JAVA

Wstęp do programowania w języku obiektowym

Opracował: Andrzej Nowak

Bibliografia: **JAVA Szkoła programowania,** D. Trajkowska
Ćwiczenia praktyczne JAVA. Wydanie III,M. Lis

Platforma JSE:

JSE (Java Standard Edition) – standardowa technologia Javy zaprojektowana przez firmę Sun Microsystem specjalnie do tworzenia aplikacji desktopowych i apletów na strony WWW.

JRE (Java Runtime Environment) – środowisko uruchomieniowe standardowej Javy w skład, którego wchodzi: JVM i Java API

JVM (Java Virtual Machine) – maszyna wirtualna Javy – jest to serce platformy JRE i służy ona wraz z **Javą API (interfejs tworzenia aplikacji)** do uruchamiania aplikacji Java.

JDK (Java Development Kit) – środowisko programistyczne zawierające kompilator i debugger, niezbędne do tworzenia aplikacji w języku Java.

Więcej na stronie dotyczącej języka Java : http://docs.Oracle.com/javase/7/docs/index.html

Środowiska programistyczne dla języka Java

1. JDK – prosty kompilator dla systemu operacyjnego Windows działający z wiersza poleceń
   Adres strony:
   http://download.oracle.com/otn-pub/java/jdk/7u7-b11/jdk-7u7-windows-i586.exe
2. NetBeans 7.2.1
   Adres strony:
   http://download.netbeans.org/netbeans/7.2.1/final/bundles/netbeans-7.2.1-ml-windows.exe
3. Eclipse
   Adresy stron:
   http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/juno/SR1/eclipse-jee-juno-SR1-win32.zip

Budowa klasy (programu) w języku JAVA

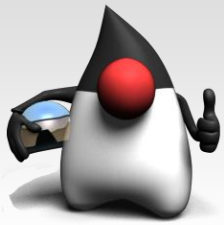
```
/* komentarz wielowierszowy
 * Miejsce na opis programu
 */
package program1; //nagłówek określający nazwę pakietu
/**
 * komentarz wielowierszowy miejsce na podanie autora
 * @author Dom
 */
public class Program1 { //nagłówek głównej klasy programu
    /**
       * komentarz wielowierszowy
       * opis metod
     * @param args the command line arguments
     */
    public static void main(String[] args) { //nagłówek głównej metody main
        // komentarz jednowierszowy treść metody głównej main
         System.out.print( "Java jest OK" ); // instrukcja wypisująca napis na ekranie
   }
}
```

Słowa kluczowe:

`public` – modyfikator dostępu

`static` – specyfikator dostępu

UWAGA: Nazwa pliku musi być taka sama jak nazwa klasy głównej z rozszerzeniem.java – np.: **`Program1.java`**



Zmienne i operatory

Zmienne

UWAGA: W języku Java każda zmienna powinna mieć nadaną wartość początkową

Typy podstawowe

Zmienne typu znakowego:

```
String tekst=”Java jest OK”;
```

Zmienne typu liczbowego – typy całkowitoliczbowe (ang. integral types):

```
byte ilosc=10;
short ilosc_2=10;
int ilosc_3=10;
long ilosc_4=10;
```

Zmienne typu liczbowego – typy zmiennopozycyjne (rzeczywiste, ang. floating-point types):

```
float ulamek=10;
double ulamek_1=10.5;
```

Zmienne typu logicznego:

```
boolean warunek=true;
```

Wczytywanie wartości zmiennych podawanych przez użytkownika

```
Scanner sc=new Scanner(System.in);

System.out.println("Wczytywanie wartosci typu string");
String str=sc.nextLine();
System.out.println("str = " +str);

System.out.println("Wczytywanie wartosci typu integer");
int i=sc.nextInt();
System.out.println("i typu integer = " +i);
```

zadanie 1

Napisz program wypisujący na ekranie nazwy zmiennych wszystkich podstawowych typów oraz ich reprezentację liczbową wprowadzoną przez użytkownika.

Typy odnośnikowe

Typy odnośnikowe (ang. reference types) możemy podzielić na dwa umowne rodzaje:

Typy tablicowe (ang. array types)

Tablice są to wektory elementów danego typu i służą do uporządkowanego przechowywania wartości tego typu. Mogą być jedno- bądź wielowymiarowe. Dostęp do elementu tablicy uzyskujemy poprzez indeks (czyli numer miejsca, w którym się dany element znajduje).

Przykłady deklarowania tablic:

```
int tablica[] = new int[10];
```

zadanie 1

Napisz program tworzący i inicjujący tablicę elementów typu całkowitego. Przypisz zerowemu elementowi tablicy dowolną wartość. Spróbuj wyświetlić zawartość tego elementu na ekranie.

zadanie 2

Napisz program tworzący i inicjujący tablicę 10-elementową typu całkowitego. Przypisz elementowi o indeksie 10 dowolną wartość. Spróbuj wyświetlić zawartość tego elementu na ekranie.

Typy klasowe (ang. class types)

- Obiektowe
- Interfejsowe

Typy klasowe pozwalają na tworzenie klas i deklarowanie zmiennych obiektowych.

Operatory

Arytmetyczne

\+ - dodawanie

\- - odejmowanie

* - mnożenie

/ - dzielenie

% - dzielenie modulo (całkowite)

zadanie 1

Zadeklaruj dwie zmienne typu całkowitego. Wykonaj na nich kilka operacji arytmetycznych. Wyniki wyświetl na ekranie.

Problem inkrementacji i dekrementacji

++x, x++

--x, x--

zadanie 2

Napisz program prezentujący użycie inkrementacji i dekrementacji.

Bitowe

Umożliwiają operacje na bitach (reprezentacja w systemie dwójkowym).

Możliwe operacje to

AND (iloczyn bitowy)	symbol **&**
OR (suma bitowa)	symbol **|**
NOT (negacja bitowa)	symbol **~**
XOR (bitowa alternatywa wykluczająca)	symbol **^**
Przesunięcie bitowe w prawo	symbol **>>**
Przesunięcie bitowe w lewo	symbol **<<**
Przesunięcie bitowe w prawo z wypełnieniem zerami	symbol **>>>**

Logiczne

Argumentami operacji logicznych muszą być wyrażenia posiadające wartość logiczną, czyli true lub false (prawda i fałsz).

Operatory to

AND - iloczyn logiczny	symbol **&&**
OR - suma logiczna	symbol **||**
negacja logiczna	symbol **!**

Przypisania

Operacje przypisania są dwuargumentowe i powodują przypisanie wartości argumentu znajdującego się z prawej strony do argumentu znajdującego się z lewej strony.

Operatory przypisania i ich znaczenie w Javie

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

Porównania (relacyjne)

Służą do porównywania argumentów. Wynikiem porównania jest wartość logiczna true lub false .

operator	Opis
==	Jeśli argumenty są sobie równe ,wynikiem jest true, w przeciwnym razie wynikiem jest false
!=	Jeśli argumenty są różne, wynikiem jest true, w przeciwnym razie wynikiem jest false
>	Jeśli argument prawostronny jest mniejszy, wynikiem jest true, w przeciwnym razie wynikiem jest false
<	Jeśli argument prawostronny jest większy, wynikiem jest true, w przeciwnym razie wynikiem jest false
>=	Jeśli argument prawostronny jest mniejszy lub równy lewostronnemu, wynikiem jest true, w przeciwnym razie wynikiem jest false
<=	Jeśli argument prawostronny jest większy lub równy lewostronnemu, wynikiem jest true, w przeciwnym razie wynikiem jest false

Operator warunkowy

Ma następującą składnię

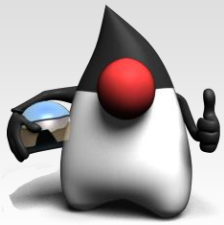
```
warunek ? wartość1 : wartość2
```

Wyrażenie przyjmuje `wartość1`, jeżeli warunek jest prawdziwy lub `wartość2` w przeciwnym wypadku.

Priorytety operatorów

Priorytet operatorów to nic innego, jak kolejność ich wykonywania. W tabeli zostały przedstawione wybrane operatory z podaniem priorytetu (ważności). Im wyższy numer w tabeli, tym priorytet wyższy.

Nr	Grupa operatorów	Symbole
1	Inkrementacja przyrostkowa	++ , --
2	Inkrementacja przyrostkowa, negacje	++ , -- , ~ , !
3	Mnożenie, dzielenie	* , / , %
4	Przesunięcia bitowe	<< , >> , >>>
5	Porównania	< , > , <= , >=
6	Porównania	== , !=
7	Bitowe AND	&
8	Bitowe XOR	^
9	Bitowe OR	|
10	Logiczne AND	&&
11	Logiczne OR	||
12	Warunkowe	?
13	Przypisania	= , += , -= , *= , /= , %= , >>= , <<= , >>>= , &= , ^= , |=



Instrukcje warunkowe

Instrukcja warunkowa if….else

```
if (wyrażenie warunkowe) {
      //instrukcje do wykonania jeżeli warunek jest prawdziwy
}
```

lub instrukcja warunkowa z alternatywą

```
if (wyrażenie warunkowe) {
      //instrukcje do wykonania jeżeli warunek jest prawdziwy
}
else{
      //instrukcje do wykonania jeżeli warunek jest nie prawdziwy
}
```

Przykład użycia:

ćwiczenie 1

Napisz program sprawdzający czy zmienna a jest większa czy mniejsza od zera

```
class Main{
     public static void main(String args[]){
          int a = -10;
          if (a > 0){
           System.out.println("Zmienna a jest wieksza od zera”);
         }
         else{
     System.out.println(“Zmienna a nie jest wieksza od zera”);
        }
    }
}
```

zadanie 1

Napisz program wyznaczający pierwiastki równania kwadratowego dla parametrów podanych bezpośrednio w kodzie programu.

Przypomnij sobie odpowiedni algorytm.

zadanie 2

Napisz program (lub zmodyfikuj program napisany wcześniej) tak, aby wyznaczał pierwiastki równania kwadratowego dla parametrów podanych przez użytkownika programu.

Instrukcja wyboru switch

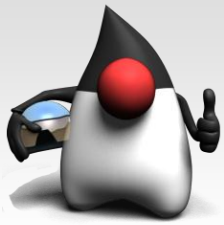
```
switch (wyrażenie) {
     case X:         // jeśli wyrażenie przyjmie wartość X
       instrukcja1; // wykona się pierwsza instrukcja
       break;       //po skończeniu działanie zostanie przerwane
     case Y:         // jeśli wyrażenie przyjmie wartość Y
       instrukcja2;
       break;
     case Z:         // jeśli wyrażenie przyjmie wartość Z
       instrukcja3;
       break;
     default:  //jeżeli wartość wyrażenie nie przyjmie X, Y lub Z
       instrukcja4; // to wykona się instrukcja domyślna
}
```

Przykład użycia:

ćwiczenie 1

Używając instrukcji `switch` napisz program sprawdzający, czy wartość zadeklarowanej zmiennej jest równa 1 czy 10. Na ekranie wyświetl odpowiedni komunikat.

```
class Main{
     public static void main(String args[]){
          int a = 10;
          switch (a){
           case 1:
             System.out.println("a jest rowne 1”);
             break;
          case 10:
             System.out.println("a jest rowne 10”);
             break;
          default:
             System.out.println("a nie jest rowne ani 1, ani 10”);
        }
     }
}
```



Pętle iteracyjne

Pętla for

Pozwala na wykonywanie powtarzających się czynności określoną liczbę razy.

```
for (wyrażenie początkowe; wyrażenie warunkowe; wyrażenie modyfikujące) {
        // instrukcje do wykonania
}
```

Przykłady użycia:

ćwiczenie 1

Używając instrukcji `for`, napisz program wyświetlający na ekranie 10 razy napis `JAVA`.

```
class Main{
     public static void main(String args[]){
          for (int i = 1; i <= 10; i++){
             System.out.println("JAVA");
          }
     }
 }
```

ćwiczenie 2

Zmodyfikuj poprzednie zadanie, tak aby wyrażenie modyfikujące znalazło się w bloku instrukcji.

```
class Main{
     public static void main(String args[]){
          for (int i = 1; i <= 10; ){
             System.out.println("JAVA");
             i++;
          }
     }
 }
```

ćwiczenie 3

Zmodyfikuj poprzednie zadanie, tak aby wyrażenie warunkowe było jednocześnie wyrażeniem modyfikującym.

```
class Main{
     public static void main(String args[]){
          for (int i = 1; i++ <= 10; ){
             System.out.println("JAVA");
          }}}
```

ćwiczenie 4

Zmodyfikuj pętlę typu `for` w taki sposób, aby wyrażenie początkowe znalazło się przed pętlą, a wyrażenie modyfikujące wewnątrz niej.

```
class Main{
     public static void main(String args[]){
          int i = 1;
          for ( ; i <= 10; ){
            System.out.println("JAVA");
            i++;
          }
     }
 }
```

ćwiczenie 5

Zmodyfikuj pętlę typu `for` w taki sposób, aby wyrażenie początkowe znalazło się przed pętlą, natomiast wyrażenie modyfikujące i warunkowe wewnątrz niej.

```
class Main{
     public static void main(String args[]){
          int i = 1;
          for ( ; ; ){
            System.out.println("JAVA");
            if (i++ >= 10) break;
          }
     }
 }
```

ćwiczenie 6

Napisz program wyświetlający na ekranie liczby od 1 do 20, które nie są podzielne przez 2. Skorzystaj z pętli `for` i instrukcji `continue`.

```
class Main{
     public static void main(String args[]){
          for (int i = 1 ;i <= 20 ; i++ ){
            if (i % 2 == 0)
              continue;
            System.out.println(i);
}}}
```

ćwiczenie 7

Zmodyfikuj kod z zadania poprzedniego tak, aby nie było konieczności użycia instrukcji `continue`.

```
class Main{
     public static void main(String args[]){
          for (int i = 1 ;i <= 20 ; i++ ){
            if (i % 2 == 0) System.out.println(i);
}}}
```

Pętla while

Pozwala na wykonywanie powtarzających się czynności dowolną liczbę razy ograniczoną spełnieniem warunku. Warunek ma wartość logiczną `true`.

```
while (wyrażenie warunkowe) {
          // instrukcje do wykonania
}
```

Przykłady użycia:

ćwiczenie 1

Używając instrukcji `while`, napisz program wyświetlający na ekranie 10 razy napis `JAVA`.

```
class Main{
     public static void main(String args[]){
          int i = 1;
          while ( i <= 10){
             System.out.println("JAVA”);
             i++;
          }
     }
 }
```

ćwiczenie 2

Zmodyfikuj kod z poprzedniego zadania tak, aby wyrażenie warunkowe zmieniało jednocześnie wartość zmiennej i.

```
class Main{
     public static void main(String args[]){
          int i = 1;
          while ( i++ <= 10){
             System.out.println("JAVA”);
          }
     }
 }
```

zadanie 1

Korzystając z pętli `while`, napisz program wyświetlający na ekranie liczby od 1 do 20, które nie są podzielne przez 2.

Pętla do…while

Pozwala na wykonywanie powtarzających się czynności dowolną liczbę razy ograniczoną spełnieniem warunku. Warunek ma wartość logiczną `true`.

```
do{
          // instrukcje do wykonania
}
while();
```

Przykłady użycia:

ćwiczenie 1

Korzystając z pętli `do…while`, napisz program wyświetlający na ekranie 10 razy napis `JAVA`.

```
class Main{
     public static void main(String args[]){
          int i = 1;
          do{
             System.out.println("JAVA”);
          }
           while (i++ <= 9);
     }
 }
```