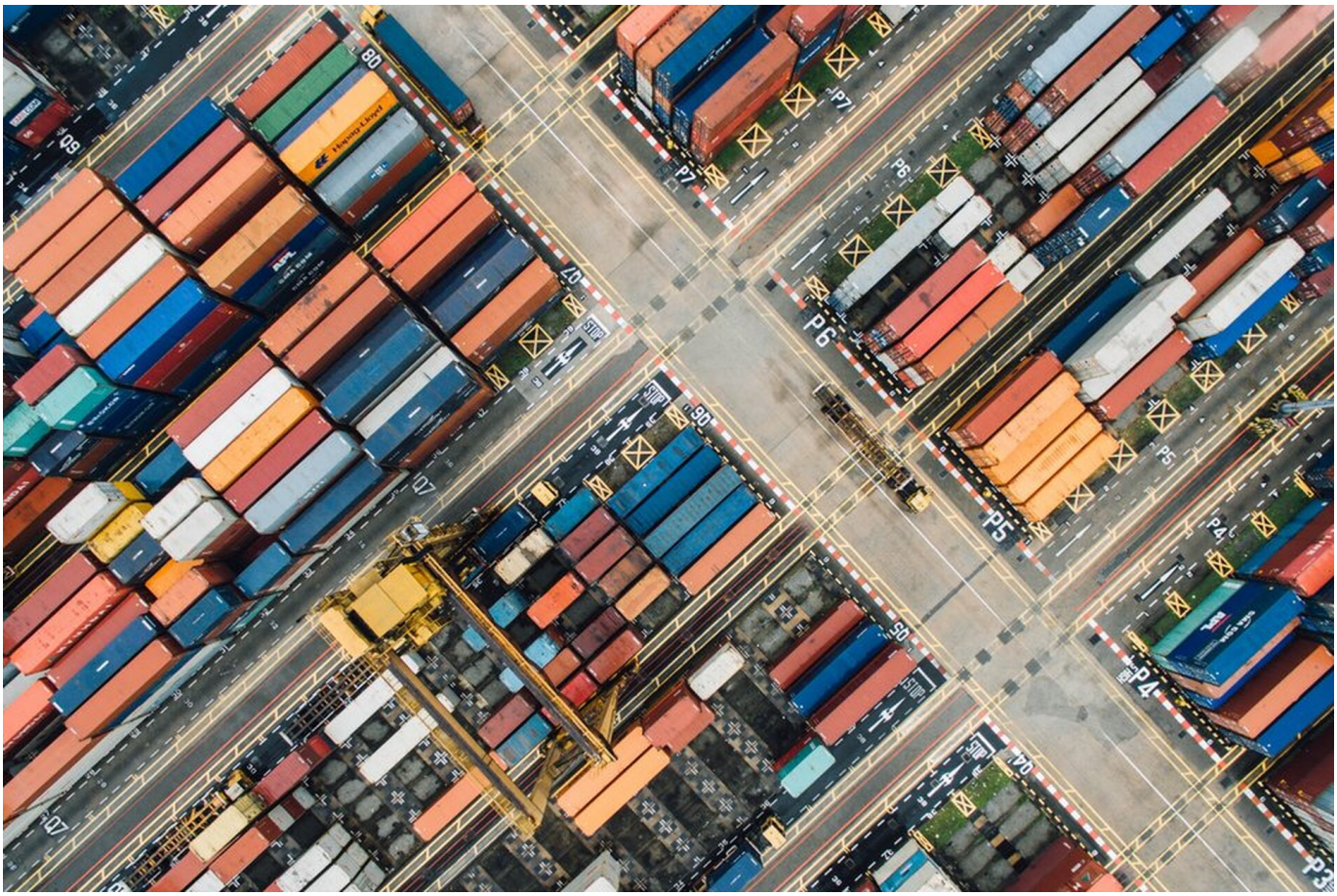


Podstawowe typy zmiennych w języku Java

Źródło: Chuttersnap, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Wiesz już, że programy wykorzystują pojemniki na dane – zmienne. Dowiedzieliśmy się tego z e-materiału Podstawowe typy zmiennych.

Czy wiesz jednak, jaka jest ich specyfika w danym języku programowania?

W tym e-materiale zapoznamy się z podstawowymi typami zmiennych w języku Java.

Charakterystykę podstawowych typów zmiennych w innych językach programowania znajdziesz w e-materiałach:

- Podstawowe typy zmiennych w języku C++,
- Podstawowe typy zmiennych w języku Python.

Więcej zadań? Sięgnij do Podstawowe typy zmiennych – ćwiczenia.

> **Twoje cele**
>
> - Przeanalizujesz różnice między zmiennymi w języku Java.
> - Dobierzesz do realizowanego algorytmu zmienne właściwego typu.
> - Rozwiążesz proste problemy z zastosowaniem zmiennych w języku Java.

Przeczytaj

Problem 1

Napisz program, który będzie przechowywał cztery zmienne zgodnie ze specyfikacją.

Specyfikacja:

Dane:

- `znakZodiaku` – zmienna typu `String`; informuje o znaku zodiaku użytkownika,
- `liczbaBraci` – zmienna typu `int`; przechowuje dane o liczbie braci użytkownika,
- `waga` – zmienna typu `double`; informuje o wyrażonej w kilogramach wadze użytkownika,
- `czySamochod` – zmienna typu `bool`; zawiera informację, czy użytkownik ma samochód.

Wynik:

Stworzone na podstawie specyfikacji zmienne, zainicjowane wybraną wartością początkową.

Polecenie 1

Porównaj swoje rozwiązanie z filmem.

Film dostępny pod adresem /preview/resource/R1IODkhO0WwZH

Film nawiązujący do treści materiału podstawowe typy zmiennych.

Kod programu zaprezentowanego w filmie:

Plik o rozmiarze 261.00 B w języku polskim

Podsumowanie informacji na temat zmiennych w języku Java

Typy danych w języku Java

Ponieważ zmienne są używane do przechowywania danych, niezbędne jest omówienie typów danych w języku Java. Należą do nich liczby, znaki lub ciągi znaków o różnym zakresie oraz długości, zajmujące różną ilość miejsca w pamięci komputera, co wpływa na wydajność programów. Typy danych dzielimy na:

- proste (prymitywne),
- złożone (obiektowe), które w uproszczeniu opiszemy jako składające się z typów prostych.

Do typów prostych zaliczamy typy numeryczne (mieszczące dane liczbowe):

- liczby całkowite:

a. `byte` (1 bajt) – zakres od -127 do 128,
b. `short` (2 bajty) – zakres od -32768 do 32767,
c. `int` (4 bajty) – zakres od -2147483648 do 2147483647,
d. `long` (8 bajtów) – zakres od -2^{63} do $2^{63}-1$ (mają przyrostek `L` lub `l`)

a. `float` (4 bajty) – maksymalnie 7 liczb po przecinku,
b. `double` (8 bajtów) – maksymalnie 15 cyfr po przecinku.

`Float` i `double` oprócz tego, że różnią się ilością zajmowanego miejsca w pamięci, są typami o różnej precyzji; `double` jest dokładniejszy.

Zazwyczaj w programach tworzonych w języku Java dla typów całkowitych używany jest `int`, a dla typów zmiennoprzecinkowych `double`. Typ `int` jest na tyle duży, że w większości przypadków pomieści potrzebne wartości, a jednocześnie nie spowolni działania programu; `double` natomiast jest dostatecznie precyzyjny.

> **Ważne!**
>
> Przy deklarowaniu zmiennych niektórych typów konieczne jest dodanie przyrostka:
>
> - `L` lub `l` dla zmiennej `long`: `long l = 6L;`
> - `F` lub `f` dla zmiennej `float`: `float f = 6.89f;`
> - `D` lub `d` dla zmiennej `double`: `double d = 34.77d;`
>
> **Dla zainteresowanych**

> Z `double` większość IDE sobie poradzi bez przyrostka `d`, jednak zawsze warto go dodać dla bezpieczeństwa i większej czytelności kodu.
> **Ważne!**
>
> Wartości liczb zmiennoprzecinkowych zapisujemy z użyciem kropki, a nie – jak w języku polskim – przecinka. Próba użycia przecinka spowoduje błąd.

Kolejnym typem prostym jest `char` (z ang. *character*) – znak. Mieści on w sobie pojedynczy znak, który musi być umieszczony pomiędzy apostrofami, np. `'c'`. Może także być zapisany w postaci liczby szesnastkowej lub dziesiętnej odpowiadającej kodowi danego znaku z tabeli unikodu, np. U + 0063 to `'c'` w unikodzie.

Pierwszy sposób zapisu zmiennych typu `char` jest zdecydowanie prostszy i dużo częściej stosowany przez programistów.

> **Ważne!**
>
> Istnieją także znaki specjalne, które deklarujemy z użyciem znaku backslash (\). Jest to znak ucieczki (ang. *escape character*), który zmienia interpretację kolejnego znaku. Dzięki temu możemy zapisać poniższe znaki specjalne:
>
> - `\n` – nowa linia,
> - `\t` – tabulacja,
> - `\r` – powrót karetki.
>
> Znak \ pozwala również zapisać w zmiennej znaki, które pełnią specjalne znaczenie w języku programowania:
>
> - `\` – backslash,
> - `\'` – apostrof,
> - `\"` – cudzysłów.

Do zmiennych typu prymitywnego zaliczamy także wartości logiczne (boolowskie) `boolean`, które mogą przyjmować tylko `true` i `false`. Możemy zarówno zapisać w nich wartość logiczną (`prawda` lub `fałsz`), jak również z ich pomocą sprawdzać, czy dane wyrażenie jest prawdziwe. Do tego celu używamy operatora porównania.

Kolejnym typem danych jest ciąg znaków, czyli `String`.

W odróżnieniu od wszystkich poprzednich `String` nie jest typem prymitywnym, a obiektowym – język Java jest obiektowy i to właśnie takie dane są jego główną składową.

`String` łatwo odróżnić od innych typów danych choćby dlatego, że jak wszystkie typy klasowe (obiektowe) jest zapisywany wielką literą. Przechowuje on ciągi dowolnych znaków, a jego wartość musi być umieszczona w cudzysłowie.

> **Ważne!**

> Jak już wiemy, wartość zmiennej typu `String` musi być zapisana w cudzysłowie. Jeśli jednak chcemy, aby w ciągu znaków znalazł się cudzysłów, musimy go poprzedzić znakiem ucieczki backslash: `"Cytuję: \"znak ucieczki jest potrzebny\"."`.

Jak tworzymy zmienne?

Zmienne w języku Java, niezależnie od typu, tworzymy według schematu:

```
typDanych nazwaZmiennej = wartość;
```

W przykładowej linii kodu utworzyliśmy zmienną w jednym kroku, dokonując jednoczesnej deklaracji i inicjalizacji.

```
int x = 10;
```

Natomiast faktycznie najpierw następuje **deklaracja** zmiennej:

```
int x;
```

kiedy zostaje zarezerwowane miejsce w pamięci komputera dla zmiennej `x`, a następnie następuje **inicjalizacja** zmiennej:

```
x = 10;
```

kiedy zostaje jej przypisana podana wartość.

Najczęściej deklaracja i inicjalizacja zmiennej następują w jednym kroku. Czasem przydaje się możliwość wcześniejszej deklaracji zmiennej i użycie jej w dalszej części programu.

Dla zmiennej typu `float` będzie to wyglądało w następujący sposób:

```
float z = 4.5f;
// lub
float z = 4.5F; // pamiętamy o przyrostku, inaczej program nie sk
```

Analogicznie dla zmiennej przechowującej wartość logiczną:

```
boolean jestPelnoletni = true;
```

W celu sprawdzenia wartości logicznej przy pomocy zmiennej `boolean`, możemy napisać:

```
int x = 4;
int y = 6;
boolean czyJestWieksze = x > y;
```

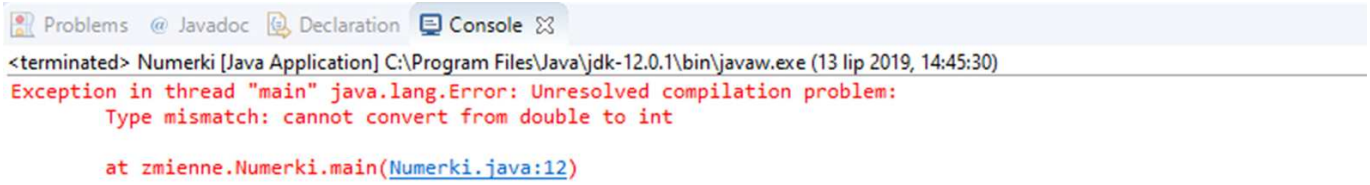
Jeśli wypiszemy wynik do konsoli:

```
System.out.println(czyJestWieksze);
```

otrzymamy oczywiście `false`.

> **Ważne!**
>
> Aby zmienna została zainicjowana, musimy pamiętać o średniku – taka jest składnia języka Java.

Java jest językiem o ścisłej kontroli typów, dlatego tworzenie w niej zmiennych jest bardziej skomplikowane pod względem zapisu niż np. w języku Python. Podajemy zawsze typ deklarowanej zmiennej, a w niektórych typach danych dodajemy przyrostki. W ten sposób kompilator dowiaduje się, że typ zmiennej jest zgodny z przypisaną wartością. Jeśli nie podamy typu lub przypiszemy wartość innego typu, program wyrzuci błąd `type mismatch`, czyli niezgodność typów.



> **Dla zainteresowanych**
>
> W języku Java od wersji 10 pojawiła się funkcjonalność o nazwie `var`. W przeciwieństwie do innych języków, np. JavaScript, nie ma ona w tym wypadku nic wspólnego z dynamicznym typowaniem. Jest to jedynie nowa funkcjonalność języka Java, skracająca zapis, np. przy tworzeniu zmiennych obiektowych.
> **Ciekawostka**

> Jeśli potrzebujemy zadeklarować wiele zmiennych tego samego rodzaju, możemy to zrobić w jednej linijce, np.
>
> ```
> double wiek, wzrost, pensja;
>
> // zamiast
> double wiek;
> double wzrost;
> double pensja;
> ```
>
> Może to skrócić kod o kilka linijek. Wartości oczywiście musimy przypisać do każdej zmiennej indywidualnie.
> **Ciekawostka**
>
> Jeśli musimy zadeklarować zmienną numeryczną o bardzo dużej wartości, to dopuszczalny jest zapis z użyciem znaku podkreślenia, np.
>
> ```
> long duzaLiczba = 2_344_334_789L
> ```
>
> co zwiększa czytelność kodu – należy jednak pamiętać, że nie można w tym celu stosować kropki (jak w języku polskim), ponieważ kropka w języku Java oddziela część całkowitą liczby od ułamkowej.

Zmienne „niezmienne”

W języku Java zmienne domyślnie są modyfikowalne, np. gdy stworzymy zmienną:

```
int wiek = 56;
```

następnie zmienimy ją na:

```
wiek = 57;
```

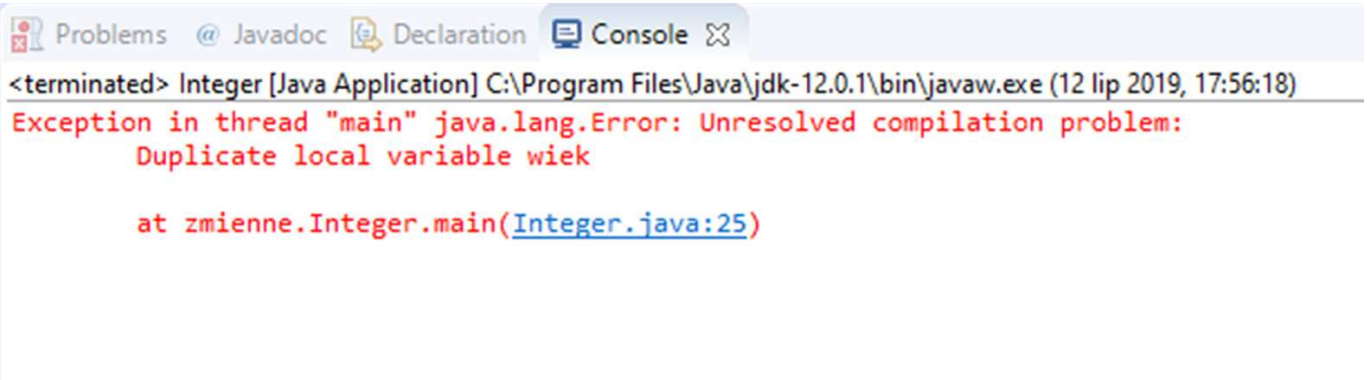
i wydrukujemy metodą `println();`

```
System.out.println(x);
```

to w konsoli zobaczymy 57, a nie 56. Do zmiennej możemy przypisywać wartości dowolną liczbę razy. Oczywiście typ wartości musi odpowiadać typowi zmiennej. Natomiast nie możemy napisać:

```
1 int wiek = 56;
2 int wiek = 57;
```

ponieważ w ten sposób dwa razy deklarujemy zmienną o takiej samej nazwie, a tego – jak już wiadomo – robić nam nie wolno. W takim wypadku pojawi się komunikat `Duplicate local variable wiek`:



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

czyli wyjątek (błąd) mówiący o powtórnym zadeklarowaniu zmiennej `wiek`.

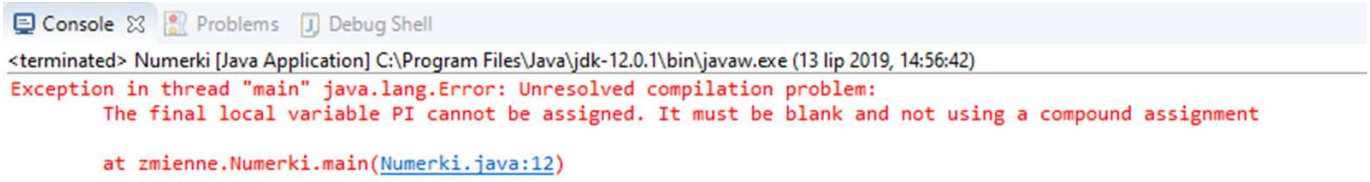
Istnieje natomiast mechanizm, który pozwala tworzyć zmienne niemodyfikowalne (niezmienne), czyli tzw. **stałe**. Należy użyć słowa kluczowego `final`:

```
1 final typDanych nazwaZmiennej = wartość;
```

Dla przykładu:

```
1 final double PI = 3.14159;
```

Zmienne finalne, czy też stałe, możemy zainicjować w programie **tylko raz**. Za każdym razem, kiedy będziemy próbowali zmienić wartość takiej zmiennej, otrzymamy błąd.



Zmienne finalne mogą być przydatne w sytuacjach, w których chcemy mieć pewność, że określona wartość nie została i nie zostanie zmieniona. Zmienne finalne wykorzystuje się przy pisaniu programu, który korzysta z pewnych stałych wartości w obliczeniach. Przykładem może być program, który wykorzysta liczbę π i pobrane od użytkownika dane, by obliczyć pole koła.

Działania na zmiennych

Jak wspomnieliśmy, na zmiennych, oprócz przypisywania im wartości, wykonujemy różne operacje. Możemy je dodawać, odejmować, mnożyć, dzielić. Możemy też zapisywać w nich wyniki działań arytmetycznych. Oczywiście wszystko to z zachowaniem odpowiednich typów zmiennych i ich wartości.

```
1 double x = 6;
2 double y = 8;
3 double z = x + y;
4
5 System.out.println(z);
6 // wydrukuje się 14.0
```

Co prawda mamy możliwość wykonania działań na różnych typach liczbowych, jednak trzeba przy tym przestrzegać pewnych zasad. Należy teraz wspomnieć o dwóch ważnych pojęciach: konwersji rozszerzającej i zawężającej.

Konwersja rozszerzająca i zawężająca

Konwersję typów w języku Java nazywa się także **rzutowaniem** (używa się także spolszczonego sformułowania kastowanie, od ang. *cast*)

Co się stanie, gdy spróbujemy wykonać takie działanie?

```
1 int liczba1 = 45;
```

```
double liczba2 = 6.67;
int wynik = liczba1 / liczba2; // / - operator dzielenia
```

Otrzymamy błąd. Aby móc zapisać wynik takiego działania w zmiennej `wynik`, musi być ona zadeklarowana jako zmienna typu bardziej ogólnego dla działania; takiego, który „pomieści" obie strony działania, w tym wynik. W tym przypadku jest to `double`, a nie `int`. Jeśli chcemy, aby program zadziałał, musimy napisać:

```
int liczba1 = 5;
double liczba2 = 6.67;
double wynik = liczba1 / liczba2;
// wynik to: 0.7496251874062969
```

Nastąpi wtedy **niejawna** konwersja **rozszerzająca** zmiennej `int liczba1` do typu `double`. Mówimy o konwersji niejawnej, ponieważ nie napisaliśmy jej sami w kodzie programu, tylko język Java wykonał ją automatycznie.

Typy zmiennoprzecinkowe są bardziej ogólne niż całkowite i konwersja niejawna (automatyczna) następuje tylko z typów całkowitoliczbowych do zmiennoprzecinkowych.

Jeśli chcemy, aby wynikiem działania na liczbach typu `int` oraz `double` był typ `int` (czyli aby wynikiem działania liczby o szerszym typie i zakresie oraz liczby o węższym typie była liczba o węższym typie), to wtedy musimy samodzielnie **jawnie** wykonać konwersję **zawężającą**, o ile oczywiście wynik będzie w zakresie typu węższego. Czyli:

```
int liczba1 = 5;
double liczba2 = 6.67;
int wynik = liczba1 / (int)liczba2; // tu następuje jawna konwers

System.out.println("Wynik to " + wynik);
```

W konsoli ukaże się napis „`Wynik to 0`".

> **Ważne!**
>
> Musimy pamiętać, że w powyższym przykładzie nie następuje żadne zaokrąglenie. Część ułamkowa wyniku jest po prostu obcinana, kiedy rzutujemy `double` do `int`. Najpierw `liczba2` jest redukowana do 6, a następnie wynik działania `liczba1 / (int)liczba2` (czyli 5 / 6) jest redukowany o część ułamkową. Jako wynik otrzymujemy wartość 0.749 . . . obciętą do części całkowitej, czyli 0.
>
> Gdybyśmy napisali program następująco:

```
int liczba1 = 5;
double liczba2 = 6.67;

System.out.println("Wynik to " + liczba1 / liczba2);
```

i nie zapisali uprzednio wyniku w zmiennej `wynik`, tylko od razu wypisali ją do konsoli, język Java również przeprowadziłby konwersję niejawną zmiennej `liczba2` i w konsoli wyświetliłoby się 0.7496251874062969.

Ciekawostka

Zmiennym można również przypisywać wartości innych zmiennych:

```
int x = 67;
int y = 89;
y = x;

System.out.println(y);
```

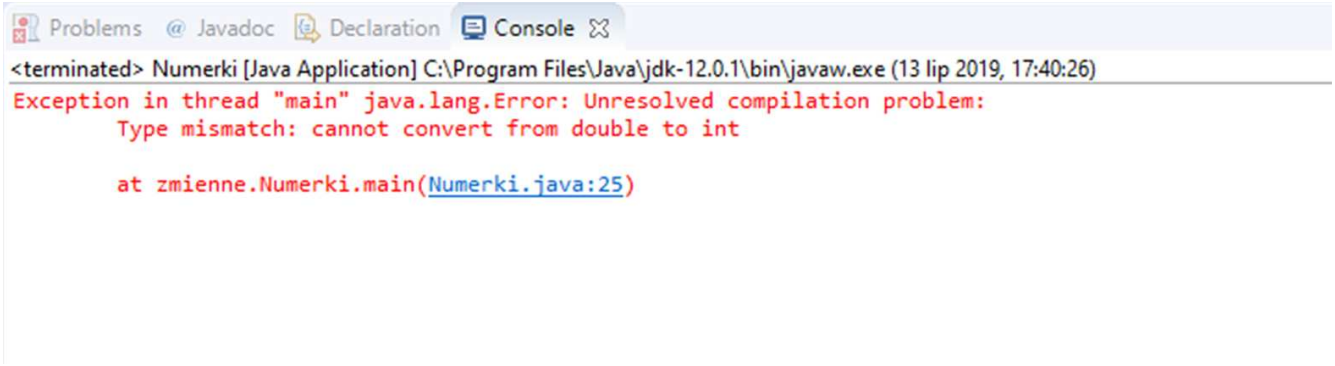
jest jak najbardziej poprawny, a w konsoli wydrukuje się wartość 67, nie 89. Musimy jednak pamiętać, że typy zmiennych muszą się zgadzać. Zatem kod skomplikuje się:

```
int x = 67;
double y = 56.7;
y = x;
```

Java wykona rzutowanie rozszerzające (z `int` na `double`) na wartość `x`. Natomiast ten kod:

```
int x = 67;
double y = 56.7;
x = y;
```

nie skompiluje się, ponieważ Java nie wykona samodzielnie rzutowania zawężającego z wartości `double` do `int` i wyrzuci błąd z rodzaju `type mismatch`.



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Aby program zadziałał, musielibyśmy wykonać jawną konwersję, czyli napisać w ostatniej linijce:

```
1 x = (int)y;
```

Język Java posiada kilka podstawowych typów danych i odpowiadających im typów zmiennych. Typy te określają, jaki rodzaj operacji może być wykonywany na zmiennych. Typy muszą być ze sobą zgodne, a Java może wykonać jedynie konwersję rozszerzającą (czyli z typu „węższego” do bardziej ogólnego).

Oczywiście ta ścisła kontrola typów ma swoje niewątpliwe zalety, dlatego język Java jest od lat tak popularny.

Słownik

powrót karetki

CRLF (CR, ang. *carriage return* + LF, ang. *line feed*) – ciąg dwóch znaków o wartościach ASCII równych 13 i 10 (szesnastkowo $0 \times 0D$ i $0 \times 0A$), oznaczający koniec bieżącej linii tekstu.

słowo kluczowe

(ang. *keyword*); w językach programowania są to słowa specjalnego rodzaju, ponieważ są interpretowane przez kompilator i spełniają określoną funkcję, np. `float`

statycznie typowany

język programowania, w którym typy są przypisywane do wartości (danych) przechowywanych w zmiennych w momencie kompilacji programu, w odróżnieniu od **typowania dynamicznego**, kiedy typy są przypisywane dopiero w momencie uruchomienia programu; typowanie statyczne pociąga za sobą ścisłą (silną) kontrolę typów

wartość logiczna

> podstawowa cecha zdania określająca jego stosunek do faktów; w logice klasycznej każde zdanie może przyjąć tylko jedną z dwóch wartości logicznych: prawda czyli zgodność zdania ze zbiorem faktów, oznaczana tradycyjnie znakiem „1”, fałsz czyli niezgodność zdania ze zbiorem faktów, oznaczana tradycyjnie znakiem „0”

Audiobook

Polecenie 1

Zapoznaj się z audiobookiem i wykonaj polecenie dotyczące przestrzegania konwencji nazewnictwa.

Audiobook można wysłuchać pod adresem: https://zpe.gov.pl/b/PSwJhUqFh

---

Nazewnictwo w języku Java

W języku Java istnieją zasady tworzenia nazw zmiennych. Nazwy powinny być samoopisujące się (z angielskiego self-descriptive), czyli powinny odzwierciedlać zawartość. Nie powinno się tworzyć zmiennej o nazwie x dla przechowania wieku osoby. Taki kod jest nieczytelny. Zamiast tego powinniśmy użyć nazwy: wiek lub age, jeśli przyjmujemy, że nazwy zmiennych będą w języku angielskim.

Kolejna zasada głosi, że w języku Java nazwy zmiennych zapisujemy systemem camelCase, to znaczy nie używamy spacji, a dla oddzielenia słów stosujemy wielkie litery. Zwyczajowo też, jeżeli piszemy program w języku polskim, nie używamy polskich znaków diakrytycznych (ą, ę, ż itp.). Użyjemy zatem zmiennej czyBedzieJutroPadalo zapisanej bez spacji, z zastosowaniem notacji camelCase, zamiast: czy będzie jutro padało – zapisanej ze spacjami i z polskimi znakami.

Warto również zaznaczyć, że nazwy zmiennych mogą zawierać wyłącznie litery i cyfry. Nie używamy zatem znaków podkreślenia, spacji lub innych znaków interpunkcyjnych albo specjalnych. Co więcej, choć nazwy mogą zawierać cyfry, to nie mogą się od nich zaczynać.

Wprawdzie język Java wybaczy użycie niektórych znaków (na przykład znaku podkreślenia _), to dobrą i stosowaną praktyką jest trzymanie się konwencji camelCase.

Nazwy zmiennych nie mogą zawierać też pewnych słów kluczowych, w tym nazw typów zmiennych. Kompilator wyświetli błąd, jeśli spróbujemy zmienną nazwać na przykład int. W takiej sytuacji zobaczymy przynajmniej dwa komunikaty dotyczące błędów.

Pierwszy będzie wskazywał typy niekompatybilne (język Java posiada przecież ścisłą kontrolę typów), a drugi będzie błędem składniowym.

Kolejną ważną zasadą, o której należy pamiętać, jest to, że nazwy zmiennych w obrębie jednej klasy muszą być unikalne, to znaczy nie mogą się powtarzać.

Konwencja nakazuje, by nazwy zmiennych rozpoczynać małą literą. Podobnie jest w przypadku nazw metod. Jednak jeśli zapisujemy nazwę klasy, rozpoczynamy ją wielką literą. Z kolei nazwy stałych piszemy wielkimi literami, a następujące po sobie elementy nazwy oddzielamy znakiem podkreślenia.

Przyjęło się również, by stawiać klamrę otwierającą klasę, metodę lub pętlę w tej samej linii co wspomniane elementy.

Dokumentacja przygotowana przez firmę Oracle jest obszerna i szczegółowo opisuje dobre praktyki związane z tworzeniem kodu w Javie. Warto się z nią zapoznać, by już na samym początku tworzyć schludny i czytelny kod źródłowy.

Polecenie 2

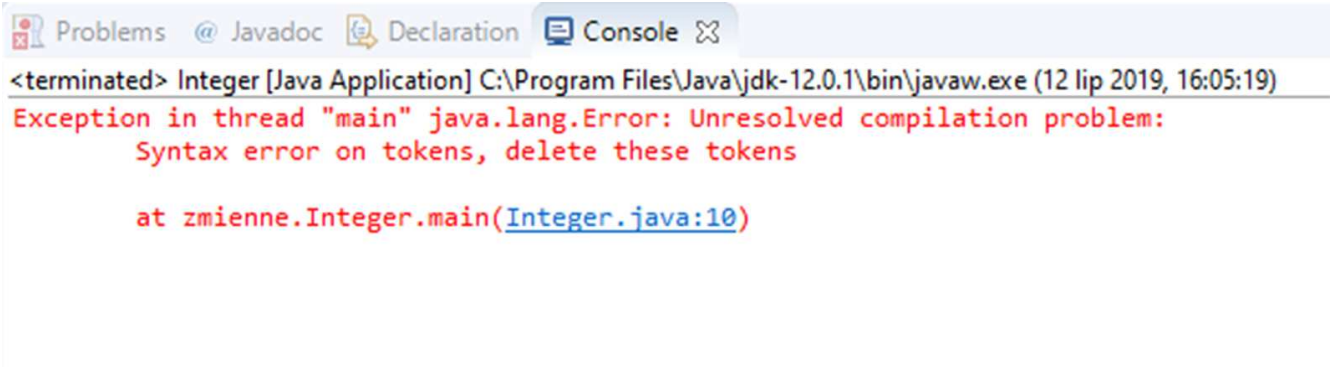
Zapisz kilka przykładów kodu, stosując się do konwencji, a następnie nie stosując się do niej.

Czy widzisz korzyści wynikające z przestrzegania konwencji nazewnictwa?

Ważne!

```
String osoba7 = "Aneta"; // kod się skompiluje
```

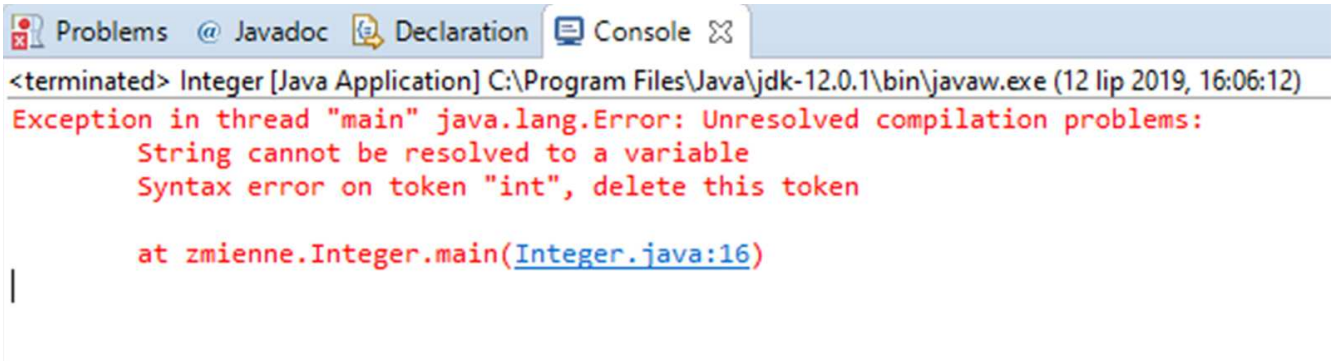
```
String 7osoba = "Aneta"; // kod się nie skompiluje
String osoba^&% = "ktoś"; // kod się nie skompiluje
double ", = 4.5d; // kod się nie skompiluje
```



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Ważne!

```
1 String int = "liczba";
```



Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Polecenie 3

Zapisz notatkę, w której podsumujesz najważniejsze informacje przedstawione w audiobooku.

Sprawdź się

Pokaż ćwiczenia:

Ćwiczenie 1

Uzupełnij tekst.

Java jest językiem [ ], który posiada [ ] kontrolę [ ]. Jest tak, ponieważ jest on [ ] typowany, co oznacza, że typy są nadawane w momencie [ ] programu.

statycznie | kompilacji | dynamicznym | mocną | dynamicznie | ścisłą | szeroką | obiektów | uruchomienia | działania | typów | obiektowo | obiektowym | zmiennych

Ćwiczenie 2

Wskaż, co nazywamy znakiem ucieczki.

- ;
- \
- ////
- \\

Ćwiczenie 3

Wskaż, co nazywamy konwersją jawną.

- konwersję z typu `int` do `double`
- konwersję zawężającą
- automatyczne rzutowanie typów

Ćwiczenie 4

Dokończ zdanie.

`String imie` jest zmienną typu...

- ◯ prymitywnego.
- ◯ tekstowego.
- ◯ obiektowego.

Ćwiczenie 5

Wskaż, który fragment kodu się nie skompiluje.

```
1
```

◯

```
1 long x = 65665478L;
2     float y = 4;
3     double z = x-y;
```

◯

```
1 double x = 6.78d;
2     int y = 4;
3     int z = x - y;
```

◯

```
1 double x = 6.78d;
2     int y = 4;
3     double z = x - y;
```

Ćwiczenie 6

Wskaż, które zmienne są zadeklarowane w poprawny sposób. Zaznacz wszystkie poprawne odpowiedzi.

- [ ] `double = 6.4324d;`
- [ ] `final double x = 4.5656f;`
- [ ] `double x = 4;`
- [ ] `final double x = 3.55523d;`

Ćwiczenie 7

Uszereguj typy danych w kolejności od najwęższego do najszerszego.

- `double`
- `float`
- `byte`
- `int`

Ćwiczenie 8

Wskaż, jakie błędy popełniono w deklaracji w tym fragmencie kodu:

```
int final typ finalny = 456 + (int)67.0.
```

- [ ] Słowo `final` umieszczono w niewłaściwym miejscu.
- [ ] Nie zachowano konwencji nazewnictwa zmiennych w języku Java.
- [ ] Błędnie zastosowano konwersję zawężającą.

Dla nauczyciela

Autor: Zespół autorski Contentplus.pl sp. z o.o.
Przedmiot: Informatyka

Temat: Podstawowe typy zmiennych w języku Java

Grupa docelowa:

Liceum ogólnokształcące i technikum, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) do realizacji rozwiązania problemu dobiera odpowiednią metodę lub technikę algorytmiczną i struktury danych;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje

poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz różnice między zmiennymi w języku Java.
- Dobierzesz do realizowanego algorytmu zmienne właściwego typu.
- Rozwiążesz proste problemy z zastosowaniem zmiennych w języku Java.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;

- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Podstawowe typy zmiennych w języku Java". Nauczyciel prosi uczniów o zapoznanie się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Nauczyciel wyświetla i odczytuje temat lekcji oraz cele zajęć. Prosi uczniów o sformułowanie kryteriów sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytania dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu, np. Chętni lub wybrani uczniowie udzielają na nie odpowiedzi.

Faza realizacyjna:

1. Nauczyciel wyświetla film zawarty w sekcji „Przeczytaj". Uczniowie wspólnie tworzą cztery zmienne, których zadaniem będzie przechowywanie różnych typów informacji. Następnie porównują swoje rozwiązanie z filmem.
2. **Ćwiczenie umiejętności.** Uczniowie realizują indywidualnie ćwiczenia nr 1-5, po ich wykonaniu porównują otrzymane wyniki z inną osobą.

Faza podsumowująca:

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku Java.

Praca domowa:

1. Uczniowie wykonują ćwiczenia nr 6-8 z sekcji „Sprawdź się".
2. Uczniowie wykonują polecenia 1 i 2 z sekcji „Audiobook".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).

- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

Wskazówki metodyczne:

- Uczniowie mogą wykorzystać treści w sekcjach: „Przeczytaj”, „Audiobook”, „Sprawdź się” jako materiał do lekcji powtórkowej.