

# Petle

While

Do while

For

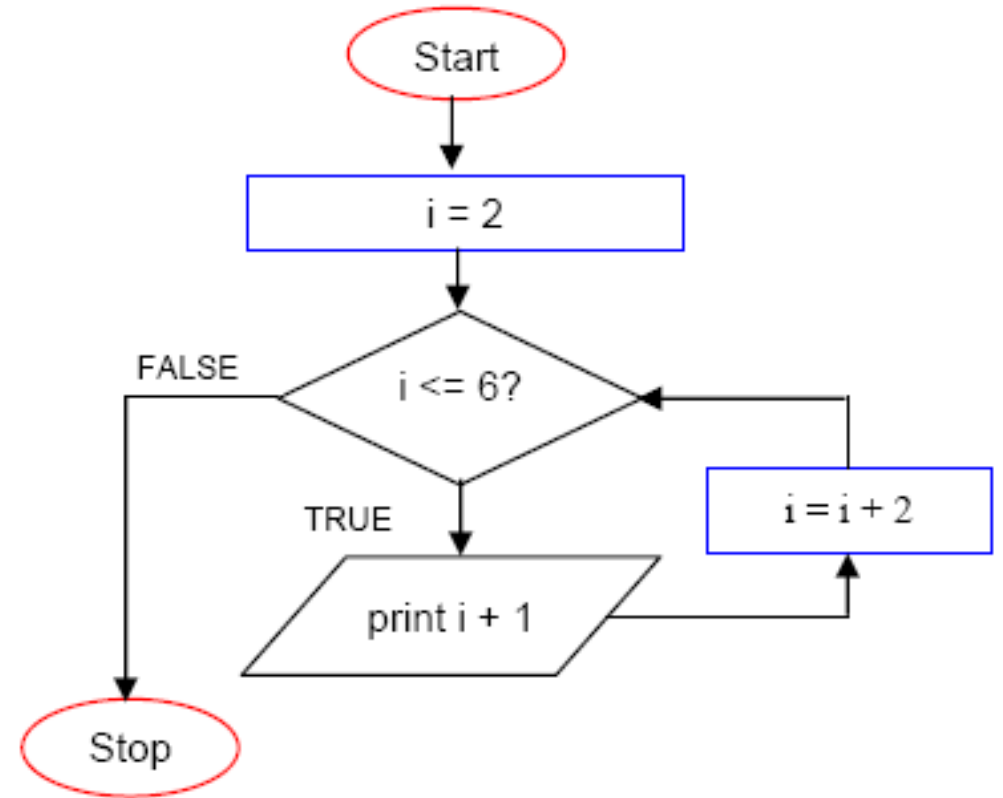
# Problem

```
public class Problem {  
    public static void main(String[] args) {  
        System.out.println(1);  
        System.out.println(2);  
        System.out.println(3);  
        System.out.println(4);  
        System.out.println(5);  
    }  
}
```

# Pętle

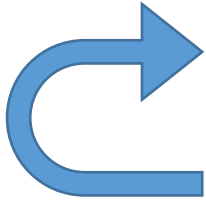
Pętle są rozwiązaniem problemu powtarzalnych operacji w naszym kodzie, np.:

- wyświetlanie/tworzenie wartości z inkrementacją o określoną wartość
- wczytywanie pliku wiersz po wierszu
- wielokrotne wczytywanie danych od użytkownika



# while

- Pętla while wykonuje się tak długo dopóki warunek w niej zawarty jest prawdziwy
- Warunek może być dowolnym wyrażeniem, które w wyniku daje wartość true/false – podobnie jak przy instrukcji if/else
- Łatwo popełnić błąd i zapisać pętlę nieskończoną – wtedy nasz program "się zawiesi"
- Ciało pętli wyznaczają nawiasy klamrowe, które można pominąć, jeśli do wykonania jest tylko jedna powtarzalna operacja



```
public class LoopWhile {  
    public static void main(String[] args) {  
        int i = 0;  
        while(i < 10) {  
            System.out.println(i);  
            i++;  
        }  
    }  
}
```

# Przykład

```
import java.util.Scanner;

public class LoopWhile {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int guessNumber = 123;
        System.out.println("Zgadnij Liczbę:");
        int userChoice = input.nextInt();


        while(userChoice != guessNumber) {
            System.out.println("Niestety nie zgadłeś, spróbuj ponownie:");
            userChoice = input.nextInt();
        }
        System.out.println("Bingo!");
        input.close();
    }
}
```

# Przykład

```
import java.util.Scanner;

public class LoopWhile {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int guessNumber = 123;
        System.out.println("Zgadnij liczbę:");
        int userChoice = input.nextInt();

        while(userChoice != guessNumber) {
            System.out.println("Niestety nie zgadłeś, spróbuj ponownie:");
            userChoice = input.nextInt();
        }
        System.out.println("Bingo!");
        input.close();
    }
}
```




# Przykład

```
import java.util.Scanner;

public class LoopWhile {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int guessNumber = 123;
        System.out.println("Zgadnij liczbę:");
        int userChoice = input.nextInt();

        while(userChoice != guessNumber) {
            System.out.println("Niestety nie zgadłeś, spróbuj ponownie:");
            userChoice = input.nextInt();
        }
        System.out.println("Bingo!");
        input.close();
    }
}
```



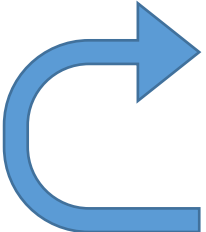
Problems Javadoc Declaration Console

<terminated> LoopWhile [Java Application] C:\Program Files\Java\j

Zgadnij liczbę:  
12  
Niestety nie zgadłeś, spróbuj ponownie:  
5  
Niestety nie zgadłeś, spróbuj ponownie:  
123  
Bingo!

## do while

- do while działa niemal identycznie jak while, również wymaga warunku logicznego i wykonuje się dopóki jest spełniony
- jedyna różnica polega na tym, że warunek jest sprawdzany po wykonaniu ciała pętli
- **Pętla do while zawsze wykona się co najmniej raz**



```
public class LoopDoWhile {  
    public static void main(String[] args) {  
        int i = 0;  
        do {  
            System.out.println(i);  
            i++;  
        } while(i < 10);  
    }  
}
```




# while vs do while

```
import java.util.Scanner;

public class LoopWhile {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int guessNumber = 123;
        System.out.println("Zgadnij Liczbę:");
        int userChoice = input.nextInt();

        while(userChoice != guessNumber) {
            System.out.println("Spróbuj ponownie");
            userChoice = input.nextInt();
        }
        System.out.println("Bingo!");
        input.close();
    }
}
```




```
import java.util.Scanner;

public class LoopDoWhile {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int guessNumber = 123;
        int userChoice = 0;

        do {
            System.out.println("Zgadnij Liczbę:");
            userChoice = input.nextInt();
        } while(userChoice != guessNumber);

        System.out.println("Bingo!");
        input.close();
    }
}
```



## Ćwiczenie

Napisz program, który wczytuje od użytkownika pięć liczb całkowitych, a na końcu wyświetla ich sumę. Wykorzystaj pętlę while w celu uniknięcia powtarzalnego kodu.

## Ćwiczenie\*

Napisz program, w którym wczytasz od użytkownika liczbę całkowitą  $N$  z przedziału  $[-100; 100]$ .

Jeśli liczba jest dodatnia, wyświetl na ekranie wszystkie liczby z przedziału  $[N, 100]$ , jeśli liczba jest ujemna to z przedziału  $[-100, N]$ .

Jeśli użytkownik poda liczbę z poza zakresu  $[-100, 100]$  wyświetl komunikat o błędnej wartości.

# for

- Pętla for dedykowana jest głównie do iterowania, gdy znamy ilość powtórzeń
- Idealnie nadaje się do przeglądania kolekcji elementów – np. tablic
- Pętla for zawiera w sobie trzy elementy:
  - **deklaracja licznika**
  - **warunek stopu**
  - **modyfikator licznika**

przy czym każdy z tych elementów jest opcjonalny

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

The diagram illustrates the three components of a C++ for loop with arrows pointing from labels to the code:

- deklaracja licznika** (counter declaration) points to `int i = 0`
- warunek stopu** (stop condition) points to `i < 10`
- modyfikacja licznika** (counter modification) points to `i++`

# Przykład

```
public class ForLoops {  
    public static void main(String[] args) {  
        int[] numbers = new int[5];  
        numbers[0] = 0;  
        numbers[1] = 10;  
        numbers[2] = 20;  
        numbers[3] = 30;  
        numbers[4] = 40;  
        System.out.println(numbers[0]);  
        System.out.println(numbers[1]);  
        System.out.println(numbers[2]);  
        System.out.println(numbers[3]);  
        System.out.println(numbers[4]);  
    }  
}
```

# Przykład

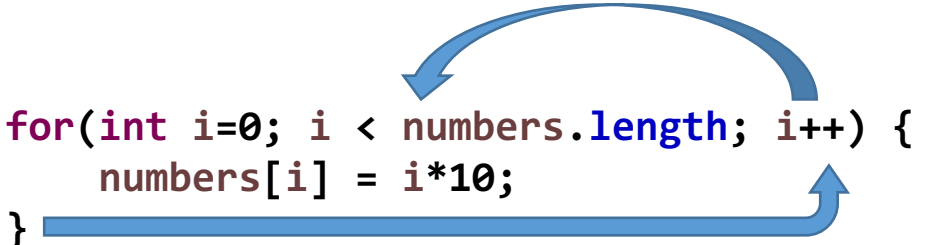
```
public class ForLoops {  
    public static void main(String[] args) {  
        int[] numbers = new int[5];  
        numbers[0] = 0;  
        numbers[1] = 10;  
        numbers[2] = 20;  
        numbers[3] = 30;  
        numbers[4] = 40;  
        System.out.println(numbers[0]);  
        System.out.println(numbers[1]);  
        System.out.println(numbers[2]);  
        System.out.println(numbers[3]);  
        System.out.println(numbers[4]);  
    }  
}
```

```
public class ForLoops {  
    public static void main(String[] args) {  
        int[] numbers = new int[5];  
  
        for(int i=0; i < numbers.length; i++) {  
            numbers[i] = i*10;  
        }  
  
        for(int i=0; i<numbers.length; i++) {  
            System.out.println(numbers[i]);  
        }  
    }  
}
```

# Przykład

```
public class ForLoops {  
    public static void main(String[] args) {  
        int[] numbers = new int[5];  
        numbers[0] = 0;  
        numbers[1] = 10;  
        numbers[2] = 20;  
        numbers[3] = 30;  
        numbers[4] = 40;  
        System.out.println(numbers[0]);  
        System.out.println(numbers[1]);  
        System.out.println(numbers[2]);  
        System.out.println(numbers[3]);  
        System.out.println(numbers[4]);  
    }  
}
```

```
public class ForLoops {  
    public static void main(String[] args) {  
        int[] numbers = new int[5];  
  
        for(int i=0; i < numbers.length; i++) {  
            numbers[i] = i*10;  
        }  
  
        for(int i=0; i<numbers.length; i++) {  
            System.out.println(numbers[i]);  
        }  
    }  
}
```



## for each

- Pętla for-each służy wyłącznie do przeglądania kolekcji obiektów (np. tablic)
- Pozwala przejść przez całą kolekcję korzystając z krótszego, bardziej opisowego zapisu
- Operuje na kopiach referencji, więc nadaje się do odczytu, ale nie umieszczania elementów w tablicy

```
public class LoopForEach {  
    public static void main(String[] args) {  
        String[] names = {"Jan", "Basia", "Wojtek"};  
  
        for(String name: names) {  
            System.out.println(name);  
        }  
    }  
}
```



# Przykład

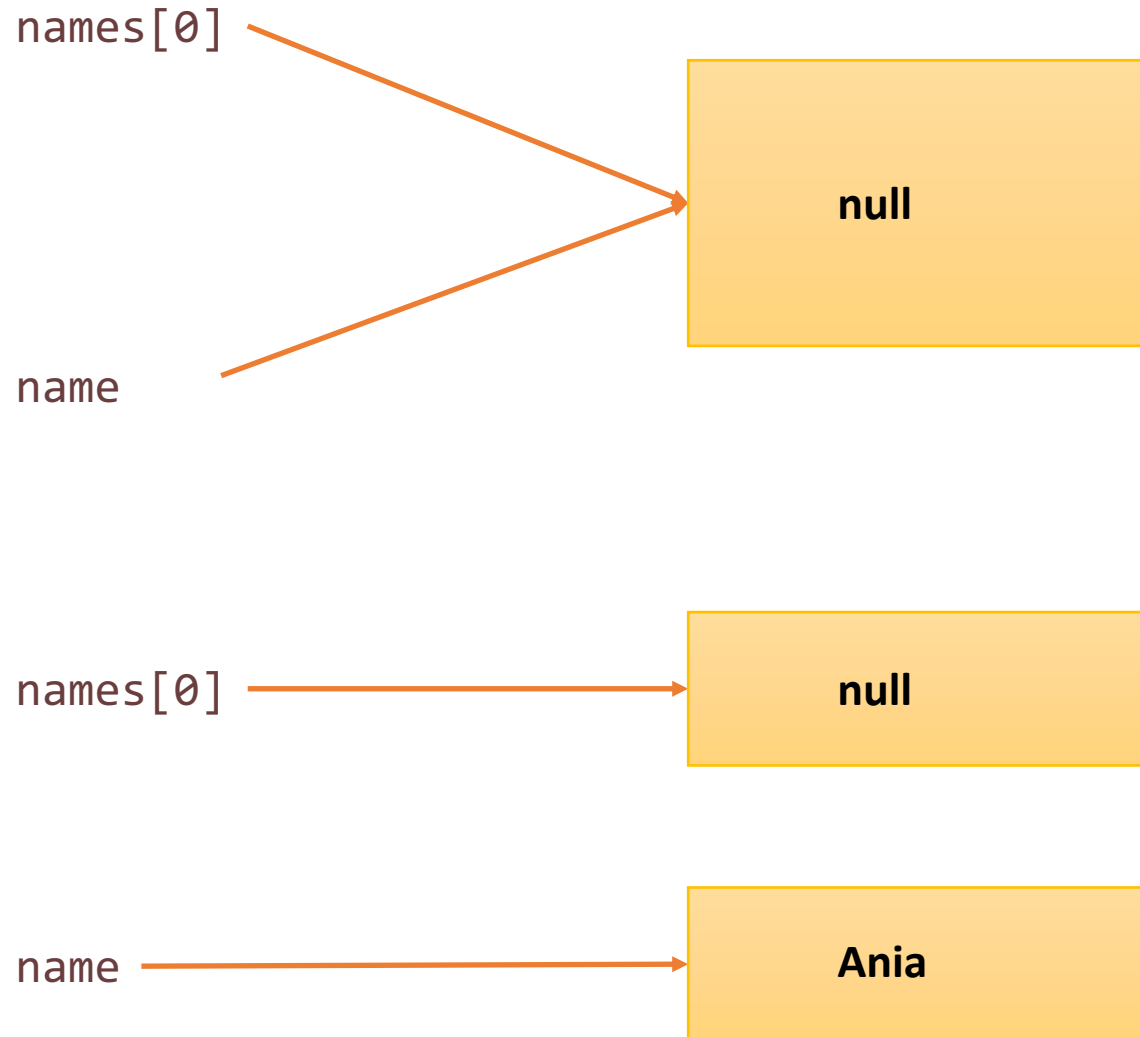
OK

```
public class LoopForEach {  
    public static void main(String[] args) {  
        String[] names = {"Jan", "Basia", "Wojtek"};  
  
        for(String name: names) {  
            System.out.println(name);  
        }  
    }  
}
```

Błąd

```
import java.util.Scanner;  
  
public class LoopForEach {  
    public static void main(String[] args) {  
        String[] names = new String[3];  
        Scanner sc = new Scanner(System.in);  
  
        for(String name: names) {  
            name = sc.nextLine();  
        }  
        sc.close();  
  
        System.out.println(names[0]); //null  
    }  
}
```

# Przykład



## Błąd

```
import java.util.Scanner;

public class LoopForEach {
    public static void main(String[] args) {
        String[] names = new String[3];
        Scanner sc = new Scanner(System.in);

        for(String name: names) {
            name = sc.nextLine(); //Ania
        }
        sc.close();

        System.out.println(names[0]); //null
    }
}
```

- **break** – przerwanie i natychmiastowe wyjście z pętli

```
int i = 0;
while (i < 100) {
    if (i == 2) {
        break;
    }
    System.out.println(i);
    i++;
}
```

0  
1  
koniec

- **continue** – przejście do kolejnej iteracji pętli

```
int i = 0;
while (i < 100) {
    if (i % 2 != 0) {
        i++;
        continue;
    }
    System.out.println(i);
    i++;
}
```

0  
2  
4  
6...

## Ćwiczenie

Napisz program, w którym wczytasz od użytkownika liczbę będącą rozmiarem tablicy liczb zmiennoprzecinkowych. Następnie korzystając z pętli for wypełnij tę tablicę liczbami wczytanymi od użytkownika, a na końcu używając pętli for each oblicz sumę kwadratów liczb zapisanych w tablicy i ją wyświetl

## Ćwiczenie\*

Stwórz i wypełnij tablicę dwuwymiarową liczbami w taki sposób jak zaprezentowano poniżej. Następnie oblicz sumę liczb na przekątnych.

0	1	4	9	16	25	36	49	64	81	100
1	4	9	16	25	36	49	64	81	100	121
4	9	16	25	36	49	64	81	100	121	144
9	16	25	36	49	64	81	100	121	144	169
16	25	36	49	64	81	100	121	144	169	196
25	36	49	64	81	100	121	144	169	196	225
36	49	64	81	100	121	144	169	196	225	256
49	64	81	100	121	144	169	196	225	256	289
64	81	100	121	144	169	196	225	256	289	324
81	100	121	144	169	196	225	256	289	324	361
100	121	144	169	196	225	256	289	324	361	400