

Mirosław J. Kubiak

# Java

*Zadania z programowania  
z przykładowymi rozwiązaniami*

*Java w analizie konkretnych przykładów*

- Proste operacje wejścia/wyjścia
- Tablice oraz iteracje
- Programowanie obiektowe i pliki tekstowe



# Spis treści

<b>Od autora</b>	<b>5</b>
<b>Rozdział 1. Proste operacje wejścia-wyjścia</b>	<b>7</b>
Operacje wejścia-wyjścia — informacje ogólne	7
<b>Rozdział 2. Podejmujemy decyzje w programie</b>	<b>19</b>
Instrukcje warunkowe w języku Java	19
<b>Rozdział 3. Iteracje</b>	<b>31</b>
Iteracje — informacje ogólne	31
Pętla for	32
Pętla do ... while	33
Pętla while	33
<b>Rozdział 4. Tablice</b>	<b>57</b>
Deklarowanie tablic jednowymiarowych	57
Dostęp do elementów tablicy	58
Tablice dwuwymiarowe	62
<b>Rozdział 5. Programowanie obiektowe</b>	<b>79</b>
Programowanie obiektowe — informacje ogólne	79
Rekurencja	92
<b>Rozdział 6. Pliki tekstowe</b>	<b>97</b>
Pliki tekstowe — informacje ogólne	97

# Od autora

Trójzbiór *Zadania z programowania z rozwiązaniami* to pierwszy w Polsce zbiór zadań adresowany do wszystkich osób zainteresowanych programowaniem, które w krótkim czasie, poprzez analizę zaproponowanych rozwiązań, chciałyby nauczyć się solidnie podstaw programowania w trzech językach: Turbo Pascalu, C++ oraz Javie.

Składa się on z trzech zbiorów zadań:

*Turbo Pascal. Zadania z programowania z przykładowymi rozwiązaniami.*

*C++. Zadania z programowania z przykładowymi rozwiązaniami.*

*Java. Zadania z programowania z przykładowymi rozwiązaniami.*

Chociaż każdy z tych zbiorów **stanowi odrębną całość**, to zostały one napisane w taki sposób, aby ten sam lub bardzo podobny problem programistyczny (np. napisz program, który oblicza pole prostokąta) został rozwiązany w **trzech** językach programowania: Turbo Pascalu, C++ i Javie, strukturalnie i obiektowo. Tak skonstruowany trójzbiór *Zadania z programowania* zyskuje zupełnie nowy wymiar dydaktyczny w nauce tych trzech języków.

*Zadania z programowania* można również wykorzystać jako uzupełnienie wiedzy zaczerpniętej z innych książek do nauki programowania. Zakres i stopień trudności zadań pokrywa się z tradycyjnym procesem nauczania wymienionych języków. Zbiór ten może też pełnić rolę podręcznej pomocy dla początkujących programistów, w której szybko znajdą oni potrzebne im rozwiązanie.

Trójzbiór adresowany jest również do maturzystów, studentów, nauczycieli informatyki oraz osób zainteresowanych programowaniem lub rozpoczynających naukę programowania w języku Java.

Uczniowie techników informatycznych mogą zbiory zadań wykorzystać do szybkiej powtórki przed egzaminem zawodowym.

W trakcie pisania tej książki korzystałem z kompilatora Java NetBeans (<http://www.netbeans.org>).

Mirosław J. Kubiak

# 1

## Proste operacje wejścia-wyjścia

*W tym rozdziale zamieszczono proste zadania wraz z przykładowymi rozwiązaniami ilustrujące, w jaki sposób komputer komunikuje się z użytkownikiem w języku Java.*

### Operacje wejścia-wyjścia — informacje ogólne

Każda aplikacja powinna posiadać możliwość komunikowania się z użytkownikiem. Wykorzystując proste przykłady, pokażemy, w jaki sposób program napisany w języku Java komunikuje się z nim poprzez standardowe operacje wejścia-wyjścia.

Operacje wejścia-wyjścia w Javie są realizowane poprzez strumienie. **Strumień** jest pojęciem abstrakcyjnym. Może on wysyłać i pobierać informacje i jest połączony z fizycznym urządzeniem (np. klawiatura, ekran) poprzez system wejścia-wyjścia. W języku tym zdefiniowano dwa typy strumieni: bajtowe i znakowe. Standardowy strumień wyjściowy w Javie jest reprezentowany przez obiekt<sup>1</sup> `out` znajdujący się

---

<sup>1</sup> Obiekty zostaną omówione w rozdziale 5.

w klasie `System`. Jest to obiekt statyczny klasy `PrintStream` zawierający metody `print()` i `println()`.

Metoda `println()` wyświetla argumenty podane w nawiasach `()`, a następnie przechodzi do początku nowej linii. Pewną jej odmianą jest metoda `print()`. Jej działanie polega na wyświetlaniu argumentów podanych w nawiasach `()` bez przemieszczania kursora do nowego wiersza.

---

**ZADANIE****1.1**

Napisz program, który oblicza pole prostokąta. Wartości boków `a` i `b` wprowadzamy z klawiatury. W programie należy przyjąć, że zmienne `a`, `b` oraz pole są typu `double` (rzeczywistego).

---

*Przykładowe rozwiązanie — listing 1.1*

---

```
package zadanie11; //Zadanie 1.1
import java.io.*;

public class Main
{
    public static void main(String[] args)
        throws IOException
    {
        double a, b, pole;

        BufferedReader br = new BufferedReader(new
            ↳InputStreamReader(System.in));

        System.out.println("Program oblicza pole prostokąta.");
        System.out.println("Podaj bok a.");
        a = Double.parseDouble(br.readLine());
        System.out.println("Podaj bok b.");
        b = Double.parseDouble(br.readLine());
        pole = a*b;

        System.out.print("Pole prostokąta o boku a = " + a + "
            ↳i boku b = " + b);
        System.out.println(" wynosi " + pole + ".");
    }
}
```

---

Klasy w Javie grupowane są w jednostki zwane pakietami (ang. *package*). **Pakiet** to zestaw powiązanych ze sobą tematycznie klas. Do jego utworzenia służy słowo kluczowe `package`, po którym następuje nazwa pakietu zakończona średnikiem, co ilustruje linijka kodu poniżej:

```
package zadanie11; //Zadanie 1.12
```

### Linijka kodu

```
double a, b, pole;
```

umożliwia deklarację zmiennych `a`, `b` i `pole` (wszystkie są typu rzeczywistego — `double`) w programie. Instrukcja

```
System.out.println("Program oblicza pole prostokąta.");
```

wyświetla na ekranie komputera komunikat *Program oblicza pole prostokąta*.

W celu czytania z klawiatury liter i cyfr należy skorzystać z dwóch klas: `InputStreamReader` oraz `BufferedReader`. Najpierw tworzymy nowy obiekt klasy `InputStreamReader`, przekazując jej konstruktorowi obiekt `System.in`. Można go następnie wykorzystać w konstruktorze klasy `BufferedReader`. Tak opisana konstrukcja ma następujący zapis:

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

Powstały obiekt klasy `BufferedReader` możemy przypisać do zmiennej referencyjnej `br` i dalej, poprzez metodę `readLine()`, możemy wykorzystać go do wczytywania zmiennej `a` typu `double` ze strumienia wejściowego. Ilustruje to następująca linijka kodu:

```
a = Double.parseDouble(br.readLine());
```

Wczytywanie liczb odbywa się tak samo jak wczytywanie tekstu, musimy jednak dokonać odpowiedniej konwersji, tzn. zamiany ciągu znaków na odpowiadającą mu wartość liczbową. Służy do tego jedna z poniższych metod statycznych:

- ❑ `parseByte` z klasy `Byte` do odczytu bajtów,
- ❑ `parseDouble` z klasy `Double` do odczytu liczb typu `double`,
- ❑ `parseFloat` z klasy `Float` do odczytu liczb typu `float`,
- ❑ `parseInt` z klasy `Int` do odczytu liczb typu `int`,
- ❑ `parseLong` z klasy `Long` do odczytu liczb typu `long`.

Aby nasz program mógł zostać skompilowany, musimy do niego dodać następujące dwie linijki kodu:

```
import java.io.*;
```

---

<sup>2</sup> Komentarze w programie oznaczamy dwoma ukośnikami `//`; *// to jest komentarz*.

oraz

throws IOException

Są one niezbędne do obsługi błędów wejścia-wyjścia. Słowo kluczowe `import` oznacza, że do programu zaimportowano wszystkie (po kropce \*) pakiety *java.io*.

Pole prostokąta zostaje obliczone w instrukcji

```
pole = a*b;
```

Za wyświetlenie wartości zmiennych `a` i `b` oraz `pole` wraz z odpowiednim opisem są odpowiedzialne następujące linijki kodu:

```
System.out.print("Pole prostokąta o boku a = " + a + " i boku b = " + b);  
System.out.println(" wynosi " + pole + ".");
```

Rezultat działania programu można zobaczyć na rysunku 1.1.

**Program oblicza pole prostokąta.**

**Podaj bok a.**

**1**

**Podaj bok b.**

**2**

**Pole prostokąta o boku a = 1.0 i boku b = 2.0 wynosi 2.0.**

*Rysunek 1.1. Efekt działania programu Zadanie 1.1*

---

## ZADANIE

### 1.2

Napisz program, który wyświetla na ekranie komputera wartość predefiniowanej stałej  $\pi = 3,14\dots$  Należy przyjąć format wyświetlania tej stałej z dokładnością do pięciu miejsc po przecinku.

#### Wskazówka

Język Java umożliwia formatowanie wyświetlanych danych w podobny sposób jak w języku C. Służy do tego metoda `printf`. Jej składnia jest następująca:

```
String format;  
System.out.printf(format, arg_1, arg_2, ..., arg_n);
```



---

Przykładowe rozwiązanie — listing 1.2

---

```
package zadanie12; //Zadanie 1.2

public class Main
{
    public static void main(String[] args)
    {
        System.out.println("Program wyświetla liczbę pi zadaną
        ↳dokładnością.");
        System.out.printf("Pi = " + "%6.5f\n", Math.PI);
    }
}
```

---

Specyfikatory typów mogą być następujące:

- ❑ %d — integer,
- ❑ %e — double,
- ❑ %f — float.

Pomiędzy znakiem % i literą przyporządkowaną danemu typowi można określić ilość pól, na których ma zostać wyświetlona liczba, np.:

%7.2f — oznacza przyznanie siedmiu pól na liczbę typu float, w tym dwóch pól na jej część ułamkową;

%4d — oznacza przyznanie czterech pól na liczbę typu całkowitego.

W programie zapis

```
System.out.printf("Pi = " + "%6.5f\n", Math.PI);
```

powoduje, że na wydruk liczby  $\pi$  przeznaczonych zostaje sześć pól, w tym pięć na część ułamkową. Znak specjalny "...  
" (ang. *new line*) oznacza przejście na początek nowego wiersza. Math jest standardową klasą Javy, która umożliwia obliczenia matematyczne.

Rezultat działania programu można zobaczyć na rysunku 1.2.

**Program wyświetla liczbę pi zadaną dokładnością.**  
**Pi = 3,14159**

*Rysunek 1.2. Efekt działania programu Zadanie 1.2*

## ZADANIE

**1.3**

Napisz program, który wyświetla na ekranie komputera pierwiastek kwadratowy z wartości predefiniowanej  $\pi = 3,14...$ . Należy przyjąć format wyświetlania pierwiastka kwadratowego ze stałej  $\pi$  z dokładnością do dwóch miejsc po przecinku.

*Przykładowe rozwiązanie — listing 1.3*

---

```
package zadanie13; //Zadanie 1.3

public class Main
{
    public static void main(String[] args)
    {
        System.out.println("Program wyświetla pierwiastek kwadratowy ");
        System.out.println("z liczby pi z dokładnością dwóch miejsc po  
↪przecinku.");
        System.out.printf("Sqrt(Pi) = " + "%2.2f\n", Math.sqrt(Math.PI));
    }
}
```

---

Metoda `sqrt()` pozwala na obliczenie pierwiastka kwadratowego. Jest ona metodą standardowej klasy `Math`.

Rezultat działania programu można zobaczyć na rysunku 1.3.

**Program wyświetla pierwiastek kwadratowy  
z liczby pi z dokładnością dwóch miejsc po przecinku.  
Sqrt(Pi) = 1,77**

*Rysunek 1.3. Efekt działania programu Zadanie 1.3*

## ZADANIE

**1.4**

Napisz program, który oblicza objętość kuli o promieniu  $r$ . Wartość promienia wprowadzamy z klawiatury. W programie należy przyjąć, że zmienne: promień  $r$  i objętość, są typu `double` (rzeczywistego). Dla tych zmiennych należy przyjąć format wyświetlania na ekranie z dokładnością do dwóch miejsc po przecinku.

---

*Przykładowe rozwiązanie — listing 1.4*

---

```
package zadanie14; //Zadanie 1.4
import java.io.*;

public class Main
{
    public static void main(String[] args)
        throws IOException
    {
        double r, objetosc;

        BufferedReader br = new BufferedReader(new
            ↳InputStreamReader(System.in));

        System.out.println("Program oblicza objętość kuli.");
        System.out.println("Podaj promień r.");
        r = Double.parseDouble(br.readLine());

        objetosc = 4.0*Math.PI*r*r*r/3;

        System.out.print("Objętość kuli o promieniu r = ");
        System.out.printf("%2.2f", r);
        System.out.print(" wynosi ");
        System.out.printf("%2.2f", objetosc);
    }
}
```

---

Objętość kuli o promieniu  $r$  oblicza następująca linijka kodu:

```
objetosc = 4.0*Math.PI*r*r*r/3;
```

gdzie potęgowanie zamieniono na mnożenie.

Rezultat działania programu można zobaczyć na rysunku 1.4.

**Program oblicza objętość kuli.**

**Podaj promień r.**

**1**

**Objętość kuli o promieniu  $r = 1,00$  wynosi 4,19.**

**Rysunek 1.4.** Efekt działania programu Zadanie 1.4

## ZADANIE

**1.5**

Napisz program, który oblicza wynik dzielenia całkowitego bez reszty dwóch liczb całkowitych:  $a = 37$  i  $b = 11$ .

**Wskazówka**

W języku Java w przypadku zastosowania operatora dzielenia / dla liczb całkowitych reszta wyniku jest pomijana (tak samo jest w C i C++).

W Turbo Pascalu należy zastosować operator dzielenia całkowitego bez reszty div.

---

*Przykładowe rozwiązanie — listing 1.5*

---

```
package zadanie15; //Zadanie 1.5

public class Main
{
    public static void main(String[] args)
    {
        int a = 37;
        int b = 11;

        System.out.println("Program wyświetla wynik dzielenia całkowitego");
        System.out.println("bez reszty dwóch liczb całkowitych.");
        System.out.println("Dla liczb a = " + a + ", b = " + b);
        System.out.println(a + "/" + b + " = " + a/b + ".");
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 1.5.

**Program wyświetla wynik dzielenia całkowitego  
bez reszty dwóch liczb całkowitych.**

**Dla liczb  $a = 37$ ,  $b = 11$**

**$37/11 = 3$ .**

*Rysunek 1.5. Efekt działania programu Zadanie 1.5*

## ZADANIE

**1.6**

Napisz program, który oblicza resztę z dzielenia całkowitego dwóch liczb całkowitych:  $a = 37$  i  $b = 11$ .

**Wskazówka**

Należy zastosować operator reszty z dzielenia całkowitego modulo, który oznaczamy w języku Java symbolem `%`. Podobnie jak w językach C i C++, operator ten umożliwia uzyskanie tylko reszty z dzielenia, natomiast wartość całkowita jest odrzucana.

*Przykładowe rozwiązanie — listing 1.6*

```
package zadanie16; //Zadanie 1.6

public class Main
{
    public static void main(String[] args)
    {
        int a = 37;
        int b = 11;

        System.out.println("Program oblicza resztę z dzielenia
        ↪całkowitego");
        System.out.println("dwóch liczb całkowitych.");
        System.out.println("Dla liczb a = " + a + ", b = " + b);
        System.out.println(a + "%" + b + " = " + a%b + ".");
    }
}
```

Rezultat działania programu można zobaczyć na rysunku 1.6.

**Program oblicza resztę z dzielenia całkowitego  
dwóch liczb całkowitych.**  
**Dla liczb a = 37, b = 11**  
**37%11 = 4.**

*Rysunek 1.6. Efekt działania programu Zadanie 1.6*

## ZADANIE

**1.7**

Napisz program, który oblicza sumę, różnicę, iloczyn i iloraz dla dwóch liczb  $x$  i  $y$  wprowadzanych z klawiatury. W programie należy założyć, że zmienne  $x$  i  $y$  są typu float (rzeczywistego). Dla zmiennych  $x$ ,  $y$ , suma, roznica, iloczyn i iloraz należy przyjąć format ich wyświetlania na ekranie z dokładnością do dwóch miejsc po przecinku.

---

*Przykładowe rozwiązanie — listing 1.7*

---

```
package zadanie17; //Zadanie 1.7
import java.io.*;

public class Main
{

    public static void main(String[] args)
        throws IOException
    {
        float x, y, suma, roznica, iloczyn, iloraz;

        BufferedReader br = new BufferedReader(new
            ↳InputStreamReader(System.in));

        System.out.println("Program oblicza sumę, różnicę, iloczyn
            ↳i iloraz ");
        System.out.println("dla dwóch liczb x i y wprowadzonych
            ↳z klawiatury.");
        System.out.println("Podaj x.");
        x = Float.parseFloat(br.readLine());
        System.out.println("Podaj y.");
        y = Float.parseFloat(br.readLine());

        suma = x+y;
        roznica = x-y;
        iloczyn = x*y;
        iloraz = x/y;

        System.out.printf("Dla liczb x = " + "%.2f",x);
        System.out.printf(" i y = " + "%.2f",y);
        System.out.println(); //wyświetlenie pustej linii
        System.out.printf("suma = " + "%.2f,\n", + suma);
        System.out.printf("różnica = " + "%.2f,\n", + roznica);
        System.out.printf("iloczyn = " + "%.2f,\n", + iloczyn);
        System.out.printf("iloraz = " + "%.2f,\n", + iloraz);

    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 1.7.

**Program oblicza sumę, różnicę, iloczyn i iloraz  
dla dwóch liczb x i y wprowadzonych z klawiatury.**

**Podaj x.**

**3**

**Podaj y.**

**2**

**Dla liczb  $x = 3,00$  i  $y = 2,00$**

**suma = 5,00,**

**różnica = 1,00,**

**iloczyn = 6,00,**

**iloraz = 1,50.**

*Rysunek 1.7. Efekt działania programu Zadanie 1.7*





# 2

## Podejmujemy decyzje w programie

*W tym rozdziale przedstawimy typowe zadania wraz z przykładowymi rozwiązaniami z wykorzystaniem instrukcji warunkowych.*

### Instrukcje warunkowe w języku Java

W Javie istnieją dwie instrukcje warunkowe:

- ❑ instrukcja warunkowa `if ... else`,
- ❑ instrukcja wyboru `switch ... case`.

Instrukcja `if ... else` służy do sprawdzania poprawności wyrażenia warunkowego i w zależności od tego, czy dany warunek jest prawdziwy, czy nie, pozwala na wykonanie różnych bloków programu.

Jej ogólna postać jest następująca:

```
if (warunek)
{
    // instrukcje do wykonania, kiedy warunek jest prawdziwy
}
else
{
    // instrukcje do wykonania, kiedy warunek jest fałszywy
}
```

Blok `else` jest opcjonalny i instrukcja warunkowa w wersji skróconej ma postać

```
if (warunek)
{
    // instrukcje do wykonania, kiedy warunek jest prawdziwy
}
```

Instrukcja wyboru `switch ... case` pozwala w wygodny i przejrzysty sposób sprawdzić ciąg warunków i wykonywać kod w zależności od tego, czy są one prawdziwe, czy fałszywe. Oto jej ogólna postać:

```
switch (wyrażenie)
{
    case wartość_1 : instrukcje_1;
    break;
    case wartość_2 : instrukcje_2;
    break;
    .....
    case wartość_n : instrukcje_n;
    break;
    default : instrukcje;
}
```

Instrukcja `break` przerywa wykonywanie całego bloku `case`.

### Uwaga

Jej brak może doprowadzić do nieoczekiwanych wyników i błędów w programie.

---

## ZADANIE

### 2.1

Napisz program, który sprawdza dla trzech boków trójkąta `a`, `b` i `c` wprowadzonych z klawiatury, czy tworzą one trójkąt prostokątny (zakładamy, że  $a > 0$ ,  $b > 0$ ,  $c > 0$ ).

---

#### Przykładowe rozwiązanie — listing 2.1

```
package zadanie21; //Zadanie 2.1
import java.io.*;

public class Main
{
    public static void main(String[] args)
        throws IOException
    {
        int a, b, c;
```

```
BufferedReader br = new BufferedReader(new
↳InputStreamReader(System.in));

System.out.println("Podaj bok a.");
a = Integer.parseInt(br.readLine());
System.out.println("Podaj bok b.");
b = Integer.parseInt(br.readLine());
System.out.println("Podaj bok c.");
c = Integer.parseInt(br.readLine());

if ((a*a+b*b) == c*c)
{
    System.out.println("Boki");
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    System.out.println("tworzą trójkąt prostokątny.");
}
else
{
    System.out.println("Boki");
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    System.out.println("nie tworzą trójkąta prostokątnego.");
}
}
```

---

**Sprawdzenie twierdzenia Pitagorasa dla wczytanych boków a, b i c zostało zawarte w następujących liniach kodu:**

```
if ((a*a+b*b) == c*c)
{
    System.out.println("Boki");
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    System.out.println("tworzą trójkąt prostokątny.");
}
else
{
    System.out.println("Boki");
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    System.out.println("nie tworzą trójkąta prostokątnego.");
}
```

Łatwo potwierdzić, że boki  $a = 3$ ,  $b = 4$ ,  $c = 5$  tworzą trójkąt prostokątny (liczby te spełniają twierdzenie Pitagorasa), i na ekranie pojawi się komunikat *Boki... tworzą trójkąt prostokątny*, natomiast boki  $a = 1$ ,  $b = 2$ ,  $c = 3$  nie tworzą trójkąta prostokątnego (podane wartości nie spełniają twierdzenia Pitagorasa), więc na ekranie zostanie wyświetlony komunikat *Boki... nie tworzą trójkąta prostokątnego*.

Rezultat działania programu dla  $a = 3$ ,  $b = 4$ ,  $c = 5$  można zobaczyć na rysunku 2.1.

**Podaj bok a.**  
**3**  
**Podaj bok b.**  
**4**  
**Podaj bok c.**  
**5**  
**Boki**  
**a = 3**  
**b = 4**  
**c = 5**  
**tworzą trójkąt prostokątny.**

*Rysunek 2.1. Efekt działania programu Zadanie 2.1*

---

## ZADANIE

### 2.2

Napisz program, który oblicza pierwiastki równania kwadratowego  $ax^2+bx+c = 0$  z wykorzystaniem instrukcji warunkowej `if`, gdzie zmienne  $a$ ,  $b$  i  $c$  to liczby rzeczywiste wprowadzane z klawiatury. Dla zmiennych  $a$ ,  $b$ ,  $c$ ,  $x_1$  oraz  $x_2$  należy przyjąć format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po przecinku.

---

#### *Przykładowe rozwiązanie — listing 2.2*

```
package zadanie22; //Zadanie 2.2
import java.io.*;

public class Main {
    public static void main(String[] args)
        throws IOException
    {
        double a, b, c, delta, x1, x2;
```

```
BufferedReader br = new BufferedReader(new
↳InputStreamReader(System.in));

System.out.println("Program oblicza pierwiastki równania
↳ $ax^2+bx+c = 0.$ ");
System.out.println("Podaj a.");
a = Double.parseDouble(br.readLine());

if (a == 0)
{
    System.out.println("Niedozwolona wartość współczynnika a.");
}
else
{
    System.out.println("Podaj b.");
    b = Double.parseDouble(br.readLine());
    System.out.println("Podaj c.");
    c = Double.parseDouble(br.readLine());

    delta = b*b-4*a*c;

    if (delta < 0)
    {
        System.out.println("Brak pierwiastków rzeczywistych.");
    }
    else
    {
        if (delta == 0)
        {
            x1 = -b/(2*a);
            System.out.printf("Dla a = " + "%.2f", a);
            System.out.printf(" b = " + "%.2f", b);
            System.out.printf(" c = " + "%.2f\n", c);
            System.out.print("trójmian ma jeden pierwiastek
↳podwójny x1 = ");
            System.out.printf("%.2f.", x1);
        }
        else
        {
            x1 = (-b-Math.sqrt(delta))/(2*a);
            x2 = (-b+Math.sqrt(delta))/(2*a);
            System.out.printf("Dla a = " + "%.2f", a);
            System.out.printf(" b = " + "%.2f", b);
            System.out.printf(" c = " + "%.2f\n", c);
            System.out.println("trójmian ma dwa pierwiastki:");
            System.out.print("x1 = ");
            System.out.printf("%.2f.",x1);
            System.out.print(" x2 = ");
            System.out.printf("%.2f.\n",x2);
        }
    }
}
```

```
    }  
  }  
}
```

---

W pierwszej części programu sprawdzamy, czy wartość współczynnika  $a$  jest równa zero. Ilustrują to następujące linijki kodu:

```
if (a == 0)  
{  
    System.out.println("Niedozwolona wartość współczynnika a.");  
}  
else  
{  
    .....  
}
```

Jeśli  $a = 0$ , to zostanie wyświetlony komunikat *Niedozwolona wartość współczynnika a* i program zostanie zakończony. Dla  $a$  różnego od zera program będzie oczekiwał na wprowadzenie wartości  $b$  i  $c$ . Po ich wprowadzeniu zostanie obliczona delta według wzoru

```
delta = b*b-4*a*c;
```

Jeśli  $delta < 0$ , zostanie wyświetlony komunikat *Brak pierwiastków rzeczywistych*.

Jeśli  $delta = 0$ , równanie kwadratowe ma jeden pierwiastek podwójny, który obliczymy ze wzoru

```
x1 = -b/(2*a);
```

Jeśli  $delta > 0$ , równanie kwadratowe ma dwa pierwiastki obliczane ze wzorów

```
x1 = (-b-Math.sqrt(delta))/(2*a);  
x2 = (-b+Math.sqrt(delta))/(2*a);
```

Dla  $a = 1$ ,  $b = 5$  i  $c = 4$  wartości pierwiastków  $x_1$  i  $x_2$  równania wynoszą odpowiednio  $-4$  i  $-1$ .

Dla  $a = 1$ ,  $b = 4$  i  $c = 4$  trójmian ma jeden pierwiastek podwójny  $x_1 = -2$ .

Dla  $a = 1$ ,  $b = 2$  i  $c = 3$  równanie nie ma pierwiastków rzeczywistych.

Rezultat działania programu dla  $a = 1$ ,  $b = 5$  oraz  $c = 4$  można zobaczyć na rysunku 2.2.

**Program oblicza pierwiastki równania  $ax^2+bx+c = 0$ .**

**Podaj a.**

**1**

**Podaj b.**

**5**

**Podaj c.**

**4**

**Dla  $a = 1,00$ ,  $b = 5,00$ ,  $c = 4,00$**

**trójkian ma dwa pierwiastki:**

**$x1 = -4,00$ ,  $x2 = -1,00$ .**

*Rysunek 2.2. Efekt działania programu Zadanie 2.2*

## ZADANIE

### 2.3

Napisz program, który oblicza pierwiastki równania kwadratowego  $ax^2+bx+c = 0$  z wykorzystaniem instrukcji wyboru switch, gdzie zmienne a, b, c to liczby rzeczywiste wprowadzane z klawiatury. Dla zmiennych a, b, c, x1 oraz x2 należy przyjąć format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po przecinku.

#### Wskazówka

Należy wprowadzić do programu zmienną pomocniczą liczba\_pierwiastkow.

### Przykładowe rozwiązanie — listing 2.3

```
package zadanie23; // Zadanie 2.3
import java.io.*;

public class Main
{
    public static void main(String[] args)
        throws IOException
    {
        double a, b, c, delta, x1, x2;
        char liczba_pierwiastkow = 0;

        BufferedReader br = new BufferedReader(new
            ↳InputStreamReader(System.in));

        System.out.println("Program oblicza pierwiastki równania
            ↳ $ax^2+bx+c = 0$ .");
```

```
System.out.println("Podaj a.");
a = Double.parseDouble(br.readLine());

if (a == 0)
{
    System.out.println("Niedozwolona wartość współczynnika a.");
}
else
{
    System.out.println("Podaj b.");
    b = Double.parseDouble(br.readLine());
    System.out.println("Podaj c.");
    c = Double.parseDouble(br.readLine());

    delta = b*b-4*a*c;

    if (delta < 0)  liczba_pierwiastkow = 0;
    if (delta == 0) liczba_pierwiastkow = 1;
    if (delta > 0)  liczba_pierwiastkow = 2;

    switch (liczba_pierwiastkow)
    {
        case 0 : System.out.println("Brak pierwiastków
↳ rzeczywistych.");
            break;
        case 1 : { x1 = -b/(2*a);
                    System.out.printf("Dla a = " + "%4.2f.", a);
                    System.out.printf(" b = " + "%4.2f.", b);
                    System.out.printf(" c = " + "%4.2f\n", c);
                    System.out.print("trójkian ma jeden
↳ pierwiastek podwójny ");
                    System.out.print("x1 = ");
                    System.out.printf("%4.2f.", x1);
                }
            break;
        case 2 : { x1 = (-b+Math.sqrt(delta))/(2*a);
                    x2 = (-b-Math.sqrt(delta))/(2*a);
                    System.out.printf("Dla a = " + "%4.2f.", a);
                    System.out.printf(" b = " + "%4.2f.", b);
                    System.out.printf(" c = " + "%4.2f\n", c);
                    System.out.print("x1 = ");
                    System.out.printf("%4.2f.", x1);
                    System.out.print(" x2 = ");
                    System.out.printf("%4.2f.\n", x2);
                }
            break;
    }
}
}
```

---



Zmienna pomocnicza `liczba_pierwiastkow` przyjmuje trzy wartości zależnie od znaku zmiennej `delta`. Ilustrują to następujące linijki kodu:

```
if (delta < 0)  liczba_pierwiastkow = 0;
if (delta == 0) liczba_pierwiastkow = 1;
if (delta > 0)  liczba_pierwiastkow = 2;
```

Rezultat działania programu dla  $a = 1$ ,  $b = 4$  i  $c = 4$  można zobaczyć na rysunku 2.3.

**Program oblicza pierwiastki równania  $ax^2+bx+c = 0$ .**

**Podaj a.**

**1**

**Podaj b.**

**4**

**Podaj c.**

**4**

**Dla  $a = 1,00$ ,  $b = 4,00$ ,  $c = 4,00$**

**trójmian ma jeden pierwiastek podwójny  $x_1 = -2,00$ .**

*Rysunek 2.3. Efekt działania programu Zadanie 2.3*

## ZADANIE

### 2.4

Napisz program, który oblicza wartość  $x$  z równania  $ax+b = c$ . Wartości  $a$ ,  $b$  i  $c$  należą do zbioru liczb rzeczywistych i są wprowadzane z klawiatury. Dodatkowo należy zabezpieczyć program na wypadek sytuacji, kiedy wprowadzona wartość  $a$  będzie równa zero. Dla zmiennych  $a$ ,  $b$ ,  $c$  oraz  $x$  należy przyjąć format wyświetlania ich na ekranie z dokładnością do dwóch miejsc po przecinku.

#### *Przykładowe rozwiązanie — listing 2.4*

```
package zadanie24; //Zadanie 2.4
import java.io.*;

public class Main
{
    public static void main(String[] args)
        throws IOException
    {
        double a, b, c, x;
```

```
BufferedReader br = new BufferedReader(new
↳InputStreamReader(System.in));

System.out.println("Program oblicza wartość x z równania
↳liniowego  $ax+b = 0$ .");
System.out.println("Podaj a.");
a = Double.parseDouble(br.readLine());

if (a == 0)
{
    System.out.println("Niedozwolona wartość współczynnika a.");
}
else
{
    System.out.println("Podaj b.");
    b = Double.parseDouble(br.readLine());
    System.out.println("Podaj c.");
    c = Double.parseDouble(br.readLine());

    x = (c-b)/a;

    System.out.print("Dla a = ");
    System.out.printf("%.2f,", a);
    System.out.print(" b = ");
    System.out.printf("%.2f,", b);
    System.out.print(" c = ");
    System.out.printf("%.2f", c);
    System.out.print(" wartość x = ");
    System.out.printf("%.2f.", x);
}
}
```

---

Rezultat działania programu można zobaczyć na rysunku 2.4.

**Program oblicza wartość x z równania liniowego  $ax+b = 0$ .**

**Podaj a.**

**1**

**Podaj b.**

**2**

**Podaj c.**

**3**

**Dla a = 1,00, b = 2,00, c = 3,00 wartość x = 1,00.**

*Rysunek 2.4. Efekt działania programu Zadanie 2.4*

## ZADANIE

**2.5**

Napisz program, w którym użytkownik zgaduje losową liczbę z przedziału od 0 do 9 generowaną przez komputer.

**Wskazówka**

W języku Java liczby pseudolosowe generujemy przy użyciu klasy

```
Random r = new Random();
```

która jest dostępna po dołączeniu do programu pakietu

```
import java.util.*;
```

*Przykładowe rozwiązanie — listing 2.5*

```
package zadanie25; // Zadanie 2.5
import java.io.*;
import java.util.*;

public class Main
{
    public static void main(String[] args)
        throws IOException
    {
        double losuj_liczbe, zgadnij_liczbe;

        BufferedReader br = new BufferedReader(new
            ↳InputStreamReader(System.in));

        System.out.println("Program losuje liczbę z przedziału
            ↳od 0 do 9. ");
        System.out.println("Zgadnij tę liczbę.");

        Random r = new Random();
        losuj_liczbe = Math.round(10*( r.nextDouble()));
        zgadnij_liczbe = Double.parseDouble(br.readLine());

        if (zgadnij_liczbe == losuj_liczbe)
        {
            System.out.println("Gratulacje! Ogdadłeś liczbę!");
        }
        else
        {
            System.out.print("Bardzo mi przykro, ale wylosowana liczba
                ↳to ");
            System.out.println((int) losuj_liczbe + ".");
        }
    }
}
```

Funkcja `round()` w linijce kodu poniżej umożliwia zaokrąglenie liczby zmiennoprzecinkowej do liczby całkowitej:

```
losuj_liczbe = Math.round(10*( r.nextDouble()));
```

Zapis `(int)` `losuj_liczbe` w linijce kodu

```
System.out.println((int) losuj_liczbe + ".");
```

oznacza tzw. rzutowanie (ang. *casting*). **Rzutowanie** jest instrukcją dla kompilatora, aby zamienił on jeden typ na drugi. Zmienna `losuj_liczbe` jest typu `double`. Zapis `(int)` `losuj_liczbe` dokonuje zamiany na typ całkowity `int`.

Rezultat działania programu można zobaczyć na rysunku 2.5.

**Program losuje liczbę z przedziału od 0 do 9. Zgadnij tę liczbę.**

**1**

**Bardzo mi przykro, ale wylosowana liczba to 0.**

***Rysunek 2.5.** Efekt działania programu Zadanie 2.5*

# 3

## Iteracje

*W tym rozdziale przedstawimy typowe zadania wraz z przykładowymi rozwiązaniami z wykorzystaniem iteracji, czyli popularnych pętli. O ile młodzi programiści nie mają problemu z programami, w których wykorzystano instrukcję `for`, to jej zamiana na instrukcje `do ... while` oraz `while` nastręcza pewne trudności. Proste przykłady z instrukcją `for` zostały tu rozwiązane zarówno z wykorzystaniem `do ... while`, jak i `while`.*

### Iteracje — informacje ogólne

**Iteracja** (łac. *iteratio* — powtarzanie) to czynność powtarzania (najczęściej wielokrotnego) tej samej instrukcji (albo wielu instrukcji) w pętli.

W Javie istnieją trzy instrukcje iteracyjne:

- ❑ `for` (dla),
- ❑ `do ... while` (wykonuj dopóki),
- ❑ `while` (dopóki).

# Pętla for

Pętla for stosujemy, kiedy dokładnie wiemy, ile razy ma ona zostać wykonana. Istnieje wiele wariantów tej pętli, ale zawsze możemy wyróżnić trzy główne części.

1. **Inicjalizacja** to zwykle instrukcja przypisania stosowana do ustawienia początkowej wartości zmiennej sterującej.
2. **Warunek** jest wyrażeniem relacyjnym określającym moment zakończenia wykonywania pętli.
3. **Inkrementacja** (zwiększanie) lub **dekrementacja** (zmniejszanie) definiuje sposób modyfikacji zmiennej sterującej pętlą po zakończeniu każdego przebiegu (powtórzenia).

Te trzy główne składowe oddzielone są od siebie średnikami.

Pętla for wykonywana jest tak długo, dopóki wartość warunku wynosi true. Gdy warunek osiągnie wartość false, działanie programu jest kontynuowane od pierwszej instrukcji znajdującej się za pętlą.

W języku Java zmienna sterująca pętlą for nie musi być typu całkowitego, znakowego czy logicznego — może być ona również np. typu float.

Pętla for może być wykonywana tyle razy, ile wartości znajduje się w przedziale:

```
inicjalizacja; warunek; zwiększanie
```

lub

```
inicjalizacja; warunek; zmniejszanie
```

Ogólna postać tej instrukcji jest następująca:

```
for (inicjalizacja; warunek; zwiększanie)
{
    // instrukcje
}
```

lub

```
for (inicjalizacja; warunek; zmniejszanie)
{
    // instrukcje
}
```

W języku Java jest możliwa zmiana przyrostu zmiennej sterującej pętlą.

# Pętla do ... while

Kolejną instrukcją iteracyjną jest instrukcja `do ... while`. Jej ogólna postać jest następująca:

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

Cechą charakterystyczną instrukcji iteracyjnej `do ... while` jest to, że bez względu na wartość zmiennej `warunek` pętla musi zostać wykonana co najmniej jeden raz. Program po napotkaniu instrukcji `do ... while` wchodzi do pętli i wykonuje instrukcje znajdujące się w nawiasach klamrowych {}, a następnie sprawdza, czy `warunek` jest spełniony. Jeśli tak, wraca na początek pętli, natomiast jeśli `warunek` osiągnie wartość `false` (nieprawda), pętla się zakończy.

# Pętla while

Ostatnią instrukcją iteracyjną jest `while`. Jej ogólna postać jest następująca:

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

Cechą charakterystyczną tej pętli jest sprawdzanie warunku jeszcze przed wykonaniem instrukcji znajdujących się w bloku {...}. W szczególnym przypadku pętla może nie zostać wcale wykonana. Instrukcja `while` powoduje wykonywanie instrukcji tak długo, dopóki `warunek` jest prawdziwy.

## ZADANIE

### 3.1

Napisz program, który za pomocą instrukcji `for` dla danych wartości  $x$  zmieniających się od 0 do 10 oblicza wartość funkcji  $y = 3x$ .

**Przykładowe rozwiązanie — listing 3.1**

---

```
package zadanie31; //Zadanie 3.1

public class Main {

    public static void main(String[] args)
    {
        int x, y;

        System.out.println("Program oblicza wartość funkcji  $y = 3x$ ");
        System.out.println("dla x zmieniającego się od 0 do 10.");

        for (x = 0; x <= 10; x++)
        {
            y = 3*x;
            System.out.println("x = " + x + '\t' + "y = " + y);
        }
    }
}
```

---

**W pętli**

```
for (x = 0; x <= 10; x++)
{
    y = 3*x;
    System.out.println("x = " + x + '\t' + "y = " + y);
}
```

kolejne wartości  $x$ , zmieniające się automatycznie od  $x = 0$  (inicjalizacja) do  $x \leq 10$  (warunek) z krokiem równym 1 (zwiększanie), są podstawiane do wzoru

$y = 3x$ ;

a następnie zostają wyświetlone na ekranie dzięki użyciu polecenia

`System.out.println("x = " + x + '\t' + "y = " + y);`

Znak `'\t'` oznacza przejście do następnej pozycji w tabulacji linii.

Rezultat działania programu można zobaczyć na rysunku 3.1.

---

**ZADANIE****3.2**

Napisz program, który za pomocą instrukcji `do ... while` dla danych wartości  $x$  zmieniających się od 0 do 10 oblicza wartość funkcji  $y = 3x$ .



**Program oblicza wartość funkcji  $y = 3 \cdot x$   
dla  $x$  zmieniającego się od 0 do 10.**

<b>x = 0</b>	<b>y = 0</b>
<b>x = 1</b>	<b>y = 3</b>
<b>x = 2</b>	<b>y = 6</b>
<b>x = 3</b>	<b>y = 9</b>
<b>x = 4</b>	<b>y = 12</b>
<b>x = 5</b>	<b>y = 15</b>
<b>x = 6</b>	<b>y = 18</b>
<b>x = 7</b>	<b>y = 21</b>
<b>x = 8</b>	<b>y = 24</b>
<b>x = 9</b>	<b>y = 27</b>
<b>x = 10</b>	<b>y = 30</b>

**Rysunek 3.1.** Efekt działania programu Zadanie 3.1

*Przykładowe rozwiązanie — listing 3.2*

```
package zadanie32; //Zadanie 3.2

public class Main {

    public static void main(String[] args)
    {
        int x = 0, y = 0; //ustalenie wartości początkowych

        System.out.println("Program oblicza wartość funkcji  $y = 3 \cdot x$ ");
        System.out.println("dla  $x$  zmieniającego się od 0 do 10.");

        do
        {
            y = 3*x;
            System.out.println(" x = " + x + "\t" + " y = " + y);
            x++;
        }
        while (x <= 10);
    }
}
```

Pętla do ... while

```
do
{
    y = 3*x;
```

```
        System.out.println(" x = " + x + "\t" + " y = " + y);
        x++;
    }
    while (x <= 10);
```

nie posiada wbudowanego mechanizmu modyfikacji sterującej nią zmiennej, dlatego musimy do niej ten mechanizm dobudować. Rolę zmiennej sterującej pełni tutaj  $x$ . Zmienną tą powinniśmy przed pętlą wyzerować, stąd zapis

```
x = 0;
```

Następnie  $x$  należy zwiększać o krok, który w naszym przypadku wynosi 1. Ilustruje to następująca linijka kodu:

```
x++;
```

Pętla będzie powtarzana tak długo, aż zostanie spełniona zależność  $x \leq 10$ . Zwróćmy uwagę, że warunek sprawdzający zakończenie działania pętli, tzn. `while (x <= 10)`, znajduje się na jej końcu.

---

**ZADANIE****3.3**

Napisz program, który za pomocą instrukcji `while` dla danych wartości  $x$  zmieniających się od 0 do 10 oblicza wartość funkcji  $y = 3x$ .

---

*Przykładowe rozwiązanie — listing 3.3*

---

```
package zadanie33; //Zadanie 3.3

public class Main {

    public static void main(String[] args)
    {
        int x = 0, y = 0; //ustalenie wartości początkowych

        System.out.println("Program oblicza wartość funkcji y = 3*x");
        System.out.println("dla x zmieniającego się od 0 do 10.");

        while (x <= 10)
        {
            y = 3*x;
            System.out.println("x = " + x + "\t" + "y=" + y);
            x++;
        }
    }
}
```

---

## Pętla while

```
while (x <= 10)
{
    y = 3*x;
    System.out.println("x = " + x + '\t' + " y=" + y);
    x++;
}
```

podobnie jak do ... while nie posiada wbudowanego mechanizmu modyfikacji sterującej nią zmiennej, musimy więc do niej ten mechanizm dobudować. Rolę zmiennej sterującej pełni tutaj `x`. Zmienną `x` powinniśmy przed pętlą wyzerować, stąd zapis

```
x = 0;
```

Następnie należy ją zwiększać o krok, który w naszym przypadku wynosi 1. Ilustruje to następująca linijka kodu:

```
x++;
```

Pętla będzie powtarzana tak długo, aż stanie się prawdziwa zależność `x <= 10`. Zwróćmy uwagę, że warunek sprawdzający zakończenie działania pętli, tzn. `while (x <= 10)`, znajduje się na jej początku.

### ZADANIE

## 3.4

Napisz program, który za pomocą instrukcji `for` wyświetla liczby całkowite od 1 do 20.

### Przykładowe rozwiązanie — listing 3.4

```
package zadanie34; //Zadanie 3.4

public class Main {

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

        System.out.println("Program wyświetla liczby całkowite
        ↪od 1 do 20.");

        for (i = 1; i <= 20; i++)
        {
            if (i < 20)
                System.out.print(i + ", ");
            else
                System.out.print(i + ".");
        }
    }
}
```

```
    }  
  }  
}
```

---

Rezultat działania programu można zobaczyć na rysunku 3.2.

**Program wyświetla liczby całkowite od 1 do 20.**  
**1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.**

***Rysunek 3.2.** Efekt działania programu Zadanie 3.4*

---

**ZADANIE****3.5**

Napisz program, który za pomocą instrukcji do ... while wyświetla liczby całkowite od 1 do 20.

*Przykładowe rozwiązanie — listing 3.5*

---

```
package zadanie35; //Zadanie 3.5  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int i = 1; // ustalenie wartości początkowej  
  
        System.out.println("Program wyświetla liczby całkowite od 1 do  
        ↪20.");  
  
        do  
        {  
            if (i < 20)  
                System.out.print(i + ", ");  
            else  
                System.out.print(i + ".");  
            i++;  
        }  
        while (i <= 20);  
    }  
}
```

---

## ZADANIE

**3.6**

Napisz program, który za pomocą instrukcji `while` wyświetla liczby całkowite od 1 do 20.

*Przykładowe rozwiązanie — listing 3.6*

```
package zadanie36; //Zadanie 3.6

public class Main {

    public static void main(String[] args)
    {
        int i = 1; // ustalenie wartości początkowej

        System.out.println("Program wyświetla liczby całkowite
        ↪od 1 do 20.");

        while (i <= 20)
        {
            if (i < 20)
                System.out.print(i + ", ");
            else
                System.out.print(i + ".");
            i++;
        }
    }
}
```

## ZADANIE

**3.7**

Napisz program, który przy użyciu instrukcji `for` sumuje liczby całkowite od 1 do 100.

*Przykładowe rozwiązanie — listing 3.7*

```
package zadanie37; //Zadanie 3.7

public class Main {

    public static void main(String[] args)
    {
        int i, suma = 0;

        System.out.println("Program sumuje liczby całkowite od 1 do 100.");

        for (i = 1; i <= 100; i++)
        {
            suma = suma + i;
        }
    }
}
```

```
    }  
    System.out.println("Suma liczb całkowitych od 1 do 100 wynosi "  
        ↳+ suma + ".");  
    }  
}
```

---

Za sumowanie liczb od 1 do 100 odpowiedzialne są następujące linijki kodu:

```
for (i = 1; i <= 100; i++)  
{  
    suma = suma + i;  
}
```

Oczywiście przed pętlą zmienna `suma` musi zostać wyzerowana:

```
suma = 0;
```

Rezultat działania programu można zobaczyć na rysunku 3.3.

**Program sumuje liczby całkowite od 1 do 100.  
Suma liczb całkowitych od 1 do 100 wynosi 5050.**

**Rysunek 3.3.** *Efekt działania programu Zadanie 3.7*

---

## ZADANIE

### 3.8

Napisz program, który korzystając z instrukcji `do ... while`, sumuje liczby całkowite od 1 do 100.

*Przykładowe rozwiązanie — listing 3.8*

---

```
package zadanie38; //Zadanie 3.8  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int i = 1, suma = 0;  
  
        System.out.println("Program sumuje liczby całkowite od 1 do 100.");  
  
        do  
        {  
            suma = suma + i;  
            i++;  
        }  
        while (i <= 100);  
    }  
}
```

```
        System.out.println("Suma liczb całkowitych od 1 do 100  
        ↳wynosi " + suma + ".");  
    }  
}
```

**ZADANIE****3.9**

Napisz program, który przy wykorzystaniu instrukcji `while` sumuje liczby całkowite od 1 do 100.

*Przykładowe rozwiązanie — listing 3.9*

```
package zadanie39; //Zadanie 3.9  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int i = 1, suma = 0;  
  
        System.out.println("Program sumuje liczby całkowite od 1 do 100.");  
  
        while (i <= 100)  
        {  
            suma = suma + i;  
            i++;  
        }  
        System.out.println("Suma liczb całkowitych od 1 do 100 wynosi "  
        ↳+ suma + ".");  
    }  
}
```

**ZADANIE****3.10**

Napisz program, który za pomocą instrukcji `for` sumuje liczby parzyste z przedziału od 1 do 100.

**Wskazówka**

Należy skorzystać z właściwości operatora modulo `%`.

*Przykładowe rozwiązanie — listing 3.10*

```
package zadanie310; // Zadanie 3.10  
  
public class Main {  
  
    public static void main(String[] args)
```

```
{
    int i, suma = 0;

    System.out.println("Program sumuje liczby parzyste z przedziału
↳od 1 do 100.");

    for (i = 1; i <= 100; i++)
    {
        if (i%2 == 0) suma = suma+i;
    }
    System.out.print("Suma liczb parzystych z przedziału od 1 do 100
↳wynosi " );
    System.out.print(suma + ".");
}
}
```

---

Za sumowanie liczb parzystych występujących w przedziale od 1 do 100 odpowiedzialne są następujące wiersze kodu:

```
for (i = 1; i <= 100; i++)
{
    if (i%2 == 0) suma = suma+i;
}
```

Do wyodrębnienia liczb parzystych wykorzystaliśmy właściwości operatora modulo oznaczonego symbolem %. Użyliśmy w tym celu zapisu `i%2 == 0` — jeśli reszta z dzielenia całkowitego zmiennej `i%2` wynosi zero, to mamy do czynienia z liczbą parzystą, którą dodajemy do zmiennej `suma`.

Rezultat działania programu można zobaczyć na rysunku 3.4.

**Program sumuje liczby parzyste z przedziału od 1 do 100.  
Suma liczb parzystych z przedziału od 1 do 100 wynosi 2550.**

*Rysunek 3.4. Efekt działania programu Zadanie 3.10*

---

## ZADANIE

### 3.11

Napisz program, który za pomocą instrukcji `do ... while` sumuje liczby parzyste z przedziału od 1 do 100.

.....  
**Wskazówka**

Należy skorzystać z właściwości operatora modulo %.



---

*Przykładowe rozwiązanie — listing 3.11*

---

```
package zadanie311; // Zadanie 3.11

public class Main {

    public static void main(String[] args)
    {
        int i = 1, suma = 0;

        System.out.println("Program sumuje liczby parzyste z przedziału
        ↪od 1 do 100.");

        do
        {
            if (i%2 == 0) suma = suma+i;
            i++;
        }
        while (i <= 100);

        System.out.print("Suma liczb parzystych w przedziale od 1 do 100
        ↪wynosi ");
        System.out.print(suma + ".");
    }
}
```

---

**ZADANIE****3.12**

Napisz program, który za pomocą instrukcji `while` sumuje liczby parzyste w przedziale od 1 do 100.

**Wskazówka**

Należy skorzystać z właściwości operatora modulo `%`.

---

*Przykładowe rozwiązanie — listing 3.12*

---

```
package zadanie312; // Zadanie 3.12

public class Main {

    public static void main(String[] args)
    {
        int i = 1, suma = 0;

        System.out.println("Program sumuje liczby parzyste z przedziału
        ↪od 1 do 100.");
    }
}
```

```
        while (i <= 100)
        {
            if (i%2 == 0) suma = suma+i;
            i++;
        }
        System.out.print("Suma liczb parzystych z przedziału od 1 do 100
        ↳wynosi ");
        System.out.print(suma + ".");
    }
}
```

---

**ZADANIE****3.13**

Napisz program, który za pomocą instrukcji `for` sumuje liczby nieparzyste z przedziału od 1 do 100.

**Wskazówka**

Należy skorzystać z właściwości operatora modulo `%` i oznaczonego symbolem `!` operatora negacji.

*Przykładowe rozwiązanie — listing 3.13*

---

```
package zadanie313; // Zadanie 3.13

public class Main {

    public static void main(String[] args)
    {
        int i, suma = 0;

        System.out.print("Program sumuje liczby nieparzyste ");
        System.out.println("z przedziału od 1 do 100.");

        for (i = 1; i <= 100; i++)
        {
            if (!(i%2 == 0)) suma = suma+i;
        }
        System.out.print("Suma liczb nieparzystych z przedziału od 1 do
        ↳100 wynosi ");
        System.out.print(suma + ".");
    }
}
```

---

Za dodawanie liczb nieparzystych znajdujących się w przedziale od 1 do 100 odpowiedzialne są następujące linijki kodu:

```
for (i = 1; i <= 100; i++)  
{  
    if (!(i%2 == 0)) suma = suma+i;  
}
```

Do wyodrębnienia liczb nieparzystych wykorzystaliśmy właściwości operatora modulo oznaczonego symbolem `%` oraz właściwości oznaczonego znakiem `!` operatora negacji. Drugi z nich przekształca warunek prawdziwy w fałszywy, a fałszywy w prawdziwy. Jeśli reszta z dzielenia całkowitego zmiennej `!(i%2 == 0)` jest różna od zera, to mamy do czynienia z liczbą nieparzystą, którą dodajemy do zmiennej `suma`.

Rezultat działania programu można zobaczyć na rysunku 3.5.

**Program sumuje liczby nieparzyste z przedziału od 1 do 100.  
Suma liczb nieparzystych z przedziału od 1 do 100 wynosi 2500.**

*Rysunek 3.5. Efekt działania programu Zadanie 3.13*

## ZADANIE

### 3.14

Napisz program, który za pomocą instrukcji `do ... while` sumuje liczby nieparzyste w przedziale od 1 do 100.

#### Wskazówka

Należy skorzystać z właściwości operatora modulo `%` i operatora negacji `!`.

#### Przykładowe rozwiązanie — listing 3.14

```
package zadanie314; // Zadanie 3.14  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int i = 1, suma = 0;  
  
        System.out.print("Program sumuje liczby nieparzyste ");  
        System.out.println("w przedziale od 1 do 100.");  
  
        do  
        {  
            if (!(i%2 == 0)) suma = suma+i;  
            i++;  
        }  
        while (i <= 100);  
    }  
}
```

```
        System.out.print("Suma liczb nieparzystych w przedziale od 1 do  
        ↪100 wynosi ");  
        System.out.print(suma + ".");  
    }  
}
```

---

**ZADANIE****3.15**

Napisz program, który za pomocą instrukcji `while` sumuje liczby nieparzyste w przedziale od 1 do 100.

**Wskazówka**

Należy skorzystać z właściwości operatora modulo `%` i operatora negacji `!`.

*Przykładowe rozwiązanie — listing 3.15*

---

```
package zadanie315; // Zadanie 3.15  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int i = 1, suma = 0;  
  
        System.out.print("Program sumuje liczby nieparzyste ");  
        System.out.println("z przedziału od 1 do 100.");  
  
        while (i <= 100)  
        {  
            if (!(i%2 == 0)) suma = suma+i;  
            i++;  
        }  
        System.out.print("Suma liczb nieparzystych w przedziale od 1 do  
        ↪100 wynosi " );  
        System.out.print(suma + ".");  
    }  
}
```

---

**ZADANIE****3.16**

Napisz program, który za pomocą instrukcji `for` znajduje największą i najmniejszą liczbę ze zbioru `n` wylosowanych liczb całkowitych od 0 do 99 (w zadaniu `n = 5`) oraz oblicza średnią ze wszystkich wylosowanych liczb.

**Wskazówka**

Przetestuj program dla  $n = 2$ .

*Przykładowe rozwiązanie — listing 3.16*

```
package zadanie316; //Zadanie 3.16
import java.util.*;

public class Main {

    public static void main(String[] args)
    {
        int ilosc_liczb = 5, i;
        double liczba, suma = 0, min, max;

        System.out.println("Program losuje " + ilosc_liczb + " liczb
        ↳całkowitych od 0 do 99,");
        System.out.println("a następnie znajduje najmniejszą i
        ↳największą");
        System.out.println("oraz oblicza średnia ze wszystkich
        ↳wylosowanych liczb.");

        Random r = new Random();
        min = Math.round(100*(r.nextDouble()));
        System.out.println();
        System.out.print("Wylosowano liczby: " + min + ", ");
        max = min;
        suma = suma+max;

        for (i = 1; i <= ilosc_liczb - 1; i++)
        {
            liczba = Math.round(100*(r.nextDouble()));
            System.out.print(liczba + ", ");

            if (max < liczba) max = liczba;
            if (liczba < min) min = liczba;

            suma = suma+liczba;
        }

        System.out.println();
        System.out.println("największa liczba to " + max + ",");
        System.out.println("najmniejsza liczba to " + min + ",");
        System.out.println("średnia = " + suma/ilosc_liczb + ".");
    }
}
```

W programie tym najpierw losujemy liczbę i przypisujemy jej wartość min:

```
min = Math.round(100*(r.nextDouble()));
```

W kolejnym kroku wartości max nadajemy wartość min:

```
max = min;
```

Następnie w pętli pomniejszonej o 1 for (i = 1; i <= ilosc\_liczb - 1; i++) sprawdzamy, czy następna wylosowana liczba jest większa od poprzedniej. Jeśli tak, to staje się ona największą liczbą (max); w przeciwnym wypadku przypisujemy jej wartość min. Ilustrują to następujące linijki kodu:

```
if (max < liczba) max = liczba;  
if (liczba < min) min = liczba;
```

Sumę wszystkich wylosowanych liczb wyliczają następujące linijki kodu: przed pętlą suma = suma+max i w pętli suma = suma+liczba, natomiast ich średnia jest obliczana i wyświetlana na ekranie przez poniższy wiersz:

```
System.out.println("średnia = " + suma/ilosc_liczb + ".");
```

Rezultat działania programu można zobaczyć na rysunku 3.6.

**Program losuje 5 liczb całkowitych od 0 do 99,  
a następnie znajduje najmniejszą i największą  
oraz oblicza średnią ze wszystkich wylosowanych liczb.**

**Wylosowano liczby: 28.0, 45.0, 12.0, 34.0, 37.0,  
największa liczba to 45.0,  
najmniejsza liczba to 12.0,  
średnia = 31.2.**

*Rysunek 3.6. Efekt działania programu Zadanie 3.16*

---

## ZADANIE

### 3.17

Napisz program, który za pomocą instrukcji do ... while znajduje największą i najmniejszą liczbę ze zbioru n wylosowanych liczb całkowitych od 0 do 99 (w zadaniu n = 5) oraz oblicza średnią ze wszystkich wylosowanych liczb.

**Wskazówka**

Przetestuj program dla  $n = 2$ .

*Przykładowe rozwiązanie — listing 3.17*

```
package zadanie317; // Zadanie 3.17
import java.util.*;

public class Main {

    public static void main(String[] args)
    {
        int ilosc_liczb = 5, i = 1;
        double liczba, suma = 0, min, max;

        System.out.println("Program losuje " + ilosc_liczb + " liczb
        ↳całkowitych od 0 do 99,");
        System.out.println("a następnie znajduje najmniejszą i największą");
        System.out.println("oraz oblicza średnia ze wszystkich
        ↳wylosowanych liczb.");

        Random r = new Random();
        min = Math.round(100*(r.nextDouble()));
        System.out.println();
        System.out.print("Wylosowano liczby: " + min + ", ");
        max = min;
        suma = suma+max;

        do
        {
            liczba = Math.round(100*(r.nextDouble()));
            System.out.print(liczba + ", ");

            if (max < liczba) max = liczba;
            if (liczba < min) min = liczba;

            suma = suma+liczba;
            i++;
        }
        while (i <= ilosc_liczb - 1);

        System.out.println();
        System.out.println("największa liczba to " + max + ",");
        System.out.println("najmniejsza liczba to " + min + ",");
        System.out.println("średnia = " + suma/ilosc_liczb + ".");
    }
}
```

## ZADANIE

**3.18**

Napisz program, który za pomocą instrukcji `while` znajduje największą i najmniejszą liczbę ze zbioru `n` wylosowanych liczb całkowitych od 0 do 99 (w zadaniu `n = 5`) oraz oblicza średnią ze wszystkich wylosowanych liczb.

**Wskazówka**

Przetestuj program dla `n = 2`.

---

*Przykładowe rozwiązanie — listing 3.18*

---

```
package zadanie318; // Zadanie 3.18
import java.util.*;

public class Main {

    public static void main(String[] args)
    {
        int ilosc_liczb = 5, i = 1;
        double liczba, suma = 0, min, max;

        System.out.println("Program losuje " + ilosc_liczb + " liczb
        ↳całkowitych od 0 do 99.");
        System.out.println("a następnie znajduje najmniejszą i
        ↳największą");
        System.out.println("oraz oblicza średnią ze wszystkich
        ↳wylosowanych liczb.");

        Random r = new Random();
        min = Math.round(100*(r.nextDouble()));
        System.out.println();
        System.out.print("Wylosowano liczby: " + min + ", ");
        max = min;
        suma = suma+max;

        while (i <= ilosc_liczb - 1)
        {
            liczba = Math.round(100*(r.nextDouble()));
            System.out.print(liczba+", ");

            if (max < liczba) max = liczba;
            if (liczba < min) min = liczba;
```



```
        suma = suma+liczba;
        i++;
    }
    System.out.println();
    System.out.println("największa liczba to " + max + ",");
    System.out.println("najmniejsza liczba to " + min + ",");
    System.out.println("średnia = " + suma/ilosc_liczb + ".");
}
}
```

---

**ZADANIE****3.19**

Napisz program wyświetlający tabliczkę mnożenia dla liczb od 1 do 100 z wykorzystaniem podwójnej pętli for.

---

*Przykładowe rozwiązanie — listing 3.19*

---

```
package zadanie319; //Zadanie 3.19

public class Main {

    public static void main(String[] args)
    {
        int n = 10, wiersze, kolumny;

        System.out.println("Program wyświetla tabliczkę mnożenia dla
        ↪ liczb od 1 do 100." );

        for (wiersze = 1; wiersze <= n; wiersze++)
        {
            for (kolumny = 1; kolumny <= n; kolumny++)
            {
                System.out.print(wiersze*kolumny + "\t");
            }
            System.out.println();
        }
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 3.7.

**Program wyświetla tabliczkę mnożenia dla liczb od 1 do 100.**

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

*Rysunek 3.7. Efekt działania programu Zadanie 3.19*

---

#### ZADANIE

### 3.20

Napisz program wyświetlający tabliczkę mnożenia dla liczb od 1 do 100 z wykorzystaniem podwójnej pętli do ... while.

*Przykładowe rozwiązanie — listing 3.20*

---

```
package zadanie320; //Zadanie 3.20

public class Main {

    public static void main(String[] args)
    {
        int n = 10, wiersze, kolumny;

        System.out.println("Program wyświetla tabliczkę mnożenia dla
        ↪liczb od 1 do 100." );

        wiersze = 1; //wartość początkowa

        do
        {
            kolumny = 1; //wartość początkowa
            do
            {
                System.out.print(wiersze*kolumny + "\t");
                kolumny++;
            }
            while (kolumny <= n);
            wiersze++;
            System.out.println();
        }
```

```
    }  
    while (wiersze <= n);  
  }  
}
```

**ZADANIE****3.21**

Napisz program wyświetlający tabliczkę mnożenia dla liczb od 1 do 100 z wykorzystaniem podwójnej pętli `while`.

*Przykładowe rozwiązanie — listing 3.21*

```
package zadanie321; // Zadanie 3.21  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int n = 10, wiersze, kolumny;  
  
        System.out.println("Program wyświetla tabliczkę mnożenia dla  
        ↪ liczb od 1 do 100." );  
  
        wiersze = 1; // wartość początkowa  
  
        while (wiersze <= n)  
        {  
            kolumny = 1; // wartość początkowa  
  
            while (kolumny <= n)  
            {  
                System.out.print(wiersze*kolumny + "\t");  
                kolumny++;  
            }  
            wiersze++;  
            System.out.println();  
        }  
    }  
}
```

**ZADANIE****3.22**

Napisz program, który wyświetla duże litery alfabetu od A do Z i od Z do A z wykorzystaniem pętli for.

*Przykładowe rozwiązanie — listing 3.22*

---

```
package zadanie322; //Zadanie 3.22

public class Main {

    public static void main(String[] args)
    {
        char znak;

        System.out.println("Program wyświetla duże litery alfabetu
        ↪od A do Z i od Z do A.");

        for (znak = 'A'; znak <= 'Z'; znak++)
        {
            if (znak < 'Z')
                System.out.print(znak + ", ");
            else
                System.out.print(znak + ".");
        }

        System.out.println();

        for (znak = 'Z'; znak >= 'A'; znak--)
        {
            if (znak > 'A')
                System.out.print(znak + ", ");
            else
                System.out.print(znak + ".");
        }
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 3.8.

**Program wyświetla duże litery alfabetu od A do Z i od Z do A.**  
**A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W,**  
**X, Y, Z.**  
**Z, Y, X, W, V, U, T, S, R, Q, P, O, N, M, L, K, J, I, H, G, F, E, D,**  
**C, B, A.**

*Rysunek 3.8. Efekt działania programu Zadanie 3.22*

## ZADANIE

**3.23**

Napisz program, który wyświetla duże litery alfabetu od A do Z i od Z do A z wykorzystaniem pętli `do ... while`.

*Przykładowe rozwiązanie — listing 3.23*

```
package zadanie323; //Zadanie 3.23

public class Main {

    public static void main(String[] args)
    {
        char znak;

        System.out.println("Program wyświetla duże litery alfabetu
        ↳ od A do Z i od Z do A.");

        znak = 'A';
        do
        {
            if (znak < 'Z')
                System.out.print(znak + ", ");
            else
                System.out.print(znak + ".");
            znak++;
        }
        while (znak <= 'Z');

        System.out.println();

        znak = 'Z';
        do
        {
            if (znak > 'A')
                System.out.print(znak + ", ");
            else
                System.out.print(znak + ".");
            znak--;
        }
        while (znak >= 'A');
    }
}
```

**ZADANIE****3.24**

Napisz program, który wyświetla duże litery alfabetu od A do Z i od Z do A z wykorzystaniem pętli `while`.

*Przykładowe rozwiązanie — listing 3.24*

---

```
package zadanie324; //Zadanie 3.24

public class Main {

    public static void main(String[] args)
    {
        char znak;

        System.out.println("Program wyświetla duże litery alfabetu od A do Z  

        ↪i od Z do A.");

        znak = 'A';
        while (znak <= 'Z')
        {
            if (znak < 'Z')
                System.out.print(znak + ", ");
            else
                System.out.print(znak + ".");
            znak++;
        }

        System.out.println();

        znak = 'Z';
        while (znak >= 'A')
        {
            if (znak > 'A')
                System.out.print(znak + ", ");
            else
                System.out.print(znak + ".");
            znak--;
        }
    }
}
```

---

# 4

## Tablice

*W tym rozdziale przedstawimy typowe zadania, wraz z przykładowymi rozwiązaniami, z wykorzystaniem tablic jedno- i dwuwymiarowych.*

### Deklarowanie tablic jednowymiarowych

**Tablice** są zbiorami zmiennych tego samego typu (np. całkowitego, rzeczywistego itd.), do których odwołujemy się poprzez jedną wspólną nazwę. Są one bardzo praktyczne, gdyż oferują wygodny sposób łączenia powiązanych ze sobą zmiennych. Ich główną zaletą jest to, że łatwo można na nich przeprowadzać różne manipulacje, np. sortowanie.

Choć tablice w języku Java mogą być używane tak jak w innych językach, mają one jeden specjalny atrybut: są zaimplementowane jako obiekty<sup>1</sup>.

---

<sup>1</sup> Więcej informacji o obiektach znajdzie czytelnik w rozdziale 6.

W deklaracji tablicy musimy określić typ wartości, jaki ma ona przechowywać, a także liczbę jej elementów. Tablice mogą być jednowymiarowe, dwuwymiarowe itd. Oto przykład zadeklarowania tablicy jednowymiarowej i związanej z nią zmiennej:

```
typ_tablicy nazwa_tablicy[] = new typ_tablicy[rozmiar_tablicy];
```

gdzie *typ\_tablicy* określa jej podstawowy typ, który jednocześnie definiuje typ każdej jej komórki. Liczba elementów, które będzie zawierać tablica, jest określona przez *rozmiar\_tablicy*. A oto przykład zadeklarowania tablicy jednowymiarowej o nazwie *dane* typu całkowitego, zawierającej 10 elementów:

```
int dane[] = new int[10]; // deklaracja tablicy typu int
```

Powyższa deklaracja działa tak jak deklaracja obiektu. Zmienna *dane[]* zawiera referencję do pamięci przydzielonej przez operator *new*. Pamięć ta jest wystarczająca, aby pomieścić 10 elementów typu *int*.

## Dostęp do elementów tablicy

Dostęp do konkretnej wartości w tablicy jest realizowany za pośrednictwem indeksu, który wskazuje dany element. Dla deklaracji

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

aby uzyskać dostęp do pierwszego elementu tablicy *dane*, powinniśmy podać indeks 0, drugi element dostępny jest poprzez indeks 1 itd. Ostatni element tablicy ma indeks równy jej wymiarowi pomniejszonemu o 1, czyli 9, co przedstawiono w reprezentacji graficznej poniżej.

0	1	2	3	4	5	6	7	8	9

Oto prosty przykład ilustrujący posługiwanie się tablicą jednowymiarową:

---

### ZADANIE

#### 4.1

Napisz program, który w 10-elementowej tablicy jednowymiarowej o nazwie *dane* umieszcza liczby od 0 do 9 (zobacz poniżej reprezentację graficzną tablicy).



Indeks tablicy	Wartość tablicy
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

---

*Przykładowe rozwiązanie — listing 4.1*

---

```
package zadanie41; // Zadanie 4.1

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i;
        int dane[] = new int [n]; // deklaracja tablicy typu int

        for (i = 0; i < 10; i++)
        {
            dane[i] = i;
            System.out.println("dane[" + i + "] = " + dane[i]);
            // wyświetlenie zawartości tablicy
        }
    }
}
```

---

Linijka kodu:

```
System.out.println("dane[" + i + "] = " + dane[i]);
//wyświetlenie zawartości tablicy
```

powoduje wyświetlenie zawartości tablicy dane.

Rezultat działania programu można zobaczyć na rysunku 4.1.

```
dane[0] = 0
dane[1] = 1
dane[2] = 2
dane[3] = 3
dane[4] = 4
dane[5] = 5
dane[6] = 6
dane[7] = 7
dane[8] = 8
dane[9] = 9
```

*Rysunek 4.1. Efekt działania programu Zadanie 4.1*

---

#### ZADANIE

### 4.2

Napisz program, który w 10-elementowej tablicy jednowymiarowej o nazwie `dane` umieszcza liczby od 9 do 0 (zobacz poniżej reprezentację graficzną tablicy).

Indeks tablicy	Wartość tablicy
0	9
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1
9	0

**Wskazówka**

Zadanie to rozwiążemy poprawnie, zamieniając w zadaniu 4.1 linijkę kodu

```
dane[i] = i;
```

na następującą:

```
dane[i] = n-1-i;
```

---

**Przykładowe rozwiązanie — listing 4.2**

---

```
package zadanie42; //Zadanie 4.2

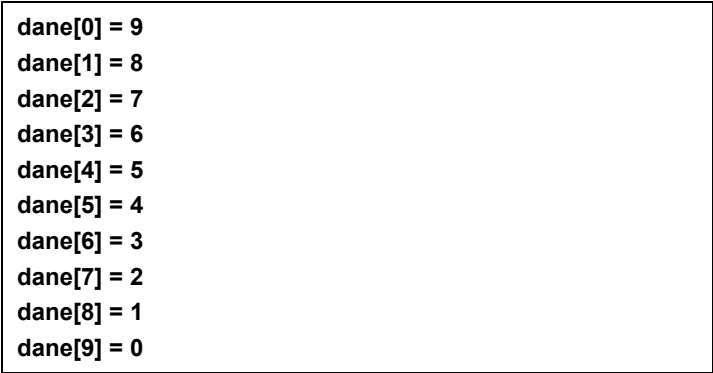
public class Main {

    public static void main(String[] args)
    {
        int n = 10, i;
        int dane[] = new int [n]; //deklaracja tablicy typu int

        for (i = 0; i < 10; i++)
        {
            dane[i] = n-1-i;
            System.out.println("dane[" + i + "] = " + dane[i]);
            //wyświetlenie zawartości tablicy
        }
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 4.2.



```
dane[0] = 9
dane[1] = 8
dane[2] = 7
dane[3] = 6
dane[4] = 5
dane[5] = 4
dane[6] = 3
dane[7] = 2
dane[8] = 1
dane[9] = 0
```

**Rysunek 4.2.** Efekt działania programu Zadanie 4.2

# Tablice dwuwymiarowe

Tablice dwuwymiarowe w języku Java deklarujemy podobnie jak jednowymiarowe. Oto przykład takiej deklaracji:

```
typ_tablicy nazwa_tablicy[][] = new
typ_tablicy[rozmiar_tablicy][rozmiar_tablicy];
```

Dostęp do elementów tablicy jest realizowany za pośrednictwem dwóch indeksów, które wskazują dany element.

Przykład poniżej ilustruje deklarację tablicy dwuwymiarowej  $10 \times 10$  typu całkowitego.

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

Tablicę dwuwymiarową o nazwie `macierz`, jako produkt o wymiarach  $10 \times 10$ , możemy sobie wyobrazić np. następująco:

1	2	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	7

Wartości liczbowe możemy wpisywać do niej wierszami lub kolumnami. Pierwszy element tablicy `macierz[0][0] = 1`, element `macierz[0][1] = 2`, element `macierz[1][0] = 3` itd. Ostatni element ma indeks równy wymiarowi tablicy minus 1, czyli 9 — w naszym przypadku `macierz[9][9] = 7`.

## ZADANIE

**4.3**

Napisz program, który w zadeklarowanej tablicy dwuwymiarowej  $10 \times 10$  o nazwie `macierz` umieszcza na przekątnej liczbę 1, a poza przekątną 0. Dodatkowo program powinien obliczać sumę elementów wyróżnionych w tablicy, tj. tych znajdujących się na jej przekątnej.

1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1

*Przykładowe rozwiązanie — listing 4.3*

```
package zadanie43; //Zadanie 4.3

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i, j, suma;
        int macierz[][] = new int[n][n];

        //wpisywanie do tablicy 1 na przekątnej, a 0 poza przekątną
        for (i = 0; i < n; i++)
        {
            for(j = 0; j < n; j++)
            {
                if (i == j)
                    macierz[i][j] = 1;
                else
                    macierz[i][j] = 0;
            }
        }

        //wyświetlenie zawartości tablicy
        for (i = 0; i < n; i++)
```

```
        {
            for(j = 0; j < n; j++)
            {
                System.out.print(macierz[i][j] + " ");
            }
            System.out.println();
        }

        suma = 0;

        for (i = 0; i < n; i++)
        {
            suma = suma+macierz[i][i];
        }
        System.out.println("Suma wyróżnionych elementów w tablicy wynosi
        ↳" + suma + ".");
    }
}
```

---

Do wpisywania danych do tablicy o nazwie `macierz` użyliśmy dwóch pętli `for`. Następujące linijki kodu instrukcji warunkowej `if`:

```
if (i == j)
    macierz[i][j] = 1;
else
    macierz[i][j] = 0;
```

powodują wpisywanie liczby 1 na przekątnej, a 0 poza przekątną.

Całość kodu odpowiedzialnego za umieszczanie na przekątnej tablicy liczby 1, a poza nią liczby 0 znajduje się poniżej.

```
// wpisywanie do tablicy 1 na przekątnej, a 0 poza przekątną
for (i = 0; i < n; i++)
{
    for(j = 0; j < n; j++)
    {
        if (i == j)
            macierz[i][j] = 1;
        else
            macierz[i][j] = 0;
    }
}
```

Za wyświetlenie zawartości tablicy na ekranie komputera odpowiadają następujące linijki kodu:

```
for (i = 0; i < n; i++)
{
    for(j = 0; j < n; j++)
```

```
    {  
        System.out.print(macierz[i][j] + " ");  
    }  
    System.out.println();  
}
```

Obliczanie sumy elementów znajdujących się na przekątnej tablicy należy do następujących linii kodu:

```
for (i = 0; i < n; i++)  
{  
    suma = suma+macierz[i][i];  
}
```

Oczywiście zmienną `suma` trzeba wcześniej wyzerować:

```
suma = 0;
```

Rezultat działania programu można zobaczyć na rysunku 4.3.

```
1 0 0 0 0 0 0 0 0 0  
0 1 0 0 0 0 0 0 0 0  
0 0 1 0 0 0 0 0 0 0  
0 0 0 1 0 0 0 0 0 0  
0 0 0 0 1 0 0 0 0 0  
0 0 0 0 0 1 0 0 0 0  
0 0 0 0 0 0 1 0 0 0  
0 0 0 0 0 0 0 1 0 0  
0 0 0 0 0 0 0 0 1 0  
0 0 0 0 0 0 0 0 0 1
```

**Suma wyróżnionych elementów w tablicy wynosi 10.**

*Rysunek 4.3. Efekt działania programu Zadanie 4.3*

#### ZADANIE

#### 4.4

Napisz program, który w zadeklarowanej tablicy dwuwymiarowej  $10 \times 10$  o nazwie `macierz` umieszcza na przekątnej liczby od 0 do 9, a poza przekątną liczbę 0. Dodatkowo program powinien obliczać sumę elementów wyróżnionych w tablicy (znajdujących się na przekątnej).

0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0	0
0	0	0	0	4	0	0	0	0	0
0	0	0	0	0	5	0	0	0	0
0	0	0	0	0	0	6	0	0	0
0	0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	0	0	8	0
0	0	0	0	0	0	0	0	0	9

**Wskazówka**

Zadanie to rozwiążemy poprawnie, zamieniając w zadaniu 4.3 linijkę kodu

```
macierz[i][j] = 1;
```

na następującą:

```
macierz[i][j] = i;
```

---

**Przykładowe rozwiązanie — listing 4.4**

---

```
package zadanie44; //Zadanie 4.4

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i, j, suma;
        int macierz[][] = new int[n][n];

        //wpisywanie do tablicy liczb od 0 do 9 na przekątnej, a 0 poza przekątną
        for (i = 0; i < n; i++)
        {
            for(j = 0; j < n; j++)
            {
                if (i == j)
                    macierz[i][j] = i;
                else
                    macierz[i][j] = 0;
            }
        }

        //wyświetlenie zawartości tablicy
        for (i = 0; i < n; i++)
```



```
{
    for(j = 0; j < n; j++)
    {
        System.out.print(macierz[i][j] + " ");
    }
    System.out.println();
}

suma = 0;

for (i = 0; i < n; i++)
{
    suma = suma+macierz[i][i];
}

System.out.println("Suma wyróżnionych elementów w tablicy
↳wynosi " + suma + ".");
}
```

Rezultat działania programu można zobaczyć na rysunku 4.4.

0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 2 0 0 0 0 0 0 0
0 0 0 3 0 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0
0 0 0 0 0 5 0 0 0 0
0 0 0 0 0 0 6 0 0 0
0 0 0 0 0 0 0 7 0 0
0 0 0 0 0 0 0 0 8 0
0 0 0 0 0 0 0 0 0 9
<b>Suma wyróżnionych elementów w tablicy wynosi 45.</b>

*Rysunek 4.4. Efekt działania programu Zadanie 4.4*

#### ZADANIE

### 4.5

Napisz program, który w zadeklarowanej tablicy dwuwymiarowej  $10 \times 10$  o nazwie `macierz` umieszcza liczby 1 i 0 zgodnie z zamieszczoną poniżej interpretacją graficzną. Program dodatkowo powinien obliczać sumę wyróżnionych elementów.

0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

**Wskazówka**

Zadanie to rozwiążemy poprawnie, zamieniając w zadaniu 4.3 linijki kodu

```
if (i == j)
    macierz[i][j] = 1;
else
    macierz[i][j] = 0;
```

na następujące:

```
if (n == i+j+1)
    macierz[i][j] = 1;
else
    macierz[i][j] = 0;
```

Sumę wyróżnionych elementów znajdujących się w tablicy obliczymy, zastępując linijki kodu z zadania 4.3:

```
for (i = 0; i < n; i++)
{
    suma = suma+macierz[i][i];
}
```

następującymi:

```
for (i = 0; i < n; i++)
{
    suma = suma+macierz[i][n-i-1];
}
```

---

*Przykładowe rozwiązanie — listing 4.5*

---

```
package zadanie45; // Zadanie 4.5

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i, j, suma;
        int macierz[][] = new int[n][n];

        for (i = 0; i < n; i++)
        {
            for(j = 0; j < n; j++)
            {
                if (n == i+j+1)
                    macierz[i][j] = 1;
                else
                    macierz[i][j] = 0;
            }
        }

        // wyświetlenie zawartości tablicy
        for (i = 0; i < n; i++)
        {
            for(j = 0; j < n; j++)
            {
                System.out.print(macierz[i][j] + " ");
            }
            System.out.println();
        }

        suma = 0;

        for (i = 0; i < n; i++)
        {
            suma = suma+macierz[i][n-i-1];
        }

        System.out.println("Suma wyróżnionych elementów w tablicy
        ↪wynosi " + suma + ".");
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 4.5.

```

0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0

```

**Suma wyróżnionych elementów w tablicy wynosi 10.**

*Rysunek 4.5. Efekt działania programu Zadanie 4.5*

#### ZADANIE

### 4.6

Napisz program, który w zadeklarowanej tablicy dwuwymiarowej  $10 \times 10$  o nazwie `macierz` umieszcza liczby od 0 do 9 zgodnie z załączoną poniżej interpretacją graficzną. Program dodatkowo powinien obliczać sumę wyróżnionych elementów.

0	0	0	0	0	0	0	0	0	<b>0</b>
0	0	0	0	0	0	0	0	<b>1</b>	0
0	0	0	0	0	0	0	<b>2</b>	0	0
0	0	0	0	0	0	<b>3</b>	0	0	0
0	0	0	0	0	<b>4</b>	0	0	0	0
0	0	0	0	<b>5</b>	0	0	0	0	0
0	0	0	<b>6</b>	0	0	0	0	0	0
0	0	<b>7</b>	0	0	0	0	0	0	0
0	<b>8</b>	0	0	0	0	0	0	0	0
<b>9</b>	0	0	0	0	0	0	0	0	0

#### Wskazówka

Zadanie to rozwiążemy poprawnie, zamieniając w zadaniu 4.5 linijki kodu

```

if (n == i+j+1)
    macierz[i][j] = 1;
else
    macierz[i][j] = 0;

```

na następujące:

```
if (n == i+j+1)
    macierz[i][j] = i;
else
    macierz[i][j] = 0;
```

---

*Przykładowe rozwiązanie — listing 4.6*

---

```
package zadanie46; // Zadanie 4.6

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i, j, suma;
        int macierz[][] = new int[n][n];

        for (i = 0; i < n; i++)
        {
            for(j = 0; j < n; j++)
            {
                if (n == i+j+1)
                    macierz[i][j] = i;
                else
                    macierz[i][j] = 0;
            }
        }

        // wyświetlenie zawartości tablicy
        for (i = 0; i < n; i++)
        {
            for(j = 0; j < n; j++)
            {
                System.out.print(macierz[i][j] + " ");
            }
            System.out.println();
        }

        suma = 0;

        for (i = 0; i < n; i++)
        {
            suma = suma+macierz[i][n-i-1];
        }

        System.out.println("Suma wyróżnionych elementów w tablicy
        ↳wynosi " + suma + ".");
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 4.6.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	2	0	0	
0	0	0	0	0	3	0	0	0	
0	0	0	0	4	0	0	0	0	
0	0	0	5	0	0	0	0	0	
0	0	6	0	0	0	0	0	0	
0	7	0	0	0	0	0	0	0	
0	8	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	
Suma wyróżnionych elementów w tablicy wynosi 45.									

*Rysunek 4.6. Efekt działania programu Zadanie 4.6*

#### ZADANIE

### 4.7

Napisz program, który w zadeklarowanej tablicy dwuwymiarowej  $10 \times 10$  umieszcza w pierwszej kolumnie liczby od 0 do 9, w drugiej kwadraty tych liczb, natomiast w pozostałych kolumnach 0 (interpretacja graficzna tablicy poniżej). Dodatkowo program powinien obliczać osobno sumę liczb znajdujących się w pierwszej oraz w drugiej kolumnie.

0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0
3	9	0	0	0	0	0	0	0	0
4	16	0	0	0	0	0	0	0	0
5	25	0	0	0	0	0	0	0	0
6	36	0	0	0	0	0	0	0	0
7	49	0	0	0	0	0	0	0	0
8	64	0	0	0	0	0	0	0	0
9	81	0	0	0	0	0	0	0	0

---

*Przykładowe rozwiązanie — listing 4.7*

---

```
package zadanie47; //Zadanie 4.7

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i, j, suma;
        int tablica[][] = new int[n][n];

        //wpisywanie liczb do tablicy
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                if (j == 0) tablica[i][j] = i;
                if (j == 1) tablica[i][j] = i*i;
                if (j > 1) tablica[i][j] = 0;
            }
        }

        //wyświetlenie zawartości tablicy
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                System.out.print(tablica[i][j] + " ");
            }
            System.out.println();
        }

        //obliczanie sumy
        suma = 0;

        for (i = 0; i < n; i++)
        {
            suma = suma+tablica[i][0];
        }
        System.out.println("Suma liczb znajdujących się w pierwszej
        ↪kolumnie to " + suma + ".");

        suma = 0;

        for (i = 0; i < n; i++)
        {
            suma = suma+tablica[i][1];
        }
        System.out.println("Suma liczb znajdujących się w drugiej
        ↪kolumnie to " + suma + ".");
    }
}
```

---

Następujące linijki kodu:

```
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        if (j == 0) tablica[i][j] = i;
        if (j == 1) tablica[i][j] = i*i;
        if (j > 1) tablica[i][j] = 0;
    }
}
```

---

są odpowiedzialne za wpisywanie liczb do tablicy. Do pierwszej kolumny wpisywane są liczby od 0 do 9:

```
if (j == 0) tablica[i][j] = i;
```

a do drugiej kwadraty tych liczb:

```
if (j == 1) tablica[i][j] = i*i;
```

Do pozostałych kolumn wprowadzana jest liczba 0:

```
if (j > 1) tablica[i][j] = 0;
```

Za sumowanie liczb znajdujących się w pierwszej kolumnie odpowiadają następujące linijki kodu:

```
for (i = 0; i < n; i++)
{
    suma = suma+tablica[i][0];
}
```

Sumowanie liczb z drugiej kolumny odbywa się w następujących linijkach kodu:

```
for (i = 0; i < n; i++)
{
    suma = suma+tablica[i][1];
}
```

Oczywiście zmienna `suma` musi zostać wcześniej wyzerowana:

```
suma = 0;
```

Rezultat działania programu można zobaczyć na rysunku 4.7.



```

0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
2 4 0 0 0 0 0 0 0 0
3 9 0 0 0 0 0 0 0 0
4 16 0 0 0 0 0 0 0 0
5 25 0 0 0 0 0 0 0 0
6 36 0 0 0 0 0 0 0 0
7 49 0 0 0 0 0 0 0 0
8 64 0 0 0 0 0 0 0 0
9 81 0 0 0 0 0 0 0 0

```

Suma liczb znajdujących się w pierwszej kolumnie to 45.  
Suma liczb znajdujących się w drugiej kolumnie to 285.

*Rysunek 4.7. Efekt działania programu Zadanie 4.7*

#### ZADANIE

### 4.8

Dane są dwie tablice dwuwymiarowe  $10 \times 10$  o nazwach a i b. Tablica a zawiera elementy przedstawione poniżej.

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

Tablica b zawiera same zera. Napisz program, który przepisuje zawartość tablicy a do tablicy b (interpretacja graficzna tablicy wynikowej poniżej), zamieniając kolumny na wiersze.

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

---

*Przykładowe rozwiązanie — listing 4.8*

```
package zadanie48; //Zadanie 4.8

public class Main {

    public static void main(String[] args)
    {
        int n = 10, i, j;
        int a[][] = new int[n][n];
        int b[][] = new int[n][n];

        //wpisywanie liczb do tablicy a
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                a[i][j] = j;
            }
        }

        //przepisywanie liczb z tablicy a do tablicy b
        for (i = 0; i < n; i++)
        {
            for (j = 0; j < n; j++)
            {
                b[i][j] = a[j][i]; //zamiana kolumn na wiersze
            }
        }

        //wyświetlenie zawartości tablicy a
        System.out.println("Wyświetlenie zawartości tablicy a:");
        System.out.println();
    }
}
```

```
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        System.out.print(a[i][j] + " ");
    }
    System.out.println();
}

//wyświetlenie zawartości tablicy b
System.out.println();
System.out.println("Wyświetlenie zawartości tablicy b:");
System.out.println();

for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        System.out.print(b[i][j] + " ");
    }
    System.out.println();
}
}
```

---

Następująca linijka kodu:

```
b[i][j] = a[j][i]; //zamiana kolumn na wiersze
```

przepisuje liczby z tablicy a do b i jest odpowiedzialna za zamianę kolumn na wiersze.

Rezultat działania programu można zobaczyć na rysunku 4.8.

**Wyświetlenie zawartości tablicy a:**

```
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
```

**Wyświetlenie zawartości tablicy b:**

```
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
```

*Rysunek 4.8. Efekt działania programu Zadanie 4.8*

# 5

## Programowanie obiektowe

*W tym rozdziale przedstawimy typowe zadania, wraz z przykładowymi rozwiązaniami, z wykorzystaniem reguł programowania obiektowego.*

### Programowanie obiektowe — informacje ogólne

Java jest obiektowym językiem programowania. **Programowanie obiektowe** (ang. *object-oriented programming*) to taki paradygmat programowania, w którym programy definiuje się za pomocą obiektów — elementów łączących *stan* (są to dane nazywane **polami**) i *zachowanie* (są to **metody** — w Javie są nimi funkcje służące do wykonywania na tych danych określonych zadań). Obiektowy program komputerowy to zbiór takich obiektów komunikujących się pomiędzy sobą w celu wykonywania zadań. Każdy program w Javie składa się ze zbioru klas. Aby program napisany w tym języku można było uruchomić, musi on posiadać publiczną i statyczną metodę `main()` w postaci:

```
public static void main(String[] args)
{
    // instrukcje do wykonania
}
```

oraz przynajmniej jedną publiczną klasę:

```
public class Main
{
    public static void main(String[] args)
    {
        //instrukcje do wykonania
    }
}
```

W języku Java podstawowym pojęciem programowania obiektowego jest **klasa** (ang. *class*), która definiuje projekt i strukturę obiektu. Schematyczny szkielek klasy ma tu następującą postać:

```
class nazwa_klasy
{
    //pola
    //metody
}
```

W klasach możemy wyróżnić m.in. następujące elementy: stałe, zmienne składowe, zmienne statyczne, metody (funkcje) i konstruktory.

Java posiada następujące cztery poziomy dostępu (modyfikatory) do pól i metod (składowych klasy): publiczny, prywatny, chroniony i pakietowy.

Jeśli przed składową klasy nie występuje żadne określenie, oznacza to, że dostęp jest pakietowy, czyli do tej składowej mają dostęp wszystkie klasy pakietu, w którym się ona znajduje.

Jeżeli dana składowa klasy jest **publiczna** (ang. *public*), to mają do niej dostęp wszystkie inne klasy i nie istnieją żadne ograniczenia w dostępności. Dla pól klasy specyfikator dostępu `public` należy umieścić przed nazwą typu według schematu

```
public nazwa_typu nazwa_pola;
```

natomiast dla metod ogólny schemat jest następujący:

```
public typ_zwracany nazwa_metody(argumenty)
```

Gdy składowa klasy jest **prywatna** (ang. *private*), dostęp do niej jest możliwy tylko w obrębie tej klasy. Oznacza to, że wszystkie metody danej klasy mogą ją dowolnie odczytywać i zapisywać, natomiast nie może tego robić żadna inna klasa. Dla pól klasy specyfikator dostępu `private` należy umieścić przed nazwą typu według wzoru

```
private nazwa_typu nazwa_pola;
```

natomiast dla metod ogólny schemat jest następujący:

```
private typ_zwracany nazwa_metody(argumenty)
```

Jeżeli składowa klasy jest **chroniona** (ang. *protected*), oznacza to, że jest ona dostępna jedynie dla metod danej klasy, klas potomnych oraz klas z tego samego pakietu. Dla pól klasy specyfikator dostępu *protected* należy umieścić przed nazwą typu według schematu

```
protected nazwa_tytu nazwa_pola;
```

natomiast dla metod wygląda on następująco:

```
protected typ_zwracany nazwa_metody(argumenty)
```

Klasy w języku Java pogrupowane są w jednostki zwane **pakietami** (ang. *package*). Tworzą one rodzaj bibliotek, czyli tematycznie powiązanych ze sobą klas. Aby utworzyć pakiet, musimy użyć słowa kluczowego *package*, po którym następuje nazwa pakietu zakończona średnikiem, według następującego schematu:

```
package nazwa_pakietu;
```

Instrukcja ta musi znajdować się na początku pliku.

```
package nazwa_pakietu;
.....
class nazwa_klasy
{
    //pola
    //metody
}
```

Jeżeli w jednej klasie chcemy skorzystać z innej klasy znajdującej się w pakiecie, musimy użyć dyrektywy *import* według wzoru

```
import nazwa_pakietu.nazwa_klasy;
```

Dyrektywa ta musi znaleźć się na początku pliku. Do zaimportowania wszystkich klas z danego pakietu stosujemy natomiast schemat

```
import nazwa_pakietu.*;
```

Aby utworzyć zmienną typu obiektowego (klasowego, referencyjnego), należy skorzystać z następującej konstrukcji:

```
nazwa_klasy nazwa_zmiennej;
```

Do tak zadeklarowanej zmiennej można następnie przypisać obiekt, który stworzymy za pomocą operatora *new*:

```
new nazwa_klasy();
```

### Natomiast konstrukcja

```
nazwa_klasy nazwa_zmiennej = new nazwa_klasy();
```

pozwała na jednoczesną deklarację zmiennej, utworzenie obiektu i przypisanie go do niej.

Po utworzeniu obiektu do jego pól można odwołać się za pomocą operatora . (kropka) według następującego schematu:

```
nazwa_obiektu.nazwa_pola;
```

Do metod utworzonego obiektu można odwołać się analogicznie jak do pól, czyli przy użyciu operatora . (kropka), zgodnie z poniższym schematem.

```
nazwa_obiektu.nazwa_metody();
```

Zadania z programowania obiektowego w dalszej części tej książki rozwiążemy, korzystając z następującego szablonu:

```
package zadanie.....;

class pole_prostokata
{
    // deklaracja zmiennych

    public void czytaj_dane() // deklaracja i definicja metody czytaj_dane()
    {
        .....
    }

    public void przetworz_dane() // deklaracja i definicja metody przetworz_dane()
    {
        .....
    }

    public void wyswietl_wynik() // deklaracja i definicja metody wyswietl_wynik()
    {
        .....
    }
}

public class Main
{
    public static void main(String[] args)
    {
        pole_prostokata pole = new pole_prostokata();
        // deklaracja zmiennej, utworzenie obiektu i przypisanie go do zmiennej
    }
}
```



```
        pole.czytaj_dane(); // wywołanie metody czytaj_dane()
        pole.przetworz_dane(); // wywołanie metody zapisz_dane_do_pliku()
        pole.wyswietl_wynik(); // wywołanie metody czytaj_dane_z_pliku()
    }
}
```

Metoda `czytaj_dane()` zajmuje się tylko czytaniem danych. Za ich przetworzenie odpowiedzialna jest metoda `przetworz_dane()`. Ostatnia z metod, `wyswietl_wynik()`, wyświetla na ekranie monitora przetworzone dane (wyniki). Metody mogą być z parametrem lub bezparametrowe w zależności od upodobań programisty.

Powyższy schemat zilustrujemy przykładem znanego nam już programu, który oblicza pole prostokąta.

#### ZADANIE

### 5.1

Napisz zgodnie z zasadami programowania obiektowego program, który oblicza pole prostokąta. Klasa powinna zawierać trzy metody:

- ❑ `czytaj_dane()` — metoda umożliwi wprowadzenie do programu długości boków `a` i `b` z klawiatury. W programie należy przyjąć, że `a` i `b` oraz zmienna `pole` są typu `double` (rzeczywistego).
- ❑ `przetworz_dane()` — metoda oblicza pole prostokąta według wzoru `pole = a*b`.
- ❑ `wyswietl_wynik()` — metoda wyświetla długości boków `a` i `b` oraz wartość zmiennej `pole` w określonym formacie. Dla zmiennych `a` i `b` oraz `pole` należy przyjąć format wyświetlania ich na ekranie z dwoma miejscami po przecinku.

#### Przykładowe rozwiązanie — listing 5.1

```
package zadanie51; //Zadanie 5.1
import java.io.*;

class pole_prostokata
{
    double a, b, pole;

    public void czytaj_dane() //deklaracja i opis metody czytaj_dane()
        throws IOException
    {
```

```
BufferedReader br = new BufferedReader(new
↳InputStreamReader(System.in));

System.out.println("Program oblicza pole prostokąta.");
System.out.println("Podaj bok a.");
a = Double.parseDouble(br.readLine());
System.out.println("Podaj bok b.");
b = Double.parseDouble(br.readLine());
}

public void przetworz_dane() //deklaracja i opis metody przetworz_dane()
{
    pole = a*b;
}

public void wyswietl_wynik() //deklaracja i opis metody wyswietl_wynik()
{
    System.out.print("Pole prostokąta o boku a = ");
    System.out.printf("%2.2f", a);
    System.out.print(" i boku b = ");
    System.out.printf("%2.2f", b);
    System.out.print(" wynosi ");
    System.out.printf("%2.2f.\n", pole);
}

}

public class Main {

    public static void main(String[] args)
        throws IOException
    {
        pole_prostokata pole = new pole_prostokata();
        //deklaracja zmiennej, utworzenie obiektu i przypisanie go do zmiennej

        pole.czytaj_dane(); //wywołanie metody czytaj_dane()
        pole.przetworz_dane(); //wywołanie metody przetworz_dane()
        pole.wyswietl_wynik(); //wywołanie metody wyswietl_wynik()
    }
}
```

---

**W naszym programie metoda czytaj\_dane():**

```
public void czytaj_dane() //deklaracja i opis metody czytaj_dane()
    throws IOException
{
    BufferedReader br = new BufferedReader(new
↳InputStreamReader(System.in));
```

```
System.out.println("Program oblicza pole prostokąta.");
System.out.println("Podaj bok a.");
a = Double.parseDouble(br.readLine());
System.out.println("Podaj bok b.");
b = Double.parseDouble(br.readLine());
}
```

wczytuje z klawiatury wartość boków a i b.

Metoda przetworz\_dane():

```
public void przetworz_dane() //deklaracja i opis metody przetworz_dane()
{
    pole = a*b;
}
```

oblicza z wykorzystaniem wzoru  $\text{pole} = a \cdot b$  pole prostokąta.

Metoda wyswietl\_wynik():

```
public void wyswietl_wynik() //deklaracja i opis metody wyswietl_wynik()
{
    System.out.print("Pole prostokąta o boku a = ");
    System.out.printf("%2.2f", a);
    System.out.print(" i boku b = ");
    System.out.printf("%2.2f", b);
    System.out.print(" wynosi ");
    System.out.printf("%2.2f.\n", pole);
}
```

wyświetla natomiast wartości zmiennych a i b oraz wartość zmiennej pole w ustalonym formacie.

Następująca linijka kodu:

```
pole_prostokata pole = new pole_prostokata();
```

pozwala na zadeklarowanie zmiennej pole, utworzenie obiektu i przypisanie go do zmiennej. W programie głównym zostają wywołane wszystkie trzy metody:

```
pole.czytaj_dane(); //wywołanie metody czytaj_dane()
pole.przetworz_dane(); //wywołanie metody przetworz_dane()
pole.wyswietl_wynik(); //wywołanie metody wyswietl_wynik()
```

Prawda, że wszystko jest teraz jasne i proste? Następne zadania, zamieszczone w dalszej części książki, spróbujemy rozwiązać według powyższego schematu.

## ZADANIE

**5.2**

Napisz zgodnie z zasadami programowania obiektowego program, który oblicza pierwiastki równania kwadratowego  $ax^2+bx+c = 0$  z wykorzystaniem instrukcji wyboru switch ... case. Klasa powinna zawierać trzy metody:

- ❑ `czytaj_dane()` — metoda jest odpowiedzialna za wczytanie danych do programu oraz obsłużenie sytuacji, kiedy  $a = 0$ . Zmienne  $a$ ,  $b$  i  $c$  to liczby rzeczywiste wprowadzane z klawiatury.
- ❑ `przetworz_dane()` — metoda odpowiada za wykonanie niezbędnych obliczeń.
- ❑ `wyswietl_wynik()` — metoda jest odpowiedzialna za wyświetlenie wyników na ekranie monitora. Dla zmiennych  $a$ ,  $b$ ,  $c$ ,  $x_1$  oraz  $x_2$  należy przyjąć format wyświetlania ich z dwoma miejscami po przecinku.

---

*Przykładowe rozwiązanie — listing 5.2*

---

```
package zadanie52; // Zadanie 5.2
import java.io.*;

class trojmian
{
    double a, b, c, delta, x1, x2;
    char liczba_pierwiastkow;

    public void czytaj_dane()
        throws IOException
    {
        BufferedReader br = new BufferedReader(new
        InputStreamReader(System.in));

        System.out.print("Program oblicza pierwiastki równania kwadratowego");
        System.out.println(" dla dowolnych wartości a, b oraz c.");
        System.out.println("Podaj a.");
        a = Double.parseDouble(br.readLine());

        if (a == 0)
        {
            System.out.println("Niedozwolona wartość współczynnika a.");
            System.exit(1); // wyjście z programu
        }
        else
        {
            System.out.println("Podaj b.");
```

```
b = Double.parseDouble(br.readLine());
System.out.println("Podaj c.");
c = Double.parseDouble(br.readLine());
}
}

public void przetworz_dane()
{
    delta = b*b-4*a*c;

    if (delta < 0) liczba_pierwiastkow = 0;
    if (delta == 0) liczba_pierwiastkow = 1;
    if (delta > 0) liczba_pierwiastkow = 2;

    switch(liczba_pierwiastkow)
    {
        case 1 : x1 = -b/(2*a);
        break;
        case 2 : {x1 = (-b-Math.sqrt(delta))/(2*a);
                  x2 = (-b+Math.sqrt(delta))/(2*a);
                  }
        break;
    }
}

public void wyswietl_wynik()
{
    System.out.println("Dla wprowadzonych liczb");
    System.out.printf("a = "+ "%2.2f,\n", a);
    System.out.printf("b = "+ "%2.2f,\n", b);
    System.out.printf("c = "+ "%2.2f,\n", c);

    switch (liczba_pierwiastkow)
    {
        case 0 : System.out.print("brak pierwiastków rzeczywistych.");
        break;
        case 1 : System.out.printf("trójmian ma jeden pierwiastek podwójny\n" +
            "x1 = " + "%2.2f.\n", x1);
        break;
        case 2 : {System.out.println("trójmian ma dwa pierwiastki");
                  System.out.printf("x1 = "+ "%2.2f,\n", x1);
                  System.out.printf("x2 = "+ "%2.2f.\n", x2);
                  }
        break;
    }
}

public class Main {
```

```
public static void main(String[] args)
    throws IOException
{
    trojmian trojmian1 = new trojmian();

    trojmian1.czytaj_dane();
    trojmian1.przetworz_dane();
    trojmian1.wyswietl_wynik();
}
```

---

**ZADANIE****5.3**

Napisz zgodnie z zasadami programowania obiektowego program, który w tablicy  $10 \times 10$  umieszcza losowo na przekątnej liczby od 0 do 9, a poza przekątną zera. Dodatkowo program oblicza sumę liczb znajdujących się na przekątnej. Klasa powinna zawierać trzy metody z parametrami:

- `czytaj_dane(double [][]macierz, int rozmiar)` — metoda umieszcza dane w tablicy.
- `przetworz_dane(double [][]macierz, int rozmiar)` — metoda oblicza i wyświetla sumę liczb znajdujących się na przekątnej.
- `wyswietl_wynik(double [][]macierz, int rozmiar)` — metoda wyświetla zawartość tablicy na ekranie monitora.

*Przykładowe rozwiązanie — listing 5.3*

---

```
package zadanie53; // Zadanie 5.3
import java.util.*;

class matrix
{
    public void czytaj_dane(double [][]macierz, int rozmiar)
    {
        int i, j;
        Random rand = new Random(); // generowanie liczby pseudolosowej

        for (i = 0; i < rozmiar; i++)
        {
            for (j = 0; j < rozmiar; j++)
            {
                if (i == j)
                    macierz[i][j] = Math.round(9*(rand.nextDouble()));
                // wpisywanie liczb pseudolosowych od 0 do 9 na przekątnej tablicy
            }
        }
    }
}
```

```
else
    macierz[i][j] = 0;
    // wpisywanie liczby 0 poza przekatną
}
}
}

public void przetworz_dane(double[][]macierz, int rozmiar)
{
    int i;
    double suma = 0;

    for (i = 0; i < rozmiar; i++)
        suma = suma+macierz[i][i];
    System.out.println("Suma elementów na przekątnej wynosi " +(int)
        ➔suma + ".");
    // rzutowanie
}

public void wyswietl_wynik(double[][]macierz, int rozmiar)
{
    int i, j;

    for (i = 0; i < rozmiar; i++)
    {
        for (j = 0; j < rozmiar; j++)
        {
            System.out.print((int) macierz[i][j] + " "); // rzutowanie
        }
        System.out.println();
    }
}

public class Main {

    public static void main(String[] args)
    {
        int rozmiar = 10;
        double [][] tablica = new double [rozmiar][rozmiar];

        matrix matrix1 = new matrix();

        matrix1.czytaj_dane(tablica, rozmiar);
        matrix1.przetworz_dane(tablica, rozmiar);
        matrix1.wyswietl_wynik(tablica, rozmiar);
    }
}
```

Wyjaśnienia wymagają następujące linijki kodu:

```
System.out.println("Suma elementów na przekątnej wynosi " +(int)
↳suma + "."); // rzutowanie
```

oraz

```
System.out.print((int) macierz[i][j] + " "); // rzutowanie
```

Zastosowano w nich tzw. rzutowania (ang. *casting*). **Rzutowanie** jest instrukcją dla kompilatora, aby zamienił on jeden typ na drugi. Może być ono wykonywane zarówno na typach prostych, jak i na typach obiektowych. Zmienna `suma` jest typu rzeczywistego — `double`. Zapis `(int) suma` powoduje zamianę na typ całkowity `int`. Podobną operację możemy przeprowadzić dla zmiennej `macierz[i][j]` — `(int) macierz[i][j]`.

---

## ZADANIE

### 5.4

Napisz zgodnie z zasadami programowania obiektowego program, który sortuje  $n$  liczb (w programie jest ich sześć). Klasa powinna zawierać trzy metody z parametrami:

- ❑ `czytaj_dane(int [] liczby, int n)` — metoda czyta dane i umieszcza je w tablicy o nazwie `liczby`.
- ❑ `przetworz_dane(int [] liczby, int n)` — metoda sortuje dane według wybranego algorytmu sortowania (w programie wykorzystano algorytm sortowania bąbelkowego).
- ❑ `wyswietl_wynik(int [] liczby, int n)` — metoda wyświetla zawartość posortowanej tablicy `liczby` na ekranie monitora.

---

#### Przykładowe rozwiązanie — listing 5.4

```
package zadanie54; //Zadanie 5.4

class sortowanie
{
    public void czytaj_dane(int [] liczby, int n)
    {
        int i;

        liczby[0] = 57;
        liczby[1] = 303;
        liczby[2] = 34;
        liczby[3] = 1025;
        liczby[4] = 8;
        liczby[5] = 20;
```



```
System.out.print("Liczby nieposortowane to: ");

for (i = 0; i < n; i++)
{
    if (i < n - 1)
        System.out.print(liczby[i] + ", ");
    else
        System.out.print(liczby[i] + ".");
}
System.out.println();
}

public void przetworz_dane(int [] liczby, int n)
{
    int i, j, x;

    for (i = 1; i <= n-1; i++)
    {
        for (j = n-1; j >= i; --j)
        {
            if (liczby[j-1] > liczby[j])
            {
                x = liczby[j-1];
                liczby[j-1] = liczby[j];
                liczby[j] = x;
            }
        } //koniec petli j
    } //koniec petli i
}

public void wyswietl_wynik(int [] liczby, int n)
{
    int i;

    System.out.print("Liczby posortowane to: ");

    for (i = 0; i < n; i++)
    {
        if (i < n - 1)
            System.out.print(liczby[i] + ", ");
        else
            System.out.print(liczby[i] + ".");
    }
    System.out.println();
}
}

public class Main {

    public static void main(String[] args)
```

```
{
    int n = 6;
    int [] liczby = new int[n];

    sortowanie babelki = new sortowanie();

    babelki.czytaj_dane(liczby, n);
    babelki.przetworz_dane(liczby, n);
    babelki.wyswietl_wynik(liczby, n);
}
}
```

---

Rezultat działania programu można zobaczyć na rysunku 5.1.

**Liczby nieposortowane to: 57, 303, 34, 1025, 8, 20.**

**Liczby posortowane to: 8, 20, 34, 57, 303, 1025.**

*Rysunek 5.1. Efekt działania programu Zadanie 5.4*

## Rekurencja

W języku Java metoda może wywołać samą siebie — proces ten nazywa się **rekurencją** (lub **rekursją**), a metoda taka nazywana jest **metodą rekurencyjną** (lub **rekursywną**). Główną zaletą rekurencji jest możliwość tworzenia metod realizujących niektóre algorytmy w sposób znacznie prostszy i bardziej czytelny niż niektóre ich odpowiedniki iteracyjne.

---

### ZADANIE

#### 5.5

Napisz program, który rekurencyjnie oblicza kolejne wartości  $n!$  dla  $n = 10$  i wynik wyświetla na ekranie komputera.

*Przykładowe rozwiązanie — listing 5.5*

---

```
package zadanie55; //Zadanie 5.5
import java.io.*;

class silnia1
{
    public long silnia(int liczba)
    {
```

```
        long zwrot = 1;
        if (liczba >= 2)
        {
            zwrot = liczba*silnia(liczba-1);
        }
        return zwrot;
    }
}

public class Main {

    public static void main(String[] args)
        throws IOException
    {
        int i, n;

        BufferedReader br = new BufferedReader(new
        ↳InputStreamReader(System.in));

        silnia1 s = new silnia1();

        System.out.println("Obliczanie silni dla dowolnej liczby
        ↳całkowitej.");
        System.out.println("Podaj n.");

        n = Integer.parseInt(br.readLine());

        for (i = 1; i <= n; i++)
        {
            System.out.println(" " + i + "! = " + s.silnia(i));
        }
    }
}
```

---

Algorytm obliczania silni został zawarty w metodzie `silnia(int liczba)`:

```
public long silnia(int liczba)
{
    long zwrot = 1;
    if (liczba >= 2)
    {
        zwrot = liczba*silnia(liczba-1);
    }
    return zwrot;
}
```

Rezultat działania programu można zobaczyć na rysunku 5.2.

**Obliczanie silni dla dowolnej liczby całkowitej.**

**Podaj n.**

**10**

**1! = 1**

**2! = 2**

**3! = 6**

**4! = 24**

**5! = 120**

**6! = 720**

**7! = 5040**

**8! = 40320**

**9! = 362880**

**10! = 3628800**

*Rysunek 5.2. Efekt działania programu Zadanie 5.5*

## ZADANIE

### 5.6

Napisz program, który rekurencyjnie znajduje 10 pierwszych liczb trójkątnych i wyświetla je na ekranie komputera.

#### Wskazówka

W matematyce liczba trójkątna to taka, którą można zapisać w postaci sumy kolejnych początkowych liczb naturalnych:  $T_n = 1 + 2 + 3 + \dots + (n - 1) + n$ . Liczby należące do tego ciągu nazywane są trójkątnymi, gdyż można je przedstawić w formie trójkąta. Na przykład  $\#6 = 21$ .

#1		1					
#2		2	1				
#3		3	2	1			
#4		4	3	2	1		
#5		5	4	3	2	1	
#6		6	5	4	3	2	1

Kolejne elementy ciągu uzyskujemy w sposób następujący:

#1 = 1

#2 = 1 + 2 = 3

#3 = 1 + 2 + 3 = 6

```
#4 = 1 + 2 + 3 + 4 = 10  
#5 = 1 + 2 + 3 + 4 + 5 = 15  
#6 = 1 + 2 + 3 + 4 + 5 + 6 = 21 itd.
```

---

*Przykładowe rozwiązanie — listing 5.6*

---

```
package zadanie56; //Zadanie 5.6  
  
class triangle  
{  
    public static int triangle(int n)  
    {  
        if (n == 1)  
            return 1;  
        else  
            return (n+triangle(n-1));  
    }  
}  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
        int i, n = 10;  
  
        triangle liczby = new triangle();  
  
        System.out.println("Program znajduje 10 pierwszych  
        ↳liczb trójkątnych.");  
  
        for (i = 1; i <= n; i++)  
        {  
            System.out.println("#" + i + " = " + liczby.triangle(i));  
        }  
    }  
}
```

Algorytm obliczania liczb trójkątnych został zawarty w metodzie `triangle(int n)`:

```
public static int triangle(int n)  
{  
    if (n == 1)  
        return 1;  
    else  
        return (n+triangle(n-1));  
}
```

Rezultat działania programu można zobaczyć na rysunku 5.3.

**Program znajduje 10 pierwszych liczb trójkątnych.**

**#1 = 1**

**#2 = 3**

**#3 = 6**

**#4 = 10**

**#5 = 15**

**#6 = 21**

**#7 = 28**

**#8 = 36**

**#9 = 45**

**#10 = 55**

***Rysunek 5.3.** Efekt działania programu Zadanie 5.6*

# 6

## Pliki tekstowe

*W tym rozdziale przedstawimy typowe zadania, wraz z przykładowymi rozwiązaniami, z wykorzystaniem plików tekstowych.*

### Pliki tekstowe — informacje ogólne

**Pliki tekstowe** zawierają informację niezakodowaną, bezpośrednio czytelną. Są one plikami o dostępie sekwencyjnym.

W języku Java operacje wejścia-wyjścia na plikach realizowane są przez strumienie. **Strumień** jest pojęciem abstrakcyjnym — może on wysyłać i pobierać informacje z fizycznego urządzenia (konsoli, dysku).

Java definiuje dwa typy strumieni: bajtowe i znakowe. Strumienie bajtów dostarczają wygodnego sposobu obsługi wejścia-wyjścia na bajtach. Korzysta się z nich podczas zapisu danych binarnych do pliku na dysku lub ich odczytu. Strumienie znakowe zostały stworzone do obsługi operacji wejścia-wyjścia na znakach. Do tego celu zostały zdefiniowane klasy **FileReader** oraz **FileWriter** oraz ich metody.

## ZADANIE

**6.1**

Napisz zgodnie z zasadami programowania obiektowego program, który wczytuje z klawiatury imię i nazwisko, zapisuje te dane do pliku tekstowego *dane.txt*, a następnie odczytuje je z tego pliku i wyświetla na ekranie komputera. Klasa powinna zawierać trzy metody:

- ❑ `czytaj_dane()` — metoda wczytuje z klawiatury imię i nazwisko.
- ❑ `zapisz_dane_do_pliku()` — metoda zapisuje imię i nazwisko do pliku tekstowego o nazwie *dane.txt*.
- ❑ `czytaj_dane_z_pliku()` — metoda odczytuje dane z pliku *dane.txt* i wyświetla je na ekranie komputera.

---

*Przykładowe rozwiązanie — listing 6.1*

---

```
package zadanie61; //Zadanie 6.1
import java.io.*;

class plik
{
    String dane, dane1;

    public void czytaj_dane()
        //deklaracja i definicja metody czytaj_dane()
        throws IOException
    {
        BufferedReader br = new BufferedReader(new
        InputStreamReader(System.in));

        System.out.println("Podaj imię i nazwisko.");
        dane = (br.readLine());
    }

    public void zapisz_dane_do_pliku()
        //deklaracja i definicja metody zapisz_dane_do_pliku()
        throws IOException
    {
        System.out.println("Zapisujemy dane do pliku dane.txt.");

        FileWriter fw = new FileWriter("dane.txt");
        fw.write(dane);
        fw.close(); //zamknięcie pliku
    }

    public void czytaj_dane_z_pliku()
        //deklaracja i definicja metody czytaj_dane_z_pliku()
        throws IOException
```



```
{
    System.out.println("Odczytujemy dane z pliku dane.txt.");
    System.out.println();
    FileReader fr = new FileReader("dane.txt");
    BufferedReader br = new BufferedReader(fr);

    while ((dane1 = br.readLine()) != null) // pętla odczytuje dane z pliku
    {
        System.out.println(dane1);
    }
    fr.close(); // zamknięcie pliku
} // koniec class plik

public class Main {

    public static void main(String[] args)
        throws IOException
    {
        plik plik1 = new plik();
        // deklaracja zmiennej, utworzenie obiektu i przypisanie go do zmiennej

        plik1.czytaj_dane(); // wywołanie metody czytaj_dane()
        plik1.zapisz_dane_do_pliku();
        // wywołanie metody zapisz_dane_do_pliku()
        plik1.czytaj_dane_z_pliku(); // wywołanie metody czytaj_dane_z_pliku()
    }
}
```

---

Zwróćmy uwagę, że w programie zadeklarowano dwie zmienne łańcuchowe:

```
String dane, dane1;
```

Zmienna o nazwie `dane` przechowuje informacje przed zapisaniem ich do pliku tekstowego, natomiast `dane1` przechowuje dane odczytane z tego pliku.

Metoda `czytaj_dane()`:

```
public void czytaj_dane()
    // deklaracja i definicja metody czytaj_dane()
    throws IOException
{
    BufferedReader br = new BufferedReader(new
    InputStreamReader(System.in));

    System.out.println("Podaj imię i nazwisko.");
    dane = (br.readLine());
}
```

wczytuje z klawiatury dane, czyli imię i nazwisko.

Metoda zapisz\_dane\_do\_pliku():

```
public void zapisz_dane_do_pliku()
    // deklaracja i definicja metody zapisz_dane_do_pliku()
    throws IOException
{
    System.out.println("Zapisujemy dane do pliku dane.txt.");

    FileWriter fw = new FileWriter("dane.txt");
    fw.write(dane);
    fw.close(); // zamknięcie pliku
}
```

poprzez właściwości klasy `FileWriter` zapisuje dane do pliku *dane.txt*.

```
FileWriter fw = new FileWriter("dane.txt");
fw.write(dane);
```

Ostatnia z metod, czytaj\_dane\_z\_pliku():

```
public void czytaj_dane_z_pliku()
    // deklaracja i definicja metody czytaj_dane_z_pliku()
    throws IOException
{
    System.out.println("Odczytujemy dane z pliku dane.txt.");
    System.out.println();
    FileReader fr = new FileReader("dane.txt");
    BufferedReader br = new BufferedReader(fr);

    while ((dane1 = br.readLine()) != null) // pętla odczytuje dane z pliku
    {
        System.out.println(dane1);
    }
    fr.close(); // zamknięcie pliku
}
```

poprzez właściwości klasy `FileReader` odczytuje `dane1` z pliku *dane.txt* i wyświetla je na ekranie monitora:

```
FileReader fr = new FileReader("dane.txt");
BufferedReader br = new BufferedReader(fr);
```

Za odczytanie wszystkich danych z pliku *dane.txt* odpowiedzialne są pętla `while` i warunek `dane1 = br.readLine() != null`:

```
while ((dane1 = br.readLine()) != null)
{
    .....
}
```



zapisuje do pliku tekstowego *dane.txt*, a następnie odczytuje z niego zapisane dane i wyświetla je na ekranie komputera. Klasa powinna zawierać trzy metody z parametrami:

- ❑ `czytaj_dane(int tablica[][], int rozmiar)` — tworzy tablicę  $10 \times 10$ .
- ❑ `zapisz_dane_do_pliku(int tablica[][], int rozmiar)` — metoda zapisuje tablicę  $10 \times 10$  do pliku tekstowego o nazwie *dane.txt*.
- ❑ `czytaj_dane_z_pliku(int tablica1[][], int rozmiar)` — odczytuje tablicę  $10 \times 10$  z pliku *dane.txt* i wyświetla ją na ekranie komputera.

---

*Przykładowe rozwiązanie — listing 6.2*

---

```
package zadanie62; //Zadanie 6.2
import java.io.*;

class matrix
{
    int rozmiar = 10;

    public void czytaj_dane(int tablica[][], int rozmiar)
    {
        int i, j;

        System.out.println("Tworzymy tablicę 10x10.");
        System.out.println();

        for (i = 0; i < rozmiar; i++) //tworzymy tablicę a 10x10
        {
            for (j = 0; j < rozmiar; j++)
            {
                if (i == j)
                    tablica[i][j] = 1;
                else
                    tablica[i][j] = 0;
                System.out.print(tablica[i][j] + " ");
            } //j
            System.out.println();
        } //i
        System.out.println();
    }

    public void zapisz_dane_do_pliku(int tablica[][], int rozmiar)
        throws IOException
    {
```

```
int i, j;

System.out.println("Zapisujemy tablicę 10x10 do pliku.");
System.out.println();

FileWriter fw = new FileWriter("dane.txt");
//tworzymy i otwieramy plik do zapisu

for (i = 0; i < rozmiar; i++)
{
    for (j = 0; j < rozmiar; j++)
    {
        fw.write((char)(tablica[i][j])); //rzutujemy i zapisujemy tablicę do pliku
        System.out.print(tablica[i][j] + " ");
    } //j
    System.out.println();
} //i
fw.close(); //zamknięcie pliku
}

public void czytaj_dane_z_pliku(int tablica1[][], int rozmiar) throws
↳IOException
{
    int i, j;

    System.out.println();
    System.out.println("Odczytujemy tablicę 10x10 z pliku.");
    System.out.println();

    FileReader fr = new FileReader("dane.txt");
    BufferedReader br = new BufferedReader(fr);

    for (i = 0; i < rozmiar; i++)
    {
        for (j = 0; j < rozmiar; j++)
        {
            tablica1[i][j] = (int) br.read(); //odczytujemy tablicę z pliku i rzutujemy
            System.out.print(tablica1[i][j]+" ");
        } //j
        System.out.println();
    } //i

    fr.close(); //zamknięcie pliku
}
} //koniec class matrix
```

```
public class Main {  
  
    public static void main(String[] args)  
        throws IOException  
    {  
        int rozmiar = 10;  
        int tab[][] = new int[rozmiar][rozmiar];  
        int tab1[][] = new int[rozmiar][rozmiar];  
  
        matrix matrix1 = new matrix(); // deklarujemy i tworzymy obiekt matrix1  
  
        matrix1.czytaj_dane(tab, rozmiar);  
        matrix1.zapisz_dane_do_pliku(tab, rozmiar);  
        matrix1.czytaj_dane_z_pliku(tab1, rozmiar);  
    }  
}
```

---

Zwróćmy uwagę, że w programie zadeklarowano dwie zmienne o nazwach `tab[][]` i `tab1[][]`. Pierwsza z nich przechowuje dane przeznaczone do zapisania w pliku, a druga dane odczytane z pliku.

Rezultat działania programu można zobaczyć na rysunku 6.2.

Tworzymy tablicę 10x10.

```
1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
```

Zapisujemy tablicę 10x10 do pliku.

```
1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
```

Odczytujemy tablicę 10x10 z pliku.

```
1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1
```

**Rysunek 6.2.** Efekt działania programu Zadanie 6.2

## ZADANIE

**6.3**

Napisz zgodnie z zasadami programowania obiektowego program, który tablicę *a* o wymiarach  $10 \times 10$  o postaci

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

przekształca w tablicę *b* o postaci

0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

i zapisuje ją do pliku tekstowego o nazwie *dane.txt*, a następnie odczytuje ją z tego pliku i wyświetla na ekranie. Klasa powinna zawierać cztery metody:

- ❑ `czytaj_dane()` — metoda tworzy tablicę *a* o wymiarach  $10 \times 10$ .
- ❑ `przetworz_dane()` — metoda przepisuje zawartość tablicy *a* o wymiarach  $10 \times 10$  do tablicy *b* o wymiarach  $10 \times 10$ .



- ❑ `zapisz_dane_do_pliku()` — zapisuje tablicę `b` o wymiarach  $10 \times 10$  do pliku.
- ❑ `czytaj_dane_z_pliku()` — metoda odczytuje z pliku tablicę `c` o wymiarach  $10 \times 10$  i wyświetla ją na ekranie.

---

*Przykładowe rozwiązanie — listing 6.3*

---

```
package zadanie63; //Zadanie 6.3
import java.io.*;

class matrix
{
    int n = 10;

    int a[][] = new int[n][n]; //deklaracja tablicy a
    int b[][] = new int[n][n]; //deklaracja tablicy b
    int c[][] = new int[n][n]; //deklaracja tablicy c

    public void czytaj_dane()
    {
        int i, j;
        System.out.println("Tworzymy tablicę a.");
        System.out.println();

        for (i = 0; i < n; i++) //tworzymy tablicę a
        {
            for (j = 0; j < n; j++)
            {
                if (i == 1)
                    a[i][j] = 1;
                else
                    a[i][j] = 0;
                System.out.print(a[i][j] + " ");
            } //j
            System.out.println();
        } //i
    }

    public void przetworz_dane()
    {
        int i, j;

        System.out.println();
        System.out.println("Przepisujemy zawartość tablicy a do tablicy b");

        for (i = 0; i < n; i++)
        {
```

```
        for (j = 0; j < n; j++)
        {
            b[i][j] = a[j][i]; //przepisujemy zawartość tablicy a do tablicy b
        }
    }
}

public void zapisz_dane_do_pliku()
    throws IOException
{
    int i, j;

    System.out.println("i zapisujemy tablicę b do pliku.");
    System.out.println();

    FileWriter fw = new FileWriter("dane.txt");
//tworzymy i otwieramy plik do zapisu

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            fw.write((char)(b[i][j])); //rztujemy i zapisujemy tablicę b do pliku
            System.out.print(b[i][j] + " ");
        } //j
        System.out.println();
    } //i
    fw.close(); //zamykamy plik
}

public void czytaj_dane_z_pliku()
    throws IOException
{
    int i, j;

    System.out.println();
    System.out.println("Odczytujemy dane z pliku i umieszczamy w tablicy c.");
    System.out.println();

    FileReader fr = new FileReader("dane.txt");
    BufferedReader br=new BufferedReader(fr);

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            c[i][j] = (int) br.read(); //odczytujemy dane z pliku i rztujemy
            System.out.print(c[i][j] + " ");
        } //j
```

---

```
        System.out.println();
    } //i

    fr.close(); //zamknięcie pliku
}
} //koniec class matrix

public class Main {

    public static void main(String[] args)
        throws IOException
    {
        matrix matrix1 = new matrix(); //deklarujemy i tworzymy obiekt matrix1

        matrix1.czytaj_dane();
        matrix1.przetworz_dane();
        matrix1.zapisz_dane_do_pliku();
        matrix1.czytaj_dane_z_pliku();
    }
}
```

---

Rezultat działania programu można zobaczyć na rysunku 6.3.

**Tworzymy tablicę a.**

```
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```

**Przepisujemy zawartość tablicy a do tablicy b  
i zapisujemy tablicę b do pliku.**

```
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
```

**Odczytujemy dane z pliku i umieszczamy w tablicy c.**

```
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
```

**Rysunek 6.3.** Efekt działania programu Zadanie 6.3