TYPY DANYCH

```
// ***** Typy proste *****
//
// Typy całkowite
byte b = 1; //-128 do 127
short s = 2; // -32 768 do 32 767
int i = 4; // -2 147 486 648 do 2 147 483 647
long l = 8; // od -9 223 372 036 775 808 do 9 223 372 036 775 807

// Typy zmiennoprzecinkowe
float f = 4.00f; // od 1.4e-045 do +3.4e+38 (3.4 * 10 do potęgi 38)
double d = 8.00; // od 4.9e-324 do +1,8e+308 (1.8 * 10 do potęgi 308)
// Domyślnie wszystkie liczby zmiennoprzecinkowe traktowane są jako double
// Aby liczba była traktowana jako float należy na końcu dopisać literę f lub F

//Typ znakowy
char c1 = 'A'; //Zmienna przechowuje kod znaku, dla A jest to 65
char c2 = 65; //zamiast znaku, możemy podać jego kod

//Typ logiczny
boolean bo = true;  // tylko dwie wartości -  true lub false

// ***** Typy obiektowe *****
//
// Typ łańcuchowy
String tekst = "Ważny tekst";
```

ZMIENNE

```
// Deklarowanie zmiennej
  int liczba;  //deklaruje zmienną o nazwie (identyfikatorze) liczba
  liczba = 100;  //nadaje wartosc zmiennej 'liczba' - 100
  System.out.println("Wartość liczby wynosi: " + liczba);

  liczba = 50;
  System.out.println("Nowa wartość liczby wynosi: " + liczba);

  int liczba2 = 75; // deklaruje zmienną i nadaje jej wrtość początkową
  System.out.println("Wartość drugiej liczby wynosi: " + liczba2);

  // Deklarowanie stałej
   //deklaruje stałą (słowo "final") o nazwie TEMPERATURA_WRZENIA
    final float TEMPERATURA_WRZENIA = 100;
  System.out.println("Temperatura wrzenia wody wynosi: " +
TEMPERATURA_WRZENIA);
    // temperaturaWrzenia = 50; Błąd, nie można zmienić wartości stałej
```

OPERATORY

```
int a = 5;
int b = 2;
int wynikArytm;

// *** Operatory arytmetyczne ***
// +  -  *  /  % ++ --
wynikArytm = a + b; // + dodawanie
wynikArytm = a - b; // - odejmowanie
wynikArytm = a * b; // * mnożenie
wynikArytm = a / b; // / dzielenie;
wynikArytm = a % b; // % reszta z dzielenia (modulo)
wynikArytm = a++; // ++ zwiększenie liczby o 1
wynikArytm = a--; // ++ zmniejszenie liczby o 1

// *** Operatory relacji ***
// ==  !=  >  <  >=  <=
boolean wynikRelacji;
wynikRelacji = a == b; // == równy
wynikRelacji = a != b; // != różny
wynikRelacji = a >= b; // > wiekszy, >= większy lub równy
wynikRelacji = a <= b; // < mniejszy, <= mniejszy lub równy

// *** Operatory logiczne ***
// &  &&  |  ||  !
boolean prawda = true;
boolean falsz = false;
boolean wynikLogiczny;
// &, && koniunkcja (i) - prawdziwe, gdy oba warunki są prawdziwe
wynikLogiczny = prawda && falsz;
// |, || alternatywa (lub) - prawdziwe, gdy choć jeden warunek jest prawdziwy
wynikLogiczny = prawda || falsz;
// ! negacja (zaprzeczenie) - warunek przeciwny do podanego
wynikLogiczny = !prawda;

// *** Operatory bitowe ***
// działają bezpośrednio na bitach …
```

INSTRUKCJE WARUNKOWE

```
int liczbaZyc = 1;
liczbaZyc--;
// instrukcja "if"
if (liczbaZyc == 0) {
    System.out.println("Koniec gry");
}

int liczba = 5;
// if - złożony warunek
if (liczba >= 0 && liczba <= 10) {
    System.out.println("Liczba należy do przedziału [0-10]");
}

// instrukcja if-else
if (liczba >= 0) {
    System.out.println("Liczba nieujemna");
} else {
    System.out.println("liczba ujemna");
}

// instrukcja if-else if
if (liczba > 0) {
    System.out.println("Liczba dodatnia");
} else if (liczba == 0) {
    System.out.println("Liczba jest równa 0");
} else {
    System.out.println("Liczba ujemna");
}

int liczbaCalkowita = -10;
// instrukcja switch
switch (liczbaCalkowita) {
    case 0://...
        break;
    case 100://...
        break;
    default: ;
}
// intrukcja Switch (z łańcuchami)
String kierunek = "South";

switch (kierunek) {
    case "East":
        System.out.println("Wschód");
        break;
    case "West":
        System.out.println("Zachód");
        break;
    case "North":
```

PĘTLE

```
// Pętle to instrukcje, które wykonują te same działania wiele razy.
-----------------------------------------------------------------------------------------------------------
// Pętla while
/*
Pętla wykonuje działania tak długo, jak długo warunek w nawiasie jest spełniony.
 */
int x = 0;
while (x < 5) {
    System.out.println("W pętli 'while: " + x);
    x++;
}
// Pętla 'do-while'
/*
 Pętla wykonuje działania tak długo, jak długo warunek w nawiasie jest spełniony,
 z tym że warunek ten jest sprawdzany na końcu.
 Taka pętla zawsze będzie wykonana przynajmniej jeden raz.
 */
x = 0;
do {
    System.out.println("W pętli 'do-while': " + x);
    x++;
} while (x < 5);
-----------------------------------------------------------------------------------------------------------
// Pętla for
for (int i = 1; i < 10; i++) {
    System.out.println(i);
}
System.out.println();
/*  (1) Wyrażenie początkowe 'int liczba = 1' wykonywane jest tylko jeden raz, na
   początku.
    Zmienna "i" zlicza ile razy pętla została wykonana.
    (2) Wyrażenie 'liczba < 10' sprawdzane jest przed każdym wykonaniem pętli,
    gdy warunek jest prawdziwy, to pętla jest wykonana, w przeciwnym wypadku
   zostaje zakończona
    (3) Wyrażenie 'liczba++' jest obliczane po każdym przebiegu pętli
 */
// Wartość licznika można zmieniać w dowolny sposób np. zmniejszać o 2.
for (int i = 10; i >0; i = i - 2) {
    System.out.println(i);
}
System.out.println();

// Zmienna "i" jest dostępna tylko w pętli
// Jeśli chcemy zmienną wykorzystać poza pętlą, to musimy ją zadeklarować
wcześniej.
int licznik;
for (licznik = 0; licznik < 5; licznik++) {
    System.out.println(licznik);
}
System.out.println("Pętlę wykonano " + licznik + " razy");
```

PĘTLE

// Instrukcje break i continue;

```
// Instrukcja 'continue' przerywa działanie pętli
// i wykonywana jest następna iteracja.

System.out.println("Instrukcja 'continue'");
int liczba = 10;
for (int dzielnik = -3; dzielnik <= 2; dzielnik++) {
    if (dzielnik == 0) {
        System.out.println("Nie można dzielić przez 0.");
        continue;
    }
    System.out.println(liczba + "/" + dzielnik + " = " + liczba / dzielnik);
}

// Instrukcja 'break' natychmiast kończy działanie pętli.

System.out.println();
System.out.println("Instrukcja 'break'");
liczba = 10;
for (int dzielnik = -3; dzielnik <= 2; dzielnik++) {
    if (dzielnik == 0) {
        System.out.println("Nie można dzielić przez 0.");
        break;
    }
    System.out.println(liczba + "/" + dzielnik + " = " + liczba / dzielnik);
}

Scanner odczyt = new Scanner(System.in);
String tekst;
do {
    System.out.print("Wpisz tekst (k - koniec):");
    tekst = odczyt.nextLine();
    System.out.println("Wpisałeś " + tekst);
} while (!tekst.equals("k"));
```

TABLICE

// Deklaracja zmiennej tablicowej
int[] tablica1; // pierwszy sposób deklarowania (czytelniejszy)
int tablica2[]; // drugi sposób deklarowania tablicy

// Deklaracja tablicy z wyszczególnieniem elementów
int[] dane = {2, 9, 5};
System.out.println(dane[0]);
System.out.println(dane[1]);
System.out.println(dane[2]);
// System.out.print(dane[3]);//Przekroczenie zakresu tablicy

// Deklaracja tablicy bez podawania danych, rezerwacja miejsca na 3 liczby
int[] liczby = new int[3];
// Przypisanie wartości poszczególnym elementom tablicy
liczby[0] = 2;
liczby[1] = 9;
liczby[2] = 5;
System.out.println("Pierwsza liczba to " + liczby[0]);

// Deklaracja tablicy łańcuchów
String[] poryRoku = {"wiosna", "lato", "jesień", "zima"};
System.out.print("To są pory roku: ");
System.out.print(poryRoku[0] + ", ");
System.out.print(poryRoku[1] + ", ");
System.out.print(poryRoku[2] + ", ");
System.out.println(poryRoku[3]);

// Dodatkowe własności
// Tablica jest obiektem i oprócz danych posiada dodatkowe własności.
// Można na przykład uzyskać informację ile elementów jest w tablicy.
System.out.println("Ilośc elementów tablicy poryRoku - " + poryRoku**.length**);

Łańcuchy

```
// Deklaracja łańcuchów
String tekst1 = "lato";        String tekst2 = "zima";

// Znaki specjalne w łańcuchach
\n - koniec linii
\" - cudzysłów
\\ - ukośnik – backslash

// Długość łańcucha
int dlugosc = tekst1.length();
System.out.println("tekst1 ma długość " + dlugosc);

// Pojedynczy znak na danej pozycji
char znak = tekst1.charAt(3);
System.out.println("Znak numer 3 z tekst1 to: " + znak);

// Sprawdzenie, czy tekst zawiera inny tekst
int pozycja = tekst1.indexOf("abc");
System.out.println("Czy tekst1 zawiera \"abc\" ? " + +pozycja);

String nowyTekst;
// Zamiana wszystkich znaków na duże
nowyTekst = tekst1.toUpperCase();
System.out.println("Zamiana na duże litery: " + nowyTekst);

// Zamiana wybranego znaku na inny
nowyTekst = tekst1.replace("o", "a");
System.out.println("Zamiana litery 'a' na 'o' " + nowyTekst);

// Podłańcuch
nowyTekst = tekst1.substring(2, 4);
System.out.println("Wycinek tekst1 od znaku 2 do znaku 4: " + nowyTekst);

// Połączenie łańcuchów
nowyTekst = tekst1 + tekst2;
System.out.println("Połączone łańcuchy 1 i 2: " + nowyTekst);

// Sprawdzenie czy dwa łańcuchy sa takie same.
// Do porównania służy metoda 'equals'. Nie wolno używać operatora ==
boolean czyTakieSame = tekst1.equals(tekst2);
System.out.println("Czy tekst1 i tekst2 są takie same? " + czyTakieSame);

// Podział łańcucha na części przy użyciu znaku rozdzielającego np. spacji, przecinka
String poryRoku = "wiosna lato jesień zima";
String[] wyrazy = poryRoku.split(" ");
System.out.println("Tekst został podzielony na " + wyrazy.length + " części");

// Zamiana łańcucha na tablicę znaków
char[] znaki = poryRoku.toCharArray();
```

Łańcuchy - formatowanie

/* Czasami chcemy, aby tekst wyglądał w zaplanowany przez nas sposób
np. był wyrównany do lewej strony albo liczby wyświetlały się do drugiego
miejsca po przecinku. Takie porządkowanie tekstu nazywa się **formatowaniem**.
Aby sformatować tekst musimy przygotować opis tego jak ma wyglądać taki tekst.
Przykładowy opis to "**%f8.2**" lub "**%10d**".
W miejsce znaku '%' będą wstawiane dane np. liczba, łańcuch itp.

W opisie używane są różne informacje:
d - liczba całkowita dziesiętna
f - liczba zmiennoprzecinkowa
s - łańcuch
c - znak
b - wartość logiczna
\n oznacza przejście do nowej linii
*/

// Drukowanie sformatowanego tekstu
double liczba = 99.1234;
// 8 znaków liczby, w tym trzy po przecinku.
System.out.printf(**"%8.3f", liczba**);
// Dwie liczby oddzielone średnikiem.
System.out.printf(**"%d ; %d\n", 11, 12**);

// Tekst można też sformatować przed jego wyświetleniem.
// ustalamy, że tekst ma zająć 10 miejsc
String tekstSformatowany1 = String.format("%10s", 10);
String tekstSformatowany2 = String.format("%10s", 100);
System.out.println(tekstSformatowany1);
System.out.println(tekstSformatowany2);

Biblioteki

// **Drukowanie informacji**
System.out.**println**("Tekst do wyświetlenia");

// **Losowanie liczby**
Random losowanie = **new Random()**;
losowanie.**nextInt(100)**; // losuje liczbę typu int od 0 do 99
losowanie.**nextFloat()**; // losuje liczbę typu float
losowanie.**nextBoolean()**; // losuje wartość true lub false

// **Pobieranie danych z klawiatury**
Scanner odczyt = **new Scanner(System.in)**; lub
Scanner odczyt = **new Scanner**(System.in, "Windows-1250");
odczyt.**nextInt()**; // odczytuje liczbę typu int
odczyt.**nextFloat()**; // odczytuje liczbę typu float
odczyt.**next()**; // odczytuje następny element jako łańcuch
odczyt.**nextLine()**; // odczytuje całą linię jako łańcuch