Java Applet — Wydanie III OddEven — TABLICE INFORMATYCZNE

Marcin Lis

Helion

BUDOWA APLIKACJI

Program w Javie składa się ze zbioru klas. Każda klasa publiczna musi być zapisana w oddzielnym pliku o takiej samej nazwie, jak nazwa tej klasy (zasada ta nie dotyczy klas wewnętrznych i pakietowych), oraz rozszerzeniu *java*. Wykonywanie programu rozpoczyna się od metody o nazwie `main`, która musi być zadeklarowana jako publiczna i statyczna. Jeżeli program składa się z wielu klas publicznych i więcej niż jedna z nich zawiera taką metodę `main`, to aplikacja ma wiele punktów wejścia (startowych), może być więc uruchamiana na kilka sposobów. Schematyczna konstrukcja programu:

```
public class Main
{
  public static void main(String args[])
  {
    //kod metody main
  }
}
```

KOMPILACJA KODU

W przypadku korzystania ze zintegrowanych środowisk programistycznych sposób kompilacji zależy od wykorzystywanego produktu. W przypadku korzystania z kompilatora pracującego w wierszu poleceń, zawartego w pakiecie Java Development Kit (JDK), kompilacja odbywa się po wydaniu następującego polecenia:

```
javac nazwa_pliku.java
```

Kompilator ten udostępnia m.in. opcje:

Opcja	Znaczenie
`g`	Włącza lub wyłącza generowanie informacji dla debugera
`nowarn`	Wyłącza wyświetlanie ostrzeżeń
`verbose`	Wyświetla dodatkowe informacje o postępie kompilacji
`deprecation`	Wyświetla informacje o użyciu przestarzałych metod API
`classpath`	Specyfikuje położenie plików bibliotecznych
`sourcepath`	Specyfikuje położenie plików źródłowych
`d`	Określa położenie plików wynikowych
`encoding`	Określa standard kodowania znaków w plikach źródłowych
`source`	Określa standard, z którym kompatybilne są pliki źródłowe
`target`	Określa kompatybilność kodów wynikowych z poszczególnymi wersjami maszyn wirtualnych
`help`	Wyświetla skróconą listę opcji kompilatora

Parametr `target` może przyjmować jedną z następujących wartości:

- 1.1 — kod zgodny z maszyną wirtualną w wersji 1.1;
- 1.2 — kod zgodny z wersją 1.2 i wyższymi;
- 1.3 — kod zgodny z wersją 1.3 i wyższymi;
- 1.4 — kod zgodny z wersją 1.4 i wyższymi;
- 1.5 — kod zgodny z wersją 1.5 i wyższymi;
- 1.6 — kod zgodny z wersją 1.6 i wyższymi;
- 1.7 — kod zgodny z wersją 1.7 i wyższymi;
- 5 — kod zgodny z wersją 1.5 i wyższymi;
- 6 — kod zgodny z wersją 1.6 i wyższymi;
- 7 — kod zgodny z wersją 1.7 i wyższymi.

TYPY DANYCH

Java udostępnia pewną liczbę wbudowanych typów danych, czyli takich, które oferuje sam język i z których można korzystać bez potrzeby ich definiowania. Określa się je jako typy proste lub podstawowe (z ang. *primitive types*).

Typ znakowy (char)

Typ `char` służy do reprezentowania wszelkich znaków, m.in. liter. Jest on 16-bitowy i opiera się na standardzie Unicode (czyli standardzie umożliwiającym przedstawienie znaków występujących w większości języków świata). Ponieważ znaki reprezentowane są tak naprawdę jako 16-bitowe kody liczbowe, typ ten można zaliczyć również do typów arytmetycznych.

Typ logiczny (boolean)

Typ `boolean` może reprezentować jedynie dwie wartości: `true` (prawda) i `false` (fałsz).

Typy arytmetyczne

Typy całkowitoliczbowe

Typy arytmetyczne całkowitoliczbowe służą do reprezentowania liczb całkowitych.
W Javie występują cztery ich rodzaje:

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

Rodzaje te różnią się zakresem liczb, które można reprezentować za ich pomocą, co przedstawia poniższa tabela.

Typ	Liczba bitów	Zakres
`byte`	8	od –128 do 127
`short`	16	od –32 768 do 32 767
`int`	32	od -2^{31} do $2^{31}-1$
`long`	64	od -2^{63} do $2^{63}-1$

Osoby programujące w innych językach programowania, takich jak C, C++ czy PHP, powinny zwrócić uwagę, że zakres wartości możliwych do przedstawienia jest z góry ustalony i nie zależy od platformy systemowej, na której uruchamiany jest program w Javie.

Typy zmiennopozycyjne

Typy zmiennopozycyjne występują w dwóch odmianach:

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

różniących się rozmiarem oraz zakresem liczb możliwych do zaprezentowania.

Typ	Liczba bitów	Zakres
`float`	32	od –3,4e38 do 3,4e38
`double`	64	od –1,8e308 do 1,8e308

KOMENTARZE

W Javie istnieją dwa rodzaje komentarzy, oba zapożyczone z języków takich, jak C i C++:

- blokowy,
- wierszowy.

Komentarz blokowy

Komentarz blokowy rozpoczyna się od znaków `/*`, a kończy się znakami `*/`. Wszystko, co znajduje się pomiędzy tymi znakami, jest traktowane przez kompilator jako komentarz i pomijane w procesie kompilacji. Umiejscowienie komentarza blokowego jest w zasadzie dowolne — może on się znaleźć nawet w środku instrukcji (pod warunkiem, że nie zostanie przedzielone żadne słowo). Komentarzy blokowych nie wolno zagnieżdżać.

```
class Main
{
  public static void main(String args[])
  {
    /*
    to jest komentarz blokowy
    */
    System.out.println("To jest napis");
  }
}
```

Komentarz wierszowy

Komentarz wierszowy zaczyna się od znaków `//` i obowiązuje do końca danej linii programu. Wszystko to, co występuje po tych dwóch znakach aż do końca bieżącej linii, jest ignorowane przez kompilator.

```
class Main
{
  public static void main(String args[])
  {
    //to jest komentarz wierszowy
    System.out.println ("To jest napis.");
  }
}
```

Komentarz wierszowy może znaleźć się w środku komentarza blokowego:

```
/*
//ta konstrukcja jest poprawna
*/
```

ZMIENNE

Deklaracja pojedynczej zmiennej

Deklaracja zmiennej polega na podaniu jej typu oraz nazwy i kończy się znakiem średnika; schematycznie:

```
nazwa_zmiennej typ_zmiennej;
```

Przykładowo:

```
class Main
{
  public static void main(String args[])
  {
    int liczba;
  }
}
```

Deklaracja może być równoczesna z przypisaniem wartości (inicjalizacją zmiennej):

```
nazwa_zmiennej typ_zmiennej = wartość;
```

Przykładowo:

```
int liczba = 100;
```

Nazwy zmiennych

Nazwa zmiennej może się składać z liter (zarówno małych, jak i dużych), cyfr oraz znaku podkreślenia, nie może jednak zaczynać się od cyfry. Dopuszczalny jest także znak dolara, ale przyjmuje się, że jest on zarezerwowany dla narzędzi przetwarzających kod i raczej nie należy go stosować w nazwach zmiennych. Można wykorzystać znaki spoza ścisłego alfabetu łacińskiego (wszelkiego rodzaju znaki narodowe).

Deklaracja wielu zmiennych

W jednym wierszu można deklarować wiele zmiennych, jeśli są one tego samego typu; schematycznie:

```
typ_zmiennej nazwa1, nazwa2, ..., nazwaN;
```

Przykładowo:

```
int pierwsza_liczba, druga_liczba, trzecia_liczba;
```

Deklaracje wielu zmiennych mogą być powiązane z ich równoczesną inicjalizacją:

```
int pierwsza_liczba = 100, druga_liczba, trzecia_liczba = 200;
```

Zmienne referencyjne

Zmienne typów referencyjnych (z ang. *reference types*), inaczej odnośnikowych, deklaruje się tak, jak przedstawiono wyżej, czyli:

```
typ_zmiennej nazwa_zmiennej;
```

z tą różnicą, że konstrukcja ta powoduje powstanie jedynie odniesienia, któremu domyślnie zostanie przypisana wartość pusta `null`. Zmiennej referencyjnej po deklaracji należy przypisać odniesienie do obiektu utworzonego oddzielną instrukcją (patrz sekcja „Klasy i obiekty").

OPERATORY

Operatory arytmetyczne

Operator	Wykonywane działanie
`*`	mnożenie
`/`	dzielenie
`+`	dodawanie
`-`	odejmowanie
`%`	dzielenie modulo (reszta z dzielenia)
`++`	inkrementacja (zwiększanie)
`--`	dekrementacja (zmniejszanie)

Operatory logiczne

Operator	Symbol
Iloczyn (AND)	`&&`
Suma (OR)	`||`
Negacja (NOT)	`!`

Operatory logiczne działają według zasad opisanych poniżej.

Iloczyn logiczny

Wynikiem operacji AND (iloczynu logicznego) jest wartość `true`, wtedy i tylko wtedy, kiedy oba argumenty mają wartość `true`. W każdym innym przypadku wynikiem jest wartość `false`.

Argument 1	Argument 2	Wynik
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

Suma logiczna

Wynikiem operacji OR (sumy logicznej) jest wartość `false`, wtedy i tylko wtedy, kiedy oba argumenty mają wartość `false`. W każdym innym przypadku wynikiem jest `true`.

Argument 1	Argument 2	Wynik
`true`	`true`	`true`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

Negacja logiczna

Operacja NOT (negacja logiczna) zamienia wartość argumentu na przeciwną. Jeśli argument miał wartość `true`, będzie miał wartość `false`, jeśli argument miał wartość `false`, będzie miał wartość `true`.

Argument	Wynik
`true`	`false`
`false`	`true`

Operatory bitowe

Operator	Symbol
Iloczyn (AND)	`&`
Suma (OR)	`|`
Negacja (NOT)	`~`
Różnica symetryczna (XOR)	`^`
Przesunięcie bitowe w prawo	`>>`
Przesunięcie bitowe w lewo	`<<`
Przesunięcie bitowe w prawo z wypełnieniem zerami	`>>>`

Operatory przypisania

Argument 1	Operator	Argument 2	Znaczenie
x	`=`	y	`x = y`
x	`+=`	y	`x = x + y`
x	`-=`	y	`x = x - y`
x	`*=`	y	`x = x * y`
x	`/=`	y	`x = x / y`
x	`%=`	y	`x = x % y`
x	`<<=`	y	`x = x << y`
x	`>>=`	y	`x = x >> y`
x	`>>>=`	y	`x = x >>> y`
x	`&=`	y	`x = x & y`
x	`|=`	y	`x = x | y`
x	`^=`	y	`x = x ^ y`

Ciąg dalszy na stronie 2 ➔