Programowanie w języku Java - Instrukcje sterujące, pętle, liczby pseudolosowe

mgr inż. Maciej Lasota <m.lasota@tu.kielce.pl>

Version 1.0, 03-03-2017

Spis treści

Operatory i wyrażenia

Operatory dostępne w języku Java możemy podzielić na **cztery grupy**. Są to operatory *arytmetyczne*, *logiczne*, *relacyjne* oraz operatory *przypisania*.

Typ operatora	Operatory
Jednoargumentowe	+ – ++ – –
Arytmetyczne oraz przesunięcia	* / % + – << >>
Relacyjne	> < >= <= == !=
Logiczne i bitowe	&& || & | ^
Warunkowy (trójargumentowy)	A > B ? X : Y
Przypisania	= (oraz przypisania złożone)

Operatory arytmetyczne

Operatory arytmetyczne (odpowiedniki dobrze znanych operacji matematycznych) to operator sumy liczb i konkatenacji stringów `+`, operator różnicy liczb `-`, operator iloczynu liczb `*`, operator ilorazu liczb `/`, operator reszty z dzielenia (modulo) `%` oraz operatory inkrementacji i dekrementacji, tj. `++`, `--`.

Wśród operatorów arytmetycznych szczególną rolę odgrywają jednoargumentowe operatory zwiększania `++` i zmniejszania `--`. Oba występują w dwóch postaciach:

- **przyrostkowej** *(postfix)* - operator po argumencie - zmiennej,
- **przedrostkowej** *(prefix)* - operator przed argumentem - zmienną.

Składnia wyrażenia arytmetycznego dla operatorów dwuargumentowych jest następująca:

```
[operand pierwszy] [operator dwuargumentowy] [operand drugi]
```

dla operatorów jednoargumentowych:

```
[operator jednoargumentowy][operand]
```

lub

```
[operand][operator jednoargumentowy]
```

Operatory logiczne

Operatory logiczne to operatory których operandami są wartości logiczne. Operatory dostępne w

języku Java to: operator koniunkcji (tj. operator and) `&`, operator alternatywy (tj. operator or) `|`, operator alternatywy wykluczającej (tj. operator xor) `^` i jednoargumentowy operator negacji `!`. Operatory `&` i `|` posiadają dodatkowo wersje tzw. *leniwe*. Są to odpowiednio operatory `&&` i `||`.

Operatory relacyjne

Operatory relacyjne to operatory których używamy do porównywania wartości liczbowych oraz dodatkowo, do badania równości wartości logicznych i tożsamości obiektów. Operatory relacyjne używane wyłącznie do porównywania wartości liczbowych to: `<`, `⇐`, `>` i `>=`. Są to odpowiedniki dobrze znanych operatorów matematycznych. Operator równości to `==` (dwa znaki =) a operator nierówności to `!=`.

Operator przypisania

Klasyczny operator przypisania to operator `=`. Operator ten przypisuje zmiennej wskazanej po swojej lewej stronie wartość wyrażenia prawej strony. Jeśli zmienna jest **zmienną typu prostego** to przypisanie wartości jest literalne. Jeśli zmienna jest **zmienną typu obiektowego** *(tj. jest referencją)*, to do zmiennej tej przypisywane jest wskazanie na obiekt wskazany po prawej stronie operatora przypisania, nie zaś sam obiekt. Zmiennej typu obiektowego można także przypisać wartość **null**.

Java oferuje także **złożone operatory przypisania**. Z pośród nich interesujące są operatory: `+=`, `-=`, `*=` oraz `/=`. Operatory te służą do skracania zapisu klasycznych operacji arytmetycznych.

Instrukcje sterujące, pętle

Instrukcje sterujące odgrywają w programowaniu bardzo istotną rolę, bowiem pozwalają na zmianę sekwencji (kolejności) wykonania instrukcji programu.

Instrukcje warunkowe i wyboru

- Instrukcja `if (wyrażenie) [instrukcje]`

```
if (i < 8) {
  //...
}
```

- Instrukcja `if (wyrażenie) [instrukcje] else [instrukcje]`

```
if (i < 8) {
  //...
} else {
  //...
}
```

- Instrukcja `switch (wyrażenie) [case instrukcje]`

```
switch (test) {
  case 1 : {
    //...
  } break;
  case 2 : {
    //...
  } break;
  //...
  default : {
    //...
  }
}
```

Instrukcje pętli

- Pętla `while (wyrażenie) [ciało pętli]`

```
while (i < 8) {
  //...
}
```

- Pętla `do [ciało pętli] while (wyrażenie)`

```
do {
  //...
} while (i < 8);
```

- Pętla `for({deklaracje zmiennych};{warunek pętli};{instrukcje inkrementacji}) [ciało pętli]`

```
for (int i = 0; i < 8; i++) {
  //...
}
```

- Pętla `for-each for({deklaracja zmiennej pętli} : {kolekcja lub tablica}) [ciało pętli]`

```
int[] myArray = { 1, 3, 5, 7, 11 };
for (int arrayElem : myArray) {
  System.out.println(arrayElem + " ");
}
```

- Pętle nieskończone

```
for (;;) {
  //...
}

while (true) {
  //...
}
```

- instrukcja **break** - przerywa iteracje (wyjście z pętli)
- instrukcja **continue** - przechodzi do następnej iteracji (wraca na początek pętli)
- instrukcja **continue [etykieta]** - skacze do etykiety i ponownie wchodzi do pętli
- instrukcja **break [etykieta]** - przechodzi poza koniec pętli oznaczonej etykietą

Składnia wszystkich instrukcji sterujących jest zgodna z **językiem C**

Porównywanie

Operator równości `==` może być użyty do sprawdzenia czy równe są dwie wartości *liczbowe* albo dwie wartości *logiczne*, lub do sprawdzenia czy dwie *zmienne referencyjne* mają tą samą wartość, tj. czy obydwie wskazują na ten sam obiekt (ten sam, nie taki sam). Za pomocą operatora równości możemy także sprawdzić czy referencja ma wartość **null**. Operator `!=` jest negacją operatora `==`.

- typy proste
  - operator relacji równości zwraca **true** lub **false** na podstawie porównania wartości zmiennych *(i==n)*.
- obiekty
  - operator relacji porównuje **referencje obiektów**,
  - aby porównać obiekty (ich właściwości) należy użyć metody `equals(Object)` z klasy **Object**

Konwersja typów

- **niejawna** *(automatyczna)* – rozszerzanie typu

```
int i = 200;
long l = i;
```

- **jawna** *(z określeniem typu w nawiasach)* – zawężanie typu

```
int k = (int) l;
```

Liczby pseudolosowe

Generator liczb pseudolosowych *(Pseudo-Random Number Generator, lub PRNG)* to program lub biblioteka, która na podstawie niewielkiej ilości informacji (ziarno, zarodek, ang. seed) generuje deterministycznie ciąg bitów, który pod pewnymi względami jest nieodróżnialny od ciągu uzyskanego z prawdziwie losowego źródła. Generatory liczb pseudolosowych nie generują ciągów prawdziwie losowych – generator inicjowany ziarnem, które może przyjąć `k` różnych wartości, jest w stanie wyprodukować co najwyżej `k` różnych ciągów liczb.

W języku Java dostępna jest klasa `java.util.Random` która umożliwia generowanie liczb pseudolosowych.

Przykład 1. Przykład losowania liczb pseudolosowych

```
public class TestRandom {

  public static void main(String[] args) {
    //Stworzenie obiektu generatora
    Random randomGenerator = new Random();
    for (int idx = 1; idx <= 10; ++idx){
      // Losowanie liczb od 0 do 99
      int randomInt = randomGenerator.nextInt(100);
      log("Generated : " + randomInt);
    }
  }
}
```

Przykład 2. Przykład losowania liczb pseudolosowych z zadanego zakresu

```
public class TestRandom {

  int MIN = 10;
  int MAX = 20;
  public static void main(String[] args) {
    //Stworzenie obiektu generatora
    Random randomGenerator = new Random();
    for (int idx = 1; idx <= 10; ++idx){
      // Losowanie liczb od 10 do 20
      // int randomNumber = randomGenerator.nextInt((MAX - MIN) + 1) + MIN;
      int range = MAX - MIN + 1;
      int fraction = randomGenerator.nextInt() % range;
      int randomNumber =  MIN + fraction;
      log("Generated : " + randomNumber);
    }
  }
}
```

Bibliografia

- **Bruce Eckels**, *"Thinking in Java. Edycja polska. Wydanie IV"*, wydawnictwo Helion.
- **Cay S. Horstmann, Gary Cornell**, *"Java. Podstawy. Wydanie IX"*, wydawnictwo Helion.
- **Cay S. Horstmann, Gary Cornell**, *"Java. Techniki zaawansowane. Wydanie IX"*, wydawnictwo Helion.
- **Krzysztof Barteczko**, *"Podstawy programowania w języku Java, PJWSTK"*, http://edu.pjwstk.edu.pl/wyklady/ppj/scb/
- **Konrad Kurczyna**, *”Laboratorium Java”*, Politechnika Świętokrzyska w Kielcach.
- **Mariusz Lipiński**, *"Nauka Javy"*, http://www.naukajavy.pl/