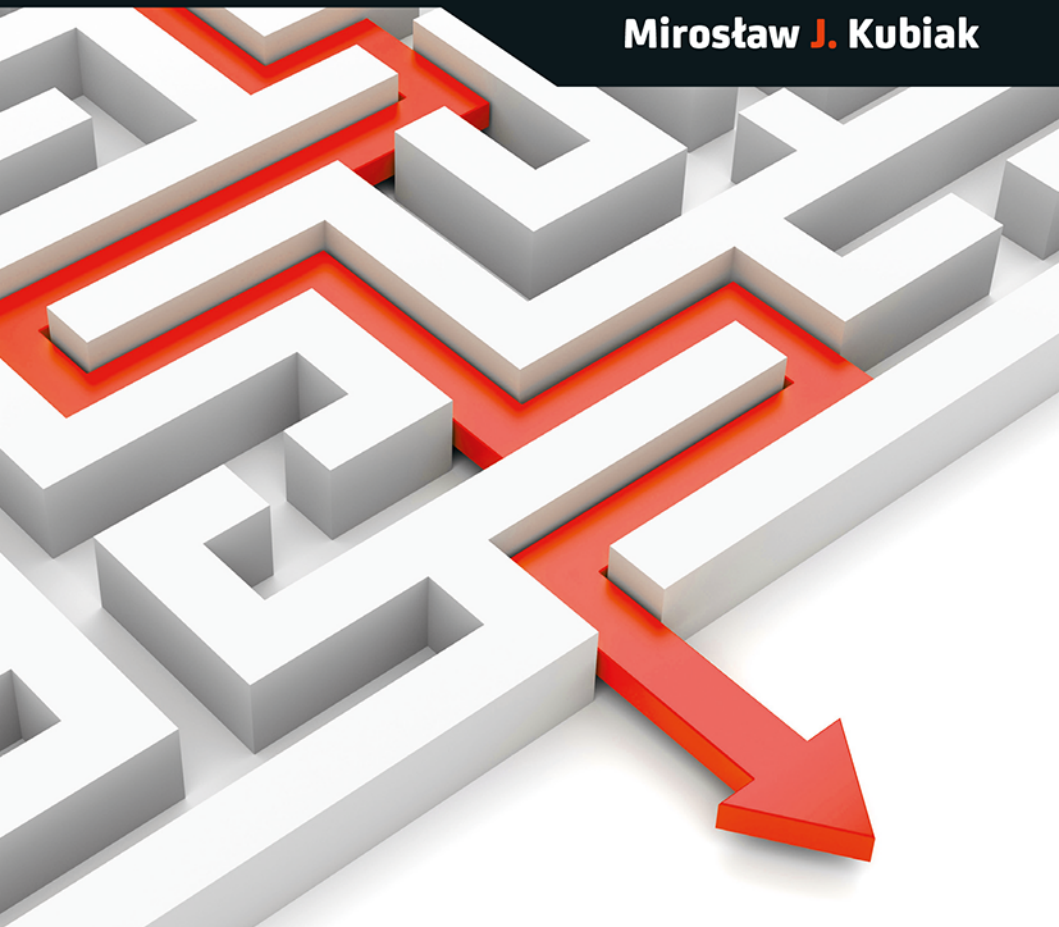
Mirosław J. Kubiak



Java

Zadania z programowania z przykładowymi rozwiązaniami

Helion

WYDANIE III

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki
Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Helion SA
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/javaz3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-6956-6

Printed in Poland.

Spis treści

	Od autora o trzecim wydaniu	**5**
Rozdział 1.	**Proste operacje wejścia-wyjścia**	**9**
	Operacje wejścia-wyjścia — informacje ogólne	9
	Obsługa sytuacji wyjątkowych	18
Rozdział 2.	**Podejmujemy decyzje w programie**	**21**
	Instrukcje warunkowe w języku Java	21
Rozdział 3.	**Iteracje**	**33**
	Iteracje — informacje ogólne	33
	Pętla for	34
	Pętla do ... while	35
	Pętla while	35
Rozdział 4.	**Tablice**	**59**
	Deklarowanie tablic jednowymiarowych	59
	Dostęp do elementów tablicy	60
	Tablice dwuwymiarowe	63
	Działania na macierzach	79
	Sortowanie bąbelkowe	87
	Łańcuchy tekstowe	90
	Konkatenacja	92
Rozdział 5.	**Programowanie obiektowe**	**95**
	Programowanie obiektowe — informacje ogólne	95
	Rekurencja	108
	Klasa osoba	113
	Dziedziczenie	114

Rozdział 6. Pliki **119**
Pliki tekstowe — informacje ogólne 119
Pliki o dostępie swobodnym — informacje ogólne 130

Rozdział 7. Wątki **135**
Podstawy wielowątkowości w Javie 135
Tworzymy pierwszy wątek 136
Tworzymy wiele wątków 140
Badamy, kiedy wątek się zakończy 143
Priorytety wątków 145
Synchronizacja 149
Korzystamy z synchronizowanych metod 150
Instrukcja synchronized 152
Komunikacja między wątkami 154
Zawieszanie, wznawianie oraz zatrzymywanie wątków 159

Rozdział 8. Rozszerzona pętla for i kolekcje **163**
Rozszerzona pętla for 163
Programowanie uogólnione — wprowadzenie 166
Kolekcje 167

Rozdział 9. Podążając w kierunku funkcyjnego paradygmatu programowania **179**
Wstęp 179
Co to jest paradygmat programowania? 180
Co to jest programowanie funkcyjne? 181

Bibliografia **183**

Rozdział 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. Omówiono też obsługę sytuacji wyjątkowych.

Operacje wejścia-wyjścia — informacje ogólne

Każda aplikacja powinna mieć 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 za pośrednictwem strumieni. **Strumień** jest pojęciem abstrakcyjnym. Może on wysyłać i pobierać informacje i jest połączony z fizycznym urządzeniem (np. klawiaturą, ekranem) poprzez system wejścia-wyjścia. W języku tym zdefiniowano dwa typy strumieni: bajtowe i znakowe. Standardowy strumień wyjściowy w Javie jest reprezentowany przez obiekt[1] `out` znajdujący się 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()`.

[1] Obiekty zostaną omówione w rozdziale 5.

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 zadanie_11; // Zadanie 1.1
import java.io.*;

public class Zadanie_11
{
    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*)[2]. **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 1.1
```

[3]

[2] Zobacz rozdział 5.

[3] Komentarze w programie oznaczamy dwoma ukośnikami `//`: `// to jest komentarz`.

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 poniższa 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.*;
```

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.

Rysunek 1.1. *Efekt działania programu Zadanie 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.
```

Zadanie

1.2 Napisz program, który wyświetla na ekranie komputera wartość predefiniowanej stałej π = 3,14… Należy przyjąć format wyświetlania tej stałej z dokładnością do pięciu miejsc po przecinku.

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

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

Przykładowe rozwiązanie — listing 1.2

```
package zadanie_12; // Zadanie 1.2

public class Zadanie_12
{
    public static void main(String[] args)
    {
        System.out.println("Program wyświetla liczbę pi z 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, na ilu polach 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 π przeznaczonych zostaje sześć pól, w tym pięć na część ułamkową. Znak specjalny `"...\n"` (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.

Rysunek 1.2. *Efekt działania programu Zadanie 1.2*

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

Zadanie

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

Przykładowe rozwiązanie — listing 1.3

```
package zadanie_13; // Zadanie 1.3

public class Zadanie_13
{
```

```
    public static void main(String[] args)
    {
       System.out.println("Program wyświetla pierwiastek kwadratowy");
       System.out.println("z liczby pi z dokładnością do 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.

Rysunek 1.3. *Efekt działania programu Zadanie 1.3*

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

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 zadanie_14; // Zadanie 1.4
import java.io.*;

public class Zadanie_14
{
    public static void main(String[] args)
        throws IOException
    {
        double r, objętość;

        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());

        objętość = 4*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.", objętość);
        System.out.println();
    }
}
```

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

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

gdzie potęgowanie zamieniono na mnożenie.

Rezultat działania programu można zobaczyć na rysunku 1.4.

Rysunek 1.4. *Efekt działania programu Zadanie 1.4*

```
Program oblicza objętość kuli.
Podaj promień r.
1
Objętość kuli o promieniu r = 1,00 wynosi 4,19.
```

Zadanie 1.5

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

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

Przykładowe rozwiązanie — listing 1.5

```
package zadanie_15; // Zadanie 1.5

public class Zadanie_15
{
    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 dla 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.

Rysunek 1.5. *Efekt działania programu Zadanie 1.5*

```
Program wyświetla wynik dzielenia całkowitego
bez reszty dla dwóch liczb całkowitych.
Dla liczb: a = 37, b = 11
37/11 = 3.
```

Zadanie

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

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 zadanie_16; // Zadanie 1.6

public class Zadanie_16
{
    public static void main(String[] args)
    {
        int a = 37;
        int b = 11;

        System.out.println("Program oblicza resztę z dzielenia
        ↪całkowitego");
        System.out.println("dla 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.

Rysunek 1.6. *Efekt działania programu Zadanie 1.6*

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

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`, `różnica`, `iloczyn` i `iloraz` należy przyjąć format ich wyświetlania na ekranie z dokładnością do dwóch miejsc po przecinku.

Przykładowe rozwiązanie — listing 1.7

```
package zadanie_17; // Zadanie 1.7
import java.io.*;

public class Zadanie_17
{
    public static void main(String[] args)
        throws IOException
    {
        float x, y, suma, różnica, 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;
        różnica = x-y;
        iloczyn = x*y;
        iloraz = x/y;

        System.out.printf("Dla liczb: x = " + "%2.2f", x);
        System.out.printf(" i y = " + "%2.2f", y);
        System.out.println(""); // wyświetlenie pustej linii
        System.out.printf("Suma = " + "%2.2f, \n", + suma);
        System.out.printf("Różnica = " + "%2.2f, \n", + różnica);
        System.out.printf("Iloczyn = " + "%2.2f, \n", + iloczyn);
        System.out.printf("Iloraz = " + "%2.2f. \n", + iloraz);
    }
}
```

Rezultat działania programu można zobaczyć na rysunku 1.7.

Rysunek 1.7. *Efekt działania programu Zadanie 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.
```

Obsługa sytuacji wyjątkowych

Jeśli do programu Zadanie 1.1 wprowadzimy z klawiatury poprawne dane, to jego działanie polegające na wprowadzeniu liczb dla dwóch boków i obliczeniu pola prostokąta się zakończy. Natomiast kiedy zamiast oczekiwanych liczb wprowadzimy dowolny znak, to program „wyłoży się" i pojawią się błędy związane z jego wykonaniem.

W celu uzyskania większej odporności programów na wszelkiego rodzaju awarie programowe i sprzętowe język Java oferuje wbudowaną **obsługę sytuacji wyjątkowych** (ang. *exception handling*). Wyjątki (ang. *exceptions*) definiują jednolity mechanizm komunikowania o błędach, które pojawiają się podczas wykonywania programu.

W Javie wszystkie wyjątki są reprezentowane przez klasy[4]. Każda klasa wyjątków jest wyprowadzona z wbudowanej klasy `Throwable`. Zarządzanie wyjątkami w Javie obsługiwane jest przez następujące słowa kluczowe: `try`, `catch`, `throw`, `throws` i `finally`.

W naszych rozważaniach obsługę wyjątków omówimy tylko pobieżnie (odsyłamy czytelników do bibliografii znajdującej się na końcu książki). Do przechwytywania wyjątku posłużymy się blokiem instrukcji `try ... catch` o następującej, schematycznej budowie:

[4] Zobacz rozdział 5.

```
try
{
   ..... // instrukcja niebezpieczna, mogąca powodować wyjątek
}

catch (TypWyjątku identyfikatorWyjątku)
{
   ..... // instrukcja obsługi wyjątku
}
```

Przykładową obsługę sytuacji wyjątkowych zilustrujemy w zadaniu poniżej, które jest modyfikacją zadania 1.1.

Zadanie

1.8 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). Dodatkowo w program wbuduj obsługę sytuacji wyjątkowych[5].

Przykładowe rozwiązanie — listing 1.8

```
package zadanie_18; // Zadanie 1.8
import java.io.*;

public class Zadanie_18
{
    public static void main(String[] args)
        throws IOException
    {
        double a, b, pole;

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

        try
        {
            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);
```

[5] Zachęcamy czytelnika, żeby w ramach dodatkowych ćwiczeń utrwalających wszystkie pozostałe programy w książce wyposażył w obsługę sytuacji wyjątkowych.

```
            System.out.println(" wynosi " + pole + ".");
        }
        catch (NumberFormatException exc)
        {
            System.out.println("Nie wczytano liczby. Koniec programu.");
        }
    }
}
```

Obsłużenie sytuacji krytycznej, związanej z wprowadzeniem do programu błędnych danych, zawarto w następujących linijkach kodu:

```
try
        {
            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 + ".");
        }
        catch (NumberFormatException exc)
        {
            System.out.println("Nie wczytano liczby. Koniec programu.");
        }
```

Wyjątek `NumberFormatException exc` znajdujący się w linijkach kodu

```
catch(NumberFormatException exc)
{
System.out.println("Nie wczytano liczby. Koniec programu.");
}
```

jest uruchamiany z chwilą, kiedy zamiast oczekiwanej dowolnej liczby typu `double` wprowadzimy dowolny znak, np. `a`. Wyjątek nie jest uruchamiany, jeśli do działającego programu wprowadzimy poprawne dane.

Rezultat działania programu, w którym wygenerowano sytuację wyjątkową, możemy zobaczyć na rysunku 1.8.

Rysunek 1.8. *Efekt działania programu Zadanie 1.8*

```
Program oblicza pole prostokąta.
Podaj bok a.
a
Nie wczytano liczby. Koniec programu.
```


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Poznaj Javę w praktyce

Java to nowoczesny, współbieżny, obiektowy język programowania, który zdobył uznanie tysięcy programistów na całym świecie. Ogromne możliwości, niezależność od platformy, niezawodność i bezpieczeństwo, a także łatwość tworzenia i przejrzystość kodu powodują, że Java od lat cieszy się niesłabnącą popularnością. Na programistów posługujących się tym językiem czekają setki atrakcyjnych ofert pracy. Java znajduje zastosowanie w najróżniejszych dziedzinach i branżach, co sprawia, że opracowane za jej pomocą programy można spotkać niemal wszędzie — wiele popularnych aplikacji sieciowych i mobilnych zostało napisanych właśnie w Javie.

Teoretyczna nauka programowania jest jak czytanie o lataniu — można się w ten sposób dużo dowiedzieć, ale z pewnością nie zapewni to doświadczenia niezbędnego, żeby naprawdę wystartować. Dlatego z językiem programowania warto zapoznać się od strony praktycznej: pisać kod, wykonywać ćwiczenia programistyczne, wykorzystywać kolejne techniki i konstrukcje języka, a przede wszystkim mierzyć się z coraz trudniejszymi zadaniami. Doskonałym wsparciem w tym działaniu będzie najnowsze wydanie książki *Java. Zadania z programowania z przykładowymi rozwiązaniami*. Dzięki niej dowiesz się, jak wykorzystać otwarte, bezpłatne środowisko NetBeans IDE 8.2 do tworzenia aplikacji o prostym i przejrzystym kodzie, i szybko opanujesz Javę!

- **Proste operacje wejścia-wyjścia**
- **Instrukcje warunkowe i iteracje**
- **Tablice, macierze i łańcuchy tekstowe**
- **Programowanie obiektowe**
- **Przetwarzanie plików tekstowych**
- **Zastosowanie wielowątkowości**
- **Kolekcje i ich możliwości**

Spraw, aby Java nie miała przed Tobą tajemnic!

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

ISBN 978-83-283-6956-6

Cena: 37,00 zł