PODSTAWY PROGRAMOWANIA W JAVA

dr inż. Michał Tomaszewski

katedra Metod Programowania
Polsko-Japońska Akademia Technik Komputerowych

INSTRUKCJE

INSTRUKCJE

Czym jest instrukcja?

INSTRUKCJE

Instrukcja jest poleceniem wydanym procesorowi komputera, przez program. source:internet

INSTRUKCJE

Instrukcje to podstawowe jednostki działania w programach komputerowych. Mówimy o nich w kontekście przekazywania poleceń do wykonania w określonym porządku. Mogą one zawierać konkretne operacje, warunki, pętle i wiele innych elementów, które wpływają na zachowanie programu.

INSTRUKCJE

Instrukcje dzielą się na **czynne** i **bierne**.

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,
- instrukcja **grupująca**,

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,
- instrukcja **grupująca**,
- instrukcja **wyrażeniowa**,

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,
- instrukcja **grupująca**,
- instrukcja **wyrażeniowa**,
- instrukcja **warunkowa**,

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,
- instrukcja **grupująca**,
- instrukcja **wyrażeniowa**,
- instrukcja **warunkowa**,
- instrukcje **iteracyjne i decyzyjne**,

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,
- instrukcja **grupująca**,
- instrukcja **wyrażeniowa**,
- instrukcja **warunkowa**,
- instrukcje **iteracyjne i decyzyjne**,
- instrukcje **zaniechania i kontynuowania**.

INSTRUKCJE

Instrukcją bierną jest

- instrukcja **deklaracyjna**

Natomiast instrukcjami czynnymi są:

- instrukcja **pusta**,
- instrukcja **grupująca**,
- instrukcja **wyrażeniowa**,
- instrukcja **warunkowa**,
- instrukcje **iteracyjne i decyzyjne**,
- instrukcje **zaniechania i kontynuowania**.

Każda instrukcja jest zakończona klamrą albo średnikiem.

INSTRUKCJA PUSTA

Instrukcja pusta ma postać:

```
;
```

INSTRUKCJA GRUPUJĄCA

Instrukcja grupująca ma postać

```
{ Ins Ins … Ins}
```

w której każde `Ins` jest dowolną instrukcją albo jest **puste**.

INSTRUKCJA GRUPUJĄCA

Instrukcja grupująca ma postać

```
{ Ins Ins … Ins}
```

w której każde `Ins` jest dowolną instrukcją albo jest **puste**.

```
{
    System.out.println(”Hello old friend”);
}
```

INSTRUKCJA WYRAŻENIOWA

Instrukcja wyrażeniowa ma postać:

```
exp;
```

w której `exp` jest: **przypisaniem**, **wywołaniem**, **zwiększeniem** albo **zmniejszeniem**.

INSTRUKCJA DEKLARACYJNA

Instrukcja:

```
Typ nazwa;
```

deklaruje zmienną `nazwa` typu `Typ`.

INSTRUKCJA DEKLARACYJNA

Instrukcja:

```
Typ nazwa;
```

deklaruje zmienną `nazwa` typu `Typ`.

W szczególności instrukcja:

```
int age = 7;
```

deklaruje zmienną `age` typu `int`(całkowitego), którą podczas deklarowania zainicjowano wartością 7 reprezentowaną przez literał.

ZMIENNE

CZYM JEST ZMIENNA?

Zmienną jest **obszar pamięci** operacyjnej, w której przechowuje się **dane**.

CZYM JEST ZMIENNA?

Zmienną jest **obszar pamięci** operacyjnej, w której przechowuje się **dane**.

Od **typu** zmiennej zależy:

CZYM JEST ZMIENNA?

Zmienną jest **obszar pamięci** operacyjnej, w której przechowuje się **dane**.

Od **typu** zmiennej zależy:

- **rozmiar** zajmowanego obszaru,

CZYM JEST ZMIENNA?

Zmienną jest **obszar pamięci** operacyjnej, w której przechowuje się **dane**.

Od **typu** zmiennej zależy:

- **rozmiar** zajmowanego obszaru,
- **zakres** wartości danych jakie można przypisywać zmiennej,

CZYM JEST ZMIENNA?

Zmienną jest **obszar pamięci** operacyjnej, w której przechowuje się **dane**.

Od **typu** zmiennej zależy:

- **rozmiar** zajmowanego obszaru,
- **zakres** wartości danych jakie można przypisywać zmiennej,
- **reprezentacja** danych.

RODZAJE ZMIENNYCH

- Zmienne można podzielić na:
  - pierwotne
  - odnośnikowe

RODZAJE ZMIENNYCH

- Zmienne można podzielić na:
  - pierwotne
  - odnośnikowe
- zmienne typów pierwotnych

RODZAJE ZMIENNYCH

- Zmienne można podzielić na:
  - pierwotne
  - odnośnikowe
- zmienne typów pierwotnych
  - numeryczne
  - orzecznikowe

RODZAJE ZMIENNYCH

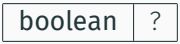
- Zmienne można podzielić na:
  - pierwotne
  - odnośnikowe
- zmienne typów pierwotnych
  - numeryczne
    - całkowite
    - rzeczywiste
    - znakowe
  - orzecznikowe

RODZAJE ZMIENNYCH

- Zmienne można podzielić na:
  - pierwotne
  - odnośnikowe
- zmienne typów pierwotnych
  - numeryczne
    - całkowite
    - rzeczywiste
    - znakowe
  - orzecznikowe
- zmienne typów odnośnikowych (złożonych)
  - tablice
  - obiekty

TYPY ZMIENNYCH

typ orzecznikowy:



typ orzecznikowy:

boolean	?

- true
- false

typ numeryczne – całkowite:

byte	1 bajt
short	2 bajty
int	4 bajty
long	8 bajtów

TYPY ZMIENNYCH

typ numeryczne – całkowite:

byte	1 bajt
short	2 bajty
int	4 bajty
long	8 bajtów

- `10` – system dziesiętny
- `0b1010` – zapis w systemie binarnym (dwójkowym)
- `010` – zapis w systemie oktalnym (ósemkowym)
- `0x5A` – zapis w systemie heksadecymalnym (szesnastkowym)

typ numeryczne – całkowite:

byte	1 bajt
short	2 bajty
int	4 bajty
long	8 bajtów

- `10L`
- `10l`

typ numeryczne – rzeczywiste:

float	4 bajty
double	8 bajty

typ numeryczne – rzeczywiste:

float	4 bajty
double	8 bajty

- `3.14`
- `3.14f` lub `3.14F`

TYPY ZMIENNYCH

typ numeryczne – znakowe:

char	2 bajty

typ numeryczne – znakowe:

char	2 bajty

- `'a'`

OPERACJE NA ZMIENNYCH

CYKL ŻYCIA ZMIENNEJ

Zmienna **żyje** od miejsca zadeklarowania do **końca bloku** w którym została zadeklarowana.

WYRAŻENIA

Wyrażenia są zestawami **operacji**.

WYRAŻENIA

Wyrażenia są zestawami **operacji**.

Do wykonywania operacji na zmiennych służą **operatory**.

Separatory (ang. punctuators):

- ( ) { } [ ] ; , . … @ ::

Separatory (ang. punctuators):

- ( ) { } [ ] ; , . … @ ::

Operatory:

- = > < ! ? : -> == >= <= != && || ++ – + - * / & | ^% « » »> += -= *= /= &= |= ^= %= «= »= »>=

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Respektowanie nawiasów

Jeśli wyrażenie zawarte w nawiasach okrągłych jest poprzedzone operatorem, to wykonanie operacji określonej przez ten operator nastąpi dopiero **po** opracowaniu wyrażenia w nawiasach.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Respektowanie nawiasów

Podczas opracowania wyrażenia:

```
a + (b - c)
```

operacja **dodawania** zostanie wykonana dopiero **po** wykonaniu operacji odejmowania.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Uwzględnienie priorytetów

Jeśli wyrażenie można uznać za argument **2** operacji o **różnych** priorytetach, to operację o wyrażoną przez operator o niższym priorytecie wykonuje się **po** operacji o wyższym priorytecie.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Uwzględnienie priorytetów

Ponieważ priorytet mnożenia jest **wyższy** od priorytetu dodawania, więc podczas opracowania wyrażenia

```
a + b * c
```

najpierw zostanie wykonana operacja **mnożenia** a dopiero potem **dodania**.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Uwzględnienie wiązań

Jeśli wyrażenie można uznać za argument **2** operacji o **równych** priorytetach, to kolejność operacji określona jest przez **wiązanie**.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Uwzględnienie wiązań

Jeśli wyrażenie można uznać za argument **2** operacji o **równych** priorytetach, to kolejność operacji określona jest przez **wiązanie**.

Jeśli wiązanie operacji jest **lewe**, to najpierw wykonuje się operację wyrażoną przez operator znajdujący się z **lewej** strony wyrażenia, a jeśli jest **prawe**, to najpierw wykonuje się operację wyrażoną przez operator znajdujący się z **prawej** strony wyrażenia.

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Uwzględnienie wiązań

Ponieważ wiązanie operatorów dodawania i odejmowania są **lewe** więc wyrażenie

```
a - b + c
```

zostanie potraktowane jako wyrażenie

```
(a - b) + c
```

W celu określenia **kolejności** w jakiej mają być wykonywane operacje, **respektuje się nawiasy** oraz odwołanie do takich pojęć jak **priorytet** i **wiązanie** operatora.

Uwzględnienie wiązań

Ponieważ wiązanie operatora przypisania jest **prawe** więc wyrażenie

```
a = b = c
```

zostanie potraktowane jako wyrażenie

```
a = (b = c)
```

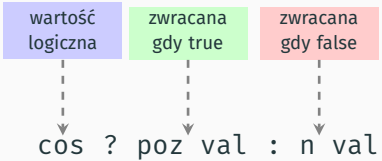
PRIORYTETY OPERATORÓW

Priorytet	Wyrażenie pierwotne
0	new (x) x,y x[y] f(args)

Priorytet	Operator	wiązanie
1	++ – (następnikowe)	lewe
2	++ – (poprzednikowe)	prawe
3	+ - ! (Type)	prawe
4	* / %	lewe
5	- +	lewe
6	« » »>	lewe
7	< <= > >= instanceof	lewe
8	== !=	lewe
9	&	lewe
10	^	lewe
11	|	lewe
12	&&	lewe
13	||	lewe
14	?:	prawe
15	= += -= *= /= %= &= ^= |= «= »= »>=	prawe

OPERATOR WYBORU

OPERATOR WYBORU



Dziękuje

Dziękuje