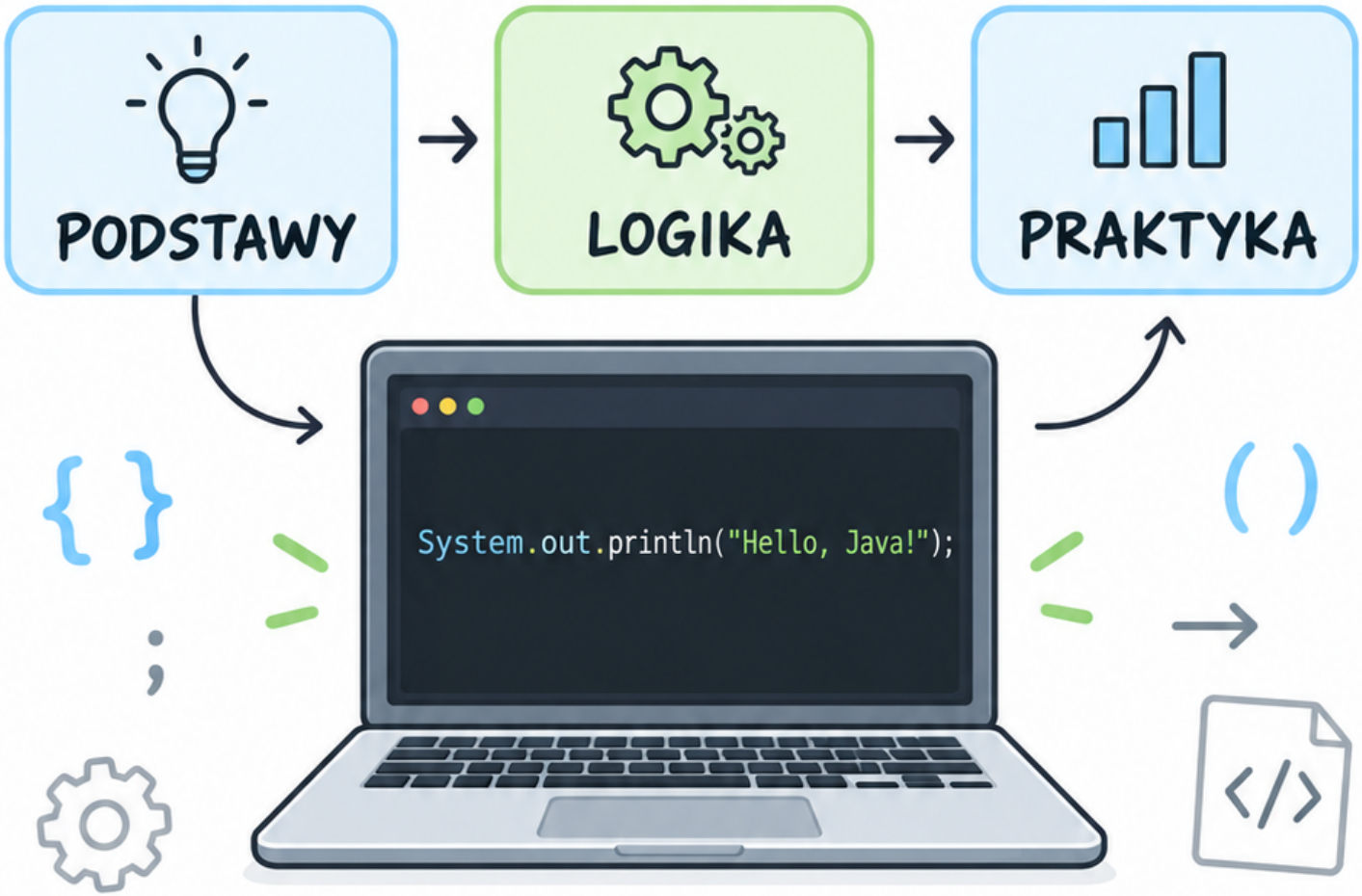
BONUS #2

50 PRAKTYCZNYCH WYZWAŃ Z JAVY

Materiał uzupełniający



Od pierwszego programu do praktycznych miniaplikacji

Ucz się przez działanie. Pisz kod, testuj rozwiązania i rozwijaj swoje umiejętności krok po kroku.

INFORMACJE O MATERIALE

50 PRAKTYCZNYCH WYZWAŃ Z JAVY

Materiał uzupełniający do samodzielnej nauki programowania.



1

O materiale

50 praktycznych wyzwań z Javy
Materiał uzupełniający do samodzielnej nauki programowania.

2

Prawa autorskie

© 2026. Wszelkie prawa zastrzeżone.
Materiał przeznaczony jest do osobistego użytku edukacyjnego. Kopiowanie, rozpowszechnianie lub odsprzedaż materiału bez zgody właściciela praw autorskich są niedozwolone.

3

Informacja edukacyjna

Ważna informacja: Przykłady kodu i zadania służą celom edukacyjnym. Zaleca się samodzielne uruchamianie, testowanie i modyfikowanie programów.

Java jest znakiem towarowym firmy Oracle i/lub jej podmiotów stowarzyszonych. Niniejszy materiał ma charakter niezależny.

SPIS TREŚCI

01

02

03

04

05



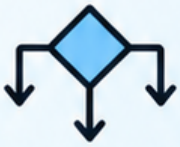
NA POCZĄTEK

01 — Informacje o materiale 01
02 — Spis treści 02
03 — Jak korzystać z wyzwań? 03



MODUŁ 01

04 — Podstawy i pierwsze programy 04



MODUŁ 02

16 — Warunki i decyzje 16



MODUŁ 03

28 — Pętle, napisy i tablice 28



MODUŁ 04

40 — Metody, kolekcje i obiekty 40



MODUŁ 05

52 — Wyjątki, pliki i miniaplikacje 52



ZAKOŃCZENIE

64 — Twoje kolejne kroki 64

GOTOWY NA 50 WYZWAŃ?

Najlepiej uczysz się programowania wtedy, gdy sam piszesz kod.

Nie musisz znać całej Javy, aby zacząć programować. Wystarczy jedno małe zadanie, kilka linijek kodu i gotowość do eksperymentowania.

Przed Tobą 50 praktycznych wyzwań. Każde z nich pomoże Ci przećwiczyć konkretną umiejętność. Zaczniemy od prostych instrukcji, a następnie przejdziemy do warunków, pętli, metod, obiektów i małych aplikacji.

JAK PRACOWAĆ Z WYZWANIAMI?

1. **PRZECZYTAJ** — Dowiedz się, jaki problem masz rozwiązać.
2. **NAPISZ** — Spróbuj samodzielnie przygotować kod.
3. **URUCHOM** — Sprawdź, czy program działa.
4. **POPRAW** — Przeanalizuj błędy i spróbuj ponownie.
5. **ROZWIŃ** — Wykonaj dodatkowe zadanie BONUS.

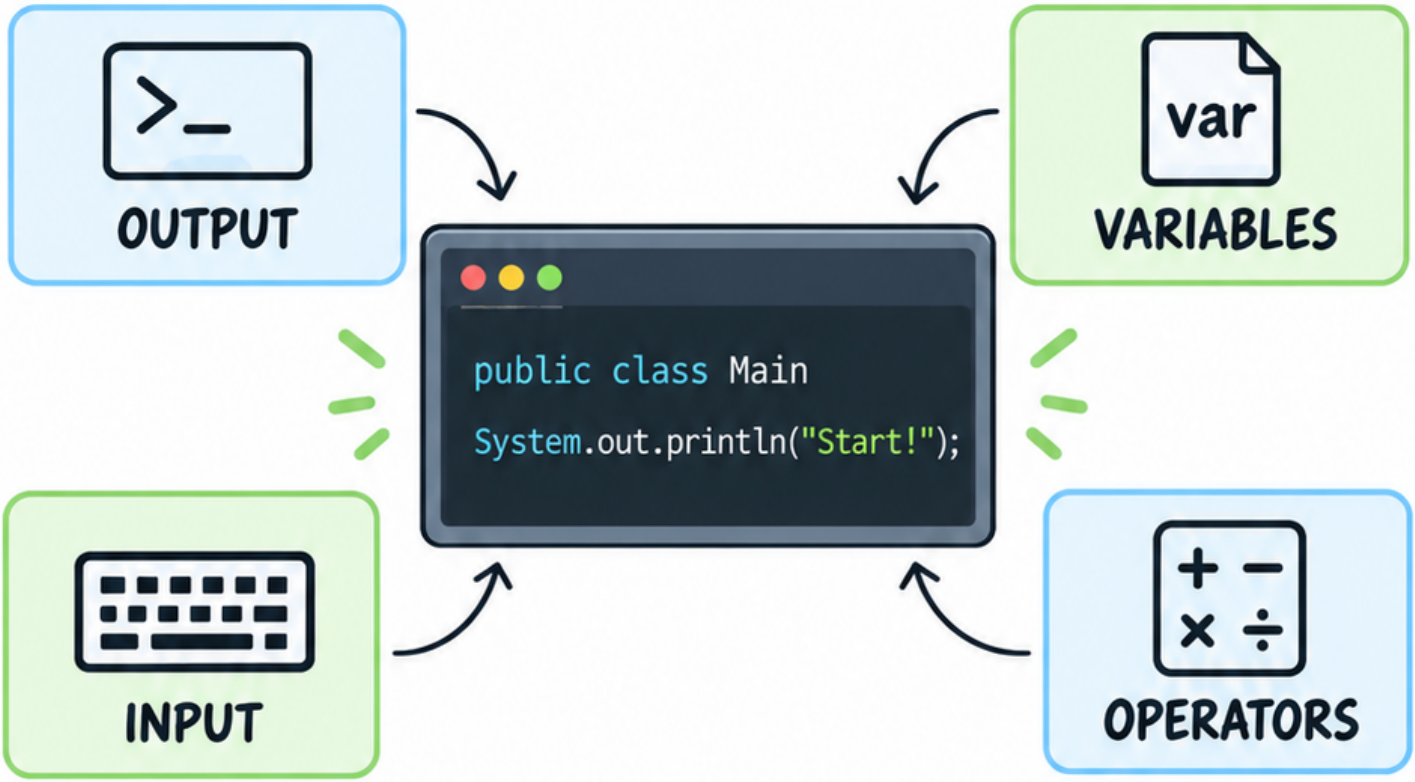


Błędy są częścią nauki.
Każda poprawiona linijka kodu to krok naprzód.

MODUŁ 01

PODSTAWY I PIERWSZE PROGRAMY

Twoje pierwsze kroki w praktycznym programowaniu.



W tym module nauczysz się:

- Wyświetlać komunikaty.
- Tworzyć i wykorzystywać zmienne.
- Pobierać dane od użytkownika.
- Wykonywać podstawowe obliczenia.
- Przekształcać dane w praktyczne wyniki.

Każdy program zaczyna się od pierwszej linijki kodu.

01 WYZWANIE 01 – TWÓJ PIERWSZY KOMUNIKAT

Napisz program, który coś wyświetla.

1. CEL WYZWANIA

Napisz program w Javie, który wyświetli dwa komunikaty w osobnych liniach.

2. TWOJE ZADANIE

Program powinien wyświetlić:

- **Cześć, Javo!**
- **To mój pierwszy program.**

3. OD CZEGO ZACZĄĆ?

```
public class Main {
    public static void main(String[] args) {
        // Wpisz tutaj dwie instrukcje
    }
}
```



4. PODPOWIEDŹ

Do wyświetlania tekstu użyj instrukcji:

```
System.out.println("Twój tekst");
```

5. SPRAWDŹ SWÓJ PROGRAM

- [ ] Czy program uruchamia się bez błędów?
- [ ] Czy oba komunikaty są widoczne?
- [ ] Czy każdy komunikat znajduje się w osobnej linii?

6. BONUS

Dodaj trzeci komunikat z własnym imieniem.

02 WYZWANIE 02 – MOJA WIZYTÓWKA

Przechowuj informacje w zmiennych.

1. CEL WYZWANIA

Utwórz prostą wizytówkę wykorzystującą trzy zmienne.

2. TWOJE ZADANIE

Zapisz w programie:

- **Imię** jako **String**.
- **Wiek** jako **int**.
- Informację, czy uczysz się Javy, jako **boolean**.

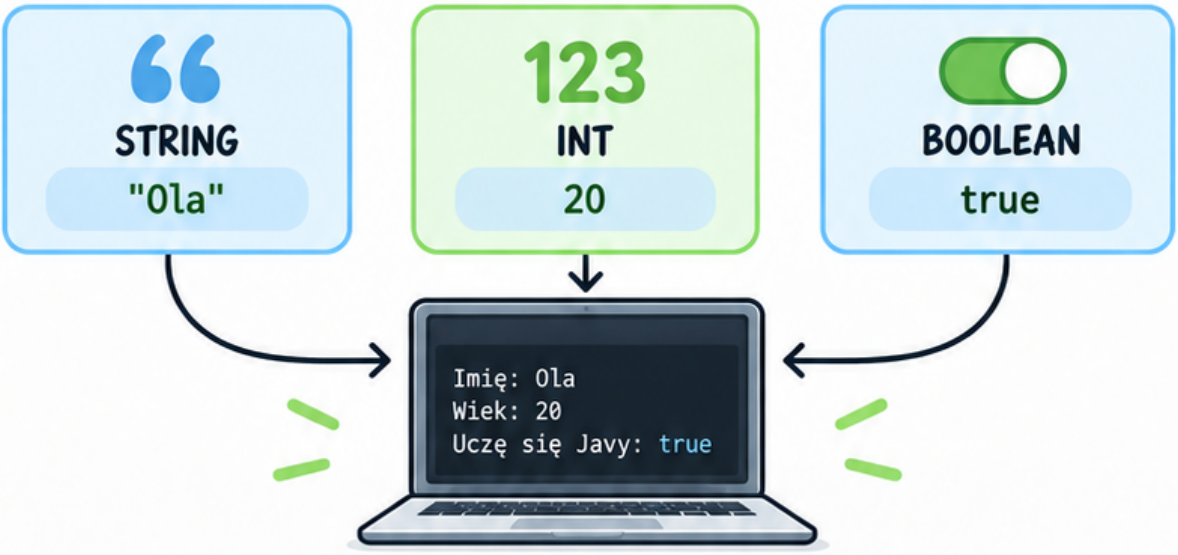
Następnie wyświetl te dane na ekranie.

3. PRZYKŁADOWE DANE

```
String imie = "Ola";
int wiek = 20;
boolean uczeSie = true;
```

4. OCZEKIWANY WYNIK

```
Imię: Ola
Wiek: 20
Uczę się Javy: true
```



5. PODPOWIEDŹ

String przechowuje tekst.
int przechowuje liczby całkowite.
boolean przechowuje wartości **true** lub **false**.

6. SPRAWDŹ SWÓJ PROGRAM

- [ ] Czy wszystkie zmienne mają odpowiednie typy?
- [ ] Czy program wyświetla wartości?
- [ ] Czy po zmianie imienia zmienia się wynik?

7. BONUS

Dodaj zmienną **double wzrost** i wyświetl wzrost w metrach.

03 WYZWANIE 03 — POWITAJ UŻYTKOWNIKA

Odczytaj dane wpisane z klawiatury.

1. CEL WYZWANIA

Napisz program, który zapyta użytkownika o imię i przywita go spersonalizowanym komunikatem.

2. TWOJE ZADANIE

1. Wyświetlić pytanie o imię.
2. Odczytać wpisaną odpowiedź.
3. Zapisać ją w zmiennej.
4. Wyświetlić powitanie.

3. PRZYKŁAD DZIAŁANIA

```
Jak masz na imię?
Ania
Cześć, Ania!
```

4. KOD POMOCNICZY

```
import java.util.Scanner;
Scanner scanner = new Scanner(System.in);
String imie = scanner.nextLine();
```

Import umieść przed klasą, a tworzenie i odczyt obiektu Scanner — wewnątrz metody main.



5. PODPOWIEDŹ

Scanner pozwala odczytywać dane wpisywane przez użytkownika. `nextLine()` odczytuje cały wiersz tekstu.

6. SPRAWDŹ SWÓJ PROGRAM

- [ ] Czy program czeka na wpisanie imienia?
- [ ] Czy wykorzystuje wpisany tekst?
- [ ] Czy działa dla różnych imion?

7. BONUS

Zapytaj również o ulubiony kolor i dodaj go do powitania.

04 WYZWANIE 04 — KALKULATOR SUMY

Dodawaj liczby podane przez użytkownika.

1. CEL WYZWANIA

Zbuduj prosty kalkulator, który odczyta dwie liczby całkowite i wyświetli ich sumę.

2. TWOJE ZADANIE

1. Poprosić o pierwszą liczbę.
2. Poprosić o drugą liczbę.
3. Obliczyć sumę.
4. Wyświetlić wynik.

3. PRZYKŁAD DZIAŁANIA

```
Podaj pierwszą liczbę: 7
Podaj drugą liczbę: 5
Suma: 12
```



4. PRZYDATNY KOD

```
int a = scanner.nextInt();
int b = scanner.nextInt();
```

5. PODPOWIEDŹ

Operator `+` pozwala dodawać wartości liczbowe. Metoda `nextInt()` odczytuje liczbę całkowitą.

6. PRZETESTUJ

Przypadek A: 7 + 5 = 12

Przypadek B: 0 + 9 = 9

Przypadek C: -3 + 8 = 5

7. BONUS

Rozbuduj program tak, aby obliczał również różnicę i iloczyn dwóch liczb.

05 WYZWANIE 05 — PRZELICZNIK TEMPERATURY

Zamień stopnie Celsjusza na Fahrenheita.

1. CEL WYZWANIA

Napisz program przeliczający temperaturę ze stopni Celsjusza na stopnie Fahrenheita.

2. TWOJE ZADANIE

1. Odczytać temperaturę w stopniach Celsjusza.
2. Wykonać odpowiednie obliczenie.
3. Wyświetlić temperaturę w stopniach Fahrenheita.

3. WZÓR

20°C → $°F = (°C \times \frac{9}{5}) + 32$ → 68°F

4. PRZYKŁAD

```
Podaj temperaturę w °C: 20
Temperatura w °F: 68.0
```

5. PODPOWIEDŹ

Wykorzystaj typ `double`, który pozwala pracować z liczbami zawierającymi część dziesiętną. W kodzie użyj `9.0 / 5`, aby uniknąć dzielenia całkowitoliczbowego.

6. SPRAWDŹ SWÓJ PROGRAM

- [ ] Czy 0°C daje 32°F?
- [ ] Czy 100°C daje 212°F?
- [ ] Czy 20°C daje 68°F?

7. BONUS

Dodaj możliwość przeliczania temperatury w odwrotnym kierunku — z Fahrenheita na Celsjusza.

06 WYZWANIE 06 — POLE PROSTOKĄTA

Dwie liczby, jedno proste obliczenie.

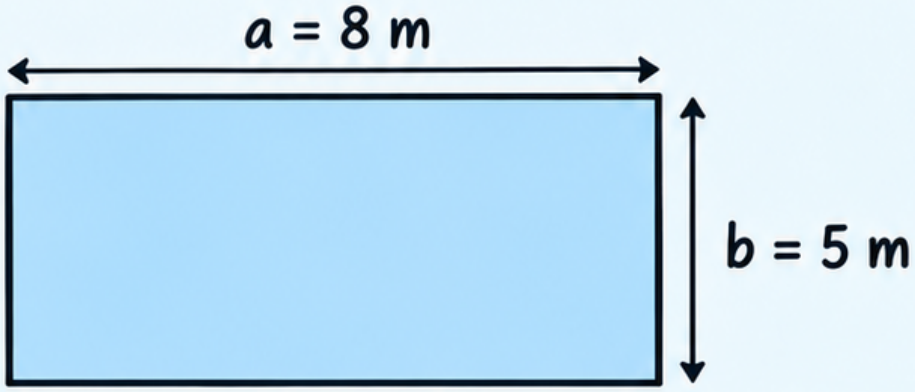
1. CEL WYZWANIA

Napisz program, który obliczy pole prostokąta na podstawie jego długości i szerokości.

2. TWOJE ZADANIE

1. Poprosić o długość prostokąta.
2. Poprosić o szerokość prostokąta.
3. Pomnożyć podane wartości.
4. Wyświetlić wynik w metrach kwadratowych.

3. ILUSTRACJA I WZÓR



$$P = a \times b \rightarrow P = 40\ m^2$$

4. PRZYKŁADOWE DANE

Długość: **8 m**

Szerokość: **5 m**

Pole: **40 m²**

5. PODPOWIEDŹ

W Javie użyj operatora `*` do mnożenia. Wartości możesz przechowywać w zmiennych typu `double`.

6. SPRAWDŹ

- [ ] Czy program poprawnie odczytuje obie wartości?
- [ ] Czy oblicza pole?
- [ ] Czy dla wymiarów 8 i 5 otrzymujesz 40?

7. BONUS

Dodaj obliczenie obwodu prostokąta.

07 WYZWANIE 07 — PROSTY PARAGON

Ile kosztują Twoje zakupy?

1. CEL WYZWANIA

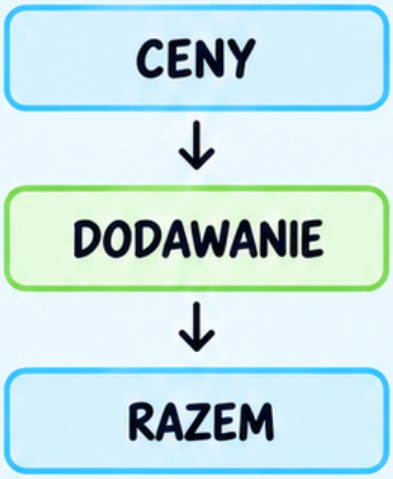
Stwórz program obliczający całkowitą wartość zakupów.

2. SYTUACJA

- Zeszyt — 8,50 zł
- Długopis — 4,00 zł
- Plecak — 75,00 zł

3. ILUSTRACJA I WZÓR





4. TWOJE ZADANIE

1. Utwórz zmienne przechowujące ceny produktów.
2. Oblicz sumę zakupów.
3. Wyświetl wynik z odpowiednim opisem.

5. PODPOWIEDŹ

Do przechowywania cen w tym ćwiczeniu możesz użyć typu `double`. Do wyświetlenia dwóch miejsc po przecinku wykorzystaj `System.out.printf("%.2f", suma);`.

W rzeczywistych aplikacjach finansowych preferuje się `BigDecimal` do dokładnych obliczeń dziesiętnych.

6. SPRAWDŹ

- [ ] Czy wszystkie ceny zostały uwzględnione?
- [ ] Czy wynik jest poprawny?
- [ ] Czy kwota jest czytelna?

7. BONUS

Dodaj liczbę kupowanych zeszytów i oblicz ich łączny koszt.

08 WYZWANIE 08 – PRZELICZNIK CZASU

Zamień sekundy na minuty

1. CEL WYZWANIA

Zbuduj program zamieniający liczbę sekund na pełne minuty i pozostałe sekundy.

2. TWOJE ZADANIE

Użytkownik podaje liczbę sekund. Program powinien obliczyć:

1. Liczbę pełnych minut.
2. Liczbę pozostałych sekund.

3. ILUSTRACJA I WZÓR



135 SEKUND

DZIELENIE

2 MIN + 15 SEK

4. JAK TO DZIAŁA?

135 / 60 = 2

135 % 60 = 15

/

Operator dzielenia (/)

przy liczbach typu int wykonuje dzielenie całkowite (bez reszty).

%

Operator modulo (%)

zwraca resztę z dzielenia dwóch liczb.

5. PRZYKŁAD

Wejście: **135** sekund

Wynik: **2** minuty i **15** sekund.

6. PODPOWIEDŹ

Operator / przy dwóch liczbach typu int wykonuje dzielenie całkowite. Operator % zwraca resztę z dzielenia.

7. SPRAWDŹ

- [ ] 60 sekund → 1 minuta i 0 sekund
- [ ] 125 sekund → 2 minuty i 5 sekund
- [ ] 45 sekund → 0 minut i 45 sekund

8. BONUS

Rozbuduj program o zamianę sekund na godziny, minuty i sekundy.

09 WYZWANIE 09 — OBLICZ ŚREDNIĄ

Oblicz średnią z trzech wyników

1. CEL WYZWANIA

Napisz program obliczający średnią arytmetyczną trzech liczb.

2. TWOJE ZADANIE

1. Odczytać trzy liczby.
2. Dodać je do siebie.
3. Podzielić sumę przez trzy.
4. Wyświetlić średnią.

3. ILUSTRACJA I WZÓR



$$\text{Średnia} = \frac{a + b + c}{3}$$

4. PRZYKŁAD

Liczba 1: **8**
Liczba 2: **7**
Liczba 3: **9**
Średnia: **8.0**

5. PODPOWIEDŹ

Użyj zmiennych typu `double`. Zwróć uwagę na nawiasy w wyrażeniu matematycznym.

6. SPRAWDŹ

- [ ] Czy dodajesz wszystkie trzy liczby?
- [ ] Czy dzielisz całą sumę przez trzy?
- [ ] Czy wynik jest poprawny dla liczb dziesiętnych?

7. BONUS

Wyświetl wynik z dokładnością do dwóch miejsc po przecinku.

10 WYZWANIE 10 – PRZELICZNIK ODLEGŁOŚCI

Kilometry, metry i centymetry

1. CEL WYZWANIA

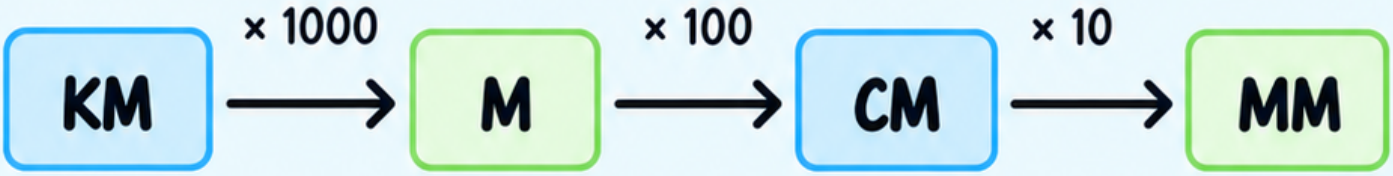
Napisz program zamieniający kilometry na metry i centymetry.

2. TWOJE ZADANIE

Użytkownik podaje odległość w kilometrach. Program oblicza:

1. Metry.
2. Centymetry.

3. ILUSTRACJA I WZÓR



Przykład: 2.5 km

2.5 km → 2500 m → 250000 cm

(kilometry) (metry) (centymetry)

4. PRZYDATNE ZALEŻNOŚCI

1 km = 1000 m
1 m = 100 cm
1 km = 100000 cm

5. PODPOWIEDŹ

Wykorzystaj zmienną typu `double` oraz mnożenie.

6. SPRAWDŹ

- [ ] Czy 1 km daje 1000 m?
- [ ] Czy 0.5 km daje 500 m?
- [ ] Czy 2 km dają 200000 cm?

7. BONUS

Dodaj przeliczanie kilometrów na milimetry.

01 MODUŁ 01 — SPRAWDŹ SWOJE UMIEJĘTNOŚCI

Co już potrafisz zrobić w Javie?

TWÓJ POSTĘP

Zaznacz wykonane wyzwania z modułu 01.

☐ 01 ☐ 02 ☐ 03 ☐ 04 ☐ 05 ☐ 06 ☐ 07 ☐ 08 ☐ 09 ☐ 10

LISTA UMIEJĘTNOŚCI

Sprawdź, co już potrafisz. Zaznacz wykonane punkty.

- ☐ 1 Potrafię wyświetlać tekst w konsoli.
- ☐ 2 Umiem tworzyć zmienne.
- ☐ 3 Rozróżniam typy `int`, `double`, `String` i `boolean`.
- ☐ 4 Potrafię odczytywać dane za pomocą `Scanner`.
- ☐ 5 Umiem dodawać i mnożyć liczby.
- ☐ 6 Rozumiem dzielenie całkowite.
- ☐ 7 Potrafię używać operatora `%`.
- ☐ 8 Umiem obliczać średnie.
- ☐ 9 Potrafię przeliczać jednostki.
- ☐ 10 Umiem sprawdzać działanie własnego programu.

MINI-SPRAWDZIAN

Napisz program, który:

1. Pobiera długość i szerokość pokoju.
2. Oblicza jego powierzchnię.
3. Oblicza koszt podłogi przy cenie 45 zł za m^2.
4. Wyświetla oba wyniki.

PRZYKŁAD

```
Długość: 4 m
Szerokość: 3 m
Powierzchnia: 12 m²
Koszt podłogi: 540 zł
```

PYTANIA DO REFLEKSJI

- ☐ Które wyzwanie było najłatwiejsze?
- ☐ Co sprawiło Ci największą trudność?
- ☐ Czy potrafisz wyjaśnić własny kod?

MODUŁ 01 — UKOŃCZONY

☐ Zaznacz, gdy ukończysz wszystkie wyzwania z modułu 01.

Świetna robota!
Każdy mały krok to duży postęp w nauce Javy!

MODUŁ 02

WARUNKI I DECYZJE

Naucz swój program podejmować decyzje.



CZEGO SIĘ NAUCZYSZ?

1. Tworzenia instrukcji if.
2. Korzystania z else.
3. Porównywania wartości.
4. Stosowania operatorów logicznych.
5. Budowania bardziej interaktywnych programów.

PRZYKŁADOWA LOGIKA

Czy użytkownik ma co najmniej 18 lat?

TAK → Wyświetl komunikat A.

NIE → Wyświetl komunikat B.

GŁÓWNA IDEA

Program może wykonywać różne instrukcje w zależności od spełnienia warunku.

Teraz Twój kod zacznie reagować na różne sytuacje.

11 WYZWANIE 11 — SPRAWDŹ WIEK

Sprawdź, czy użytkownik jest pełnoletni

1. CEL WYZWANIA

Napisz program, który sprawdzi, czy użytkownik ma co najmniej 18 lat.

2. TWOJE ZADANIE

1. Zapytaj o wiek.
2. Odczytaj liczbę.
3. Sprawdź, czy wiek jest większy lub równy 18.
4. Wyświetl odpowiedni komunikat.

3. ILUSTRACJA I WZÓR



4. PRZYKŁADOWE DANE

PRZYKŁAD 1
Wiek: **20**
Wynik: Jesteś osobą pełnoletnią.

PRZYKŁAD 2
Wiek: **15**
Wynik: Nie jesteś jeszcze osobą pełnoletnią.

5. PODPOWIEDŹ

```
if (wiek >= 18) {
    // Komunikat dla osoby pełnoletniej
} else {
    // Komunikat dla osoby niepełnoletniej
}
```

6. SPRAWDŹ

- [ ] Czy program poprawnie działa dla wieku 18?
- [ ] Czy działa dla wieku 17?
- [ ] Czy działa dla wieku 25?

7. BONUS

Dodaj osobny komunikat dla wieku równego dokładnie 18 lat.

12 WYZWANIE 12 — TRZY MOŻLIWOŚCI

Rozpoznaj znak liczby

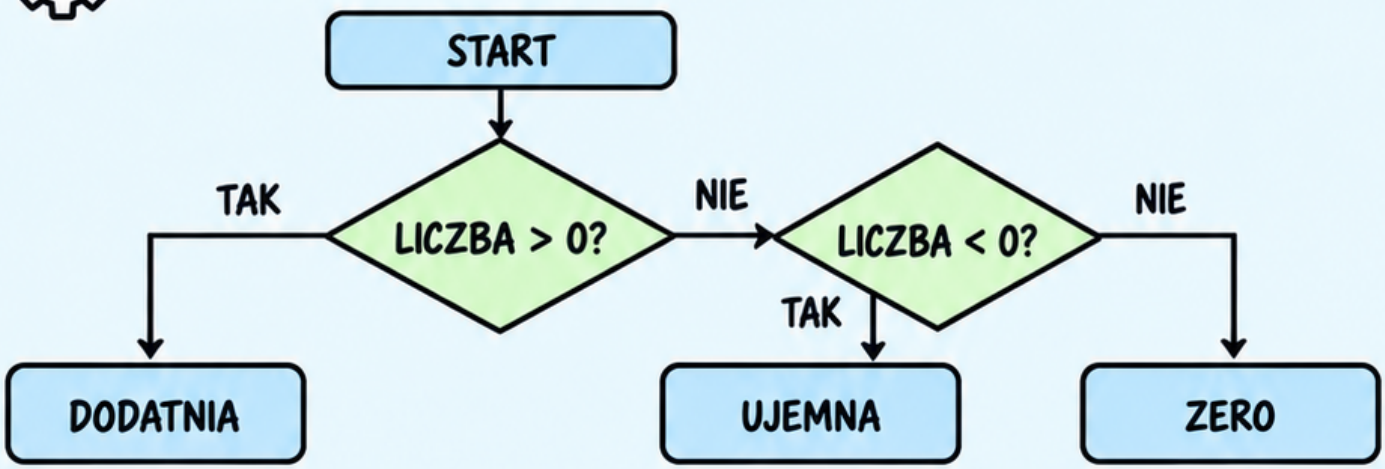
1. CEL WYZWANIA

Napisz program sprawdzający, czy liczba jest dodatnia, ujemna czy równa zero.

2. TWOJE ZADANIE

1. Pobrać liczbę od użytkownika.
2. Porównać ją z zerem.
3. Wyświetlić odpowiednią informację.

3. SCHEMAT DECYZJI



4. PRZYKŁADOWE WYNIKI

Liczba: **12**	Liczba: **−5**	Liczba: **0**
Wynik: **Liczba dodatnia.**	Wynik: **Liczba ujemna.**	Wynik: **Liczba równa zero.**

5. PODPOWIEDŹ

Wykorzystaj:

```
if else if else
```

6. WAŻNE OPERATORY

- `>` — większe niż
- `<` — mniejsze niż
- `==` — równe

7. SPRAWDŹ

Przetestuj liczby dodatnie, ujemne i zero.

8. BONUS

Dodaj komunikat informujący o wartości bezwzględnej liczby, wykorzystując `Math.abs()`.

13 WYZWANIE 13 – OBLICZ RABAT

Kiedy klient otrzymuje zniżkę?

1. CEL WYZWANIA

Napisz program, który przyznaje rabat w zależności od wartości zakupów.

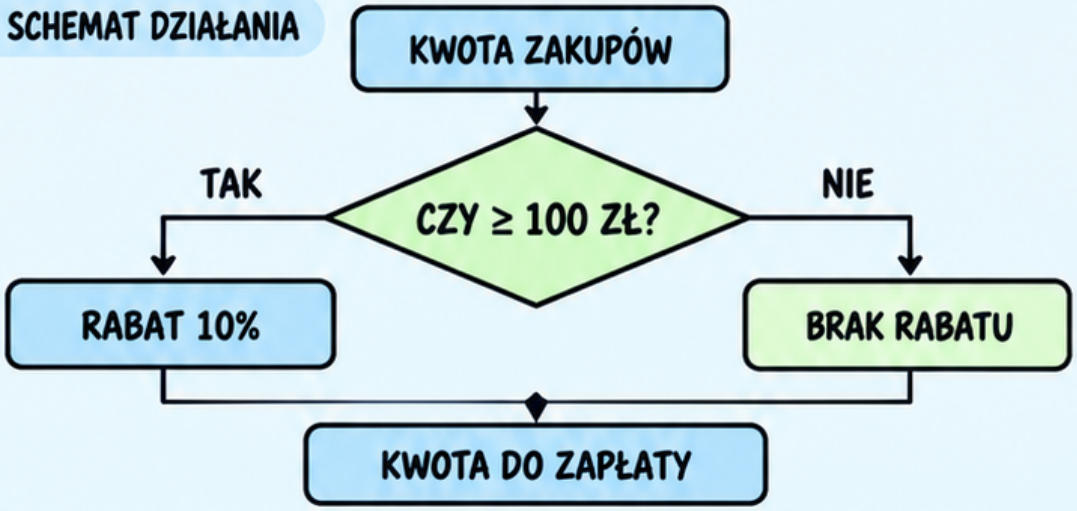
2. ZASADA

Jeśli zakupy wynoszą co najmniej 100 zł, klient otrzymuje 10% rabatu. W przeciwnym przypadku płaci pełną cenę.

3. TWOJE ZADANIE

1. Pobierz wartość zakupów.
2. Sprawdź, czy kwota wynosi co najmniej 100 zł.
3. Jeśli tak, oblicz rabat 10%.
4. Wyświetl ostateczną kwotę do zapłaty.

4. SCHEMAT DZIAŁANIA



5. PRZYKŁADOWE WYNIKI

Przykład 1

Zakupy: **150 zł**
Rabat: **15 zł**

Do zapłaty: 135 zł

Przykład 2

Zakupy: **80 zł**
Rabat: **0 zł**

Do zapłaty: 80 zł

6. PODPOWIEDŹ

Kwotę rabatu obliczysz, mnożąc cenę przez **0.10**.
Do wykonania różnych działań użyj `if` i `else`.

7. SPRAWDŹ

- [ ] Czy dla 100 zł przyznawany jest rabat?
- [ ] Czy dla 99 zł nie ma rabatu?
- [ ] Czy dla 200 zł klient płaci 180 zł?

8. BONUS

Dodaj drugi poziom promocji:
Zakupy od **300 zł** → rabat **15%**.

14 WYZWANIE 14 — PARZYSTA CZY NIEPARZYSTA?

Sprawdź, czy liczba jest parzysta

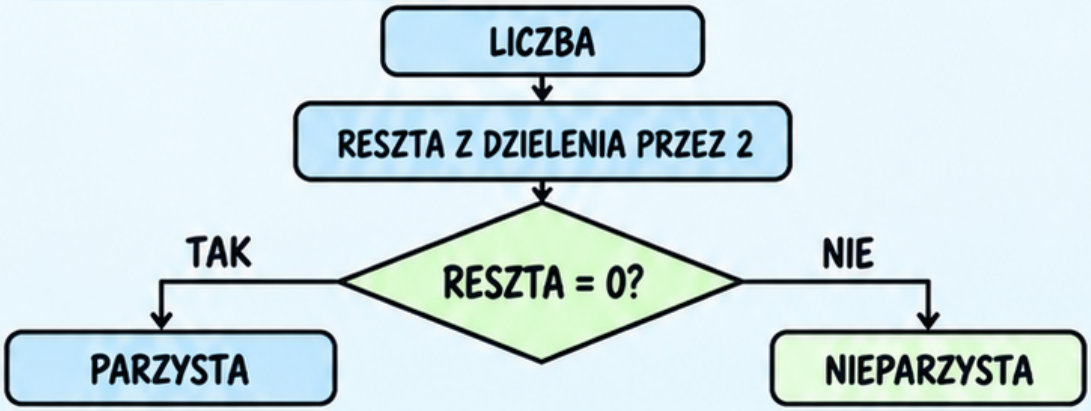
1. CEL WYZWANIA

Napisz program, który sprawdza, czy podana liczba całkowita jest parzysta, czy nieparzysta.

2. TWOJE ZADANIE

1. Poprosić użytkownika o liczbę całkowitą.
2. Obliczyć resztę z dzielenia przez 2.
3. Sprawdzić wynik za pomocą instrukcji `if`.
4. Wyświetlić odpowiedni komunikat.

3. SCHEMAT DZIAŁANIA



4. PRZYKŁAD DZIAŁANIA

PRZYKŁAD 1
Wejście: Podaj liczbę: **14**
Wynik: Liczba 14 jest parzysta.

PRZYKŁAD 2
Wejście: Podaj liczbę: **9**
Wynik: Liczba 9 jest nieparzysta.

PRZYKŁAD 3
Wejście: Podaj liczbę: **8**
Wynik: Liczba 8 jest parzysta.

PRZYKŁAD 4
Wejście: Podaj liczbę: **13**
Wynik: Liczba 13 jest nieparzysta.

5. PODPOWIEDŹ

Operator % zwraca resztę z dzielenia. Jeżeli liczba `% 2 == 0`, liczba jest parzysta. W przeciwnym przypadku jest nieparzysta.

6. SPRAWDŹ

- [ ] Czy 8 jest rozpoznawane jako parzyste?
- [ ] Czy 13 jest rozpoznawane jako nieparzyste?
- [ ] Czy 0 jest rozpoznawane jako parzyste?
- [ ] Czy program działa dla liczb ujemnych?

7. BONUS

Dodaj sprawdzenie, czy liczba jest również podzielna przez 5.

15 WYZWANIE 15 — ZNAJDŹ MAKSIMUM

Znajdź największą liczbę

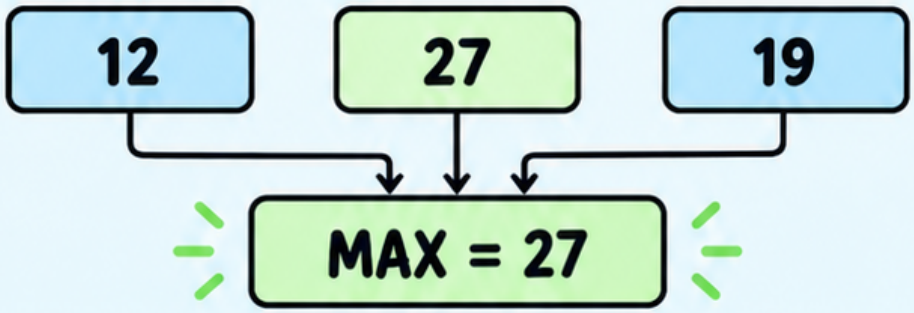
1. CEL WYZWANIA

Napisz program, który porównuje trzy liczby i wskazuje największą.

2. TWOJE ZADANIE

1. Pobierz trzy liczby od użytkownika.
2. Porównaj pierwszą liczbę z pozostałymi.
3. Sprawdź, czy druga jest największa.
4. Jeśli żadna z nich nie jest większa od pozostałych, wykorzystaj trzecią.
5. Wyświetl największą wartość.

3. ILUSTRACJA I WZÓR



4. PRZYKŁAD DZIAŁANIA

```
Pierwsza liczba: 12
Druga liczba: 27
Trzecia liczba: 19
Największa liczba: 27
```

5. PODPOWIEDŹ

Możesz wykorzystać instrukcje `if`, `else if` i `else`. Operator && oznacza logiczne „i”.

```
a >= b && a >= c
```

Oznacza to, że a jest większe lub równe obu pozostałym liczbom.

6. SPRAWDŹ

- [ ] 5, 9, 3 → 9
- [ ] 20, 10, 15 → 20
- [ ] 4, 4, 4 → 4
- [ ] -2, -7, -1 → -1

7. BONUS

Rozbuduj program, aby wskazywał również najmniejszą liczbę.

WAŻNE OPERATORY

>	<	>=	&&
większe od	mniejsze od	większe lub równe	logiczne „i” (oraz)

16 WYZWANIE 16 — SYSTEM OCENIANIA

Zamień punkty na ocenę

1. CEL WYZWANIA

Stwórz program, który zamienia wynik procentowy testu na ocenę.

2. ZASADY OCENIANIA

Wynik	Ocena
90–100%	5
75–89%	4
60–74%	3
40–59%	2
0–39%	1



3. TWOJE ZADANIE

1. Odczytać wynik procentowy.
2. Sprawdzić, do którego przedziału należy wynik.
3. Przypisać odpowiednią ocenę.
4. Wyświetlić wynik.

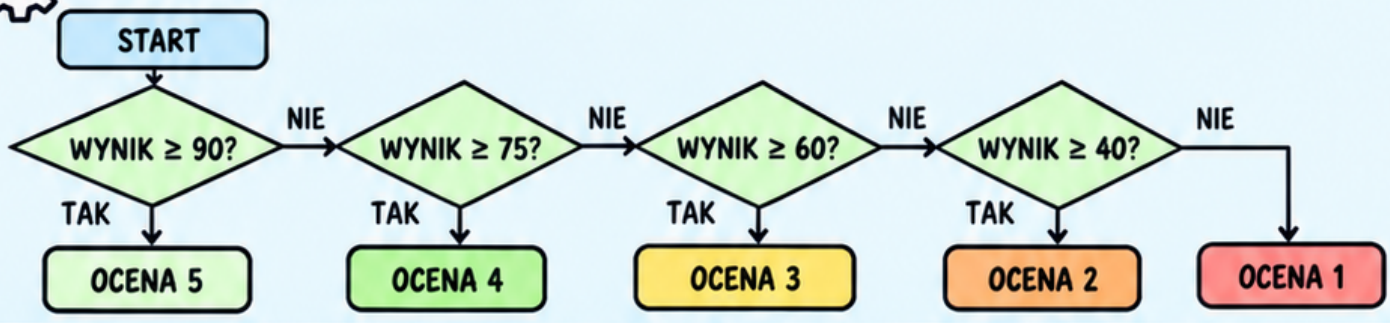
4. PRZYKŁAD DZIAŁANIA

```
Podaj wynik procentowy: 82
Twoja ocena: 4
```

5. PODPOWIEDŹ

Sprawdzaj warunki od najwyższego progu do najniższego. Pamiętaj, że instrukcja `else if` zostanie sprawdzona tylko wtedy, gdy wcześniejszy warunek nie został spełniony.

6. SCHEMAT DECYZJI



7. SPRAWDŹ

- [ ] 95% → 5
- [ ] 75% → 4
- [ ] 60% → 3
- [ ] 39% → 1

8. BONUS

Dodaj sprawdzenie, czy wartość mieści się w przedziale od 0 do 100. Dla wyniku spoza zakresu wyświetl komunikat: `Nieprawidłowy wynik.`

17 WYZWANIE 17 — KOD DOSTĘPU

Sprawdź poprawność kodu PIN

1. CEL WYZWANIA

Napisz prosty program, który sprawdza wpisany kod PIN.

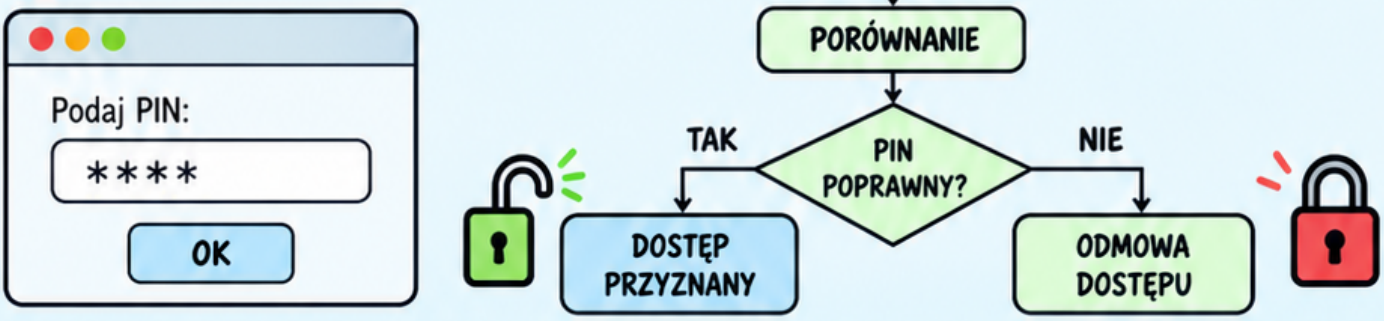
2. SYTUACJA

Program ma zapisany przykładowy kod: **1234**. Użytkownik musi podać prawidłową wartość, aby otrzymać komunikat o uzyskaniu dostępu.

3. TWOJE ZADANIE

1. Utwórz zmienną przechowującą kod.
2. Poproś użytkownika o wpisanie PIN-u.
3. Porównaj wpisaną wartość z zapisaną.
4. Wyświetl odpowiedni komunikat.

4. ILUSTRACJA I SCHEMAT



5. PRZYKŁADY

PRZYKŁAD 1

```
Podaj PIN: 1234
Dostęp przyznany.
```

PRZYKŁAD 2

```
Podaj PIN: 4321
Nieprawidłowy PIN.
```

6. PODPOWIEDŹ

Do porównania dwóch wartości typu `int` możesz wykorzystać operator `==`.

```
if (wpisanyPin == zapisanyPin)
```

7. WAŻNE

To wyłącznie ćwiczenie z instrukcji warunkowych. Prawdziwe aplikacje nie powinny przechowywać PIN-u na stałe w kodzie źródłowym.

8. SPRAWDŹ

- [ ] Czy poprawny PIN daje dostęp?
- [ ] Czy niepoprawny PIN jest odrzucany?
- [ ] Czy program działa bez błędów?

9. BONUS

Dodaj osobny komunikat, jeśli użytkownik wpisze liczbę spoza zakresu od 1000 do 9999.

18 WYZWANIE 18 — BILET DO KINA

Oblicz cenę biletu na podstawie wieku

1. CEL WYZWANIA

Napisz program, który ustala cenę biletu kinowego w zależności od wieku użytkownika.

2. ZASADY

Dzieci poniżej 7 lat	Osoby od 7 do 17 lat	Osoby od 18 do 64 lat	Osoby od 65 lat
0 zł	15 zł	25 zł	12 zł

3. TWOJE ZADANIE

1. Zapytaj użytkownika o wiek.
2. Sprawdź odpowiedni przedział.
3. Przypisz cenę biletu.
4. Wyświetl końcową kwotę.

4. PRZYKŁAD

```
Podaj wiek: 16
Cena biletu: 15 zł
```

5. PODPOWIEDŹ

Możesz połączyć dwa warunki za pomocą `&&`.
Przykład: `wiek >= 7 && wiek < 18`.
Ten warunek sprawdza, czy wiek mieści się w określonym przedziale.

6. SCHEMAT

WIEK → PRZEDZIAŁ WIEKOWY → CENA



7. SPRAWDŹ

- [ ] 5 lat → 0 zł
- [ ] 12 lat → 15 zł
- [ ] 30 lat → 25 zł
- [ ] 70 lat → 12 zł

8. BONUS

Dodaj zabezpieczenie przed wpisaniem wieku ujemnego.

19 WYZWANIE 19 — WYBIERZ DZIEŃ TYGODNIA

Poznaj instrukcję switch

1. CEL WYZWANIA

Napisz program, który wyświetla nazwę dnia tygodnia na podstawie liczby od 1 do 7.

2. ZASADY

1. — Poniedziałek
2. — Wtorek
3. — Środa
4. — Czwartek
5. — Piątek
6. — Sobota
7. — Niedziela

3. TWOJE ZADANIE

1. Poprosić o liczbę od 1 do 7.
2. Odczytać wpisaną liczbę.
3. Wykorzystać instrukcję `switch`.
4. Wyświetlić odpowiedni dzień tygodnia.

4. PRZYKŁAD DZIAŁANIA

```
Podaj numer dnia: 5
Dzień tygodnia: Piątek
```

5. PODPOWIEDŹ

```
switch (dzien) {
  case 1: System.out.println("Poniedziałek"); break;
  default: System.out.println("Nieprawidłowy numer");
}
```

Dodaj pozostałe przypadki.

6. ILUSTRACJA I SCHEMAT

1	2	3	4	5	6	7
Poniedziałek	Wtorek	Środa	Czwartek	Piątek	Sobota	Niedziela

LICZBA → SWITCH → WYBRANY PRZYPADEK

7. SPRAWDŹ

- [ ] Czy 1 oznacza poniedziałek?
- [ ] Czy 7 oznacza niedzielę?
- [ ] Czy 9 wyświetla informację o błędzie?

8. BONUS

Dodaj komunikat informujący, czy wybrany dzień jest częścią weekendu.

20 WYZWANIE 20 — MINI KALKULATOR

Stwórz własny kalkulator

1. CEL WYZWANIA

Zbuduj program, który wykonuje jedno z czterech działań matematycznych wybranych przez użytkownika.

2. DOSTĘPNE DZIAŁANIA

- `+` Dodawanie
- `-` Odejmowanie
- `*` Mnożenie
- `/` Dzielenie

3. TWOJE ZADANIE

1. Pobierz pierwszą liczbę.
2. Pobierz drugą liczbę.
3. Zapytaj o symbol działania.
4. Wykorzystaj switch.
5. Oblicz i wyświetl wynik.

4. PRZYKŁAD DZIAŁANIA

```
Pierwsza liczba: 20
Druga liczba: 4
Wybierz działanie: /
Wynik: 5.0
```

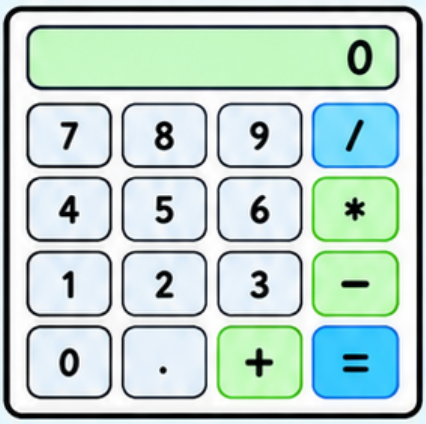
5. PODPOWIEDŹ

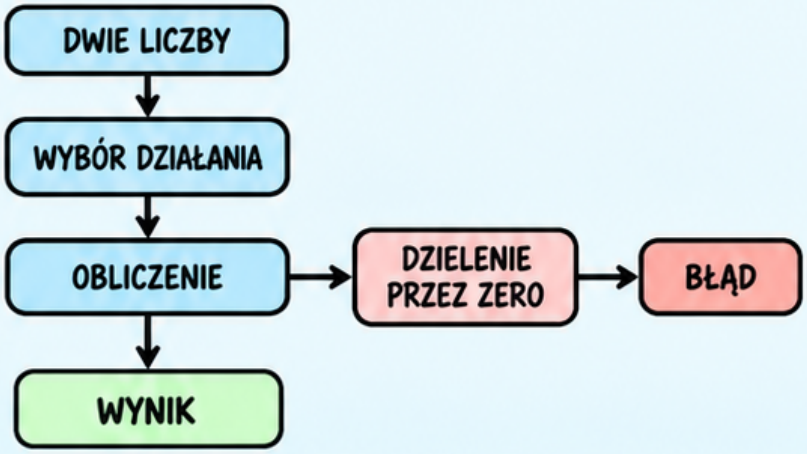
Do przechowywania liczb użyj typu `double`. Do odczytania symbolu działania jako tekstu możesz wykorzystać `scanner.next()`. Instrukcja `switch` pozwala wybrać odpowiednią operację.

6. WAŻNE

Przed dzieleniem sprawdź, czy druga liczba nie jest równa zero. Jeśli użytkownik wybierze nieznany operator, wyświetl komunikat o nieprawidłowym działaniu.

7. ILUSTRACJA I SCHEMAT





8. SPRAWDŹ

- [ ] `8 + 2 → 10`
- [ ] `8 - 2 → 6`
- [ ] `8 * 2 → 16`
- [ ] `8 / 2 → 4`
- [ ] `8 / 0 →` komunikat o błędzie

9. BONUS

Dodaj możliwość obliczenia reszty z dzielenia za pomocą `%`.

02 MODUŁ 02 – SPRAWDŹ SWOJE UMIEJĘTNOŚCI

Sprawdź, czy potrafisz podejmować decyzje w kodzie

TWÓJ POSTĘP

Zaznacz wykonane wyzwania z modułu 02.

11 12 13 14 15 16 17 18 19 20

CO JUŻ POTRAFISZ?

Sprawdź, co już potrafisz. Zaznacz wykonane punkty.

- [ ] 1 Potrafię używać instrukcji if.
- [ ] 2 Rozumiem działanie else.
- [ ] 3 Umiem tworzyć warunki z else if.
- [ ] 4 Potrafię porównywać liczby.
- [ ] 5 Rozumiem operatory `>`, `<`, `>=`, `<=` i `==`.
- [ ] 6 Potrafię wykorzystać operator `&&`.
- [ ] 7 Umiem sprawdzać zakresy liczb.
- [ ] 8 Potrafię korzystać z instrukcji `switch`.
- [ ] 9 Umiem wykrywać nieprawidłowe dane.
- [ ] 10 Potrafię stworzyć prosty kalkulator.

PODPOWIEDŹ

Wykorzystaj operator `&&`, aby połączyć dwa warunki.

PYTANIA DO REFLEKSJI

- [ ] Kiedy warto używać if/else?
- [ ] Czym różni się switch od wielu instrukcji if?
- [ ] Jak sprawdzić więcej niż jeden warunek?

MINI-SPRAWDZIAN

PROSTY SYSTEM REZERWACJI

Napisz program, który ustala możliwość rezerwacji miejsca na podstawie wieku i liczby dostępnych miejsc.

ZASADY

- Minimalny wiek uczestnika: 16 lat.
- Rezerwacja jest możliwa tylko wtedy, gdy liczba wolnych miejsc jest większa od zera.

1. Odczytać wiek.
2. Odczytać liczbę wolnych miejsc.
3. Sprawdzić oba warunki.
4. Wyświetlić decyzję.

PRZYKŁAD

```
Wiek: 18
Wolne miejsca: 3
Wynik: Rezerwacja możliwa.
```

MODUŁ 02 – UKOŃCZONY

- [ ] Zaznacz, gdy ukończysz wszystkie wyzwania z modułu 02.

Świetna robota!
Każda decyzja w kodzie to krok do lepszego programowania.

MODUŁ 03

PĘTLE, NAPISY I TABLICE

Naucz program powtarzać działania i pracować z wieloma danymi.



CZEGO SIĘ NAUCZYSZ?

1. Wykorzystywać pętlę for.
2. Korzystać z pętli while.
3. Powtarzać operacje określoną liczbę razy.
4. Pracować z napisami typu `String`.
5. Tworzyć i odczytywać tablice.
6. Przetwarzać wiele danych w jednym programie.

GŁÓWNA IDEA

Zamiast pisać tę samą instrukcję wiele razy, możesz wykorzystać pętlę.

PRZYKŁAD KONCEPCYJNY

Bez pętli

```
System.out.println("Cześć");
System.out.println("Cześć");
System.out.println("Cześć");
```

Z pętlą

```
for (int i = 0; i < 3; i++) {
  System.out.println("Cześć");
}
```

WYNIK

```
Cześć
Cześć
Cześć
```

Mniej powtarzania kodu. Więcej możliwości.

21 WYZWANIE 21 — LICZNIK LICZB

Wyświetl liczby za pomocą pętli for

1. CEL WYZWANIA

Napisz program, który wyświetla liczby od 1 do 10 za pomocą pętli `for`.

2. TWOJE ZADANIE

1. Utworzyć pętlę `for`.
2. Rozpocząć liczenie od 1.
3. Powtarzać instrukcję do osiągnięcia 10.
4. Wyświetlać każdą liczbę w osobnej linii.

3. PODPOWIEDŹ

```
for (int i = 1; i <= 10; i++) {
    System.out.println(i);
}
```

4. JAK DZIAŁA PĘTLA?

1. START

`int i = 1`

Ustawienie wartości początkowej.

2. WARUNEK

`i <= 10`

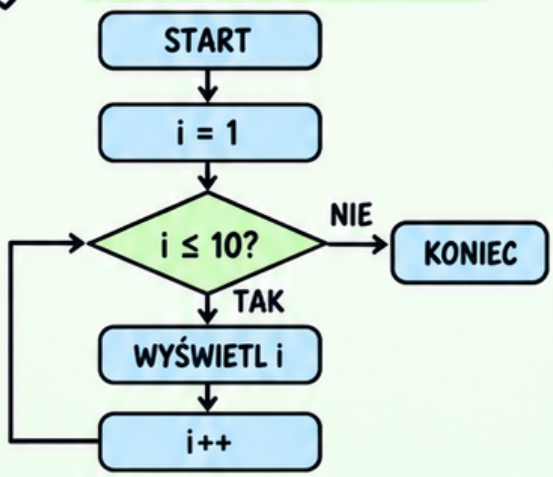
Pętla działa, dopóki warunek jest prawdziwy.

3. ZMIANA

`i++`

Zwiększenie licznika o jeden po każdym wykonaniu.

5. SCHEMAT DZIAŁANIA



6. PRZYKŁAD DZIAŁANIA

```
1
2
3
4
5
6
7
8
9
10
```

7. SPRAWDŹ

- [ ] Czy program wyświetla dokładnie dziesięć liczb?
- [ ] Czy zaczyna od 1?
- [ ] Czy kończy na 10?
- [ ] Czy liczby pojawiają się we właściwej kolejności?

8. BONUS

Zmodyfikuj program, aby:

- Wyświetlał liczby od 10 do 1.
- Wyświetlał tylko liczby parzyste od 2 do 20.

22 WYZWANIE 22 — TABLICZKA MNOŻENIA

Jedna pętla, dziesięć obliczeń.

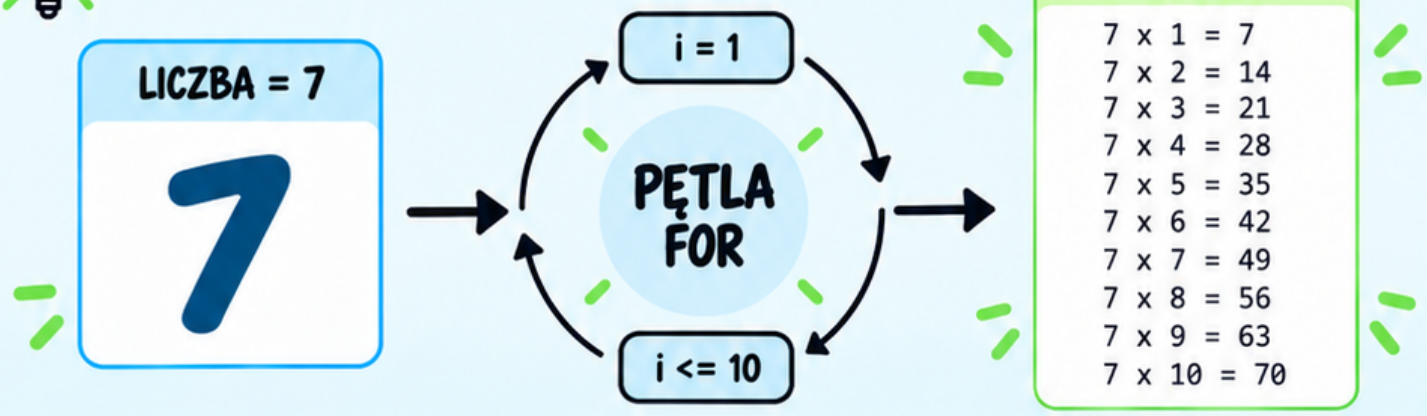
1. CEL WYZWANIA

Napisz program, który wyświetla tabliczkę mnożenia dla liczby podanej przez użytkownika.

2. TWOJE ZADANIE

1. Poprosić użytkownika o liczbę całkowitą.
2. Zapisać ją w zmiennej.
3. Uruchomić pętlę for od 1 do 10.
4. Pomnożyć podaną liczbę przez aktualny licznik.
5. Wyświetlić każde działanie i jego wynik.

TABLICZKA MNOŻENIA W PRAKTYCE



3. PRZYKŁAD DZIAŁANIA

```
Wejście: Podaj liczbę: 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

4. PODPOWIEDŹ

```
for (int i = 1; i <= 10; i++) {
    int wynik = liczba * i;
    System.out.println( liczba + " x " + i + " = " + wynik );
}
```

Zmienna `i` będzie kolejno przyjmowała wartości od 1 do 10.

5. SPRAWDŹ

- [ ] Czy program wyświetla dokładnie dziesięć działań?
- [ ] Czy wyniki mnożenia są prawidłowe?
- [ ] Czy działa dla liczby 5?
- [ ] Czy działa dla liczby 0?

6. BONUS

Rozbuduj program tak, aby użytkownik mógł zdecydować, do jakiej liczby ma zostać wygenerowana tabliczka mnożenia.

SCHEMAT DZIAŁANIA



23 WYZWANIE 23 — SUMA LICZB

Oblicz sumę liczb za pomocą pętli

1. CEL WYZWANIA

Napisz program, który oblicza sumę wszystkich liczb całkowitych od 1 do liczby podanej przez użytkownika.

2. TWOJE ZADANIE

1. Poprosić użytkownika o dodatnią liczbę całkowitą.
2. Utworzyć zmienną suma z wartością początkową 0.
3. Wykorzystać pętlę for.
4. Dodawać kolejne liczby do zmiennej suma.
5. Wyświetlić wynik końcowy.

3. PRZYKŁAD DZIAŁANIA

```
Wejście:     Podaj liczbę: 5
Obliczenie.  1 + 2 + 3 + 4 + 5 = 15
Wynik:       Suma liczb od 1 do 5
             wynosi 15.
```

4. PODPOWIEDŹ

```
int suma = 0;
for (int i = 1; i <= n; i++) {
    suma += i;
}
```

Zapis `suma += i` oznacza to samo co:
`suma = suma + i;`

5. JAK DZIAŁA AKUMULATOR?

Start: suma = 0	Krok 1: suma = 1	Krok 2: suma = 3	Krok 3: suma = 6	Krok 4: suma = 10	Krok 5: suma = 15
0 →	1 →	3 →	6 →	10 →	15

AKUMULATOR — zmienna przechowująca rosnący wynik obliczeń.

6. SPRAWDŹ

- [ ] Czy dla 3 otrzymujesz 6?
- [ ] Czy dla 5 otrzymujesz 15?
- [ ] Czy dla 10 otrzymujesz 55?
- [ ] Czy program odrzuca wartości mniejsze od 1?

7. BONUS

Oblicz osobno sumę liczb parzystych od 1 do n.

24 WYZWANIE 24 — ODLICZANIE

Stwórz odliczanie za pomocą while

1. CEL WYZWANIA

Napisz program, który odlicza od wybranej liczby do 1, a następnie wyświetla komunikat startowy.

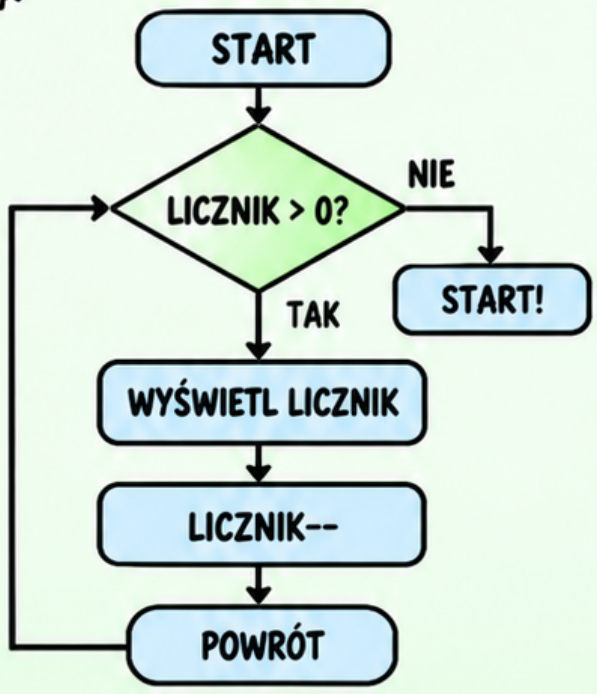
2. TWOJE ZADANIE

1. Poprosić użytkownika o dodatnią liczbę.
2. Zapisać tę wartość w zmiennej.
3. Utworzyć pętlę `while`.
4. Wyświetlać aktualną wartość licznika.
5. Zmniejszać licznik o jeden.
6. Po zakończeniu odliczania wyświetlić napis START!.

3. PRZYKŁAD DZIAŁANIA

```
Podaj liczbę: 5
5
4
3
2
1
START!
```

SCHEMAT DZIAŁANIA



4. PODPOWIEDŹ

```
while (licznik > 0) {
    System.out.println(licznik);
    licznik--;
}
System.out.println("START!");
```

5. JAK DZIAŁA WHILE?

Warunek jest sprawdzany przed każdym wykonaniem pętli. Jeśli jest prawdziwy, instrukcje zostają wykonane. Jeśli jest fałszywy, program opuszcza pętlę.

6. WAŻNE

Pamiętaj o zmianie wartości licznika. Bez `licznik--` pętla może działać bez końca.

8. BONUS

Zmień program tak, aby wyświetlał wyłącznie liczby parzyste podczas odliczania.

7. SPRAWDŹ

- [ ] Czy dla 3 otrzymujesz 3, 2, 1?
- [ ] Czy komunikat START! pojawia się tylko raz?
- [ ] Czy licznik zmniejsza się prawidłowo?
- [ ] Czy program obsługuje wartość 1?

25 WYZWANIE 25 — SPRAWDŹ DANE

Powtarzaj pytanie, aż użytkownik poda poprawną wartość

1. CEL WYZWANIA

Napisz program, który prosi użytkownika o liczbę od 1 do 10 i powtarza pytanie, dopóki użytkownik nie poda poprawnej wartości.

2. TWOJE ZADANIE

1. Wyświetlić prośbę o podanie liczby.
2. Odczytać wpisaną wartość całkowitą.
3. Sprawdzić, czy należy do przedziału od 1 do 10.
4. Jeśli nie, wyświetlić komunikat o błędzie.
5. Ponownie poprosić o liczbę.
6. Zakończyć powtarzanie po otrzymaniu poprawnej wartości.

3. PRZYKŁAD DZIAŁANIA

```
Podaj liczbę od 1 do 10: 15
Nieprawidłowa liczba.
Podaj liczbę od 1 do 10: -2
Nieprawidłowa liczba.
Podaj liczbę od 1 do 10: 7
Dziękuję! Wybrano liczbę 7.
```

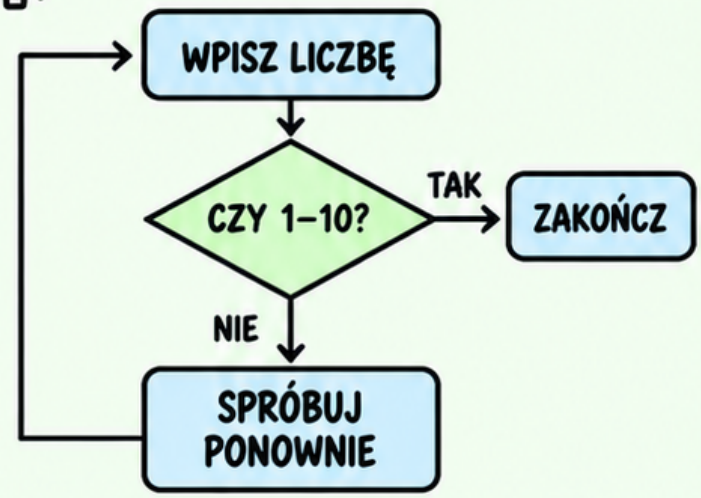
4. PODPOWIEDŹ

```java
int liczba;
do {
   System.out.print( "Podaj liczbę od 1 do 10: ");
   liczba = scanner.nextInt();
} while (liczba < 1 || liczba > 10);
```

5. CO OZNACZA ||?

Operator `||` oznacza logiczne „lub”. Warunek jest prawdziwy, jeśli przynajmniej jedna jego część jest prawdziwa.

6. SCHEMAT DZIAŁANIA



7. SPRAWDŹ

- [ ] Czy liczba 0 jest odrzucana?
- [ ] Czy liczba 11 jest odrzucana?
- [ ] Czy liczba 5 zostaje zaakceptowana?
- [ ] Czy program poprawnie powtarza pytanie?

8. BONUS

Dodaj licznik informujący, ile prób potrzebował użytkownik.

DO-WHILE — instrukcje wykonują się przynajmniej jeden raz.

26 WYZWANIE 26 — POZNAJ STRING

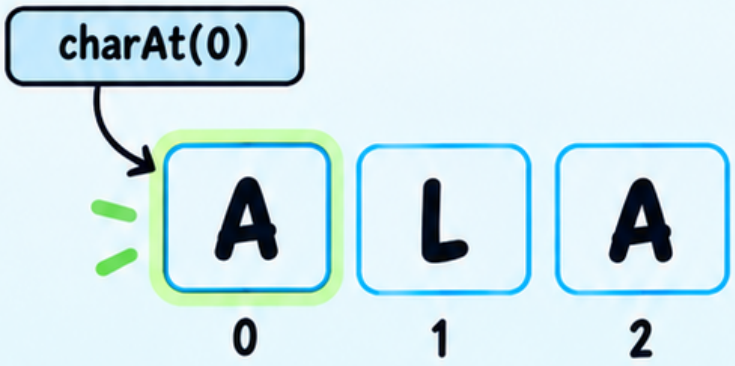
Sprawdź długość tekstu

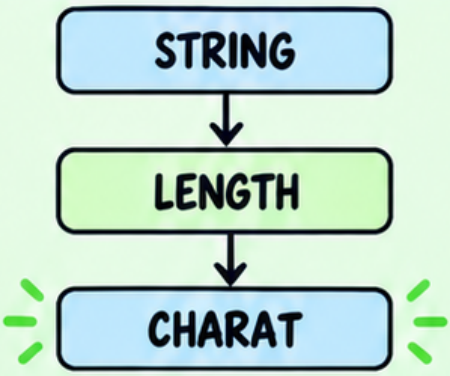
1. CEL WYZWANIA

Napisz program, który analizuje imię wpisane przez użytkownika.

2. TWOJE ZADANIE

1. Poprosić użytkownika o imię.
2. Odczytać tekst.
3. Usunąć zbędne spacje z początku i końca.
4. Obliczyć liczbę znaków.
5. Wyświetlić pierwszą literę imienia.





3. PRZYKŁAD DZIAŁANIA

```
Podaj imię: Aleksandra
Liczba znaków: 10
Pierwsza litera: A
```

4. PODPOWIEDŹ

Przydatne metody:

- `trim()` — usuwa spacje z początku i końca tekstu.
- `length()` — zwraca długość napisu.
- `charAt(0)` — zwraca pierwszy znak.

```
String imie = scanner.nextLine().trim();
if (!imie.isEmpty()) {
     System.out.println(imie.length());
     System.out.println(imie.charAt(0));
}
```

5. WAŻNE

Przed użyciem `charAt(0)` sprawdź, czy napis nie jest pusty. Pierwszy znak ma indeks 0, a nie 1.

6. SPRAWDŹ

- [ ] Czy dla Ala długość wynosi 3?
- [ ] Czy dla Jan pierwszą literą jest J?
- [ ] Czy program poprawnie usuwa zbędne spacje?
- [ ] Czy obsługuje pusty tekst bez błędu?

7. BONUS

Dodaj informację o ostatniej literze imienia.

Wykorzystaj indeks `imie.length() - 1`.

27 WYZWANIE 27 — LICZNIK SAMOGŁOSEK

Policz samogłoski w podanym słowie

1. CEL WYZWANIA

Napisz program, który sprawdza, ile samogłosek znajduje się w podanym słowie.

2. TWOJE ZADANIE

1. Poprosić użytkownika o słowo.
2. Zamienić litery na małe.
3. Przejść przez wszystkie znaki za pomocą pętli.
4. Sprawdzić, czy dany znak jest samogłoską.
5. Zwiększyć licznik po znalezieniu samogłoski.
6. Wyświetlić wynik końcowy.

3. SAMOGŁOSKI W JĘZYKU POLSKIM

a, ą, e, ę, i, o, ó, u, y

J A V A

SŁOWO → ZNAKI → SPRAWDŹ SAMOGŁOSKĘ → LICZNIK

PRZYDATNE METODY

`length()`
Zwraca długość tekstu.

`charAt()`
Zwraca znak o podanym indeksie.

`indexOf()`
Sprawdza, czy znak znajduje się w tekście (zwraca indeks).

4. PODPOWIEDŹ

```
String tekst = slowo.toLowerCase();
String samogloski = "aąeęioóuy";
int licznik = 0;
for (int i = 0; i < tekst.length(); i++) {
    char znak = tekst.charAt(i);
    if (samogloski.indexOf(znak) >= 0) {
        licznik++;
    }
}
```

5. JAK TO DZIAŁA?

Pętla odczytuje każdy znak tekstu. `indexOf(znak)` sprawdza, czy znak występuje na liście samogłosek. Jeśli tak, licznik zwiększa się o jeden.

6. PRZYKŁAD DZIAŁANIA

```
Podaj słowo: Java
Liczba samogłosek: 2
```

7. SPRAWDŹ

- Java → 2
- kot → 1
- programowanie → 6
- xyz → 1

8. BONUS

Rozbuduj program, aby obliczał również liczbę spółgłosek w słowie składającym się wyłącznie z polskich liter.

28 WYZWANIE 28 — PRACUJ Z TABLICĄ

Przechowuj wiele wartości w jednej tablicy

1. CEL WYZWANIA

Napisz program, który przechowuje temperatury z pięciu dni i wyświetla wszystkie wartości.

2. TWOJE ZADANIE

1. Utworzyć tablicę pięciu temperatur.
2. Przechować w niej wartości liczbowe.
3. Wykorzystać pętlę `for`.
4. Odczytać każdy element tablicy.
5. Wyświetlić temperaturę wraz z numerem dnia.

3. PRZYKŁADOWE DANE

```
double[] temperatury = { 18, 20, 22, 19, 21 };
```

4. OCZEKIWANY WYNIK

```
Dzień 1: 18.0°C
Dzień 2: 20.0°C
Dzień 3: 22.0°C
Dzień 4: 19.0°C
Dzień 5: 21.0°C
```

5. JAK DZIAŁA TABLICA?

Tablica przechowuje wiele wartości tego samego typu. Indeks pierwszego elementu wynosi 0. Właściwość `length` określa liczbę elementów tablicy.

INDEKS →	0	1	2	3	4
WARTOŚĆ →	18	20	22	19	21

TABLICA → INDEKS → WARTOŚĆ

`temperatury[2] = 22`

6. PODPOWIEDŹ

```
for (int i = 0; i < temperatury.length; i++) {
    System.out.println( "Dzień " + (i + 1) + ": " + temperatury[i] + "°C" );
}
```

7. SPRAWDŹ

- [ ] Czy program wyświetla pięć temperatur?
- [ ] Czy pierwszy element ma indeks 0?
- [ ] Czy ostatni element ma indeks 4?
- [ ] Czy wszystkie wartości są poprawne?

8. BONUS

Dodaj obliczenie sumy temperatur ze wszystkich dni.

29 WYZWANIE 29 — ZNAJDŹ MAKSIMUM

Znajdź największą wartość w tablicy

1. CEL WYZWANIA

Napisz program, który znajduje największą liczbę w tablicy.

2. TWOJE ZADANIE

1. Utworzyć niepustą tablicę liczb całkowitych.
2. Przyjąć pierwszy element jako aktualne maksimum.
3. Przejść przez pozostałe elementy pętlą.
4. Porównywać każdą liczbę z aktualnym maksimum.
5. Aktualizować maksimum, gdy znajdziesz większą liczbę.
6. Wyświetlić wynik.

3. PRZYKŁADOWA TABLICA

```
int[] liczby = {12, 7, 25, 9, 18};
```

4. OCZEKIWANY WYNIK

```
Największa liczba: 25
```

5. PODPOWIEDŹ

```
int maksimum = liczby[0];
for (int i = 1; i < liczby.length; i++) {
    if (liczby[i] > maksimum) {
        maksimum = liczby[i];
    }
}
```

6. JAK DZIAŁA ALGORYTM?

1. Początkowe maksimum: 12
2. Porównaj z 7 → zostaje 12
3. Porównaj z 25 → nowe maksimum 25
4. Porównaj z 9 → zostaje 25
5. Porównaj z 18 → zostaje 25
6. Wynik: 25

TABLICA I KOLEJNE PORÓWNANIA

12	7	25	9	18

Wartość aktualnego maksimum po każdej iteracji:

12 → 12 → 25 → 25 → 25

AKTUALNE MAKSIMUM

Podczas przechodzenia przez tablicę przechowujemy aktualnie największą wartość. Jeśli napotkamy większą liczbę, aktualizujemy maksimum.

9 > 25

nowe maksimum

7. SPRAWDŹ

- [ ] Czy program działa dla tablicy z jedną liczbą?
- [ ] Czy znajduje maksimum wśród liczb ujemnych?
- [ ] Czy poprawnie obsługuje powtarzające się wartości?
- [ ] Czy działa, jeśli największa liczba jest na końcu?

8. BONUS

Dodaj wyszukiwanie najmniejszej liczby w tej samej tablicy.

30 WYZWANIE 30 — ANALIZA AKTYWNOŚCI

Przeanalizuj aktywność z pięciu dni

1. CEL WYZWANIA

Napisz program analizujący liczbę kroków wykonanych podczas pięciu dni.

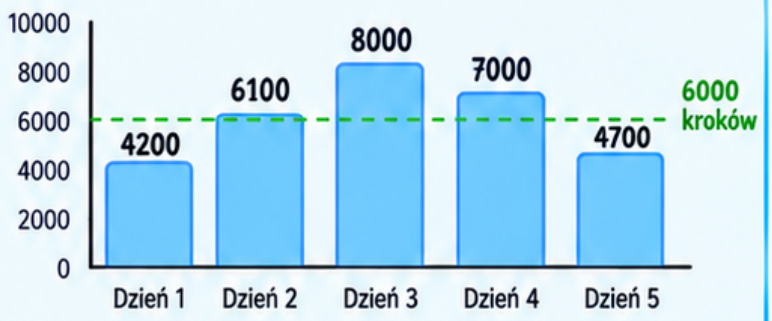
2. SYTUACJA

1. Dzień 1: 4200 kroków.
2. Dzień 2: 6100 kroków.
3. Dzień 3: 8000 kroków.
4. Dzień 4: 7000 kroków.
5. Dzień 5: 4700 kroków.

3. TWOJE ZADANIE

1. Utworzyć tablicę z pięcioma wynikami.
2. Obliczyć sumę wszystkich kroków.
3. Obliczyć średnią dzienną.
4. Znaleźć największy wynik.
5. Policzyć dni, w których wykonano co najmniej 6000 kroków.
6. Wyświetlić podsumowanie.

ANALIZA AKTYWNOŚCI



4. PRZYKŁADOWA TABLICA

```
int[] kroki = { 4200, 6100, 8000, 7000, 4700 };
```

5. OCZEKIWANY WYNIK

```
Suma kroków: 30000
Średnia dzienna: 6000.0
Największy wynik: 8000
Dni z wynikiem >= 6000: 3
```

6. PODPOWIEDŹ

Do obliczeń wykorzystaj:

- Pętlę `for`.
- Zmienną sumą.
- Zmienną `maksimum`.
- Warunek `if`.
- Zmienną zliczającą dni.

7. OBLICZENIE ŚREDNIEJ

```
double srednia = (double) suma / kroki.length;
```

Rzutowanie na `double` zapewni wynik zmiennoprzecinkowy.

8. SPRAWDŹ

- [ ] Czy suma wynosi 30000?
- [ ] Czy średnia wynosi 6000?
- [ ] Czy maksimum wynosi 8000?
- [ ] Czy trzy dni spełniają wymagany próg?

9. BONUS

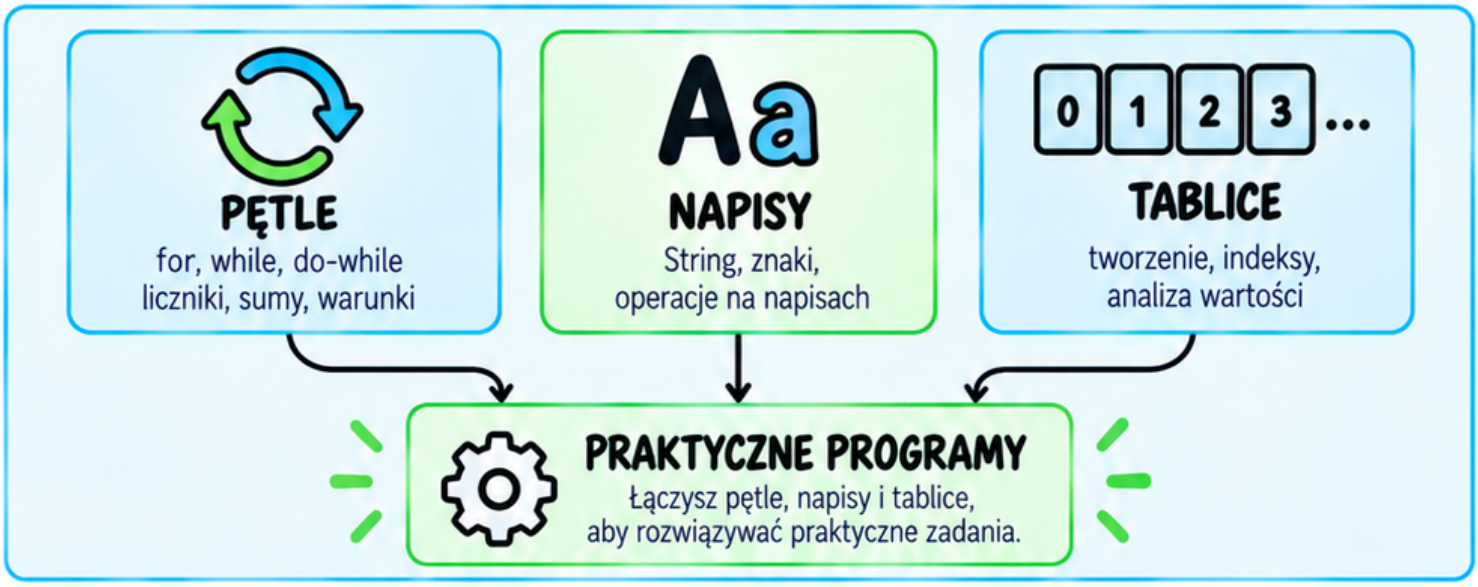
Dodaj obliczenie liczby dni, w których wynik był niższy od średniej.

PODSUMOWANIE (dla podanych danych)

SUMA	ŚREDNIA	MAKSIMUM	DNI ≥ 6000
30000	6000	8000	3

03 MODUŁ 03 — SPRAWDŹ SWOJE UMIEJĘTNOŚCI

Sprawdź swoje umiejętności pracy z pętlami i tablicami



CO JUŻ POTRAFISZ?

Sprawdź, co już potrafisz. Zaznacz wykonane punkty.

- [ ] 1 Potrafię korzystać z pętli for.
- [ ] 2 Rozumiem działanie pętli while.
- [ ] 3 Umiem wykorzystywać do-while.
- [ ] 4 Potrafię tworzyć liczniki.
- [ ] 5 Umiem sumować wartości w pętli.
- [ ] 6 Rozumiem podstawowe metody klasy `String`.
- [ ] 7 Potrafię odczytywać pojedyncze znaki.
- [ ] 8 Umiem tworzyć tablice.
- [ ] 9 Potrafię korzystać z indeksów.
- [ ] 10 Umiem analizować wartości zapisane w tablicy.

PODPOWIEDŹ

Wykorzystaj jedną pętlę do przeanalizowania elementów tablicy. Możesz aktualizować sumę, maksimum i licznik w trakcie tej samej pętli.

PYTANIA DO REFLEKSJI

- [ ] Czym różni się pętla `for` od while?
- [ ] Dlaczego indeksowanie tablicy zaczyna się od 0?
- [ ] Jak obliczyć średnią z elementów tablicy?
- [ ] Co może się stać, jeśli warunek zakończenia pętli nigdy nie zostanie spełniony?

MINI-SPRAWDZIAN

ZADANIE: ANALIZA WYNIKÓW

Napisz program, który analizuje wyniki pięciu testów.

```
int[] wyniki = {6, 8, 5, 9, 7};
```

PROGRAM POWINIEN:

1. Obliczyć sumę wyników.
2. Obliczyć średnią.
3. Znaleźć największy wynik.
4. Policzyć, ile wyników wynosi co najmniej 7.
5. Wyświetlić podsumowanie.

OCZEKIWANE REZULTATY

```
Suma: 35
Średnia: 7.0
Największy wynik: 9
Wyników >= 7: 3
```

TWÓJ POSTĘP

Zaznacz wykonane wyzwania z modułu 03.

21 22 23 24 25 26 27 28 29 30

MODUŁ 03 — UKOŃCZONY

- [ ]

Potrafisz już analizować wiele danych i automatyzować powtarzalne działania. To ważny krok w nauce programowania!

04 MODUŁ 04
METODY, KOLEKCJE I OBIEKTY

Twórz uporządkowany kod i buduj własne struktury danych.

WPROWADZENIE DO MODUŁU

W tym module przechodzisz od małych, prostych programów do bardziej uporządkowanych projektów. Poznasz metody, które pozwalają dzielić kod na mniejsze części, kolekcje do przechowywania wielu danych oraz klasy i obiekty, które łączą dane i zachowania w jedną spójną całość.

CZEGO SIĘ NAUCZYSZ?

1. **METODY**
   Dziel program na mniejsze części.
2. **PARAMETRY**
   Przekazuj dane do metod.
3. **WARTOŚCI ZWRACANE**
   Odbieraj wyniki działania metod.
4. **KOLEKCJE**
   Przechowuj wiele danych i zarządzaj nimi.
5. **OBIEKTY**
   Łącz dane i zachowania w jednej strukturze.

GŁÓWNA IDEA

Dobry program to nie tylko kod, który działa. To również kod, który łatwo zrozumieć, poprawić i rozbudować.

PRZYKŁAD KONCEPCYJNY

DUŻY PROGRAM
↓
MNIEJSZE METODY
↓
UPORZĄDKOWANY KOD

JAK TO WSZYSTKO ŁĄCZY SIĘ W PRAKTYCE?

METODY
Metody pozwalają dzielić kod na mniejsze i czytelne części.

Przykładowa klasa w Javie

```
import java.util.ArrayList;
import java.util.List;

public class Uczen {
    private String imie;
    private List<Integer> oceny;

    public Uczen(String imie) {
        this.imie = imie;
        this.oceny = new ArrayList<>();
    }
    public void dodajOcene(int ocena) {
        oceny.add(ocena);
    }
    public double srednia() {
        if (oceny.isEmpty()) return 0;
        int suma = 0;
        for (int o : oceny) suma += o;
        return (double) suma / oceny.size();
    }
}
```

KOLEKCJE
Kolekcje umożliwiają przechowywanie wielu danych, np. ocen.

OBIEKTY
Obiekty łączą dane (pola) i zachowania (metody) w jednej strukturze.

Teraz nauczysz się tworzyć programy, które łatwiej rozwijać i utrzymywać.

31 WYZWANIE 31 — PIERWSZA METODA

Wydziel fragment kodu do osobnej metody

1. CEL WYZWANIA

Napisz program, który korzysta z osobnej metody do wyświetlania powitania.

2. TWOJE ZADANIE

1. Utworzyć metodę `powitanie()`.
2. Umieścić w niej komunikat powitalny.
3. Wywołać metodę z funkcji `main`.
4. Wywołać tę samą metodę trzy razy.

3. PRZYKŁAD DZIAŁANIA

```
Witaj w świecie Javy!
Witaj w świecie Javy!
Witaj w świecie Javy!
```

5. PODPOWIEDŹ

```
public class Main {
   static void powitanie() {
      System.out.println("Witaj w świecie Javy!");
   }

   public static void main(String[] args) {
      powitanie();
      powitanie();
      powitanie();
   }
}
```

5. JAK TO DZIAŁA?

METODA

To nazwany fragment kodu, który można uruchomić wielokrotnie.

VOID

Oznacza, że metoda nie zwraca wartości.

WYWOŁANIE

Zapis `powitanie();` uruchamia instrukcje znajdujące się wewnątrz metody.



6. SPRAWDŹ

- [ ] Czy program wyświetla trzy powitania?
- [ ] Czy kod powitania znajduje się w osobnej metodzie?
- [ ] Czy metoda jest wywoływana z `main`?

7. BONUS

Utwórz drugą metodę `pozegnanie()`, która wyświetli komunikat:

```
Do zobaczenia przy kolejnym wyzwaniu!
```

32 WYZWANIE 32 — OBLICZ KWADRAT LICZBY

Przekaż liczbę do metody i odbierz wynik

1. CEL WYZWANIA

Napisz metodę, która otrzymuje liczbę całkowitą i zwraca jej kwadrat.

2. TWOJE ZADANIE

1. Utworzyć metodę `kwadrat`.
2. Dodać parametr typu `int`.
3. Pomnożyć liczbę przez samą siebie.
4. Zwrócić wynik za pomocą `return`.
5. Wywołać metodę z `main`.

3. PRZYKŁAD DZIAŁANIA

```
Podaj liczbę: 6
Kwadrat liczby: 36
```

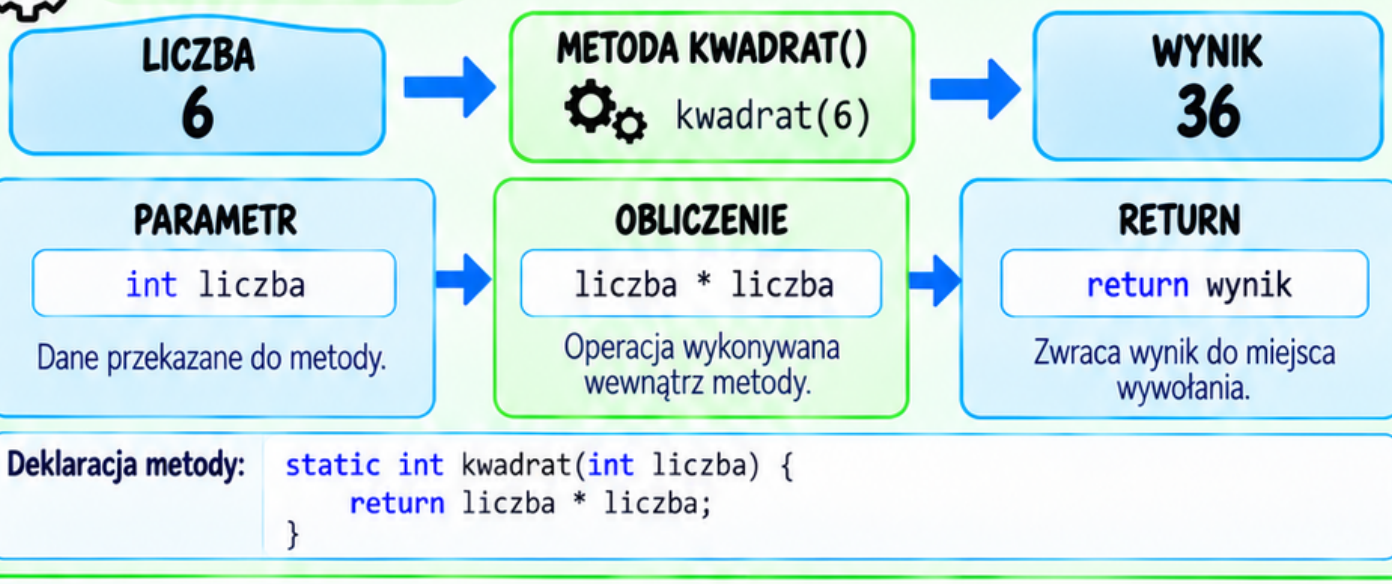
4. PODPOWIEDŹ

```
static int kwadrat(int liczba) {
    return liczba * liczba;
}
```

Wywołanie w metodzie main:

```
int wynik = kwadrat(6);
System.out.println(wynik);
```

5. JAK TO DZIAŁA?



Deklaracja metody:

```
static int kwadrat(int liczba) {
    return liczba * liczba;
}
```

6. SPRAWDŹ

- [ ] Czy kwadrat 3 wynosi 9?
- [ ] Czy kwadrat 5 wynosi 25?
- [ ] Czy kwadrat 0 wynosi 0?
- [ ] Czy metoda zwraca poprawną wartość?

7. BONUS

Utwórz metodę `szescian(int liczba)`, która oblicza trzecią potęgę liczby.

33 WYZWANIE 33 — SPRAWDŹ WIEK

Zwracaj odpowiedź true lub false

1. CEL WYZWANIA

Napisz metodę, która sprawdza, czy użytkownik ma co najmniej 18 lat.

2. TWOJE ZADANIE

Program powinien:

1. Utworzyć metodę czyPelnoletni.
2. Przyjmować parametr wiek.
3. Porównać go z liczbą 18.
4. Zwrócić true lub false.
5. Wyświetlić komunikat zależny od wyniku.

3. PRZYKŁAD DZIAŁANIA

```
Podaj wiek: 21
Czy pełnoletni? true
```

4. PODPOWIEDŹ

```
static boolean czyPelnoletni(int wiek) {
    return wiek >= 18;
}
```

Przykład wywołania:

```
boolean wynik = czyPelnoletni(21);
```

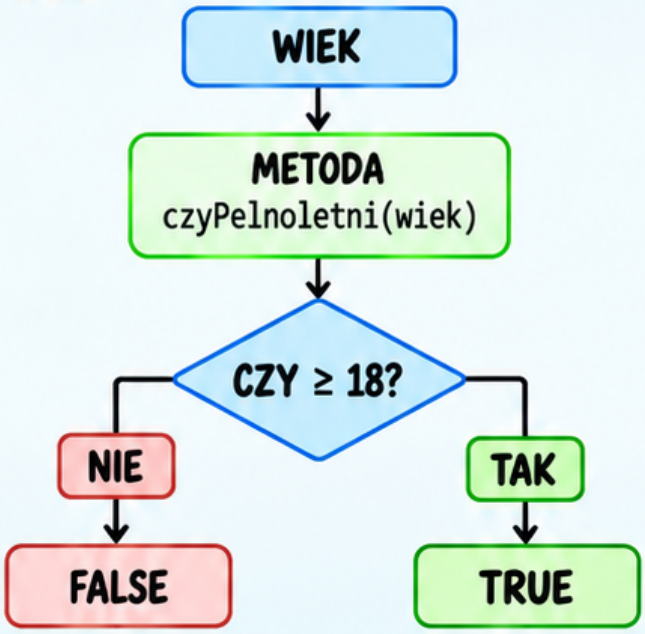
5. JAK TO DZIAŁA?

boolean to typ danych, który przechowuje tylko dwie wartości:

- **true** — prawda.
- **false** — fałsz.

Wyrażenie `wiek >= 18` samo zwraca wartość logiczną.

SCHEMAT DZIAŁANIA



BOOLEAN = TRUE / FALSE

PRZYKŁADY:

17 → false

21 → true

6. SPRAWDŹ

- [ ] Czy 18 zwraca true?
- [ ] Czy 17 zwraca false?
- [ ] Czy 25 zwraca true?
- [ ] Czy wynik można wykorzystać w instrukcji if?

7. BONUS

Utwórz metodę `czyParzysta(int liczba)`, która zwraca true dla liczb parzystych.

34 WYZWANIE 34 – DWIE METODY SUMA

Jedna nazwa, różne parametry

1. CEL WYZWANIA

Napisz dwie metody o nazwie suma, które dodają różną liczbę argumentów.

2. TWOJE ZADANIE

1. Utworzyć metodę dodającą dwie liczby.
2. Utworzyć drugą metodę dodającą trzy liczby.
3. Nadać obu metodom nazwę suma.
4. Wywołać obie metody.
5. Wyświetlić otrzymane wyniki.

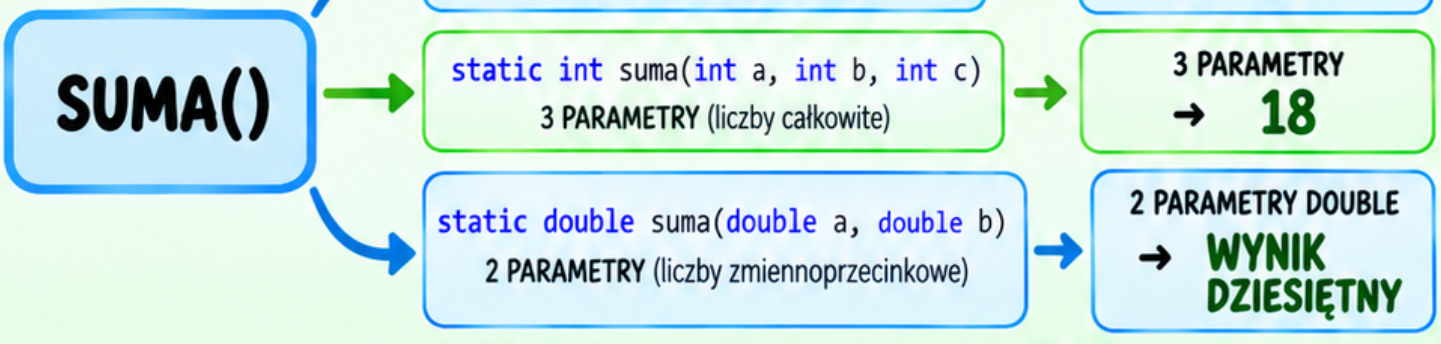
3. PRZYKŁADOWY KOD

```
static int suma(int a, int b) {
    return a + b;
}
static int suma(int a, int b, int c) {
    return a + b + c;
}
```

4. PRZYKŁAD DZIAŁANIA

```
suma(4, 6) → 10
suma(4, 6, 8) → 18
```

5. SUMA() – JEDNA NAZWA, RÓŻNE PARAMETRY



6. PODPOWIEDŹ

Java rozpoznaje odpowiednią metodę na podstawie liczby i typów parametrów. To zjawisko nazywa się przeciążaniem metod. Nie wystarczy zmienić wyłącznie typu zwracanego wyniku.

7. SPRAWDŹ

- [ ] Czy obie metody mają tę samą nazwę?
- [ ] Czy różnią się liczbą parametrów?
- [ ] Czy program wywołuje odpowiednią wersję?
- [ ] Czy wyniki obliczeń są poprawne?

8. BONUS

Utwórz trzecią wersję metody suma, przyjmującą dwie wartości typu double.

35 WYZWANIE 35 — KALKULATOR MODUŁOWY

Podziel kalkulator na mniejsze części

1. CEL WYZWANIA

Zbuduj kalkulator korzystający z osobnych metod do wykonywania działań matematycznych.

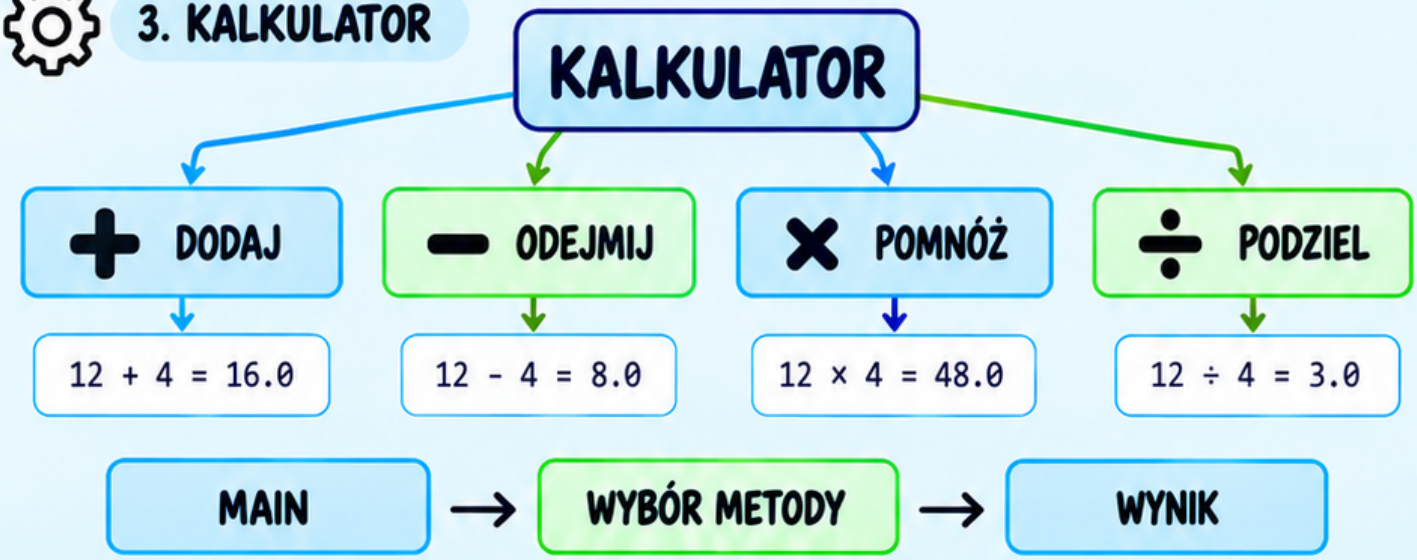
2. TWOJE ZADANIE

Utworzyć cztery metody:

1. `dodaj(a, b)`
2. `odejmij(a, b)`
3. `pomnoz(a, b)`
4. `podziel(a, b)`

Każda metoda powinna przyjmować dwie liczby i zwracać wynik działania.

3. KALKULATOR



4. PRZYKŁAD DZIAŁANIA

Dane:

```
a = 12
b = 4
```

Wyniki:

```
Dodawanie: 16.0
Odejmowanie: 8.0
Mnożenie: 48.0
Dzielenie: 3.0
```

5. PODPOWIEDŹ

```
static double dodaj(double a, double b) {
    return a + b;
}
```

Pozostałe metody utwórz według podobnego schematu.

6. WAŻNE

Przed wywołaniem metody dzielenia sprawdź, czy druga liczba nie jest równa zero. Dzięki temu program uniknie wyświetlenia nieoczekiwanego wyniku.

7. SPRAWDŹ

- [ ] Czy każda operacja ma osobną metodę?
- [ ] Czy metody zwracają prawidłowe wyniki?
- [ ] Czy program korzysta z wartości zwracanych?
- [ ] Czy sprawdzane jest dzielenie przez zero?

8. BONUS

Dodaj menu pozwalające użytkownikowi wybrać tylko jedno działanie.

36 WYZWANIE 36 — LISTA ZAKUPÓW

Stwórz dynamiczną listę z ArrayList

1. CEL WYZWANIA

Napisz program, który przechowuje nazwy produktów na liście zakupów.

2. TWOJE ZADANIE

1. Utworzyć `ArrayList<String>`.
2. Dodać trzy produkty do listy.
3. Wyświetlić wszystkie produkty.
4. Usunąć jeden wybrany produkt.
5. Wyświetlić zaktualizowaną listę.

3. PRZYKŁADOWE PRODUKTY

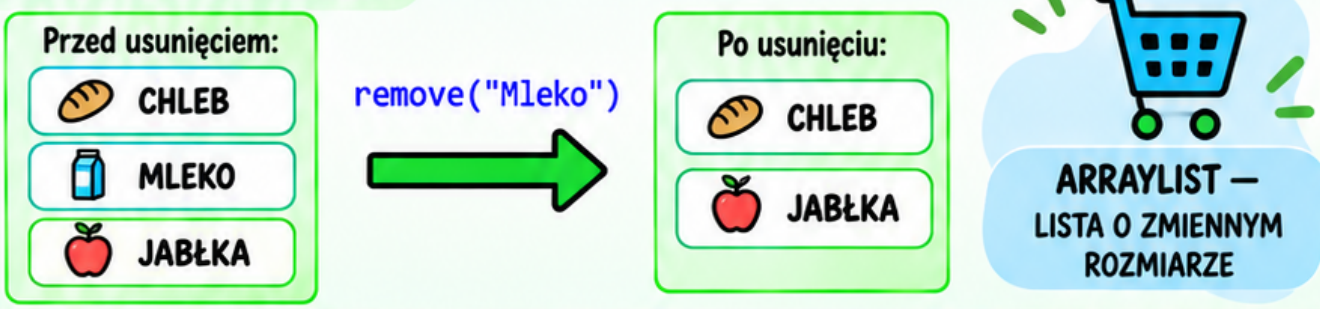
- Chleb
- Mleko
- Jabłka

4. PODPOWIEDŹ

```
import java.util.ArrayList;
ArrayList<String> zakupy =
     new ArrayList<>();
zakupy.add("Chleb");
zakupy.add("Mleko");
zakupy.add("Jabłka");

Aby usunąć produkt:
zakupy.remove("Mleko");
```

5. PRZYKŁAD DZIAŁANIA



6. JAK DZIAŁA ARRAYLIST?

- `add()` — dodaje element.
- `remove()` — usuwa element.
- `get()` — odczytuje element.
- `size()` — zwraca liczbę elementów.

7. SPRAWDŹ

- [ ] Czy lista przechowuje trzy produkty?
- [ ] Czy można dodać kolejny produkt?
- [ ] Czy usuwanie działa poprawnie?
- [ ] Czy liczba elementów zmienia się automatycznie?

8. BONUS

Dodaj możliwość wpisania nowego produktu przez użytkownika przy pomocy `Scanner`.

37 WYZWANIE 37 — ANALIZA LISTY

Oblicz średnią i znajdź najlepszy wynik

1. CEL WYZWANIA

Napisz program analizujący wyniki zapisane w `ArrayList<Integer>`.

2. TWOJE ZADANIE

1. Utworzyć listę wyników.
2. Dodać pięć liczb.
3. Obliczyć ich sumę.
4. Obliczyć średnią.
5. Znaleźć największy wynik.
6. Wyświetlić podsumowanie.

3. PRZYKŁADOWE DANE

```
ArrayList<Integer> wyniki =
    new ArrayList<>();
wyniki.add(7);
wyniki.add(9);
wyniki.add(6);
wyniki.add(10);
wyniki.add(8);
```

4. OCZEKIWANY WYNIK

```
Suma: 40
Średnia: 8.0
Największy wynik: 10
```

5. PRZYKŁADOWA ANALIZA

Lista wyników: **7** **9** **6** **10** **8**

SUMA = 40 | **ŚREDNIA = 8** | **MAX = 10**

FOR-EACH

— przechodzenie przez kolejne elementy kolekcji.

6. PODPOWIEDŹ

```
for (int wynik : wyniki) {
    suma += wynik;
}
```

Jest to pętla for-each, która odczytuje kolejne elementy kolekcji.

7. JAK OBLICZYĆ ŚREDNIĄ?

```
double srednia = (double) suma / wyniki.size();
```

Podziel sumę przez liczbę elementów.

8. WAŻNE

Przed obliczaniem średniej sprawdź, czy lista nie jest pusta.

9. SPRAWDŹ

- [ ] Czy suma wynosi 40?
- [ ] Czy średnia wynosi 8.0?
- [ ] Czy największą wartością jest 10?
- [ ] Czy program poprawnie przechodzi przez listę?

10. BONUS

Dodaj obliczenie liczby wyników większych od średniej.

38 WYZWANIE 38 — HASHMAP W PRAKTYCE

Połącz imiona z ocenami za pomocą HashMap

1. CEL WYZWANIA

Napisz program przechowujący imiona uczniów i ich oceny.

2. TWOJE ZADANIE

1. Utworzyć HashMap<String, Integer>.
2. Dodać trzech uczniów.
3. Przypisać każdemu ocenę.
4. Odczytać ocenę wybranego ucznia.
5. Zmienić jedną ocenę.
6. Wyświetlić zaktualizowany wynik.

3. PRZYKŁADOWE DANE

- Anna → 5
- Piotr → 4
- Marek → 3

PRZYKŁADOWA TABLICA

KLUCZ	WARTOŚĆ
Anna	5
Piotr	4
Marek	3

AKTUALIZACJA OCENY

Piotr:

4 → 5

4. PODPOWIEDŹ

```
import java.util.HashMap;
HashMap<String, Integer> oceny =
    new HashMap<>();
oceny.put("Anna", 5);
oceny.put("Piotr", 4);
oceny.put("Marek", 3);

Odczytywanie:
oceny.get("Anna");

Zmiana oceny:
oceny.put("Piotr", 5);
```

5. PRZYKŁAD DZIAŁANIA

```
Ocena Anny: 5
Nowa ocena Piotra: 5
```

6. JAK DZIAŁA HASHMAP?

KLUCZ
Pozwala odnaleźć informację.

WARTOŚĆ
Jest informacją przypisaną do klucza.

put()
dodaje lub aktualizuje parę.

get()
odczytuje wartość.

7. SPRAWDŹ

- [ ] Czy można odczytać ocenę Anny?
- [ ] Czy ponowne użycie put() aktualizuje ocenę?
- [ ] Czy różni uczniowie mogą mieć tę samą ocenę?
- [ ] Czy program prawidłowo obsługuje brakujące imię?

8. BONUS

Dodaj sprawdzenie `containsKey()`, aby wykrywać uczniów, których nie ma w kolekcji.

JAK TO DZIAŁA?

IMIĘ → HASHMAP → OCENA

39 WYZWANIE 39 — KLASA I OBIEKT

Stwórz klasę reprezentującą ucznia

1. CEL WYZWANIA

Napisz program wykorzystujący klasę **Uczen** do przechowywania i wyświetlania informacji o uczniu.

2. TWOJE ZADANIE

1. Utworzyć klasę `Uczen`.
2. Dodać pola `imie` i `wiek`.
3. Utworzyć konstruktor.
4. Dodać metodę `przedstawSie()`.
5. Utworzyć obiekt klasy `Uczen`.
6. Wywołać metodę obiektu.

3. PRZYKŁADOWY KOD

```
class Uczen {
    String imie;
    int wiek;

    Uczen(String imie, int wiek) {
        this.imie = imie;
        this.wiek = wiek;
    }

    void przedstawSie() {
        System.out.println(
            "Mam na imię " + imie
          + " i mam " + wiek + " lat."
        );
    }
}
```

KLASA UCZEN

POLA:
- imie
- wiek

METODA:
- przedstawSie()

OBIEKT 1



Anna, 16 lat

OBIEKT 2



Piotr, 17 lat

JEDNA KLASA → WIELE OBIEKTÓW

4. UTWORZENIE OBIEKTU

W metodzie main:

```
Uczen uczen = new Uczen("Anna", 16);
uczen.przedstawSie();
```

5. OCZEKIWANY WYNIK

```
Mam na imię Anna i mam 16 lat.
```

6. PODSTAWOWE POJĘCIA

KLASA	OBIEKT	POLE	KONSTRUKTOR	METODA
Opisuje strukturę i zachowanie obiektów.	Konkretny egzemplarz utworzony na podstawie klasy.	Przechowuje informację o obiekcie.	Pomaga przygotować nowy obiekt.	Określa czynność, którą obiekt może wykonać.

7. SPRAWDŹ

- [ ] Czy klasa posiada dwa pola?
- [ ] Czy konstruktor zapisuje przekazane wartości?
- [ ] Czy obiekt został utworzony przez new?
- [ ] Czy metoda wyświetla prawidłowe informacje?

8. BONUS

Utwórz drugi obiekt klasy Uczen z innym imieniem i wiekiem. Wywołaj metodę przedstawSie() dla obu obiektów.

40 WYZWANIE 40 — KATALOG PRODUKTÓW

Połącz klasy, obiekty i kolekcje w jednym programie.

CEL WYZWANIA

Napisz prostą aplikację, która przechowuje informacje o produktach i oblicza ich łączną wartość.

TWOJE ZADANIE

1. Utworzyć klasę `Produkt`.
2. Dodać pola `nazwa` i `cena`.
3. Utworzyć konstruktor przyjmujący nazwę i cenę.
4. Utworzyć trzy obiekty klasy `Produkt`.
5. Dodać je do `ArrayList<Produkt>`.
6. Wyświetlić produkty wraz z cenami.
7. Obliczyć łączną wartość produktów.

PODPOWIEDŹ

Przykładowa struktura klasy:

```
class Produkt {
    String nazwa;
    double cena;
    Produkt(String nazwa, double cena) {
        this.nazwa = nazwa;
        this.cena = cena;
    }
}
```

Tworzenie obiektu:

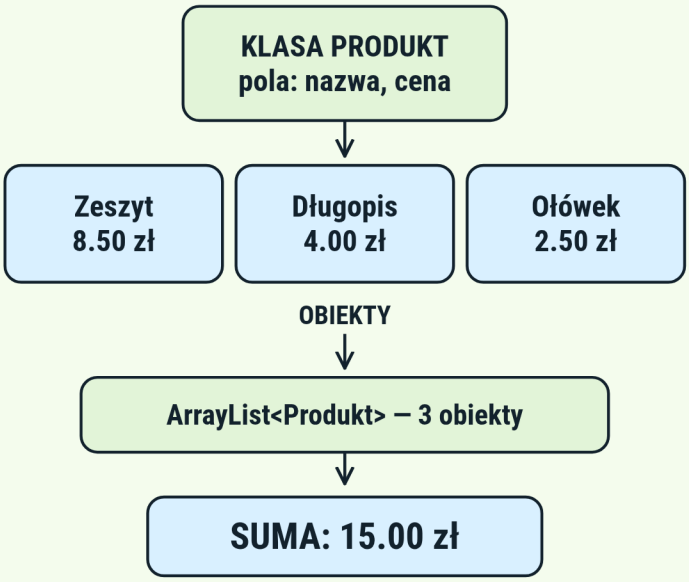
```
Produkt p1 = new Produkt("Zeszyt", 8.50);
```

Tworzenie listy:

```
ArrayList<Produkt> produkty =
    new ArrayList<>();
produkty.add(p1);
```

Do obliczenia sumy wykorzystaj pętlę for-each.

OD KLASY DO KOLEKCJI



DANE I OCZEKIWANY WYNIK

```
Lista produktów:
Zeszyt — 8.50 zł
Długopis — 4.00 zł
Ołówek — 2.50 zł
Łączna wartość: 15.00 zł
```

JAK TO DZIAŁA?

KLASA
Określa, jakie informacje przechowuje produkt.

OBIEKT
Reprezentuje konkretny produkt.

ARRAYLIST
Przechowuje wiele obiektów.

PĘTLA
Pozwala przeglądać i analizować wszystkie produkty.

SPRAWDŹ

- [ ] Czy utworzono trzy obiekty?
- [ ] Czy wszystkie produkty znajdują się na liście?
- [ ] Czy program wyświetla poprawne ceny?
- [ ] Czy suma wynosi 15.00 zł?

BONUS

Dodaj metodę `wyswietlInformacje()` w klasie `Produkt`, która wyświetla nazwę i cenę produktu. Wywołaj ją dla każdego elementu listy.

JEDNA KLASA → WIELE OBIEKTÓW → JEDNA KOLEKCJA

04 MODUŁ 04 — SPRAWDŹ SWOJE UMIEJĘTNOŚCI

Sprawdź, czy potrafisz organizować kod.

TWÓJ POSTĘP

Zaznacz wykonane wyzwania z modułu 04.

☐ 31 ☐ 32 ☐ 33 ☐ 34 ☐ 35 ☐ 36 ☐ 37 ☐ 38 ☐ 39 ☐ 40

CO JUŻ POTRAFISZ?

- ☐ Potrafię tworzyć własne metody.
- ☐ Rozumiem znaczenie słowa `void`.
- ☐ Umiem przekazywać parametry.
- ☐ Potrafię zwracać wartości przez `return`.
- ☐ Rozumiem przeciążanie metod.
- ☐ Umiem korzystać z `ArrayList`.
- ☐ Potrafię dodawać i usuwać elementy listy.
- ☐ Umiem przechowywać dane w `HashMap`.
- ☐ Rozumiem różnicę między klasą a obiektem.
- ☐ Potrafię tworzyć obiekty za pomocą konstruktora.

UPORZĄDKOWANY PROGRAM

METODY → **KOLEKCJE** → **OBIEKTY**

Małe metody, przemyślane kolekcje i własne obiekty tworzą czytelny program.

MINI-SPRAWDZIAN

ZADANIE: KATALOG KSIĄŻEK

Napisz program, który przechowuje informacje o książkach. Utwórz klasę `Ksiazka` zawierającą:

tytul	autor	liczbaStron
Tytuł książki.	Autor książki.	Liczba stron.

PROGRAM POWINIEN:

1. Utworzyć trzy obiekty książek.
2. Zapisać je w `ArrayList<Ksiazka>`.
3. Wyświetlić wszystkie tytuły.
4. Znaleźć książkę z największą liczbą stron.
5. Wyświetlić informacje o wybranej książce.

DANE I OCZEKIWANY WYNIK

Książka A — 120 stron
Książka B — 250 stron
Książka C — 180 stron

```
Liczba książek: 3
Najdłuższa książka: Książka B
Liczba stron: 250
```

PODPOWIEDŹ

Wykorzystaj:

- Klasę `Ksiazka`.
- Konstruktor.
- `ArrayList`.
- Pętlę for-each.
- Instrukcję `if`.

PYTANIA DO REFLEKSJI

- Czym różni się klasa od obiektu?
- Dlaczego warto dzielić program na metody?
- Kiedy lepiej wykorzystać ArrayList, a kiedy HashMap?
- Jakie zadanie wykonuje konstruktor?

TWOJE OSIĄGNIĘCIE

☐ MODUŁ 04 — UKOŃCZONY

Zaznacz po ukończeniu wszystkich wyzwań.

Potrafisz już tworzyć własne klasy, metody i kolekcje. Twoje programy stają się coraz bardziej uporządkowane!

05 MODUŁ 05
WYJĄTKI, PLIKI I MINIAPLIKACJE

Naucz swoje programy reagować na błędy i zapisywać dane.



CZEGO SIĘ NAUCZYSZ?

1. **WYJĄTKI**
   Obsługuj błędy podczas działania programu.
2. **TRY-CATCH**
   Reaguj na sytuacje wyjątkowe.
3. **ZAPIS DO PLIKU**
   Zachowuj informacje na dysku.
4. **ODCZYT PLIKU**
   Wczytuj wcześniej zapisane dane.
5. **MINIAPLIKACJE**
   Łącz poznane umiejętności w praktyczne projekty.

GŁÓWNA IDEA

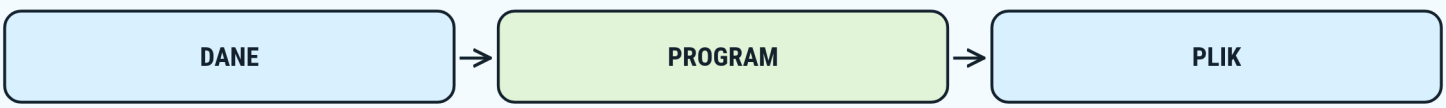
Dobrze przygotowany program potrafi poradzić sobie z błędnymi danymi i przechowywać potrzebne informacje.

Informacje zapisane w pliku mogą być dostępne także po zakończeniu programu.

PRZYKŁAD KONCEPCYJNY

1. UŻYTKOWNIK
2. WPROWADZENIE DANYCH
3. SPRAWDZENIE POPRAWNOŚCI
4. PRZETWARZANIE
5. ZAPIS DO PLIKU

OD DANYCH DO PLIKU



To ostatni etap! Wykorzystaj wszystko, czego nauczyłeś się do tej pory.

41 WYZWANIE 41 — TRY I CATCH

Zabezpiecz program przed nieprawidłową liczbą.

CEL WYZWANIA

Napisz program, który próbuje zamienić wpisany tekst na liczbę całkowitą i wyświetla komunikat, jeśli konwersja się nie powiedzie.

TWOJE ZADANIE

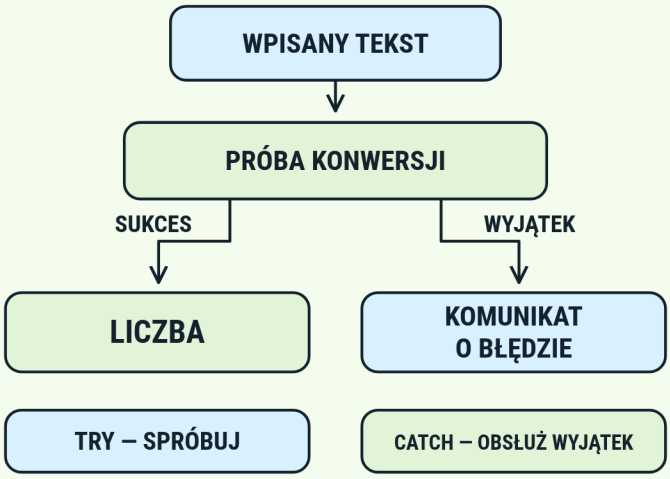
1. Poproś użytkownika o wpisanie liczby.
2. Odczytaj dane jako `String`.
3. Spróbuj zamienić tekst na `int`.
4. Jeśli konwersja się powiedzie, wyświetl liczbę.
5. Jeśli pojawi się błąd, wyświetl przyjazny komunikat.

PODPOWIEDŹ

```
try {
    int liczba = Integer.parseInt(tekst);
    System.out.println(liczba);
} catch (NumberFormatException e) {
    System.out.println(
        "Błąd: wpisz poprawną liczbę."
    );
}
```

Kod umieść w metodzie `main`, po odczytaniu tekstu od użytkownika.

SCHEMAT DZIAŁANIA



PRZYKŁAD 1

```
Podaj liczbę: 25
Wpisano liczbę: 25
```

PRZYKŁAD 2

```
Podaj liczbę: abc
Błąd: wpisz poprawną liczbę całkowitą.
```

CO OZNACZA TRY-CATCH?

TRY

Zawiera instrukcje, które mogą zgłosić wyjątek.

CATCH

Pozwala obsłużyć wskazany wyjątek, zamiast kończyć program nieobsłużonym błędem.

NumberFormatException

Wyjątek występujący między innymi wtedy, gdy tekst nie jest prawidłową reprezentacją liczby całkowitej.

SPRAWDŹ

- [ ] Czy 10 jest poprawnie odczytywane?
- [ ] Czy -5 jest akceptowane?
- [ ] Czy abc wyświetla komunikat o błędzie?
- [ ] Czy 12.5 jest odrzucane jako liczba całkowita?

BONUS

Dodaj komunikat informujący, że wpisana liczba jest dodatnia, ujemna lub równa zero.

42 WYZWANIE 42 — MENU ODPORNE NA BŁĘDY

Powtarzaj pytanie po błędnym wyborze.

CEL WYZWANIA

Stwórz menu, które pozwala użytkownikowi wybrać jedną z trzech opcji i nie kończy działania po wpisaniu nieprawidłowej wartości.

TWOJE ZADANIE

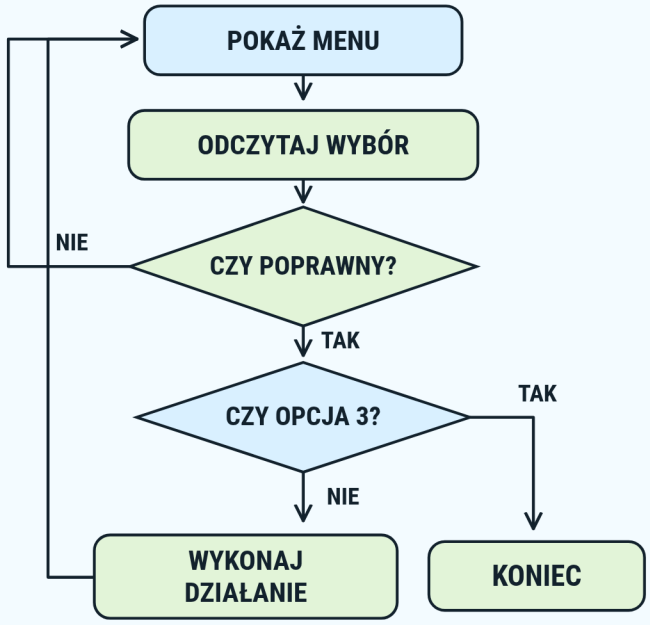
Program powinien wyświetlać:

```
1. Powitanie
2. Informacja o programie
3. Zakończ
Wybierz opcję:
```

ZASADY DZIAŁANIA

- Opcja 1 wyświetla powitanie.
- Opcja 2 wyświetla opis programu.
- Opcja 3 kończy działanie.
- Nieznany numer powoduje ponowne pokazanie menu.
- Tekst zamiast liczby nie powoduje zakończenia programu przez nieobsłużony wyjątek.

SCHEMAT MENU



PODPOWIEDŹ

Do odczytu wyboru użyj:

```
String tekst = scanner.nextLine();
```

Następnie spróbuj przekonwertować go na liczbę:

```
int wybor = Integer.parseInt(tekst);
```

Umieść konwersję w bloku try-catch. Wykorzystaj do-while, aby powtarzać menu do momentu wybrania opcji 3.

PRZYKŁAD DZIAŁANIA

```
Wybierz opcję: abc              Wybierz opcję: 1
Nieprawidłowa wartość.          Witaj w programie!
Wybierz opcję: 7                Wybierz opcję: 3
Nie ma takiej opcji.            Do zobaczenia!
```

SPRAWDŹ

- [ ] Czy program prawidłowo obsługuje opcje 1, 2 i 3?
- [ ] Czy wpisanie tekstu nie kończy programu błędem?
- [ ] Czy nieznany numer wyświetla odpowiedni komunikat?
- [ ] Czy menu pojawia się ponownie?
- [ ] Czy opcja 3 kończy działanie?

BONUS

Dodaj opcję 4:

```
4. Wyświetl aktualny rok
```

Po wykonaniu tej opcji wróć do menu.

43 WYZWANIE 43 — MOJA PIERWSZA NOTATKA

Zapisz swoją pierwszą notatkę w pliku tekstowym.

CEL WYZWANIA

Napisz program, który pobiera tekst od użytkownika i zapisuje go w pliku `notatka.txt`.

TWOJE ZADANIE

1. Poprosić użytkownika o wpisanie notatki.
2. Odczytać cały tekst.
3. Utworzyć plik tekstowy.
4. Zapisać wpisaną treść.
5. Wyświetlić komunikat o powodzeniu lub błędzie.

PRZYKŁAD DZIAŁANIA

```
Wpisz notatkę:
Dzisiaj uczę się zapisu plików w Javie.
Notatka została zapisana!
```

ZAWARTOŚĆ PLIKU

notatka.txt

Dzisiaj uczę się zapisu plików w Javie.

PODPOWIEDŹ

Wykorzystaj:

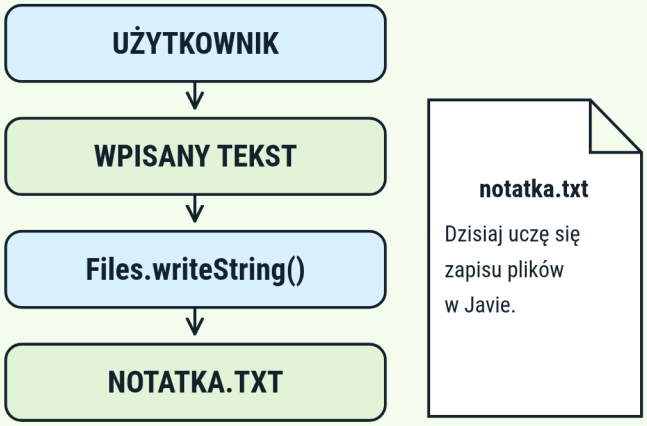
```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.charset.StandardCharsets;
import java.io.IOException;
```

Przykładowy zapis umieszczony w bloku `try`:

```
Files.writeString(
    Path.of("notatka.txt"),
    tekst,
    StandardCharsets.UTF_8
);
```

Obsłuż `IOException` za pomocą `catch`.

OD TEKSTU DO PLIKU



WAŻNE

`Files.writeString()` zapisuje tekst do pliku. Jeżeli plik już istnieje, domyślny zapis zastępuje jego wcześniejszą zawartość.

Ścieżka względna `notatka.txt` odnosi się do katalogu roboczego programu.

SPRAWDŹ

- [ ] Czy program tworzy plik?
- [ ] Czy plik zawiera wpisaną notatkę?
- [ ] Czy polskie znaki są poprawnie zapisane?
- [ ] Czy ponowny zapis aktualizuje treść?
- [ ] Czy błędy zapisu są obsługiwane?

BONUS

Dodaj możliwość wpisania własnej nazwy pliku, zamiast zawsze używać `notatka.txt`.

44 WYZWANIE 44 — WCZYTAJ NOTATKĘ

Odczytaj zapisaną notatkę.

CEL WYZWANIA

Napisz program, który odczytuje zawartość pliku `notatka.txt` i wyświetla ją na ekranie.

TWOJE ZADANIE

1. Przygotuj plik `notatka.txt`.
2. Sprawdź, czy plik istnieje.
3. Odczytaj jego zawartość.
4. Wyświetl tekst w konsoli.
5. Obsłuż błędy odczytu.

PRZYKŁADOWA ZAWARTOŚĆ

Uczę się programowania.
Java daje wiele możliwości.

OCZEKIWANY WYNIK

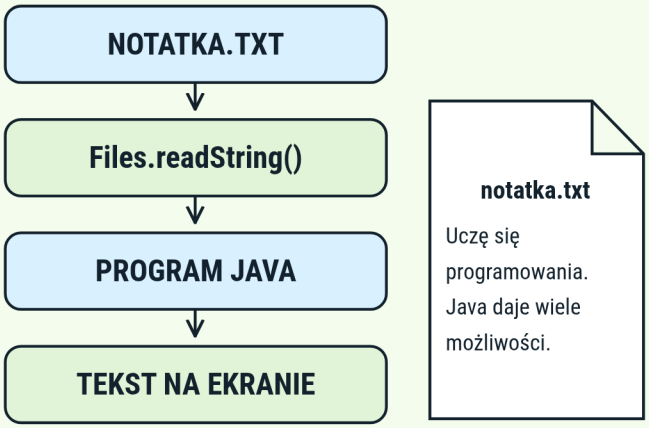
```
Zawartość notatki:
Uczę się programowania.
Java daje wiele możliwości.
```

PODPOWIEDŹ

```
Path sciezka = Path.of("notatka.txt");
try {
    String tresc = Files.readString(
        sciezka,
        StandardCharsets.UTF_8
    );
    System.out.println(tresc);
} catch (IOException e) {
    System.out.println(
        "Nie udało się odczytać pliku."
    );
}
```

Wymagane importy: `java.nio.file.Files`, `java.nio.file.Path`, `java.nio.charset.StandardCharsets` oraz `java.io.IOException`.

OD PLIKU DO PROGRAMU



JAK TO DZIAŁA?

Path.of()
Wskazuje położenie pliku.

Files.readString()
Odczytuje tekst.

IOException
Pozwala obsłużyć problemy występujące podczas operacji wejścia i wyjścia.

SPRAWDŹ

- [ ] Czy program odczytuje poprawny plik?
- [ ] Czy wyświetla całą zawartość?
- [ ] Czy zachowuje polskie znaki?
- [ ] Czy informuje o błędzie, gdy plik nie istnieje?

BONUS

Dodaj możliwość wybrania nazwy pliku przez użytkownika.

WRITE	READ
ZAPIS	ODCZYT

45 WYZWANIE 45 — PROSTY DZIENNIK

Dodawaj nowe wpisy bez usuwania poprzednich.

CEL WYZWANIA

Napisz program, który zapisuje kolejne wpisy w pliku `dziennik.txt`, zachowując wcześniejsze notatki.

TWOJE ZADANIE

1. Poprosić użytkownika o nowy wpis.
2. Odczytać tekst.
3. Otworzyć lub utworzyć plik `dziennik.txt`.
4. Dopisać nowy tekst na końcu pliku.
5. Zakończyć wpis znakiem nowej linii.
6. Wyświetlić informację o zapisie.

PRZYKŁAD

Pierwszy wpis:
Ukończyłem wyzwanie 43.

Drugi wpis:
Ukończyłem wyzwanie 44.

ZAWARTOŚĆ PLIKU

```
Ukończyłem wyzwanie 43.
Ukończyłem wyzwanie 44.
```

PODPOWIEDŹ

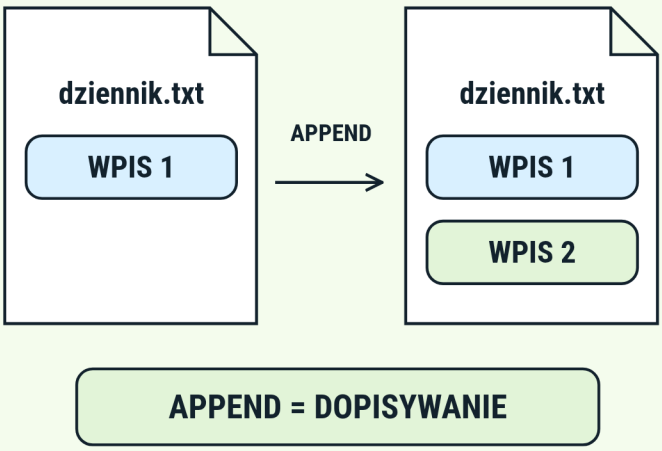
Wykorzystaj opcje `StandardOpenOption.CREATE` i `StandardOpenOption.APPEND`.

Przykładowy zapis w bloku `try`:

```
Files.writeString(
    Path.of("dziennik.txt"),
    wpis + System.lineSeparator(),
    StandardCharsets.UTF_8,
    StandardOpenOption.CREATE,
    StandardOpenOption.APPEND
);
```

Dodaj odpowiednie importy, w tym `java.nio.file.StandardOpenOption`, oraz obsługę `IOException`.

ZACHOWAJ POPRZEDNIE WPISY



JAK TO DZIAŁA?

CREATE
Tworzy plik, jeśli jeszcze nie istnieje.

APPEND
Dodaje dane na końcu pliku.

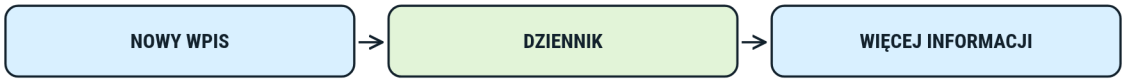
LINESEPARATOR
Wstawia znak końca linii odpowiedni dla systemu.

SPRAWDŹ

- [ ] Czy pierwszy wpis tworzy plik?
- [ ] Czy drugi wpis nie usuwa pierwszego?
- [ ] Czy każdy wpis zajmuje osobną linię?
- [ ] Czy polskie znaki są poprawne?

BONUS

Dodaj do każdego wpisu bieżącą datę, wykorzystując `LocalDate.now()`.



46 WYZWANIE 46 — STATYSTYKI TEKSTU

Policz linie i słowa w pliku.

CEL WYZWANIA

Napisz program, który analizuje plik tekstowy i oblicza liczbę linii oraz słów.

TWOJE ZADANIE

1. Odczytać wszystkie linie pliku.
2. Obliczyć liczbę linii.
3. Przejść przez każdą linię.
4. Podzielić niepuste linie na słowa.
5. Zsumować liczbę słów.
6. Wyświetlić wyniki.

PODPOWIEDŹ

Wykorzystaj:

```
List<String> linie = Files.readAllLines(
    Path.of("tekst.txt"),
    StandardCharsets.UTF_8
);
```

Do przetwarzania linii:

```
int liczbaSlow = 0;
for (String linia : linie) {
    String tekst = linia.trim();
    if (!tekst.isEmpty()) {
        liczbaSlow +=
            tekst.split("\\s+").length;
    }
}
```

Dodaj wymagane importy (`List`, `Files`, `Path`, `StandardCharsets`, `IOException`) oraz obsługę `IOException`.

PRZYKŁADOWY PLIK

tekst.txt

Java jest ciekawa.

Uczę się pisać kod.

Każde zadanie pomaga.

LINIE 3

SŁOWA 10

3 + 4 + 3 → **10 SŁÓW**

OCZEKIWANY WYNIK

```
Liczba linii: 3
Liczba słów: 10
```

JAK TO DZIAŁA?

readAllLines()
Odczytuje linie pliku do listy.

trim()
Usuwa zbędne białe znaki na początku i końcu tekstu.

split("\\s+")
Dzieli tekst według sekwencji białych znaków.

length
Pozwala policzyć elementy otrzymanej tablicy.

SPRAWDŹ

- [ ] Czy trzy linie są poprawnie liczone?
- [ ] Czy puste linie nie dodają słów?
- [ ] Czy przykład zawiera 10 słów?
- [ ] Czy program radzi sobie z pustym plikiem?

BONUS

Dodaj obliczenie liczby wszystkich znaków w pliku, bez znaków końca linii.

PLIK → **LINIE** → **SŁOWA** → **STATYSTYKI**

47 WYZWANIE 47 — LISTA ZADAŃ TO-DO

Zbuduj program, który pomaga organizować codzienne obowiązki.

CEL WYZWANIA

Napisz aplikację konsolową, która pozwala dodawać, przeglądać i usuwać zadania.

TWOJE ZADANIE

1. Utworzyć `ArrayList<String>` do przechowywania zadań.
2. Wyświetlić menu główne.
3. Pozwalać dodawać nowe zadania.
4. Wyświetlać listę zapisanych zadań.
5. Umożliwiać usuwanie zadania na podstawie numeru.
6. Obsługiwać błędne wybory.
7. Powtarzać menu aż do zakończenia programu.

MENU APLIKACJI

```
=== LISTA ZADAŃ ===
1. Dodaj zadanie
2. Pokaż zadania
3. Usuń zadanie
4. Zakończ
Wybierz opcję:
```

PRZYKŁAD DZIAŁANIA

Użytkownik wybiera opcję 1 i dodaje dwa zadania, następnie wybiera opcje 2 i 3:

```
Podaj nowe zadanie:
Powtórzyć pętle w Javie
Zadanie zostało dodane.

Następne zadanie:
Rozwiązać trzy ćwiczenia

Po wybraniu opcji 2:
Twoje zadania:
1. Powtórzyć pętle w Javie
2. Rozwiązać trzy ćwiczenia

Po wybraniu opcji 3:
Podaj numer zadania: 1
Zadanie zostało usunięte.
```

PODPOWIEDŹ

Do przechowywania danych użyj:

```
ArrayList<String> zadania =
    new ArrayList<>();
```

Do dodawania:

```
zadania.add(noweZadanie);
```

Do usuwania według numeru wyświetlanego użytkownikowi:

```
zadania.remove(numer - 1);
```

Przed usunięciem sprawdź, czy numer mieści się w zakresie od 1 do `zadania.size()`.



PODZIAŁ NA METODY

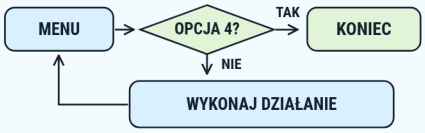
pokazMenu()	dodajZadanie()	pokazZadania()	usunZadanie()
Wyświetla dostępne opcje.	Dodaje nowy element.	Wyświetla listę zadań.	Usuwa wybrane zadanie.

Metody mogą przyjmować jako parametry listę oraz — w razie potrzeby — obiekt `Scanner`.

WAŻNE

Na razie zadania są przechowywane tylko w pamięci programu. Po jego zamknięciu lista zostanie utracona. W kolejnych wyzwaniach dodasz możliwość zapisywania i odczytywania zadań z pliku.

POWRÓT DO MENU



SPRAWDŹ

- [ ] Czy można dodać kilka zadań?
- [ ] Czy usuwanie działa poprawnie?
- [ ] Czy pusta lista jest obsługiwana?
- [ ] Czy opcja 4 kończy program?
- [ ] Czy zadania pojawiają się w prawidłowej kolejności?
- [ ] Czy program odrzuca nieprawidłowy numer?
- [ ] Czy menu wyświetla się ponownie?

BONUS

Dodaj możliwość oznaczania zadania jako wykonanego, na przykład przez umieszczenie znacznika `[X]` przed jego nazwą.

[ ]	[X]
Do zrobienia	Wykonane

48 WYZWANIE 48 – ZAPISYWANIE ZADAŃ

Twoje dane mogą pozostać dostępne nawet po zamknięciu programu.

CEL WYZWANIA

Rozbuduj aplikację listy zadań tak, aby potrafiła zapisywać wszystkie zadania do pliku tekstowego.

SYTUACJA

Masz program przechowujący zadania w `ArrayList<String>`. Podczas działania możesz dodawać i usuwać zadania. Problem pojawia się po zamknięciu aplikacji — informacje zapisane wyłącznie w pamięci zostają utracone. Twoim celem jest rozwiązanie tego problemu.

TWOJE ZADANIE

1. Utworzyć listę zadań.
2. Pozwalać na dodanie kilku przykładowych zadań.
3. Utworzyć metodę `zapiszZadania()`.
4. Zapisywać zawartość listy do pliku `zadania.txt`.
5. Umieszczać każde zadanie w osobnej linii.
6. Informować o powodzeniu lub błędzie zapisu.

PRZYKŁADOWA LISTA

- Nauczyć się metod.
- Powtórzyć kolekcje.
- Rozwiązać zadania z Javy.

ZAWARTOŚĆ PLIKU

zadania.txt

```
Nauczyć się metod.
Powtórzyć kolekcje.
Rozwiązać zadania z Javy.
```

PODPOWIEDŹ

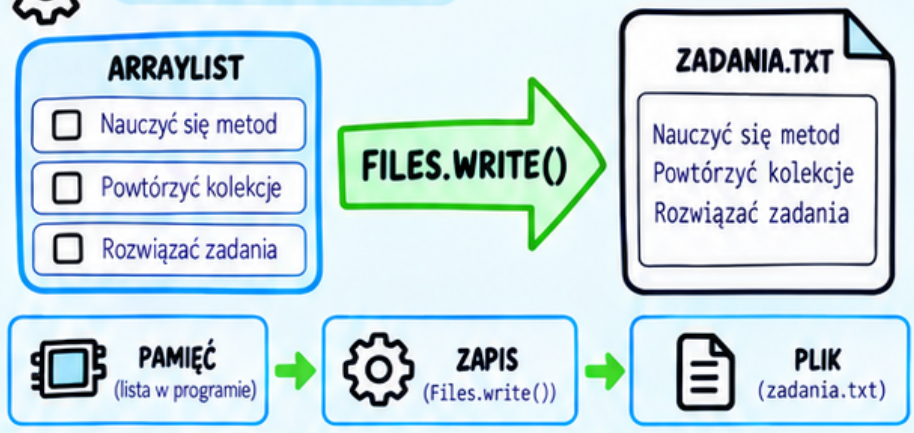
Wykorzystaj metodę: `Files.write()`

```
static void zapiszZadania(
    ArrayList<String> zadania
) {
    try {
        Files.write(
            Path.of("zadania.txt"),
            zadania,
            StandardCharsets.UTF_8
        );
        System.out.println(
            "Zadania zostały zapisane."
        );
    } catch (IOException e) {
        System.out.println(
            "Błąd podczas zapisywania."
        );
    }
}
```

Dodaj wymagane importy:

- `java.util.ArrayList`
- `java.nio.file.Files`
- `java.nio.file.Path`
- `java.nio.charset.StandardCharsets`
- `java.io.IOException`

SCHEMAT DZIAŁANIA



JAK TO DZIAŁA?

1. **ARRAYLIST** — Przechowuje zadania podczas działania programu.
2. **FILES.WRITE()** — Zapisuje elementy listy jako kolejne linie pliku tekstowego.
3. **UTF-8** — Pozwala prawidłowo zapisywać polskie znaki.
4. **TRY-CATCH** — Obsługuje problemy związane z dostępem do pliku.

WAŻNE

Domyślnie zapis zastępuje wcześniejszą zawartość pliku. W tym ćwiczeniu jest to pożądane, ponieważ plik ma odzwierciedlać aktualny stan listy. Każde zadanie powinno być pojedynczym wierszem tekstu.

SPRAWDŹ

- [ ] Czy program tworzy zadania.txt?
- [ ] Czy wszystkie zadania zostają zapisane?
- [ ] Czy każde zadanie znajduje się w osobnej linii?
- [ ] Czy polskie znaki wyświetlają się prawidłowo?
- [ ] Czy ponowny zapis aktualizuje zawartość?
- [ ] Czy błędy zapisu są obsługiwane?

BONUS

Dodaj do menu aplikacji opcję:

`5. Zapisz zadania do pliku`

Po jej wybraniu program powinien zapisać aktualną listę.

ZAPISANE DANE POZOSTAJĄ DOSTĘPNE PO ZAMKNIĘCIU PROGRAMU.

49 WYZWANIE 49 – WCZYTYWANIE ZADAŃ

Otwórz aplikację i odzyskaj swoją wcześniejszą listę.

CEL WYZWANIA

Rozbuduj menedżer zadań tak, aby wczytywał informacje z pliku `zadania.txt`.

SYTUACJA

W poprzednim wyzwaniu zapisałeś listę zadań do pliku. Teraz chcesz uruchomić program ponownie i zobaczyć te same zadania. Aplikacja powinna automatycznie odczytać zapisane informacje.

TWOJE ZADANIE

1. Sprawdzić, czy istnieje plik `zadania.txt`.
2. Jeśli plik istnieje, odczytać wszystkie linie.
3. Zapisać odczytane informacje w `ArrayList<String>`.
4. Wyświetlić liczbę odzyskanych zadań.
5. Pozwolić na dalszą pracę z listą.
6. Jeśli plik nie istnieje, rozpocząć od pustej listy.

PRZYKŁADOWY PLIK

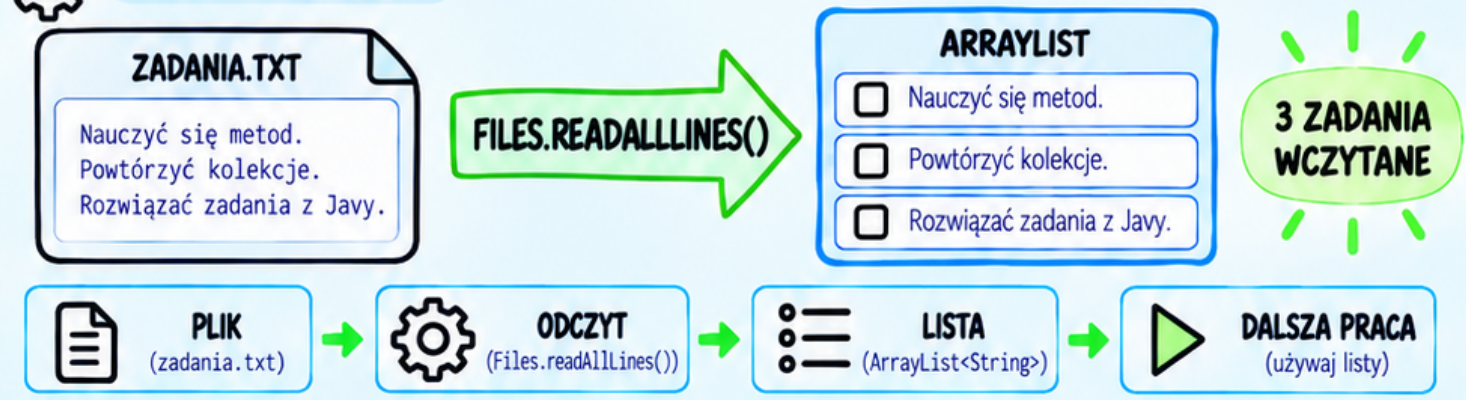
zadania.txt

```
Nauczyć się metod.
Powtórzyć kolekcje.
Rozwiązać zadania z Javy.
```

OCZEKIWANY WYNIK

```
Wczytano zapisane zadania.
Liczba zadań: 3
1. Nauczyć się metod.
2. Powtórzyć kolekcje.
3. Rozwiązać zadania z Javy.
```

SCHEMAT DZIAŁANIA



PODPOWIEDŹ

Wykorzystaj:

- `Files.exists()`
- `Files.readAllLines()`

```
static ArrayList<String> wczytajZadania() {
    Path plik = Path.of("zadania.txt");
    if (!Files.exists(plik)) {
        return new ArrayList<>();
    }
    try {
        return new ArrayList<>(
            Files.readAllLines(
                plik,
                StandardCharsets.UTF_8
            )
        );
    } catch (IOException e) {
        System.out.println(
            "Nie udało się odczytać zadań."
        );
        return new ArrayList<>();
    }
}
```

JAK TO DZIAŁA?

1. **FILES.EXISTS()** – Sprawdza, czy plik znajduje się we wskazanym miejscu.
2. **FILES.READALLLINES()** – Odczytuje wszystkie linie pliku.
3. **NEW ARRAYLIST** – Tworzy listę, którą można dalej modyfikować.
4. **RETURN** – Zwraca przygotowaną listę do programu głównego.

WAŻNE

Jeśli plik nie istnieje, program powinien utworzyć pustą listę, zamiast zgłaszać nieobsłużony błąd. Jeśli odczyt istniejącego pliku się nie powiedzie, poinformuj użytkownika o problemie. Nie nadpisuj uszkodzonego lub niedostępnego pliku pustą listą automatycznie.

SPRAWDŹ

- [ ] Czy program odczytuje istniejący plik?
- [ ] Czy wszystkie zadania wracają na listę?
- [ ] Czy zachowana jest kolejność elementów?
- [ ] Czy aplikacja działa bez pliku?
- [ ] Czy odzyskane zadania można usuwać?
- [ ] Czy program informuje o błędzie odczytu?

BONUS

Dodaj automatyczne wczytywanie zadań zaraz po uruchomieniu programu.

Wyświetl komunikat:

```
Witaj ponownie!
Twoje zadania są gotowe.
```

ZAPISANE DANE ZAWSZE MOGĄ WRÓCIĆ – WYKORZYSTAJ TO W SWOICH PROGRAMACH.

50 WYZWANIE 50 – TWOJA PIERWSZA KOMPLETNA APLIKACJA

50 wyzwań. Jedna aplikacja. Twoje własne rozwiązanie.

CEL WYZWANIA

Zbuduj kompletny menedżer zadań, który pozwala zarządzać listą obowiązków i zachowuje informacje po zamknięciu programu.

TWÓJ PROJEKT

1. Dodawanie nowych zadań.
2. Wyświetlanie wszystkich zadań.
3. Usuwanie wybranych zadań.
4. Zapisywanie listy do pliku.
5. Odczytywanie zapisanych zadań.
6. Obsługę nieprawidłowych danych.
7. Bezpieczne zakończenie działania.

MENU GŁÓWNE

```
=== MENEDŻER ZADAŃ ===
1. Dodaj zadanie
2. Pokaż wszystkie zadania
3. Usuń zadanie
4. Zapisz zadania
5. Wczytaj zadania
6. Zakończ
Wybierz opcję:
```

MENEDŻER ZADAŃ

```
=== MENEDŻER ZADAŃ ===
1. Dodaj zadanie
2. Pokaż wszystkie zadania
3. Usuń zadanie
4. Zapisz zadania
5. Wczytaj zadania
6. Zakończ

Wybierz opcję:
```

```
TWOJE ZADANIA (2)
-------------------------
1. Powtórzyć klasy i obiekty
2. Ukończyć projekt Java
```

JAK TO WSZYSTKO DZIAŁA?

SCANNER
Odczytuje dane od użytkownika.

↓

MENU I WARUNKI
Wyświetla menu i obsługuje wybór.

↓

METODY
Realizują poszczególne operacje.

↓

ARRAYLIST
Przechowuje listę zadań w pamięci.

↓

ZAPIS / ODCZYT PLIKU
Zapisuje i odczytuje dane z pliku.

ETAP 01 STRUKTURA PROGRAMU

1. Utwórz klasę: `MenedzerZadan`.
2. Wykorzystaj w niej: `ArrayList<String>` do przechowywania zadań.
3. Utwórz obiekt `Scanner` do odczytywania danych użytkownika.

ETAP 02 PODZIAŁ NA METODY

- `pokazMenu()` — Wyświetla dostępne opcje.
- `dodajZadanie()` — Pobiera i dodaje nowe zadanie.
- `pokazZadania()` — Wyświetla wszystkie zadania wraz z numerami.
- `usunZadanie()` — Usuwa zadanie na podstawie numeru.
- `zapiszZadania()` — Zapisuje listę do pliku tekstowego.
- `wczytajZadania()` — Odczytuje zadania z pliku.

ETAP 03 GŁÓWNA PĘTLA

Użyj pętli `do-while` lub `while`, aby menu było wyświetlane wielokrotnie. Pętla powinna zakończyć się po wybraniu opcji 6.

ETAP 04 OBSŁUGA BŁĘDÓW

- Wpisaniem tekstu zamiast numeru menu.
- Wybraniem nieistniejącej opcji.
- Usunięciem zadania o nieprawidłowym numerze.
- Dodaniem pustego zadania.
- Problemami podczas zapisu i odczytu pliku.

ETAP 05 PRZECHOWYWANIE DANYCH

- Przy uruchomieniu aplikacji spróbuj odczytać plik: `zadania.txt`.
- Po pomyślnym odczycie wyświetl odzyskane informacje.
- Przed zamknięciem programu zapisz aktualną listę. Jeśli zapis się nie powiedzie, poinformuj użytkownika i pozwól ponowić próbę zamiast bez ostrzeżenia utracić zmiany.

PRZYKŁAD DZIAŁANIA

```
=== MENEDŻER ZADAŃ ===
Wybierz opcję: 1
Podaj nowe zadanie:
Powtórzyć klasy i obiekty
Zadanie zostało dodane.

Wybierz opcję: 1
Podaj nowe zadanie:
Ukończyć projekt Java
Zadanie zostało dodane.

Wybierz opcję: 2
TWOJE ZADANIA:
1. Powtórzyć klasy i obiekty
2. Ukończyć projekt Java

Wybierz opcję: 4
Zadania zostały zapisane.
```

PODPOWIEDŹ

- `Scanner` — wejście danych.
- `if/else` — warunki.
- `switch` — wybór operacji.
- `while` — menu programu.
- `ArrayList` — lista zadań.
- Metody — organizacja kodu.
- `try-catch` — obsługa wyjątków.
- `Files.write()` — zapis.
- `Files.readAllLines()` — odczyt.

SPRAWDŹ SWOJĄ APLIKACJĘ

1. Czy można dodawać zadania?
2. Czy można wyświetlać listę?
3. Czy można usuwać elementy?
4. Czy zadania mają prawidłowe numery?
5. Czy błędne dane są obsługiwane?
6. Czy puste zadania są odrzucane?
7. Czy lista jest zapisywana do pliku?
8. Czy lista jest odczytywana po ponownym uruchomieniu?
9. Czy program umożliwia bezpieczne zakończenie?
10. Czy kod jest podzielony na czytelne metody?

BONUS – POZIOM WYŻEJ

Dodaj status każdego zadania:

```
[ ] -- Niewykonane.
[X] -- Wykonane.
```

- Umożliwiaj użytkownikowi oznaczanie zadań jako ukończonych.
- Aby zachować status po ponownym uruchomieniu, zmodyfikuj sposób zapisu i odczytu danych.

50 / 50 WYZWAŃ

Gratulacje! Jeśli Twój program działa, stworzyłeś własną aplikację wykorzystującą wiele podstawowych mechanizmów Javy.

05 MODUŁ 05 — SPRAWDŹ SWOJE UMIEJĘTNOŚCI

Sprawdź, czego nauczyłeś się o plikach i wyjątkach

CO JUŻ POTRAFISZ?

- [ ] Potrafię obsługiwać wyjątki za pomocą try-catch.
- [ ] Rozumiem znaczenie NumberFormatException.
- [ ] Umiem sprawdzać poprawność danych.
- [ ] Potrafię utworzyć menu odporne na błędne wybory.
- [ ] Umiem zapisywać tekst do pliku.
- [ ] Potrafię odczytywać zawartość pliku.
- [ ] Rozumiem różnicę między zapisem a dopisywaniem.
- [ ] Umiem analizować dane z pliku tekstowego.
- [ ] Potrafię połączyć metody i kolekcje w jednej aplikacji.
- [ ] Umiem stworzyć prosty program zapisujący i odczytujący zadania.

MINI-SPRAWDZIAN KOŃCOWY

ZADANIE: ZAPISYWANIE WYDATKÓW

Napisz program, który umożliwia zapisywanie codziennych wydatków.

1. Pobierać nazwę wydatku.
2. Pobierać jego kwotę.
3. Dodawać dane do kolekcji.
4. Wyświetlać wszystkie wydatki.
5. Obliczać ich łączną wartość.
6. Zapisywać dane do pliku.
7. Odczytywać zapisane wydatki.

PRZYKŁADOWE DANE

```
Chleb — 6.50 zł.
Mleko — 4.80 zł.
Owoce — 12.20 zł.
```

OCZEKIWANY WYNIK

```
Lista wydatków:
Chleb: 6.50 zł
Mleko: 4.80 zł
Owoce: 12.20 zł

Łączna kwota: 23.50 zł
```

PODPOWIEDŹ

- Możesz wykorzystać klasę Wydatek zawierającą: nazwa, kwota.
- Do przechowywania danych użyj: ArrayList<Wydatek>.
- Do zapisu wybierz prosty, ustalony format tekstowy, na przykład oddzielając nazwę i kwotę średnikiem.
- Dla obliczeń finansowych w rzeczywistej aplikacji warto zastosować BigDecimal.

PYTANIA DO REFLEKSJI

1. Dlaczego program powinien obsługiwać wyjątki?
2. Czym różni się zapis danych od ich odczytu?
3. Dlaczego warto dzielić aplikację na metody?
4. Jakie informacje powinny być przechowywane w pliku?
5. Co możesz zrobić, aby Twój kod był bardziej czytelny?

TWÓJ POSTĘP

41 42 43 44 45 46 47 48 49 50

☐ MODUŁ 05 — UKOŃCZONY

PODSUMOWANIE — WSZYSTKO ŁĄCZY SIĘ W JEDNĄ CAŁOŚĆ

WYJĄTKI
Pomagają tworzyć bezpieczne i odporne na błędy programy.

PLIKI
Umożliwiają trwałe przechowywanie i odczytywanie danych.

MINIAPLIKACJE
Łączą poznane elementy w praktyczne projekty, które rozwiązują realne problemy.

SAMODZIELNE PROGRAMOWANIE

Właśnie zakończyłeś ostatni moduł! Potrafisz już budować programy, które przetwarzają dane, obsługują błędy i zapisują informacje.

GRATULACJE!
UKOŃCZYŁEŚ 50 WYZWAŃ Z JAVY!

To dopiero początek Twojej programistycznej drogi.

50 WYZWAŃ ZA TOBĄ!

Przeszedłeś drogę od pierwszego komunikatu w konsoli do własnej aplikacji zapisującej dane w pliku.
Poznałeś podstawowe mechanizmy języka Java i nauczyłeś się wykorzystywać je do rozwiązywania praktycznych problemów.
Każde wyzwanie było kolejnym krokiem w rozwijaniu Twoich umiejętności.
Najważniejsze jest to, że nie tylko czytałeś o programowaniu — samodzielnie tworzyłeś kod.

CZEGO SIĘ NAUCZYŁEŚ?

01 PODSTAWY
Zmienne, dane, operatory i pierwsze programy.

→

02 WARUNKI
Podejmowanie decyzji za pomocą instrukcji warunkowych.

→

03 PĘTLE I TABLICE
Automatyzowanie działań i analizowanie wielu danych.

→

04 METODY I OBIEKTY
Organizowanie kodu, tworzenie klas i korzystanie z kolekcji.

→

05 PLIKI I APLIKACJE
Obsługa błędów, zapisywanie informacji i budowanie własnych projektów.

TWÓJ KOLEJNY KROK

Nie zatrzymuj się na tych ćwiczeniach. Spróbuj teraz:

1. **ROZBUDOWAĆ SWÓJ PROJEKT**
   Dodaj nowe funkcje do menedżera zadań.
2. **STWORZYĆ NOWĄ APLIKACJĘ**
   Zbuduj kalkulator, dziennik wydatków lub katalog książek.
3. **WRÓCIĆ DO TRUDNIEJSZYCH WYZWAŃ**
   Spróbuj rozwiązać je ponownie bez podpowiedzi.
4. **POZNAWAĆ KOLEJNE ZAGADNIENIA**
   Rozwijaj znajomość programowania obiektowego, kolekcji, testów i narzędzi programistycznych.
5. **ĆWICZYĆ REGULARNIE**
   Nawet kilkanaście minut praktyki dziennie może pomóc utrwalać umiejętności.

TWOJE OSIĄGNIĘCIE

50 / 50
PRAKTYCZNYCH WYZWAŃ

5 MODUŁÓW UKOŃCZONYCH

MÓJ NAJWIĘKSZY SUKCES

.....

.....

MÓJ NASTĘPNY PROJEKT

.....

.....

PAMIĘTAJ!

Nie musisz znać wszystkiego, aby tworzyć wartościowe programy.
Wystarczy, że będziesz dalej ćwiczyć, eksperymentować i rozwijać swoje pomysły.

Powodzenia w dalszej nauce Javy!