Programowanie komputerów

Wykład 2: „Elementy języka Java”

dr inż. Walery Susłow
walery.suslow@ie.tu.koszalin.pl

Komentarze w Javie

```
//To jest komentarz wierszowy

/* A to jest komentarz blokowy. Możemy przekazać dłuższe
komunikaty – kilka bądź kilkanaście wierszy.*/

/*
 * A to komentarz dokumentacyjny.
 * W taki sposób umieszczamy notatki,
 * wykorzystywane później do łatwego opracowania
*dokumentacji projektowej.
*/
```

Tworzenie dokumentacji

- Do opisu kodu źródłowego używa się komentarzy. Koniecznie opisuje się klasy, metody, interfejsy i moduły, komentarz umieszcza się bezpośrednio przed dokumentowanym elementem programu.
- Program *javadoc* rozpoznaje komentarz, jeśli został on umieszczony pomiędzy sekwencjami znaków /** i */. Początkowe znaki * w kolejnych wierszach są pomijane.
- Każdy wiersz zawierający znak @, po którym następuje jeden ze znaczników dokumentacyjnych, powoduje utworzenie w dokumentacji oddzielnego paragrafu.
- Polecenie wygenerowania dokumentacji ma postać:

  javadoc nazwa_pliku.java

Jego wynikiem jest zbiór plików z opisem w formacie HTML.

Znaczniki dokumentacyjne javadoc

@author – informacje o autorze programu,

@version – informacje o wersji programu,

@return – opis wyniku zwracanego przez metodę,

@serial – opis typu danych i możliwych wartości przyjmowanych przez zmienną,

@see – tworzy łącze do innego tematu,

@since – opis wersji, od której zaistniał określony fragment kodu,

@deprecated – informacje o elementach zdeprecjonowanych (które nie są zalecane),

@param – opis parametru wywołania metody,

@exception – identyfikator wyjątku.

Słowa kluczowe Javy

abstract	boolean	break	byte
byvalue	case	cast	catch
char	class	const	continue
default	do	double	else
extends	final	finally	float
for	future	generic	goto
if	implements	import	inner
instanceof	int	interface	long
native	new	null	operator
outer	package	private	protected
public	rest	return	short
static	strictfp	super	switch
synchronized	this	throw	throws
transient	try	var	void
		volatile	while

Operatory i ich priorytety

Priorytet	Operator	Typ argumentu	Nazwa
1	++ --	Arytmetyczny	Inkrementacja, dekrementacja
	+ -	Arytmetyczny	Unarny +, unarny -
	~	Całkowity	Uzupełnienie bitowe
	!	Boole'owski	Negacja logiczna
	(typ)	Dowolny	Konwersja (rzutowanie
2	* /	Arytmetyczny	Mnożenie, dzielenie
	%	Arytmetyczny	Modulo (reszta)
3	+ -	Arytmetyczny	Modulo (reszta)

Operatory i ich priorytety, cd.

Priorytet	Operator	Typ argumentu	Nazwa
3	+	Łańcuchowy	Konkatenacja
4	<< >> >>>	Całkowity	Przesunięcie bitowe
5	< > <= >=	Arytmetyczny	Operatory relacji
	instanceof	Obiektowy	Stwierdzenie typu
6	== !=	Podstawowy	Równe nierówne
	== !=	Obiektowy	Równe nierówne
7	&	Całkowity	Bitowe AND
8	^	Całkowity	Bitowe XOR

Operatory i ich priorytety, cd.

Priorytet	Operator	Typ argumentu	Nazwa
9	|	Całkowity	Bitowe OR
10	&&	Boole'owski	Logiczne AND
11	||	Boole'owski	Logiczne OR
12	?:	Boole'owski	Operator warunku
13	= *= /= += -= %=	Dowolna zmienna	Operatory przypisania

Znaki specjalne

Opis	Literał
New line	\n
Horizontal tab	\t
Backspace	\b
Carriage return	\r
From feed	\f
Single quote	\'
Double quote	\"
Backslash	\\

Typy danych w Javie

- Java jest językiem ze ścisłą kontrolą typów, w którym rozmiar i postać danych są określone bardzo precyzyjnie.
- Typy danych w Javie można podzielić na typy proste i typy referencyjne (klasy, interfejsy i tablice).
- Do przechowywania liczb całkowitych przeznaczone są cztery typy: byte (8), short (16), int (32) oraz long (64).
- Rzeczywiste typy liczbowe to: float (32) i double (64).
- Dane znakowe zapisywane są zgodnie ze standardem Unicode - są to 16-bitowe liczby całkowite bez znaku. Do ich przechowywania służy typ char.
- Typ boolean (1 bit) umożliwia przechowywanie wartości logicznych. Może on przyjmować tylko dwie wartości: true i false.

Tablice

- Tablica - to ciąg zmiennych tego samego typu, opisanych jedną wspólną nazwą. Elementy tablicy identyfikuje się za pomocą indeksów.
- Dostęp do poszczególnych elementów tablicy odbywa się za pomocą operatora indeksowania [ ]. Indeksy są liczone od zera.
- Tablica jednowymiarowa odpowiada matematycznemu pojęciu wektora, dwuwymiarowa – macierzy.
- Tablice w języku Java są zaimplementowane jako obiekty, więc nie mogą być deklarowane statycznie. Tworzenie tablicy składa się z dwóch etapów:
  - deklaracja zmiennej referencyjnej tablicy;
  - utworzenie nowego obiektu tablicy i przypisanie go do danej zmiennej tablicowej.

Tablice jednowymiarowe

- Przykład instrukcji tworzących tablicę:

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

- Tablicę można też utworzyć w dwóch etapach, najpierw deklarujemy zmienną referencyjną tablicy:

```
int[ ] mojaTablica ; //lub int mojaTablica[ ];
```

następnie tworzymy nowy obiekt tablicy:

```
mojaTablica = new int[10];
// Musimy określić rozmiar tablicy, aby zaalokować
//potrzebny dla niej obszar pamięci.
```

Tablice wielowymiarowe

- Java obsługuje tablice wielowymiarowe, kóre posiadają dwa lub więcej indeksów. Ogólna postać instrukcji tworzącej tablicę wielowymiarową ma postać:

typ[ ][ ]...[ ] nazwa = new typ[roz1][roz2]...[rozN];

- Przykład instrukcji tworzącej tablicę o dwóch wymiarach (jest to najczęściej używana forma tablicy wielowymiarowej):

```
int[ ][ ] A = new int[10][10];
//lub int A[ ][ ] = new int[10][10];
```

Wyrażenia logiczne (warunki)

Wyrażenia logiczne można przypisywać zmiennym typu boolean. Ich wartością może być true lub false.

Proste wyrażenia logiczne konstruuje się za pomocą operatorów relacji:

- <, >, <=, >=, == (czy równe), != (czy różne).

Do budowy bardziej złożonych wyrażeń używa się operatorów logicznych:

- && (koniunkcji – logiczne AND),
- || (alternatywy – logiczne OR),
- ! (negacji – logiczne NOT).

Funkcje matematyczne

Funkcje matematyczne Javy są zawarte w klasie Math.

Funkcje te są zadeklarowane jako statyczne, więc można ich używać bez tworzenia egzemplarza obiektu.

Funkcje matematyczne (metody klasy Math) zwykle operują na liczbach rzeczywistych typu double i zwracają wartości typu double.

Wybrane funkcje matematyczne

Metoda	Opis zwracanej wartości
abs(x)	Wartość bezwzględna liczby x
min(a, b)	Mniejsza z liczb a i b
max(a, b)	Większa z liczb a i b
ceil(double x)	Najmniejsza liczba całkowita większa lub równa x
floor(double x)	Największa liczba całkowita mniejsza lub równa x
exp(double x)	Liczba e podniesiona do potęgi x
log(double x)	Logarytm naturalny liczby x
pow(double x, double n)	Liczba x podniesiona do potęgi n

Wybrane funkcje matematyczne c.d.

Metoda	Opis zwracanej wartości
random()	Liczba losowa z zakresu od 0.0 do 1.0
round(float x)	Liczba całkowita powstałą z zaokrąglenia liczby typu float
sqrt(double x)	Pierwiastek kwadratowy z liczby x
sin(double alfa)	sinus kąta alfa podanego w radianach
cos(double alfa)	Cosinus kąta alfa podanego w radianach
tan(double alfa)	Tangens kąta alfa podanego w radianach
toRadians(double alfa)	Zamienia stopnie na radiany
toDegrees(double alfa)	Zamienia radiany na stopnie

Zaokrąglanie liczb w Java

Załóżmy, że mamy zadeklarowane dwie zmienne

```
double x = 12.2,  y = 12.8;
```

Jaką wartość przyjmą zmienne a, b, c, d, e, f w wyniku wykonania następujących instrukcji?

- `int a = (int) x;` // 12, wynik rzutowanie typu
- `int b = (int) y;` // 12
- `double c = (int) x;` // 12.0
- `double d = (int) y;` // 12.0
- `double e = Math.round(x);` // 12.0, wynik metody bib.
- `double f = Math.round(y);` // 13.0

Łańcuchy znaków w Java

Łańcuch (String), jest to ciąg dowolnych znaków, np. słowo, wyrażenie lub zdanie. Ciąg znaków ujęty w podwójny cudzysłów traktowany jest przez kompilator jako wartość typu String.

Do przechowywania pojedynczych znaków Unicode służą dwubajtowe zmienne typu char. Np. deklaracja postaci:

```
char ch;
```

tworzy zmienną znakową ch.

Podczas deklaracji zmiennej znakowej, można jej nadać wartość początkową:

```
char ch = 'a';  // lub poprzez kod znaku: char ch = 97;
```

Obiekty klasy String

Klasa String jest wbudowaną klasą Javy.

```
String s;    //utworzenie zmiennej referencyjnej
s = "abc";  //utworzenie obiektu zawierającego napis abc
```

Instrukcje te można połączyć, pisząc:

```
String s = "abc";
```

- Nazwa typu String zaczyna się od dużej litery, ponieważ jest to nazwa typu obiektowego.
- Operacje przeprowadzane na łańcuchach tworzą nowe obiekty klasy String. Nieużywane łańcuchy są automatycznie usuwane z pamięci.

Operacje na łańcuchach

Operacje na łańcuchach wykonuje się za pomocą:

- operatorów konkatenacji (łączenia):
  - `+` podstawowy operator konkatenacji,
  - `+=` rozszerzony operator konkatenacji.
- funkcji łańcuchowych.

Każdy z operatorów tworzy nowy obiekt klasy String.

```
String napis = "Ala ";
napis = napis + "ma kota ";
napis += "i psa";
```

Operatory konkatenacji

Operatory konkatenacji + i += są przeciążone w Javie – oprócz sklejania łańcuchów pozwalają również na ich łączenie z liczbami lub obiektami innych klas.

- int n = 10;
- String str, str1, str2, str3, str4;
- str = "napis";
- str1 = str + n; // napis10
- str2 = str + n + n; // napis1010
- str3 = str + (n + n); // napis20
- str4 = n + n + str; // 20napis

Metody klasy String

Operacje na łańcuchach wykonuje się za pomocą funkcji łańcuchowych.

- String s="aBCDe", s1, s2; //obiekty
- int dl=s.length(); //dl=5
- int k=s.indexOf('C'); //k=2
- int l=s.indexOf('Z'); //l=-1
- s1=s.toLowerCase(); //s1=abcde
- s2=s.toUpperCase(); //s2=ABCDE
- boolean b1=s1.equals(s2); //b1=false
- boolean b2=s1.equals(s1); //b2=true
- String s3=s.substring(0,2); //s3=aB
- char znak=s.charAt(2); //znak=C

Porównywanie łańcuchów

Do porównywania zawartości obiektów typu String służą metody

equals();

compareTo();

Druga metoda zwraca zero, gdy łańcuchy są równe.

Operator == pozwala jedynie stwierdzić, czy zmienne referencyjne odwołują się do tego samego obiektu.

```
String s = "abc";
s = s+12;
String s1 = "abc12";
boolean b1 = s==s1;            //false
boolean b2 = s.equals(s1);     // true
int k = s.compareTo(s1);       // 0
```

Przekształcanie liczb w łańcuchy

Konwersję liczb do postaci łańcuchów znakowych przeprowadza się za pomocą metody valueOf() klasy String.

```
double x = 12.31;
int n = 100;
String s1 = String.valueOf(x);
String s2 = String.valueOf(n);
```

Bardziej kosztownym rozwiązaniem jest użycie operatorów konkatenacji.

```
double x = 12.31;
int n = 100;
String s1 = "" + x;
String s2 = "" + n;
```

Przekształcanie łańcuchów w liczby

Do przekształcania łańcuchów w liczby stosuje się metodę valueOf() klasy Double lub Integer.

```
String s="3.14", t="10";
double x; int i;
x=Double.valueOf(s).doubleValue();
i=Integer.valueOf(t).intValue();
```

Można też posłużyć się metodą parseInt().

```
String t="125";
int i = Integer.parseInt(t);
```

Przykłady konwersji

```
String s1 = "12.3";
String s2 = "7.5";

double a=Double.valueOf(s1).doubleValue();
double b=Double.valueOf(s2).doubleValue();
double suma=a+b;

String wynik = String.valueOf(suma);

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