# ELEMENTY JĘZYKA JAVA

**komentarze w Javie, słowa kluczowe i operatory,**
**proste typy danych, tablice,**
**podstawowy zestaw instrukcji.**

FUW

# Komentarze w Javie

- **Komentarz wierszowy**

```
// Program wypisujący tekst powitania
```

- **Komentarz blokowy**

```
/* Program wypisujący tekst powitania
   Warszawa, 13 listopada 2002 r. */
```

- **Komentarz dokumentacyjny**

```
/**
 * Klasa rysująca wykres. Typ wykresu
 * zależy od naciśniętego przycisku.
 * @version 1.0
 */
class Wykres {...
```

# Tworzenie dokumentacji

**Do opisu fragmentów kodu źródłowego programu używa się komentarzy. Na ich podstawie, używając programu javadoc można później wygenerować dokumentację. Najczęściej opisuje się elementy takie jak klasy, metody, interfejsy czy obiekty. Komentarze powinny być krótkie, perecyzyjne. Należy je umieszczać bezpośrednio przed dokumentowanym elementem programu.**

- **Aby tekst komentarza został rozpoznany przez javadoc, musi być 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 | widefp | |

# 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 | Dodawanie, odejmowanie |

# Operatory i ich priorytety

| 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

| 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 | =<br>*=<br>/=<br>+=<br>-=<br>%= | Zmienna, dowolny | 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 dwa rodzaje: 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**.

# Proste typy danych

| *Typ danych* | *Rozmiar (w bitach)* | *Wartość domyślna* | *Opis* |
|---|---|---|---|
| **boolean** | 8 | false | przyjmuje wartosci logiczne **true** lub **false** |
| **byte** | 8 | 0 | wartość całkowita 8-bitowa ze znakiem |
| **char** | 16 | 'x0' | kod znaku w 16-bitowym kodzie Unicode |
| **short** | 16 | 0 | wartość całkowita 16-bitowa ze znakiem |
| **int** | 32 | 0 | wartość całkowita 32-bitowa ze znakiem |
| **long** | 64 | 0 | wartość całkowita 32-bitowa ze znakiem |
| **float** | 32 | 0.0F | wartość zmiennoprzecinkowa 32-bitowa |
| **double** | 64 | 0.0D | wartość zmiennoprzecinkowa 64-bitowa |

# Tablice

- Tablica - to  ciąg zmiennych tego samego typu, opisanych jedną wspólną nazwą.
- Elementy tablicy identyfikuje się je 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 tablice:**

```
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:**

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

# 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];
```

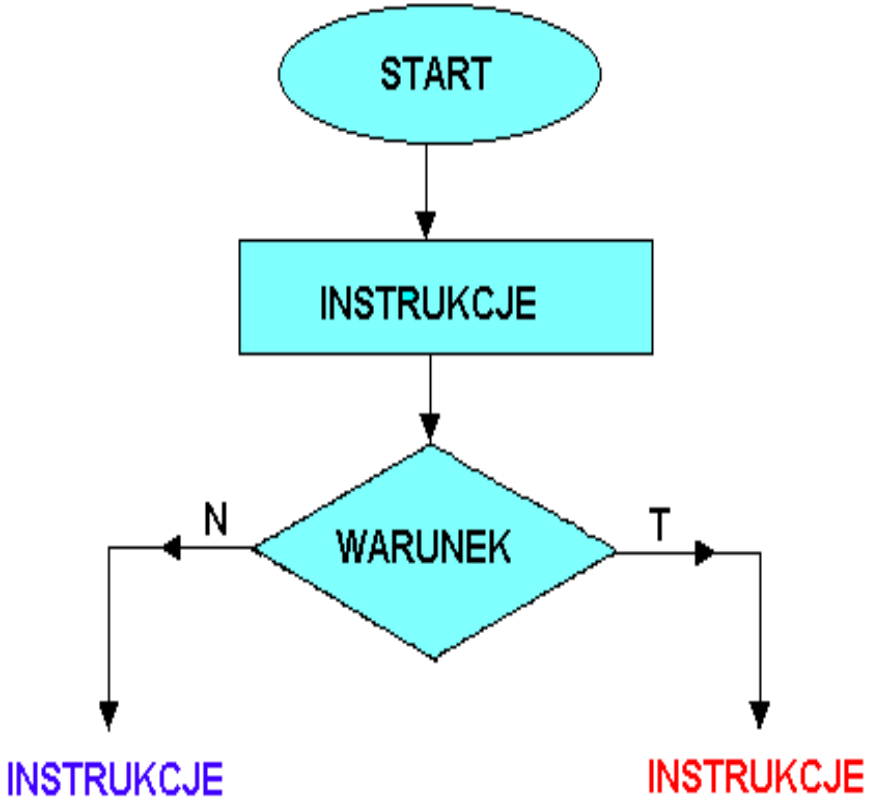
- **Przykład instrukcji tworzącej tablicę o dwóch wymiarach (jest to najczęściej używana forma tablicy wielowymiarowej) :**
  - `int[][] A = new int[10][10];`
    lub
  - `int A[][] = new int[10][10];`

# Podstawowy zestaw instrukcji

- Instrukcja warunkowa if
- Pętla while
- Pętla do while
- Pętla for
- Instrukcja switch
- Instrukcje break i continue

# Postać instrukcji if

```
if (warunek)
{
    instrukcje
}
else
{
    instrukcje
}
```



# Wyrażenia logiczne (warunki)

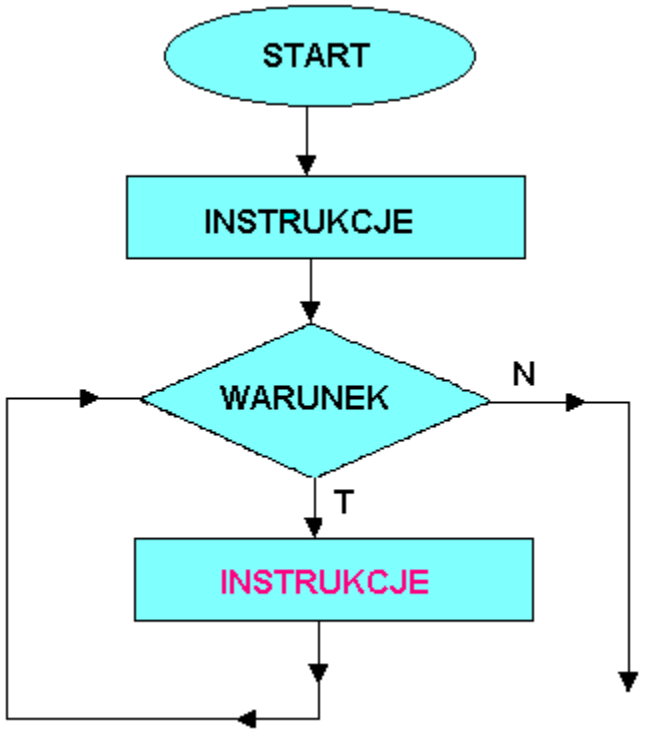
- Wyrażenia logiczne można przypisywać zmiennym typu boolean. Ich wartością może być **`true`** lub **`false`**.
- Proste wyrażenia logiczne konstruuje się za pomocą operatorów relacji:
  - <, >, <=, >=, == (czy równe), != (czy różne).
- Do budowy bardziej złożonych wyrażeń używa się operatorów logicznych:
  - **&&** (koniunkcji – logiczne AND),
  - || (alternatywy – logiczne OR),
  - **!** (negacji – logiczne NOT).

# Przykład wielowarunkowej instrukcji if

```
double x = 723*221-658*243;
if (x > 0)
{
    System.out.println("x jest dodatnie");
}
else if (x == 0)
{
    System.out.println("x jest równe zero");
}
else
{
    System.out.println("x jest ujemne");
}
```

# Instrukcja while

```
while ( warunek )
{
    instrukcje
}
```



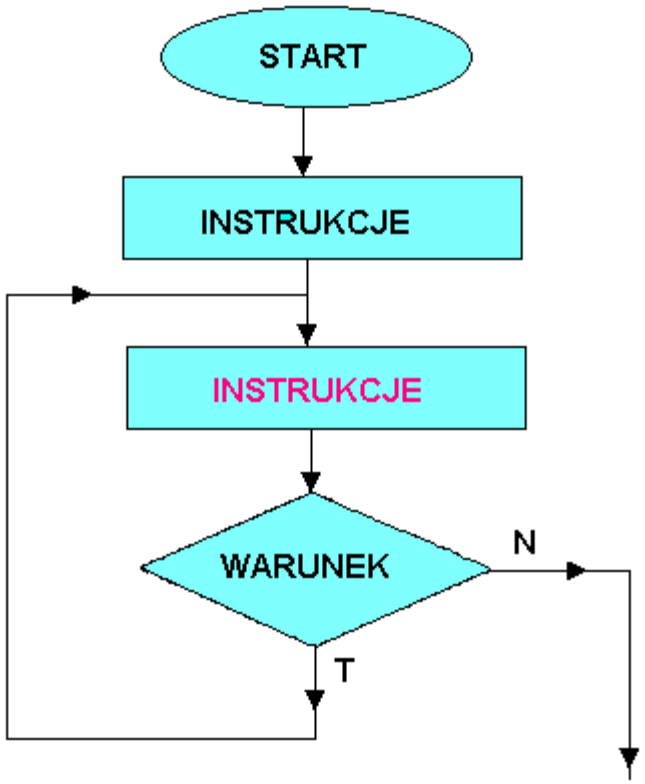

Przykład pętli **while (**dopóki**)**

```
while ( samochód jest brudny )
{
    myj_samochód( );
}
```

Ile razy wykona się instrukcja wewnętrzna pętli, gdy samochód jest czysty?

# Instrukcja do while

```
do
{
    instrukcje
}
while ( warunek );
```




Przykład pętli **do (**wykonuj dopóki**)**

```
do
{
    myj_samochód( );
}
while (samochód jest brudny);
```

Ile razy wykona się instrukcja wewnętrzna pętli, gdy samochód jest czysty?

FUW

# Instrukcja for

```
for ( wyrażenie1 ; warunek ; wyrażenie2 )
{
   instrukcje
}
```

Przykład pętli **for (**dla**)**

```
for  ( int i=0; i<10; i++)
{
    myj_samochód( );
}
```

Zmienną i nazywamy zmienną sterującą pętli.

Zasięg działania zmiennej **i** jest ograniczony do pętli **for**.

FUW

# Jaka jest końcowa wartość zmiennej i?

```
class MyApp
{
  public static void main(String args[])
  {
    int i;

    for (i=0;i<100;i++)
    {
       i=3*i+2;
    }
    System.out.println("i = " + i);

  }
}
```

| i | i<100 |
|---|---|
| 0 | 1 |
| 2 | |
| 3 | 1 |
| 11 | |
| 12 | 1 |
| 38 | |
| 39 | 1 |
| 119 | |
| 120 | 0 |

# Instrukcja switch

```
switch ( wyrażenie)
{
   case Wartość1 :
   /* ... */
   break;

   case Wartość2 :
   /* ... */
   break;

   default :
   /* ... */
   break;
}
```

# Instrukcje break i continue

```
etykieta:
for (int i=1; i<10; i++)
{
    /* po continue program zacznie
    wykonywać się tutaj */
    for(int j=1; j<100; j++)
    {
       if (j==20)
       {
         continue etykieta;
       }
    }
}
```

# Ćwiczenie 1

- **Napisz program, który znajduje wszystkie trzycyfrowe liczby, których suma sześcianów poszczególnych cyfr jest równa danej liczbie.**

## Wskazówka

Skorzystaj z operatorów arytmetycznych % i / dla argumentów całkowitych, gdzie:

% - operator modulo - reszta z dzielenia.

Na przykład wartością wyrażenia `937%10` jest liczba 7;

/ - operator dzielenia całkowitego.

Na przykład wartością wyrażenia `937/10` jest liczba 93 (ale wartością `937/10.0` jest liczba 93.7).

# Rozwiązanie

```
class Liczby
{
  public static void main(String args[])
  {
    int cj, cd, cs;
    for ( int i=100; i<1000; i++ )
    {
       cj = i%10;          // cyfra jedności liczby i
       cd = (i/10)%10;     // cyfra dziesiątek
       cs = (i/100)%10;    // cyfra setek
       if (cj*cj*cj + cd*cd*cd + cs*cs*cs == i)
       {
          System.out.print(i + "  ");
       }
    }
    System.out.println();
  }
}
```

# Zadania

- **Wypisz wszystkie liczby dwucyfrowe o takiej własności: Podwojony sześcian danej liczby jest wielokrotnością sumy jej cyfr.**
- **Napisz program, który wypisuje na ekranie wszystkie czterocyfrowe liczby palindromiczne (są to takie liczby, które czytane normalnie i wspak mają taką samą wartość, np. 1221**
- **Napisz program, który wypisuje 20 początkowych liczb czterocyfrowych o takiej własności, że: suma pierwszej i drugiej cyfry równa się sumie trzeciej i czwartej cyfry. Przykładem takiej liczby jest 3764, bo 3+7=6+4.**

# Ćwiczenie 2

- **Napisz program obliczający iloczyn skalarny wektorów a i b, gdzie**
  - **a = [1, 5, 8, 9, 11]**
  - **b = [3, 4, 7, 15, 32]**

## Wskazówka

Niewielkie tabele inicjalizuje się wartościami początkowymi, np.:

```
int a[]={1, 5, 8, 9, 11};
int b[]={3, 4, 7, 15, 32};
```

Rozmiar tabel wyznaczany jest wtedy na podstawie liczby wstawionych elementów i nie może być zmieniany w dalszej części programu.

# Rozwiązanie

```
class Wektory {
    public static void main(String args[])
    {
        int a[]={1, 5, 8, 9, 11};
        int b[]={3, 4, 7, 15, 32};
        int s=0;
        for (int i=0; i<5; i++)
        {
            s=s+a[i]*b[i];
        }
        System.out.println("a * b = " + s);
    }
}
```

# Ćwiczenie 3

- **Napisz program, który:**
  - **losuje n=10 liczb całkowitych z zakresu od 1 do 30 i zapisuje je w tablicy jednowymiarowej a,**
  - **wypisuje elementy tablicy a.**

## Wskazówka

Do wygenerowania liczby losowej z zakresu od 0.0 do 1.0 służy funkcja random() zawarta w klasie Math.

Instrukcja przypisania zmiennej x wartości losowej z przedziału od 1 do 30 ma postać:

```
int x = (int)(1+Math.random()*30);
```

# Rozwiązanie

```
class Tablica
{
   static final int n = 10;
   public static void main(String args[])
   {
      int a[] = new int[n];
      for ( int i=0; i<n; i++ )
      {
        a[i] = (int)(1+Math.random()*30);
      }
     for ( int i=0; i<n; i++ )
     {
        System.out.print(a[i] + "\t");
     }
 }
}
```

# Modyfikacja 1

- **Zmodyfikuj program tak, aby korzystał z funkcji typu void, przeznaczonych do:**
  - **losowania liczb do wektora,**
  - **wypisywania elementów wektora.**
- **Oto nagłówki tych funkcji:**
  - `public static void losuj(int[] a)`
  - `public static void wypisz(int[] a)`
- **Dopisz funkcję, która wypisze elementy wektora od końca. Oto nagłówek tej funkcji:**
  - `public static void wypisz_wspak(int[] a)`

# Rozwiązanie

```
class Tablica
{
    static final int n = 10;
    public static void main(String args[])
    {
        int a[] = new int[n];
        losuj(a);
        wypisz(a);
        wypisz_wspak(a);
    }

    public static void losuj(int a[])
    {
        for ( int i=0; i<n; i++ )
        {
            a[i] = (int)(1+Math.random()*30);
        }
    }
```

# Rozwiązanie

```
    public static void wypisz(int a[])
    {
        System.out.println("Elementy wektora");
        for ( int i=0; i<n; i++ )
        {
            System.out.print(a[i] + "\t");
        }
    }
    public static void wypisz_wspak(int a[])
    {
        System.out.println("Elementy wspak");
        for ( int i=n-1; i>=0; i-- )
        {
            System.out.print(a[i] + "\t");
        }
    }
}
```

# Modyfikacja 2

- **Zmodyfikuj kod programu tak, aby wyświetlał dodatkowo informację o liczbie wylosowanych liczb parzystych i nieparzystych. W tym celu napisz, a następnie wykorzystaj w programie funkcję typu boolean, służącą do sprawdzania parzystości danej liczby.**
  **Oto nagłówek tej funkcji:**

```
public static boolean czy_p(int k)
```

# Rozwiązanie

## Definicja funkcji

```
public static boolean czy_p(int k)
{
   boolean b = k%2==0;
   return b;
}
```

## Wywołanie funkcji

```
int liczp=0, licznp=0;
for (int i=0; i<n; i++)
{
   if (czy_p(a[i])) liczp++;
   else licznp++;
}
```

FUW

# Modyfikacja 3

- **Napisz funkcję, która posortuje elementy wektora w porządku rosnącym.**
  **Oto nagłówek tej funkcji:**

```
public static void sortuj(int[] a)
```

# Rozwiązanie

## Definicja funkcji sortuj()

```
public static void sortuj(int[] a)
{
    int i, j, pom;
    for (i=0; i<n-1; i++)
    for (j=0; j<n-i-1; j++)
        if (a[j]>a[j+1])
        {
            pom = a[j];
            a[j] = a[j+1];
            a[j+1] = pom;
        }
}
```

# Ćwiczenie 4

- **Napisz program, w którym do jednowymiarowej tablicy składającej się z 10 elementów zostaną wylosowane liczby z zakresu od 1 do 30.**
- **Wypisz tablicę na ekranie, a następnie komunikat "TAK" jeśli w tablicy są elementy powtarzające się, lub "NIE" w przeciwnym przypadku.**
- **Policz również liczbę powtarzających się par.**

# Rozwiązanie

```
class Tablica
{
      public static void main(String args[])
      {
            int a[] = new int[10];

            // Losowanie liczb do tablicy
             for ( int i=0; i<10; i++ )
            {
               a[i] = 1 + (int)(Math.random()*30);
               System.out.print(a[i] + "\t");
            }
            System.out.println();
```

# Rozwiązanie

```
        int s = 0; // suma par

        for ( int i=0; i<10; i++ )
        for ( int j=i+1; j<10; j++ )
        {
           if (a[i] == a[j]) s++;
        }

        if (s!=0)
        {
          System.out.println("TAK\t" + s);
        }
        else
        {
          System.out.println("NIE");
        }
    }
}
```