Wyższa Szkoła Bankowa
Gdańsk Gdynia

Analiza danych w języku Java I

Zajęcia 4 - Operatory

Anna Pakeizer

Data

Operatory

4 + 3

gdzie:

3,**4** operandy

+ operator

= operator przypisania

Przykłady

Przykład 1 - kolejność obliczeń

Co wydrukuje poniższy kod?

```
class ABC {
    public static void main(String[] args) {
        int wynik1 = 1 + 1 * 2 + 1 * 3 - 1;
        System.out.println(wynik1);
        int wynik2 = (1 + 1) * (2 + 1) * (3 - 1);
        System.out.println(wynik2);
    }
}
```

Stos/sterta

- stos (stack), sterta (heap)
- Typy podstawowe przechowywane są na stosie.
- Stos przechowuje również referencje do obiektów.
- Obiekty przechowywane są na stercie.

Stos	Sterta
	
`int x = 7;`	
`Samochod s =`	`new Samochod();`

Co wydrukuje poniższy kod?

Przykład 2 - kolejność obliczeń

```
int x = 4;
int y = x;
x = 5;
System.out.println("x: " + x);
System.out.println("y: " + y);
```

Co wydrukuje poniższy kod?

Przykład 2 - kolejność obliczeń

```
int x = 4;
int y = x;
x = 5;
System.out.println("x: " + x);
System.out.println("y: " + y);
```

x: 5
y: 4

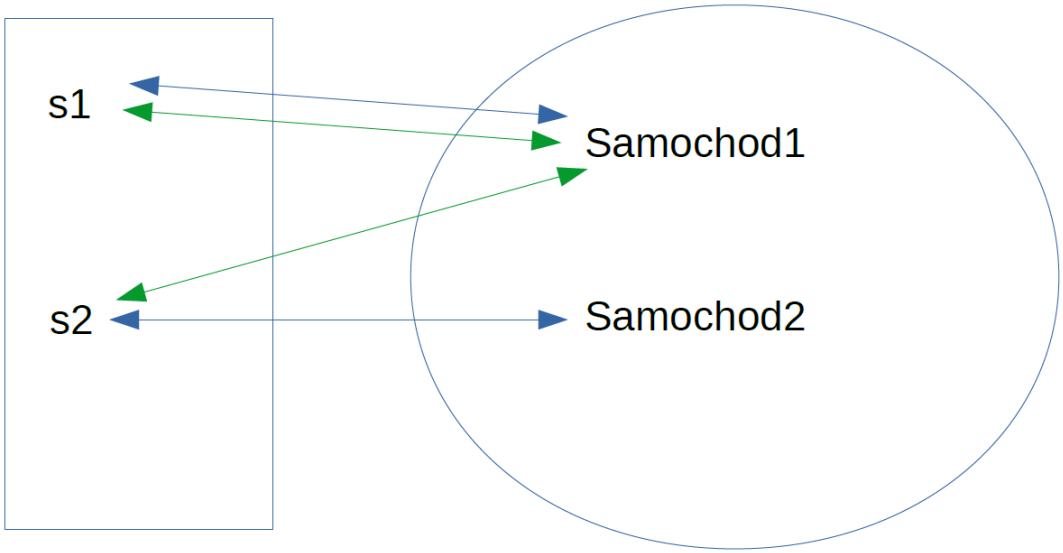
- Typy proste przechowują wartość a nie referencję do obiektu.
- Co wydrukuje poniższy kod? (Przykład 3)

```
class TEST {
    public static void main(String[] args) {
        Samochod s1 = new Samochod();
        Samochod s2 = new Samochod();
        s1.pojemnosc = 1998;
        s2 = s1;
        s1.pojemnosc = 998;
        System.out.println("s1: " + s1.pojemnosc);
        System.out.println("s2: " + s2.pojemnosc);
    }
}

class Samochod{
    double pojemnosc;
}
```

- Typy proste przechowują wartość a nie referencję do obiektu.
- Co wydrukuje poniższy kod?

```
class TEST {
    public static void main(String[] args) {
        Samochod s1 = new Samochod();
        Samochod s2 = new Samochod();
        s1.pojemnosc = 1998;
        s2 = s1;
        s1.pojemnosc = 998;
        System.out.println("s1: " + s1.pojemnosc);
        System.out.println("s2: " + s2.pojemnosc);
    }
}

class Samochod{
    double pojemnosc;
}
```

s1: 998.0
s2: 998.0



Kolor niebieski strzałek:

s1 wskazuje na obiekt Samochod1.

s2 wskazuje na obiekt Samochod2.

s2 = s1 sprawia, ze s2 zaczyna wskazywać na obiekt Samochod1.

Kolor zielony strzałek:

s1 wciąż wskazuje na Samochod1

s2 również wskazuje na Samochod1. s2 jest *aliasem* dla Samochod1.

Operatory matematyczne

+ - / * %

Skrócony zapis, jednoczesna operacja i przypisanie

+=

Przykład 4

```
int a = 1, b = 2, c = 3;
int w;
w = a+b;     System.out.println("a+b: " + w);
w = a-b;     System.out.println("a-b: " + w);
w = a*b;     System.out.println("a*b: " + w);
w = a/b;     System.out.println("a/b: " + w);
w = b/c;     System.out.println("b/c: " + w);
w = 8/9;     System.out.println("8/9: " + w);
w = 3%3;     System.out.println("3%3: " + w);
w = 4%3;     System.out.println("4%3: " + w);
w = 10%7;    System.out.println("10%7: " + w);
w = 2017%2; System.out.println("2017%2: " + w);
w = 2018%2; System.out.println("2018%2: " + w);
w = 2019%2; System.out.println("2019%2: " + w);
w = 5;
w %=2;       System.out.println("w%2: " + w); // najpierw operacja, potem
przypisanie
w = 5;
w +=2;       System.out.println("w+=2: " + w);
w = 5;
w -=2;       System.out.println("w-=2: " + w);
w = 5;
w *=2;       System.out.println("w*=2: " + w);
w = 5;
w /=2;       System.out.println("w/=2: " + w);
```

Jednoargumentowe operatory minus i plus

Przykład 5

```
int x = 2;
x = -2;
System.out.println(x);
x =-2;
System.out.println(x);
x = 2;
x =- 2;
System.out.println(x);
```

Operatory zwiększania i zmniejszania

Preinkrementacja i postinkrementacja

Przykład 6

```
int i = 10;
System.out.println(++i);
i = 10;
System.out.println(i++);
System.out.println(i);
```

Operatory relacji

- operatory, które zwracają wartości logiczne, tj. typu boolean (true, false) (Przykład 7)

```
System.out.println("10 == 10: " + (10 == 10));
System.out.println("10 != 10: " + (10 != 10));
System.out.println("true == true: " + (true == true));
System.out.println("true != true: " + (true != true));
System.out.println("'a' == 'a': " + ('a' == 'a'));
System.out.println("'a' != 'a': " + ('a' != 'a'));
System.out.println("500L == 500L: " + (500L == 500L));
System.out.println("500L != 500L: " + (500L != 500L));
```

Równość obiektów

Przykład 8

```
class ABC {
    public static void main(String[] args) {
        int x = 10;
        int y = 10;
        System.out.println(x == y);
        y = 20;
        System.out.println(x == y);

        Integer a = 10;
        Integer b = 10;
        System.out.println(a == b);

        Samochod s1 = new Samochod();
        Samochod s2 = new Samochod();
        System.out.println(s1 == s2); // sprawdzane referencje, nie wartości

    }

}

class Samochod {

}
```

Równość obiektów c.d.

- Używając operatora == do sprawdzenia równości obiektów uzyskamy błędne rezultaty.
- W przypadku obiektów operator == porównuje referencje obiektów (adresy na stercie).
- Mając dwie różne instancje obiektów mają one dwa różne adresy w pamięci w związku z tym zawsze ich adresy są różne.
- W przypadku obiektów przy pomocy operatora == możemy sprawdzić czy dwie referencje wskazują na ten sam obiekt.

Równość obiektów

Przykład 9

```
int x = 10;
int y = 10;
System.out.println(x != y);
```

Równość zawartości obiektów

Przykład 10

```
class ABC {
    public static void main(String[] args) {
        Samochod s1 = new Samochod();
        Samochod s2 = new Samochod();
        System.out.println(s1.equals(s2)); // sprawdzane referencje, nie wartości
    }

}

class Samochod {
}
```

Operatory logiczne, koniunkcja (&&), alternatywa (||), negacja (!)

Przykład 11

```
int x = 10;
int y = 10;
int z = 20;
System.out.println(x != y);
System.out.println((x == y) && (y < z));
System.out.println((x == y) && (y > z));
System.out.println((x == y) || (y > z));
```

Operator trójargumentowy if-else (ang. ternary operator)

wyrażenie_logiczne ? wylicz1 : wylicz2

Czytaj: Czy *wyrażnie_logiczne* jest prawdą? Jeśli tak to *wylicz1*. Jeśli nie to *wylicz2*.

Przykład 12

```
double x = 0.1;
System.out.println( x < 1 ? x * 100  + " %" : x * 1 + " %");
x = 20;
System.out.println( x < 1 ? x * 100  + " %" : x * 1 + " %");
```

+, += dla klasy String

Przykład 13

```
String s = "a";
int x = 3;
System.out.println(s+"b");
s = "2";
System.out.println(s+"2");
System.out.println(x + 7);
x = 3;
System.out.println("" + x + 7);

s = "a";
System.out.println(s+="b");
```

+= dla klasy String

```
s = "a";
System.out.println(s+="b");
```

s -> ab

Immutability (niezmienność) Stringów

- Raz utworzony obiekt typu string nigdy nie zmieni już swojej wartości.
- Klasa String nie posiada żadnego settera, którym moglibyśmy zmienić jego wartość, dlatego chcąc zmienić tę wartość, tworzymy nowy obiekt oraz modyfikujemy referencje przechowującą odwołanie do tego stringa.
- W powyższym przykładzie utworzyliśmy nowy obiekt String o wartości *“ab”* i ustawiliśmy na niego zmienną referencyjną *s*.