o dane dwóch kolejnych osób, czyli dodaj dwa komplety zmiennych (imie, wiek, stan cywilny), każdy określający inną osobę (mający inne wartości) i wyświetl je na ekran w takim sam spoób jak robimy do to tej pory.

Rozwiązanie

Więcej danych

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

        String name = "Paweł";
        int age = 35;
        boolean married = true;

        System.out.println("Imię: " + name);
        System.out.println("Wiek: " + age);
        System.out.println("Żonaty/a: " + married);

        String secondName = "Kinga";
        int secondAge = 36;
        boolean secondMarried = true;

        System.out.println("Imię: " + secondName);
        System.out.println("Wiek: " + secondAge);
        System.out.println("Żonaty/a: " + secondMarried);

        String thirdName = "Tomek";
```

```
21          int thirdAge = 25;
22          boolean thirdMarried = false;
23
24          System.out.println("Imię: " + thirdName);
25          System.out.println("Wiek: " + thirdAge);
26          System.out.println("Żonaty/a: " + thirdMarried);
27
28      }
29  }
```