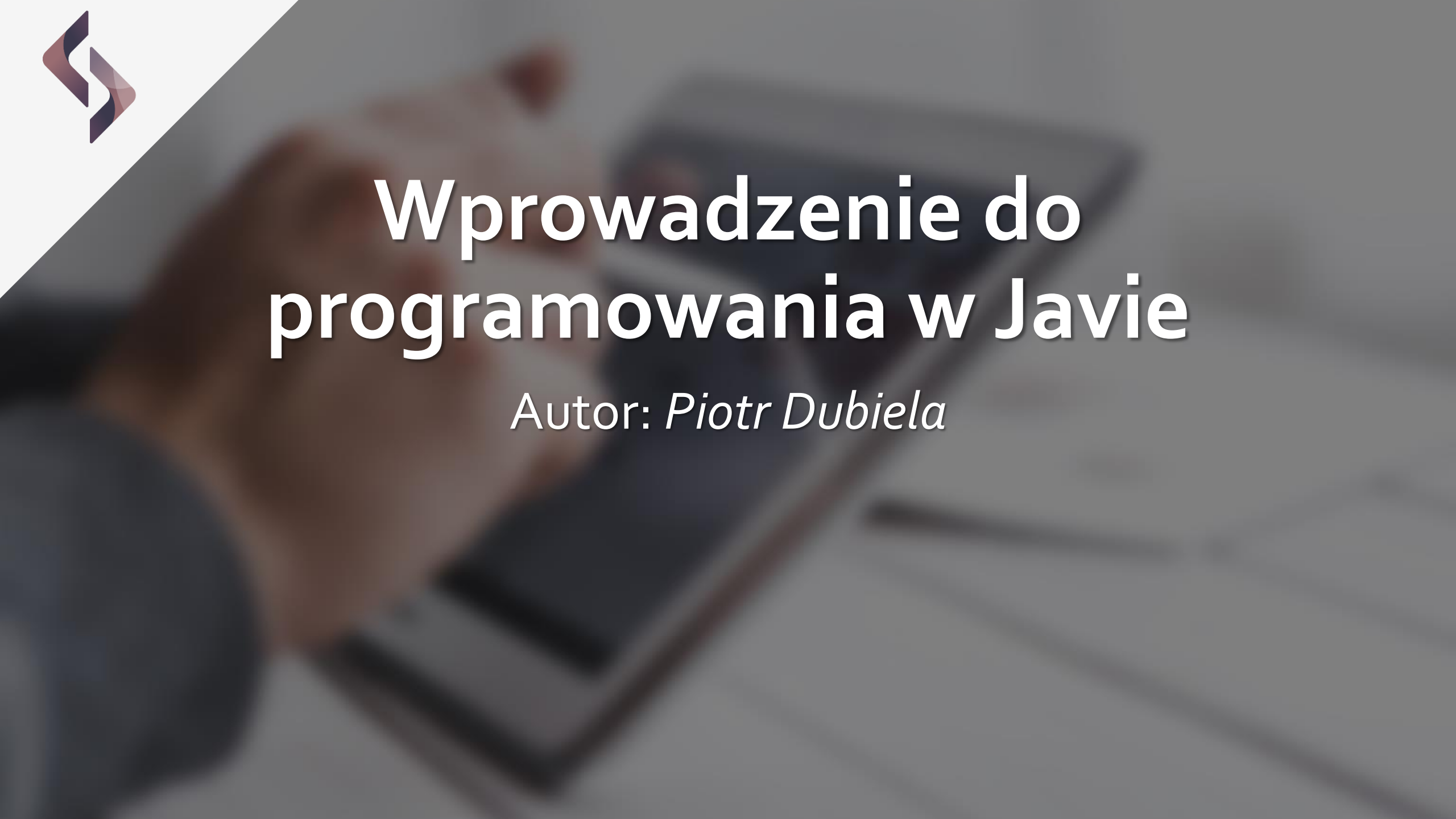


Wprowadzenie do programowania w Javie

Autor: *Piotr Dubiela*

Co to jest programowanie?

Co to jest programowanie?

Programowanie komputerów – proces **projektowania**, **tworzenia**, **testowania** i **utrzymywania kodu źródłowego** programów komputerowych lub urządzeń mikroprocesorowych (mikrokontrolery). Kod źródłowy jest napisany w języku programowania, z użyciem określonych reguł, może on być modyfikacją istniejącego programu lub czymś zupełnie nowym. Programowanie wymaga dużej wiedzy i doświadczenia w wielu różnych dziedzinach, jak projektowanie aplikacji, algorytmika, struktury danych, języki programowania i narzędzia programistyczne, kompilatory, czy sposób działania podzespołów komputera. W inżynierii oprogramowania programowanie (implementacja) jest tylko jednym z etapów powstawania programu.

Dlaczego Java?

- Język obiektowy
- Niezależność od platformy (*Write Once, Run Anywhere)*
- Automatyczne zarządzanie pamięcią
- Prostota
- Popularność
  - Duża społeczność
  - Olbrzymia ilość literatury
  - Duże zapotrzebowanie na rynku pracy

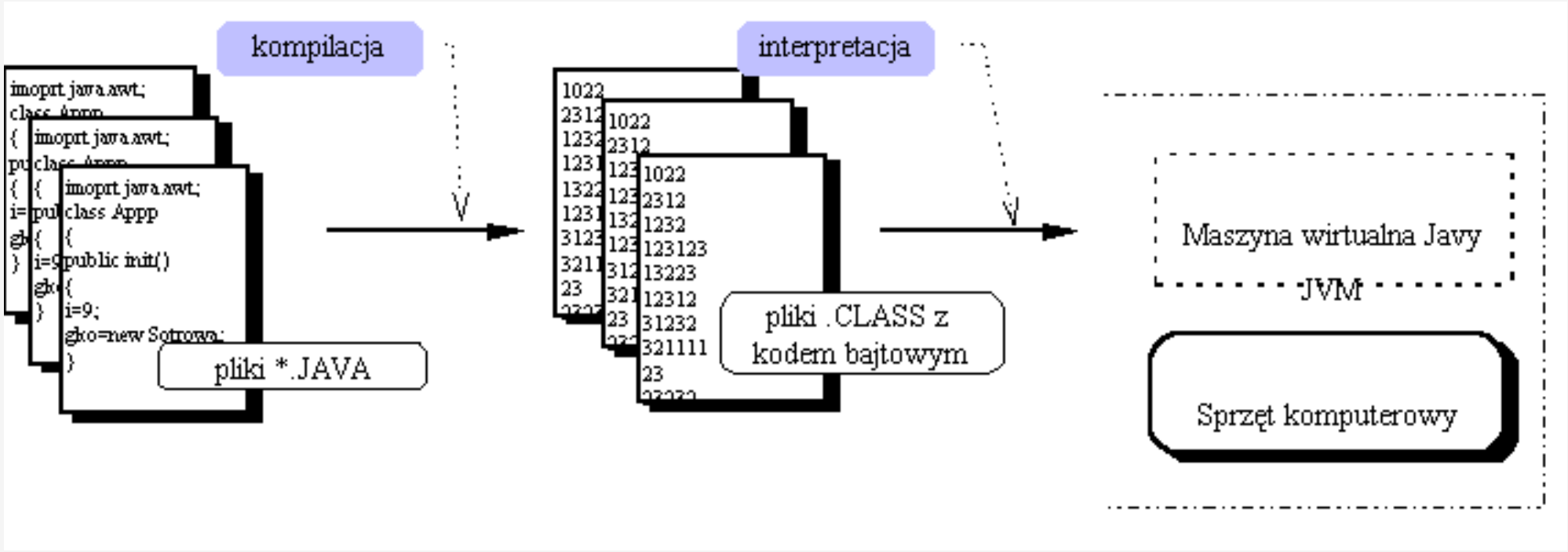
Historia języka Java

Początek języka **Java** możemy określić jako rok 1991. Wtedy firma Sun z Patrickiem Naughtonem oraz Jamesem Goslingiem na czele postanowili stworzyć prosty i niewielki język, który mógłby być uruchamiany na wielu platformach z różnymi parametrami. Projekt zatytułowano **Green**.

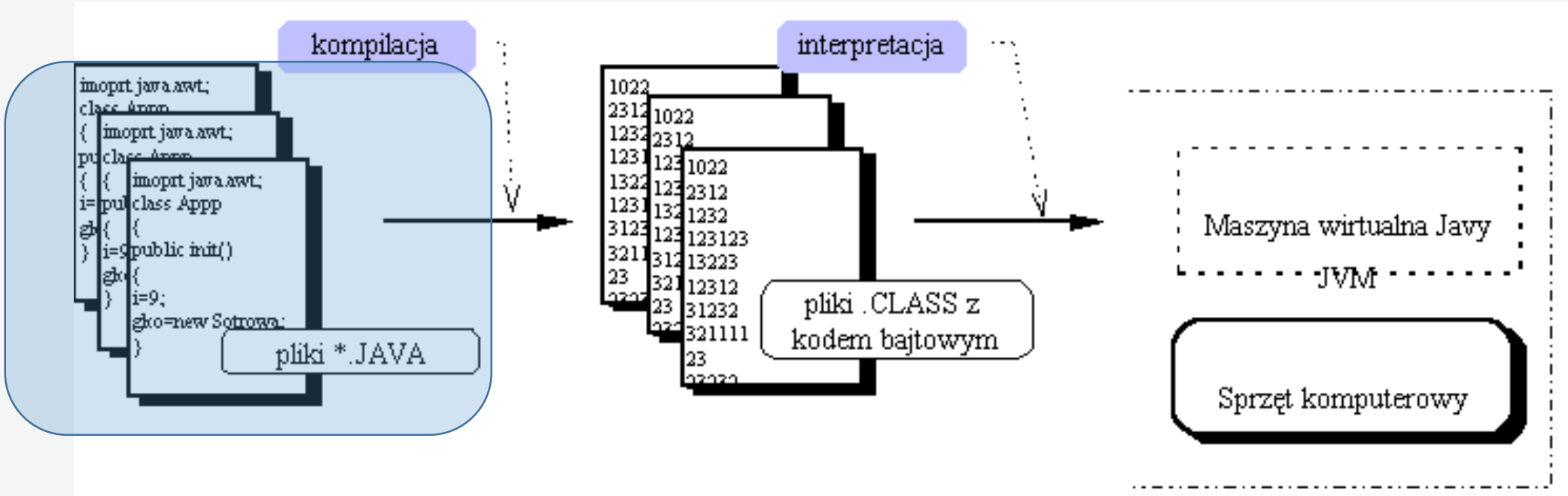

Historia wersji języka JAVA

JDK Beta	1994	
JDK 1.0	1996	Wersja inicjalna
JDK 1.1	1997	Klasy wewnętrzne, JDBC, RMI, refleksja, JIT, AWT
J2SE 1.2	1998	Swing, kolekcje
J2SE 1.3	2000	JNDI, debugger
J2SE 1.4	2002	Nowa obsługa operacji Input/Output, wyrażenia regularne, parser XML, logowanie, exception chaining
J2SE 5.0	2005	Typy generyczne, typy wyliczeniowe, autoboxing, varargs, pętla foreach, importy statyczne, zmiany w wielowątkowości
Java SE 6	2006	Zakończenie obsługi Windows 9x, zmiany w Swingu, usprawnienie JDBC, JAX-WS, zmiany w JVM
Java SE 7	2011	Zmiana obsługi plików, zmiany w języku (switch, operator diamond, catch wielu wyjątków). Nowości w wielowątkowości.
Java SE 8	2014	Wyrażenia lambda, strumienie, nowe api dla dat i czasu. Zakończenie obsługi Windows XP
Java SE 9	2017	Prywatne metody w interfejsach, Ulepszenia strumieni, klient HTTP2

Jak działa programowanie w Javie?

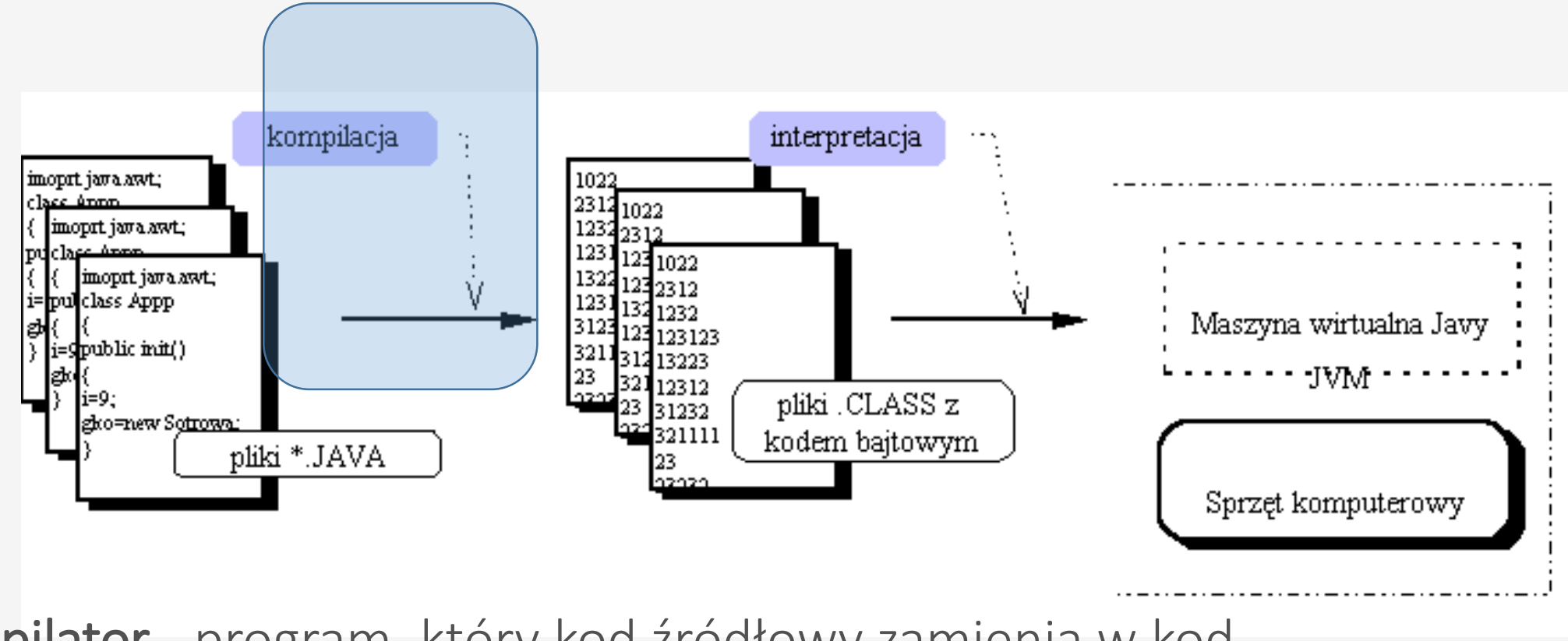


Jak działa programowanie w Javie?



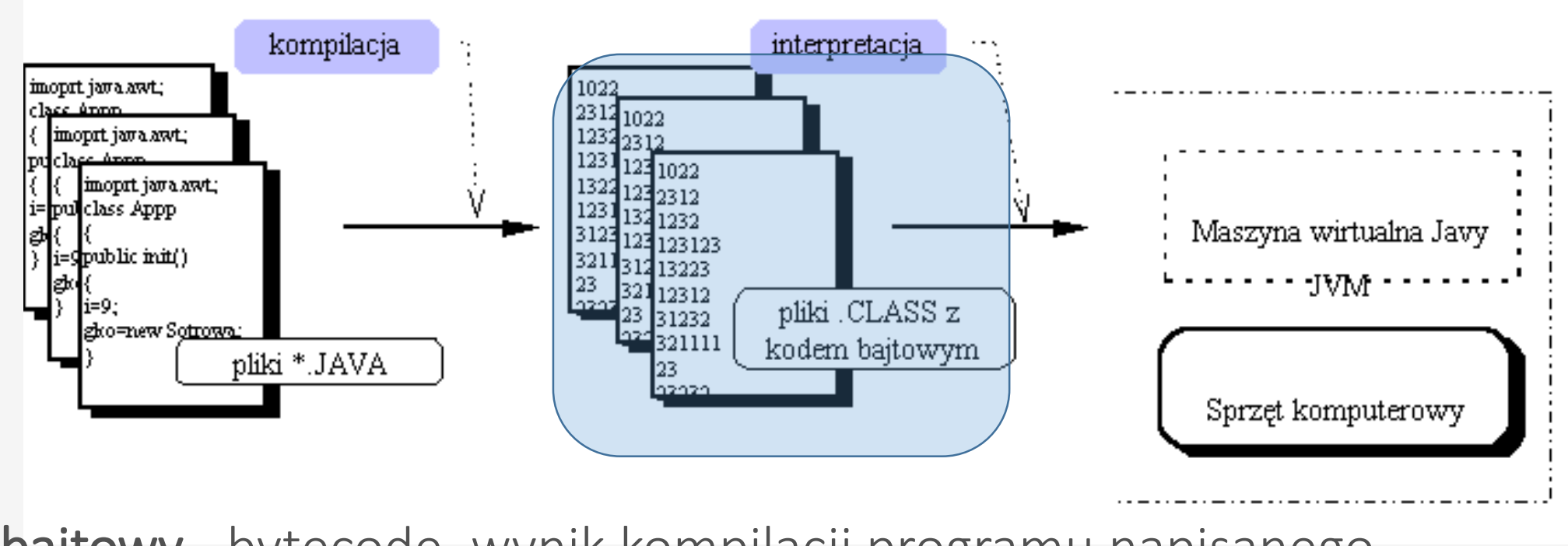
Kod źródłowy programu - zapis programu komputerowego przy pomocy określonego języka programowania, opisujący operacje jakie powinien wykonać komputer na zgromadzonych lub otrzymanych danych. Kod źródłowy jest wynikiem pracy programisty i pozwala wyrazić w czytelnej dla człowieka formie strukturę oraz działanie programu komputerowego.

Jak działa programowanie w Javie?



Kompilator - program, który kod źródłowy zamienia w kod napisany w innym języku. Oprócz tego kompilator ma za zadanie odnaleźć błędy leksykalne i semantyczne oraz dokonać optymalizacji kodu.

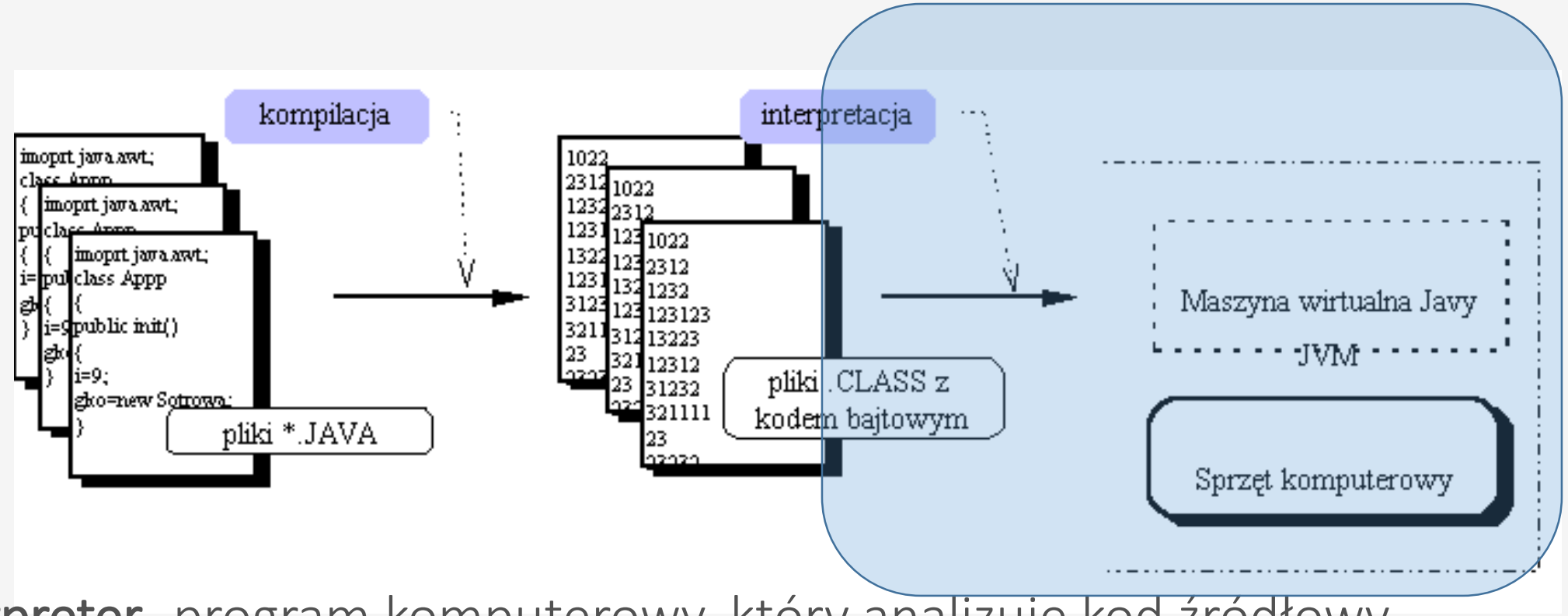
Jak działa programowanie w Javie?



Kod bajtowy - bytecode, wynik kompilacji programu napisanego

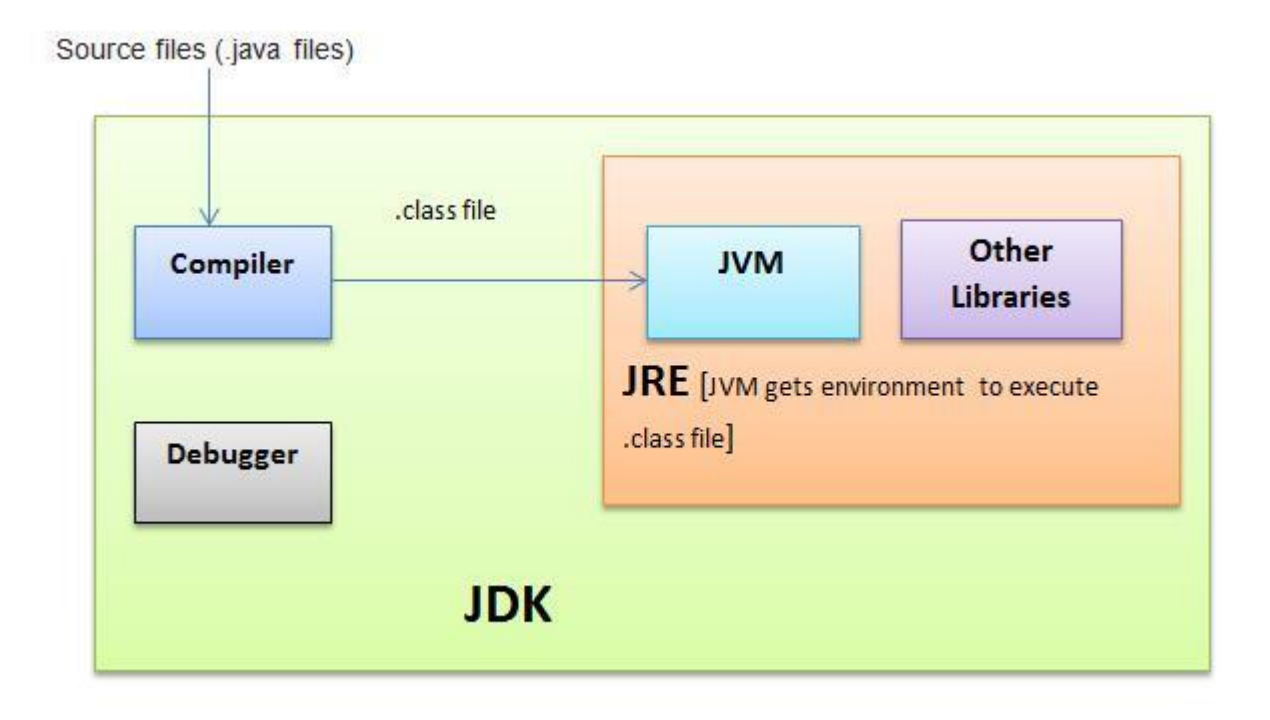
w Javie, kod ten jest zrozumiały dla środowiska uruchomieniowego Java (JVM)

Jak działa programowanie w Javie?



Interpreter- program komputerowy, który analizuje kod źródłowy programu, a przeanalizowane fragmenty wykonuje

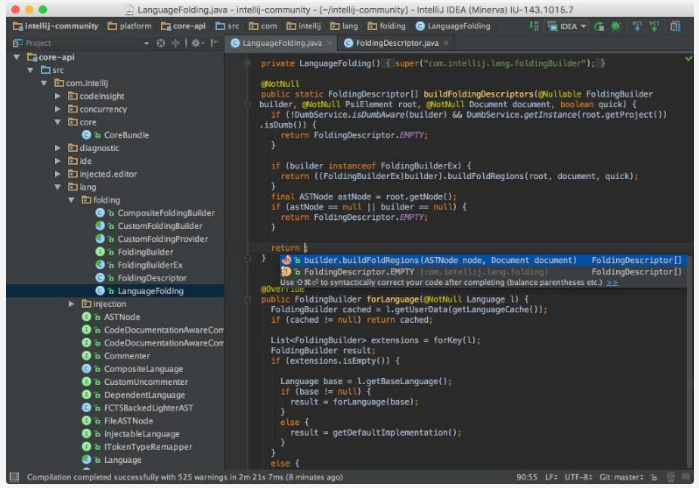
Jak działa programowanie w Javie?

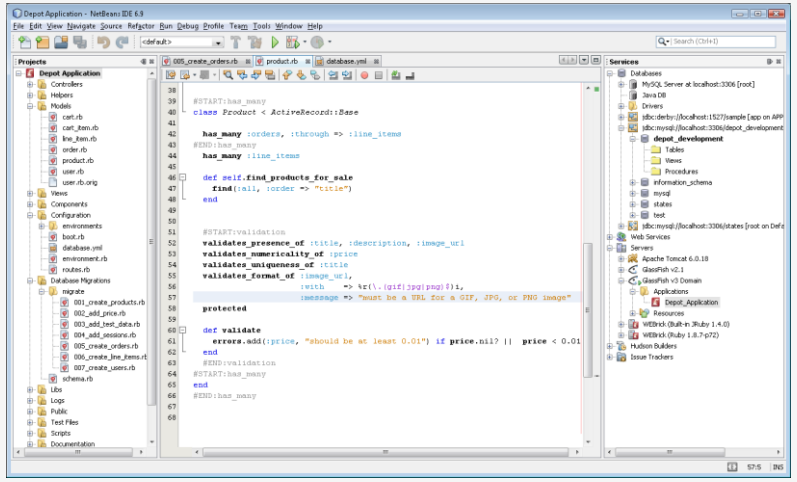


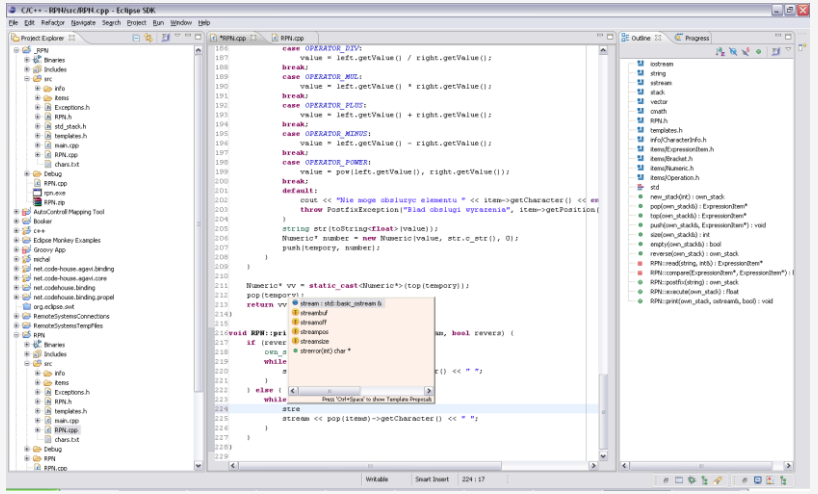
Środowisko do programowania

- Zainstalowane Java Development Kit w odpowiedniej wersji
  Sposób instalacji zależny od systemu operacyjnego. JDK można pobrać z strony Oracle lub OpenJDK.
  Ustawienie zmiennej środowiskowej JAVA_HOME

- Środowisko uruchomieniowe Java (JRE)
- Środowisko deweloperskie - IDE
  Darmowe narzędzia: IntelliJ IDEA, Eclipse, NetBeans



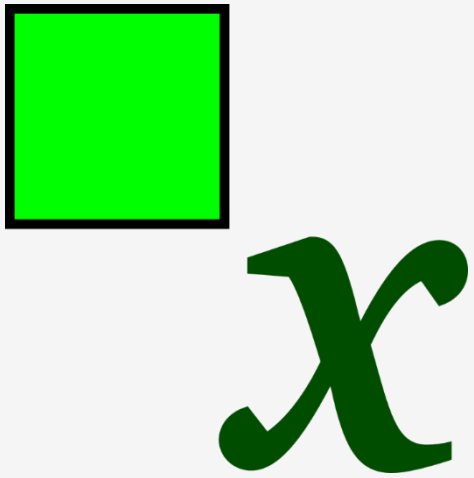




Pierwszy program ‚Hello World!'

1. *Uruchomienie IDE*
2. *Utworzenie nowej klasy*
3. *Kompilacja w IDE*
4. *Eksport do pliku .jar*
5. *Uruchomienie programu w eksploratorze systemowym*

Co to jest zmienna?



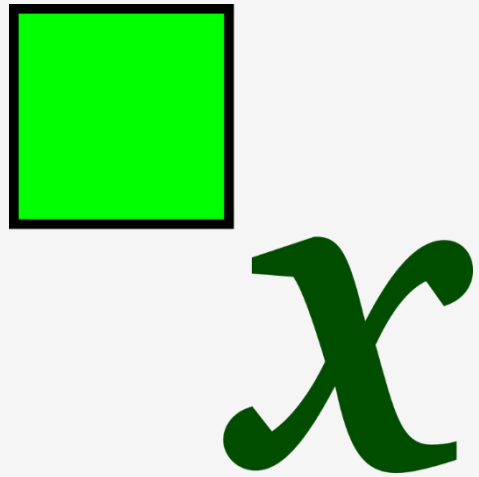
Co to jest zmienna?

Zmienna - konstrukcja programistyczna posiadająca trzy podstawowe atrybuty: symboliczną **nazwę**, **miejsce przechowywania** i **wartość**; pozwalająca w kodzie źródłowym odwoływać się przy pomocy nazwy do wartości lub miejsca przechowywania. Nazwa służy do identyfikowania zmiennej w związku z tym często nazywana jest identyfikatorem. Miejsce przechowywania przeważnie znajduje się w pamięci komputera i określane jest przez adres i długość danych. Wartość to zawartość miejsca przechowywania.

Jak wygląda zmienna w Javie?

Zmienna w Javie składa się z następujących pól:

{akcesor*} {typ zmiennej} {identyfikator zmiennej}

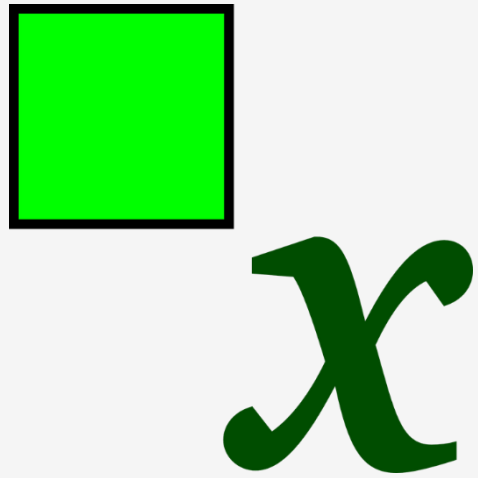


Jak wygląda zmienna w Javie?

Zmienna w Javie składa się z następujących pól:

{akcesor*} {typ zmiennej} {identyfikator zmiennej}

```
private int liczba;
```

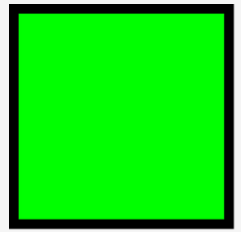


Cykl życia zmiennej

Zmienna w Javie przechodzi przez cykl życia składający się z następujących faz:

1) Deklaracja zmiennej

```
private int liczba;
```

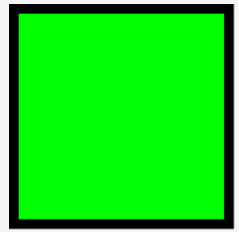


Cykl życia zmiennej

Zmienna w Javie przechodzi przez cykl życia składający się z następujących faz:

1) Deklaracja zmiennej
2) Inicjalizacja zmiennej

```
liczba = 5;
```

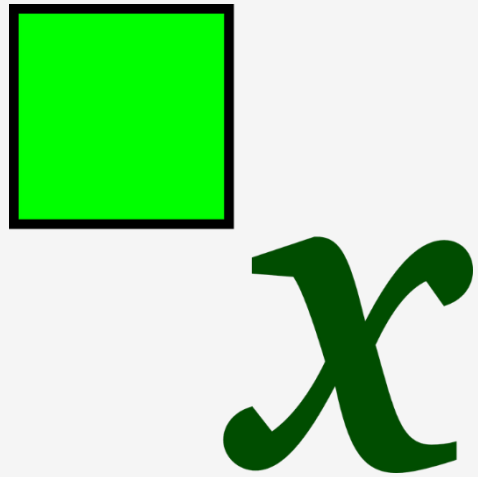


Cykl życia zmiennej

Zmienna w Javie przechodzi przez cykl życia składający się z następujących faz:

1) Deklaracja zmiennej
2) Inicjalizacja zmiennej
3) Użycie zmiennej

```
System.out.println(liczba);
```

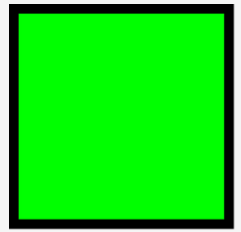


Cykl życia zmiennej

Zmienna w Javie przechodzi przez cykl życia składający się z następujących faz:

1) Deklaracja zmiennej
2) Inicjalizacja zmiennej
3) Użycie zmiennej
4) (opcjonalne) Nadpisanie zmiennej

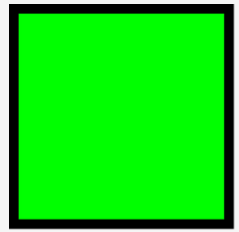
```
liczba = 1000;
```



Cykl życia zmiennej

Zmienna w Javie przechodzi przez cykl życia składający się z następujących faz:

1) Deklaracja zmiennej
2) Inicjalizacja zmiennej
3) Użycie zmiennej
4) (opcjonalne) Nadpisanie zmiennej
5) Śmierć zmiennej (usunięcie z pamięci komputera)

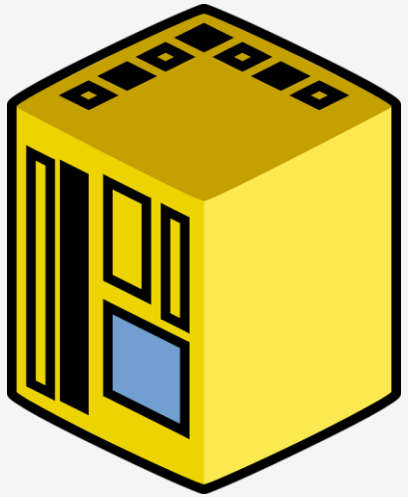


Pierwszy program ‚Hello World!'

1. *Uruchomienie IDE*
2. *Aktualizacja klasy HelloWorld:*
   - ❑ *Dodanie zmiennej ‚imie' (typu* String)
   - ❑ *Przypisanie jej swojego imienia*
   - ❑ *Użycie zmiennej w celu wyświetlenia jej w konsoli*
3. *Utworzenie nowej klasy PierwszyProgram*
   - ❑ *Korzystając z 3 zmiennych wyświetlić poniższy tekst :*

   Hello World!
   Mam na imię
   {Twoje imię}
4. ** Aktualizacja klasy PierwszyProgram, aby wyświetlić powyższy tekst przy użyciu jedynie 1 zmiennej*

Typy danych



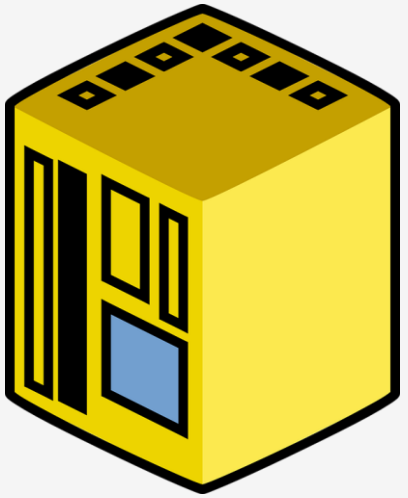
Co to jest typ danych?

Typ danych –opis rodzaju, struktury i zakresu wartości, jakie może przyjmować dany literał, zmienna, stała, argument, wynik funkcji lub wartośc.

Rodzaje typów danych w Javie

W Javie wyróżniamy następujące typy danych:

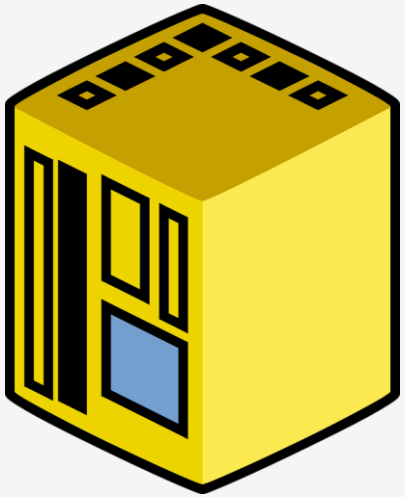
1) Typ liczbowy
    1) Całkowity
    2) Zmiennoprzecinkowy
2) Typ logiczny
    1) Prawda
    2) Fałsz
3) Typ znakowy
4) Typ tekstowy



Rodzaje typów danych w Javie

Java jest językiem programowania typowanym statycznie

- Typy zmiennych nadawane są w czasie kompilacji programu
- Łatwość wykrycia błędów w czasie kompilacji
- Konieczność deklaracji typów zmiennych przed ich inicjalizacją

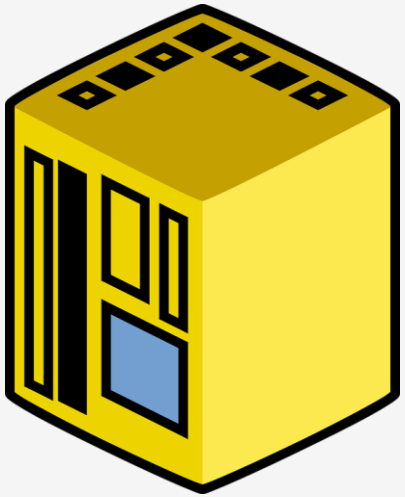


Rodzaje typów danych w Javie

Java jest językiem programowania typowanym statycznie

```
String liczba;

liczba = 2;
```

Incompatible types.
Required: java.lang.String
Found: int

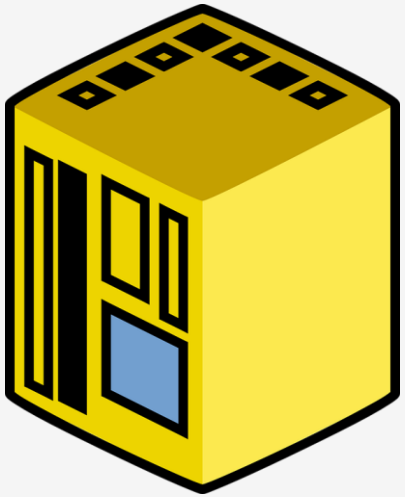


Rodzaje typów danych w Javie

Java jest językiem programowania typowanym statycznie

```
String liczba;

liczba = 2;
```

Incompatible types.
Required: java.lang.String
Found: int



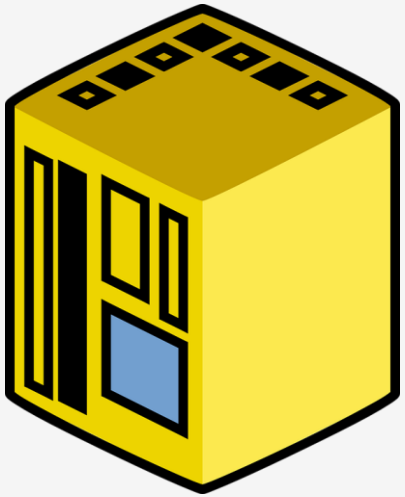
Rodzaje typów danych w Javie

Java jest językiem programowania typowanym statycznie

```
String liczba;

liczba = 2;
```

Incompatible types.
Required: java.lang.String
Found: int

```
String liczba;

liczba = "2";
```



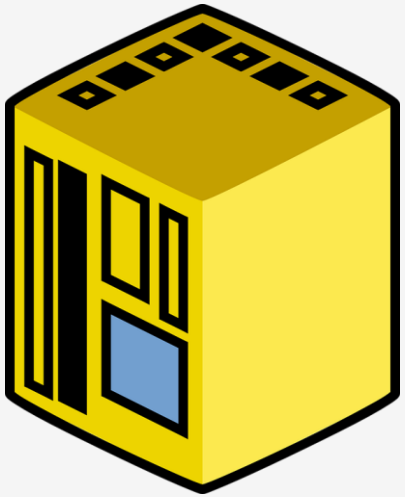
Rodzaje typów danych w Javie

Java jest językiem programowania typowanym statycznie

```
String liczba;

liczba = 2;
```

Incompatible types.
Required: java.lang.String
Found: int

```
String liczba;

liczba = "2";
```

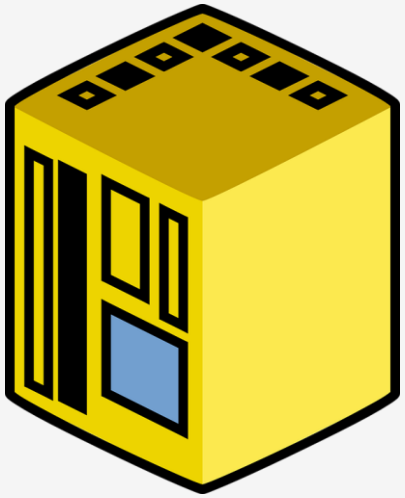


Podstawowe typy danych – liczby całkowite

Byte:

- 8 bitów pojemności
- $2^8 = 256$
- Przechowuje liczby od -128 do 127

```
byte sto = -128;
```

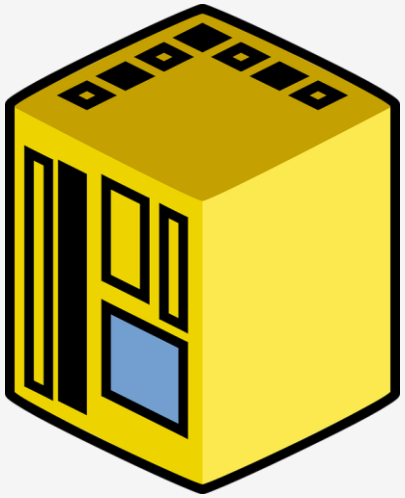


Podstawowe typy danych – liczby całkowite

Short:

- 16 bitów pojemności
- 2^{16} =65535
- Przechowuje liczby od -32768 do 32767

```
short dziesiecTysiecy = 10000;
```

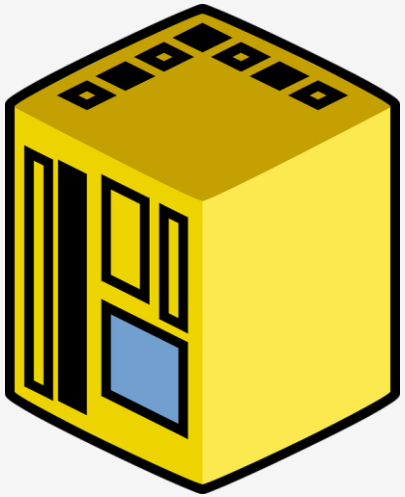


Podstawowe typy danych – liczby całkowite

Int:

- 32 bitów pojemności
- $2^{32} = 2\ 147\ 483\ 468$
- Przechowuje liczby od -2 147 483 648 do 2 147 483 647
- Najczęściej stosowany typ zmiennej liczbowej

```
int miliard = 100000000;
```

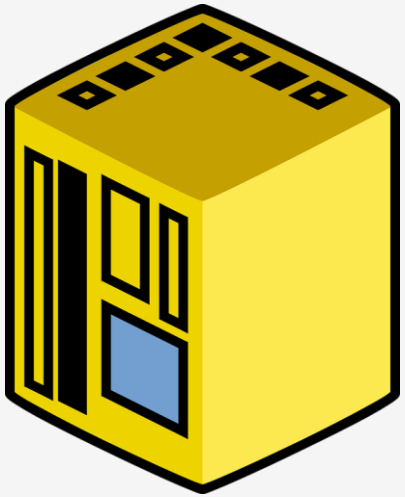


Podstawowe typy danych – liczby całkowite

Long:

- 64 bitów pojemności
- $2^{64} = dużo$:)
- W sytuacji gdy musimy przechowywać naprawdę duże liczby
- Np. do przechowywania identyfikatorów encji w bazie danych

```
long pesel = 90090809514L;
```



Programy do obliczeń matematycznych

1. *Utwórz nową klasę ‚Dodawanie'*
    1. *Utwórz metodę główną*
    2. *Utwórz 3 zmienne a,b,c typu int*
    3. *Przypisz zmiennym odpowiednio: a=1 b=2 c=3*
    4. *Wydrukuj na ekranie poniższe operacje:*
        1. *a+2*
        2. *a+b*
        3. *1+1*
        4. *a+(b+c)*
    5. *Przypisz do zmiennej c wynik dodawania a i b*
2. *Postępując analogicznie utwórz klasy ‚Odejmowanie', ‚Mnożenie' i ‚Dzielenie'*
3. *Co się stanie jeśli podzielisz przez 0?*

Pakiety

Co to jest pakiet w Javie?

Pakiet - Java nie jest monolitem, lecz składa się z szeregu klas definiujących obiekty różnego typu. Dla przejrzystości klasy te pogrupowane są w hierarchicznie ułożone pakiety.

Pakiety

Cechy pakietów:

- Przyjętą praktyką jest aby nazwy pakietu stanowiły odwrócone znaki domenowe np. com.example.main
- Każda klasa może być elementem pakietu
- Klasa może należeć tylko do jednego pakietu
- Pakiety mogą zawierać podpakiety
- Pakiety mają na celu grupowanie klas
  - o podobnych funkcjach
  - współpracujących ze sobą
  - stanowiących samodzielny moduł (pod)systemu

```
package nazwaPakietu[.nazwaPodpakietu]*;
```

Utworzenie pakietu

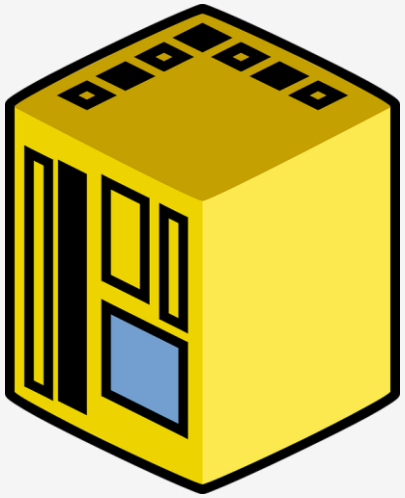
1. *Utwórz pakiet dla programu Dodawanie, Odejmowanie z poprzedniego zadania*
    1. *Nazwij go stosując odwrócona nazwę domenową*
    2. *Nadaj znaczącą nazwę końcówce pakietu, które określi jego przeznaczenie*
    3. *Przenieś tam utworzone klasy*

Podstawowe typy danych – liczby zmiennoprzecinkowe

Float:

- 32 bity pojemności
- Precyzja do 7 cyfr

```
float zero = 0.0f;
```

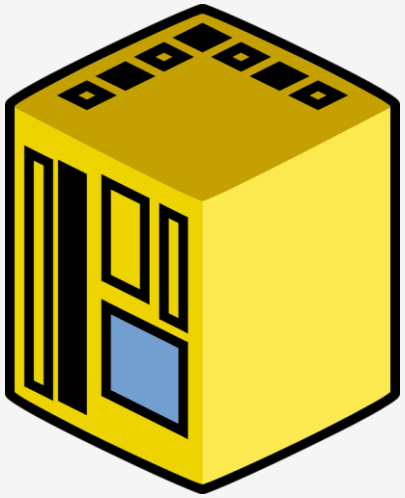


Podstawowe typy danych – liczby zmiennoprzecinkowe

Double:

- 64 bity pojemności
- Precyzja do 16 cyfr

```
double cenaBiletu = 3.20d;
```

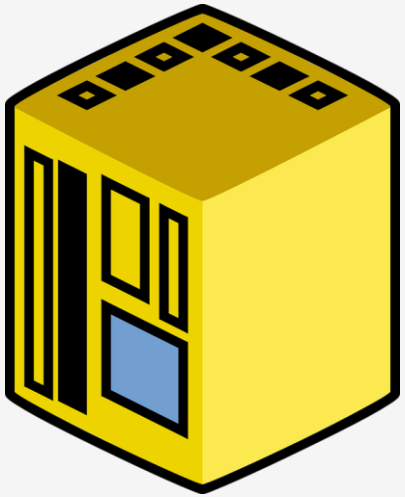


Podstawowe typy danych – typ znakowy

Char:

- Pojedynczy znak
- Ujęty w pojedynczym cudzysłowiu ,’

```
char p = 'p';
```

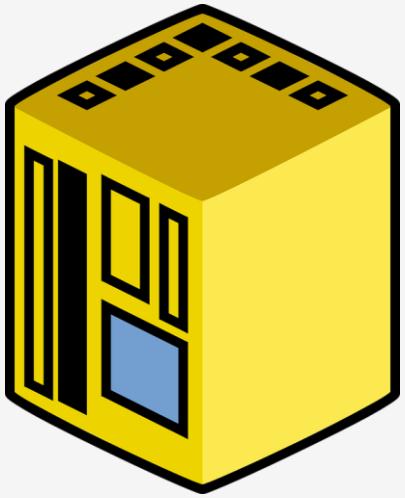


Podstawowe typy danych – typ tekstowy

String:

- Służy do przechowywania ciągu znaków
- Niezmienny stan obiektu (*immutable*)
- Ujęty w podwójnym cudzysłowiu „”

```
String ala = "Ala ma kota.";
```



Podstawowe typy danych – typ tekstowy

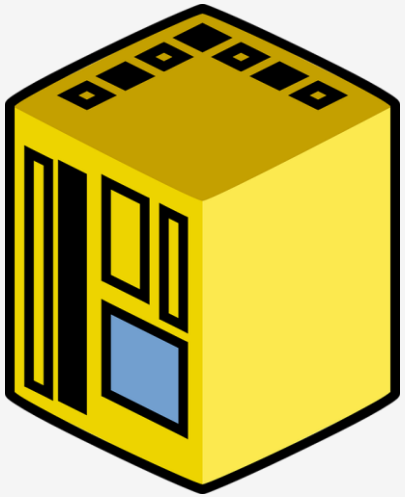
Konkantacja:

- Dodawanie łańcuchów znaków do siebie

```
String rzeczownik = "Samolot";
String przymiotnik = "pasażerski";
String spacja =" ";

String opis = rzeczownik + spacja + przymiotnik;
```



```
Samolot pasażerski
```



Typy danych - Zadania do samodzielnego wykonania

Wykonaj zadania z pliku

1. Wypisz na ekran wyniki poniższych działań
    a. 2+3
    b. 2-4
    c. 5/2
    d. 5.0/2
    e. 5/2.0
    f. 5.0/2.0
    g. 100L - 10
    h. 2f -3
    i. 5f/2
    j. 5d/2
    k. ‘A’+2
    l. ‘a’+2
    m. “a”+2
    n. “a”+”b”
    o. ‘a’+’b’
    p. “a”+’b’
    q. "a" + 'b' + 3
    r. 'b' + 3 + "a"
    s. 9%4 *(spróbuj wykonać to działanie bez %)

Zastanów się nad otrzymanymi wynikami. Czy zgadzają się z Twoimi oczekiwaniami?

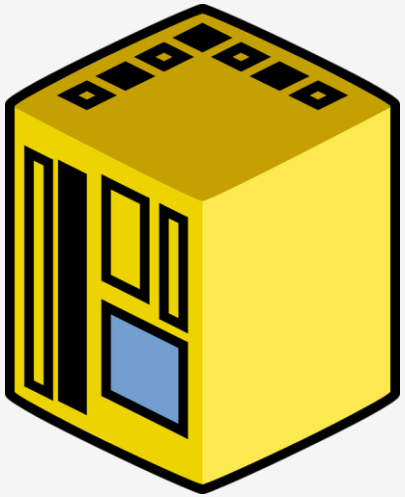
Określ typ wyników tych działań i przypisz je przed wyświetleniem do zmiennych odpowiedniego typu.

Podstawowe typy danych – typ logiczny

Typ logiczny:

- Stosuje się jako wynik funkcji lub flagę
- Zwiększa czytelność kodu
- Przyjmuje dwie wartości:
  - true
  - false

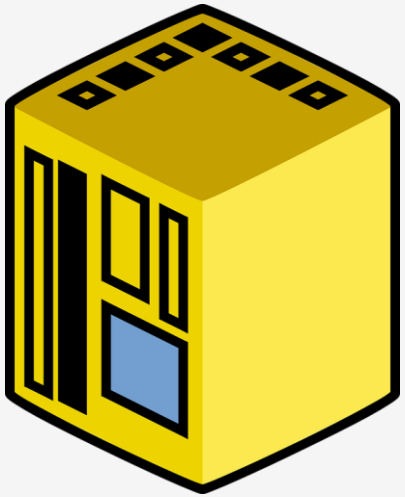
```
boolean prawda = true;
boolean falsz = false;

boolean prawdaIFalsz = prawda && falsz;
boolean prawdaLubFalsz = prawda || falsz;
```



Operatory

Wyróżniamy w Javie następujące operatory:

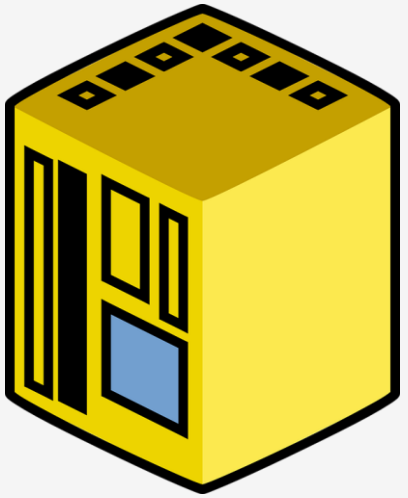
- Arytmetyczne
  - +,-,*,/,%,++,--
- Przypisania
  - =, +=, -=, *=, /=
- Relacji
  - <,<=,>,>=,==, !=
- Logiczne
  - &&, ||, !



Operatory

Operatory arytmetyczne

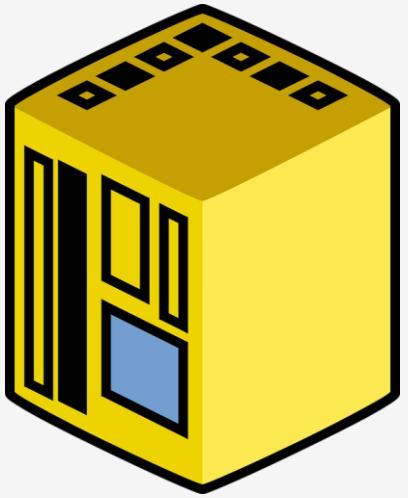
Operator	Zapis w kodzie
Dodaj	+
Odejmij	-
Pomnóż	*
Podziel	/
Modulo (reszta z dzielenia)	%
Powiększ zmienną o jeden	++
Pomniejsz zmienną o jeden	--



Operatory

Operatory przypisania

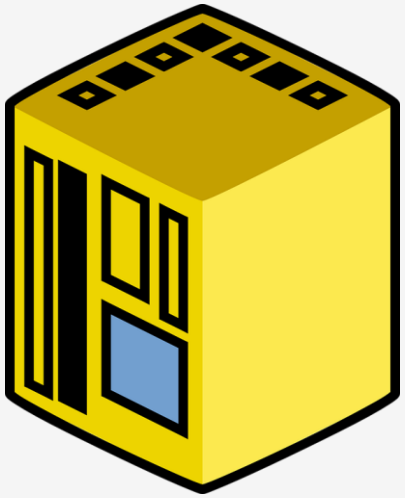
Operator	Zapis w kodzie
Przypisz	=
Zwiększ zmienną po lewej o wartość po prawej stronie	+=
Zmniejsz zmienną po lewej o wartość po prawej stronie	-=
Pomnóż zmienną po lewej stronie przez wartość po prawej stronie*	*=
Podziel zmienną po lewej stronie przez wartość po prawej stronie*	/=



Operatory

Operatory relacji

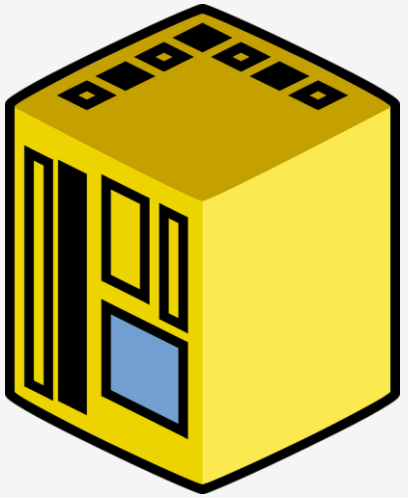
Operator	Zapis w kodzie
Równy	==
Nierówny	!=
Większy niż	>
Mniejszy niż	<
Większy bądź równy	>=
Mniejszy bądź równy	<=



Operatory

Operatory logiczne

Operator logiczny	Zapis w kodzie
Negacja (NOT)	!
Koniunkcja (AND)	&&
Alternatywa (OR)	||



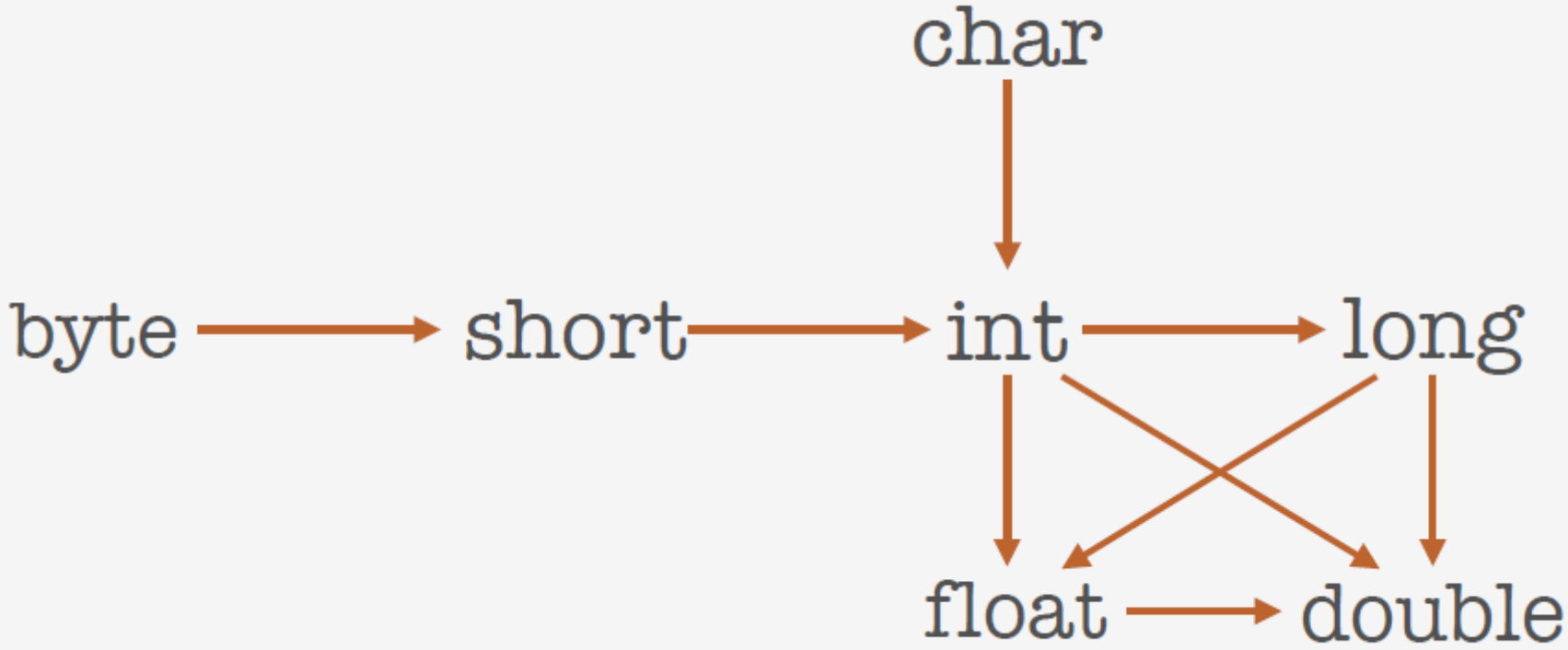
Typy danych - Zadania do samodzielnego wykonania

Wykonaj zadania :

1. Wypisz na ekran wartości poniższych wyrażeń logicznych
    a. false == false
    b. false != true
    c. !true
    d. 2 > 4
    e. 3 > 5
    f. 3 == 3 && 3 == 4
    g. 3 != 5 || 3 == 5
    h. (2+4) > (1+3)
    i. "cos".equals("cos")
    j. "cos" == "cos"

Konwersje typów

Możliwe konwersje typów (zmiany typu A na typ B) bez ryzyka utraty danych:

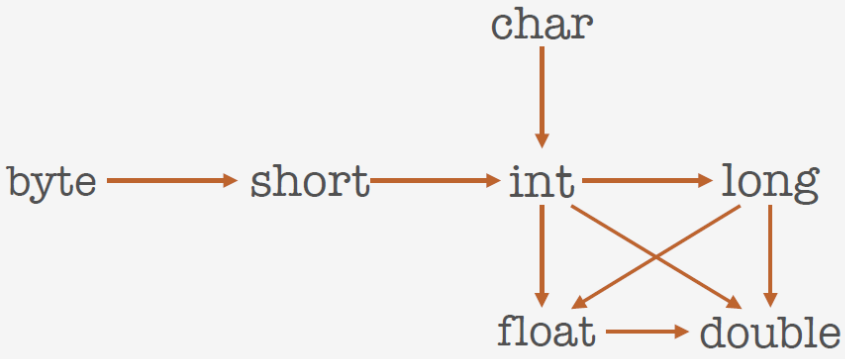


Konwersje typów

Możliwe konwersje typów (zmiany typu A na typ B):

- Typ o mniejszej pojemności konwertujemy na typ większej pojemności bitowej
  - Rzutowanie odbywa się samoistnie

```
byte sto = 100;
short stoShort = sto;
```

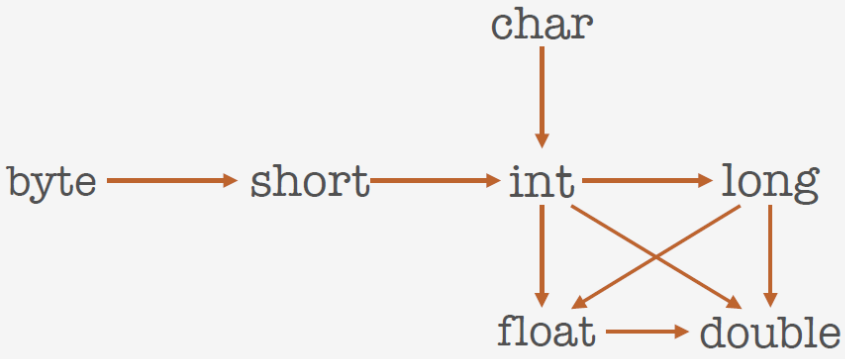


Konwersje typów

Możliwe konwersje typów (zmiany typu A na typ B):

- Typ o większej pojemności konwertujemy na typ o mniejszej pojemności bitowej
- Ogólna formuła:
  - (typ_konwertowany) zmienna_mniejszej_pojemności

```
short tysiac = 1000;
byte tysiacByte = (byte)tysiac;
```

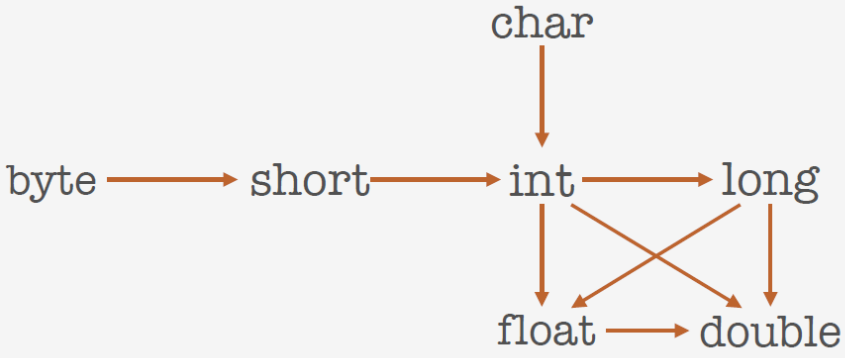


Konwersje typów

Możliwe konwersje typów (zmiany typu A na typ B):

- Typ o większej pojemności konwertujemy na typ o mniejszej pojemności bitowej
- Ogólna formuła:
  - (typ_konwertowany) zmienna_mniejszej_pojemności

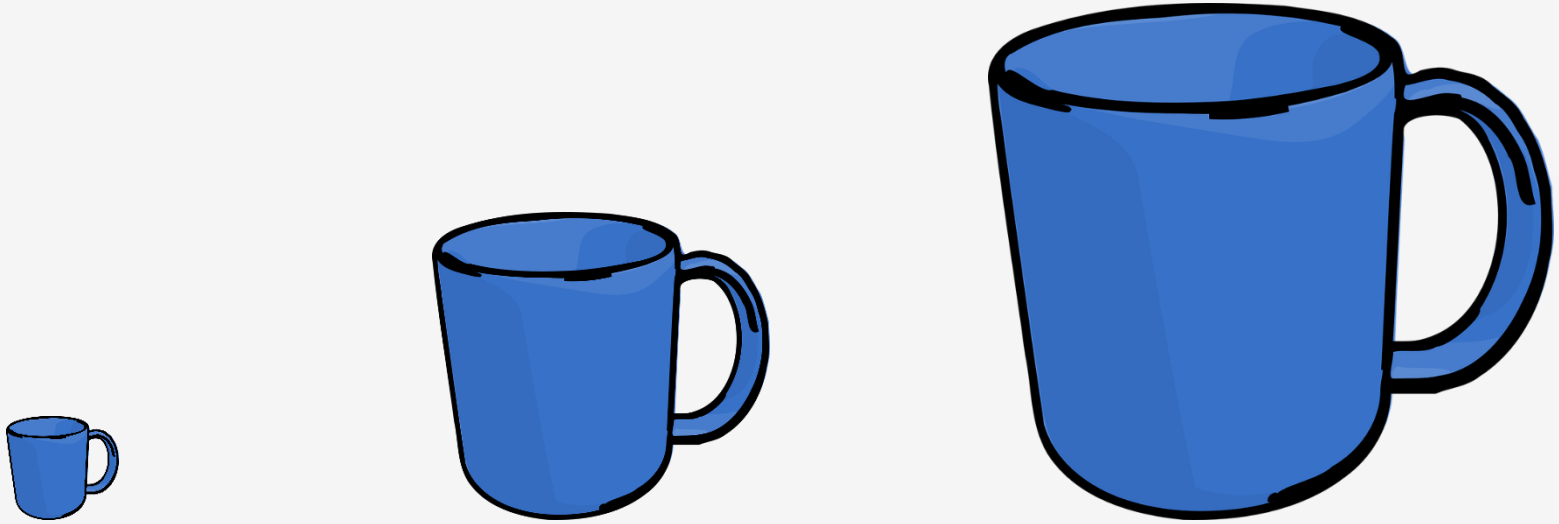
```
byte sto = 100; //100
byte dwa = 2; //2
short stoDwa = (short) (sto + dwa); //102
```



Konwersje typów

Możliwe konwersje typów (zmiany typu A na typ B):

- Analogia do kubków



Typy danych - Zadania do samodzielnego wykonania

Konwersja typów

1. Spróbuj przekonwertować następujące wartości:
   A. short - > int
   B. short -> long
   C. int -> float
   D. int -> double
   E. long -> int
   F. short -> byte
   G. char -> int
2. Przeanalizuj otrzymane rezultaty
3. *Czym jest otrzymany wynik z działania G?

TESTY WIEDZY

Powtórzenie wiedzy z poprzednich zajęć

Wykonaj poniższe zadania:

1. Utwórz nowy projekt o nazwie ‚TestWiedzy’
2. Utwórz nowy pakiet pamiętając o konwencji Javy
3. Utwórz klasę ‚TypyDanych’
4. Utwórz główną metodę wejścia programu
5. Utwórz 3 zmienne: tekstową, znakową oraz liczbową
6. Dwukrotnie zmień wartość zmiennej liczbowej
7. Wydrukuj poniższy tekst korzystając z zmiennych:
    a. {liczba}{litera}{tekst}
    b. {liczba}{KodAscii litery}{tekst}
8. * Wydrukuj poniższy tekst:
    1. Kod ASCII litery ‚{litera}’ to: {kodAscii litery}

Instrukcje sterujące

Co to jest instrukcja sterująca?

Instrukcja sterująca - **instrukcja zdefiniowana** w **składni** określonego **języka** programowania, umożliwiająca wyznaczenie i zmianę kolejności wykonania instrukcji zawartych w kodzie źródłowym

Instrukcje sterujące – instrukcja warunkowa If

Wyrażenie if

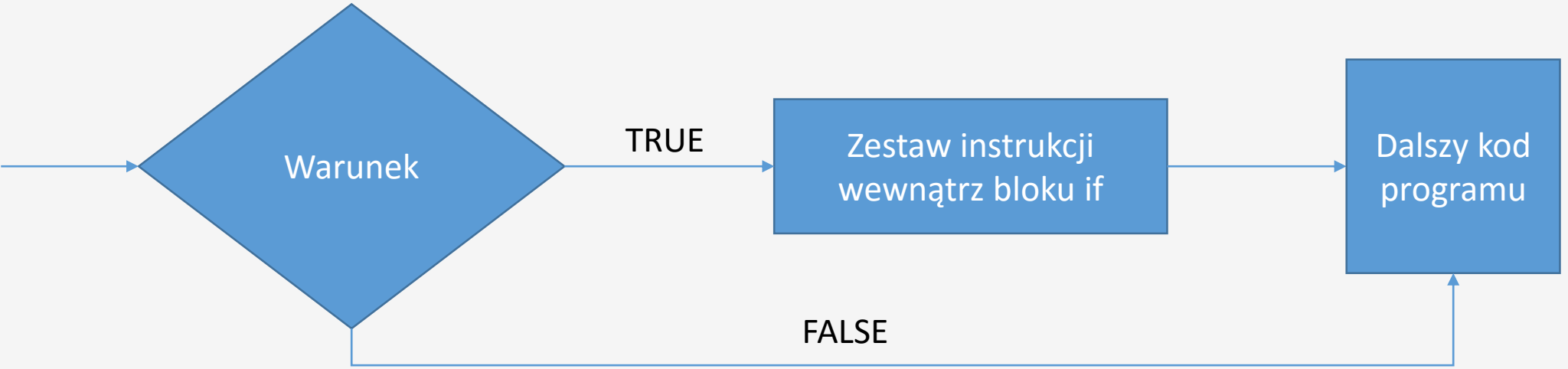
- Przy spełnieniu logicznego warunku wykonuje instrukcje znajdujące się wewnątrz bloku

```
boolean warunekLogiczny = true;

if(warunekLogiczny){
    //Wykonaj kod
}
```

Instrukcje sterujące – instrukcja warunkowa If

Wyrażenie if

- Przebieg działania



Wyrażenia warunkowe- Zadanie

Wyrażenie If

1. Napisać program, który sprawdza poniższe warunki i przy spełnieniu warunku wypisuje „:)”
   a. 2 > 3
   b. 4 < 5
   c. (2 - 2) == 0
   d. true
   e. 9%2 == 0
   f. 9%3 == 0

Instrukcje sterujące – instrukcja warunkowa If

Wyrażenie if...else

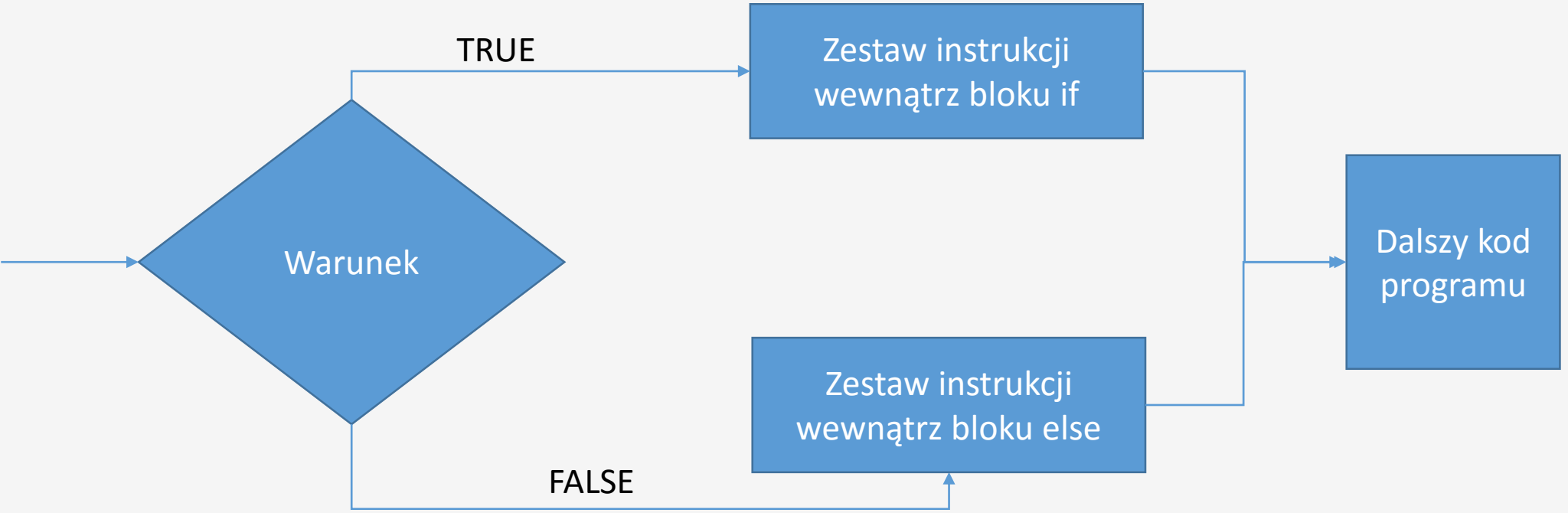
- Kod w bloku else jest wykonywany wtedy i tylko wtedy gdy warunek logiczny zadeklarowany w nawiasie if(..) nie zostaje spełniony

```
if(warunekLogiczny){
    //Wykonaj kod dla spełnionego warunku
}else{
    //Wykonaj kod dla niespełnionego warunku
}
```

Instrukcje sterujące – instrukcja warunkowa If

Wyrażenie if...else

- Przebieg działania



Wyrażenia warunkowe- Zadanie

Wyrażenie If...else

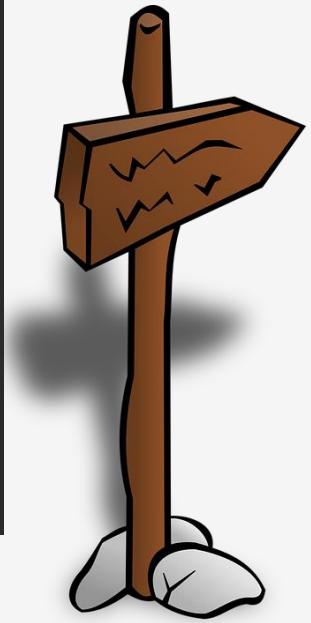
1. Rozbuduj poprzedni program o wypisywanie „:(„ w przypadku niespełnienia warunku
   a. 2 > 3
   b. 4 < 5
   c. (2 - 2) == 0
   d. true
   e. 9%2 == 0
   f. 9%3 == 0

Instrukcje sterujące – instrukcja switch

Instrukcja switch

- Pokrywa wielokrotne wyrażenia if-else
- Składa się z wielu warunków
- Domyślny kod przy braku spełnienia pozostałych warunków
- Od Java SE7 możliwe jest użycie zmiennej typu String

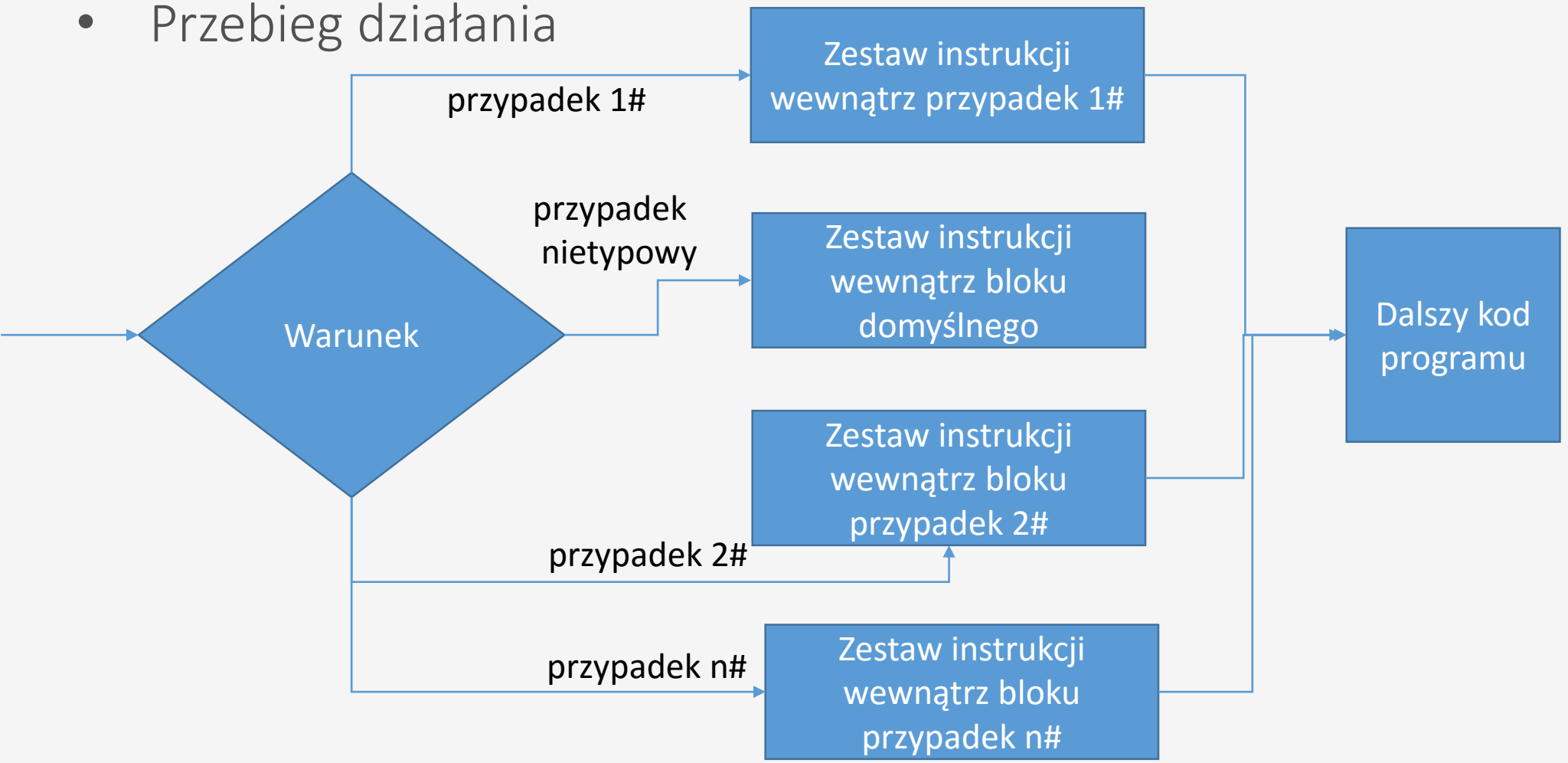
```
int zmienna=1;

switch (zmienna){
    case 1:
        //wykonaj akcje dla przypadku 1.
        break;
    case 2:
        //wykonaj akcje dla przypadku 2.
        break;
    //... other cases
    default:
        //wykonaj wtedy i tylko wtedy
        //gdy poprzednie warunki nie zostały spełnione
        break;
}
```



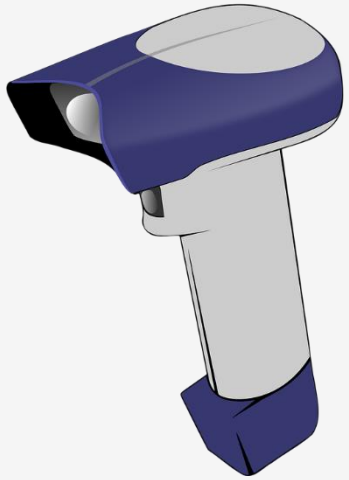
Instrukcje sterujące – instrukcja switch

Wyrażenie switch

- Przebieg działania



Scanner



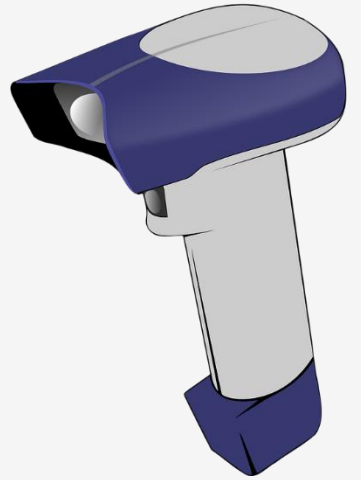
Czym jest biblioteka java.util?

java.util- podstawowa biblioteka Javy posiadająca różne klasy do wykorzystania w własnych programach takie jak: **Random**, **Date** czy **Scanner**

Scanner

Scanner:

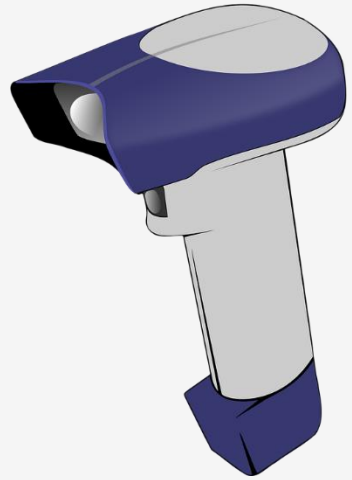
- Klasa umożliwiająca interakcję użytkownika z programem
- Korzysta z metod niestatycznych
- Jako argument konstruktora należy podać źródło strumienia danych jak np. klawiatura, czy plik tekstowy
- Przydatne metody:
  - nextLine():String – zwraca wpisaną linię tekstu (do wciśnięcia entera)
  - nextInt():int – zwraca pierwszy napotkany int (jeśli przedzielimy spacją, to następne inty są ignorowane)



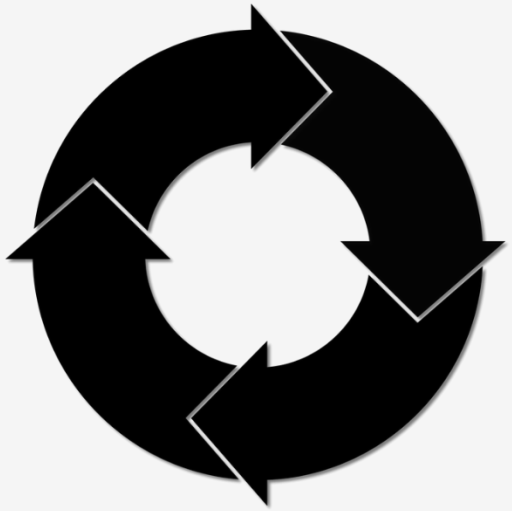
Scanner

Użycie w kodzie:

```java
import java.util.Scanner;

public class Echo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String zczytanyTekst = scanner.nextLine();
        System.out.println(zczytanyTekst);
    }
}

```



Pętle



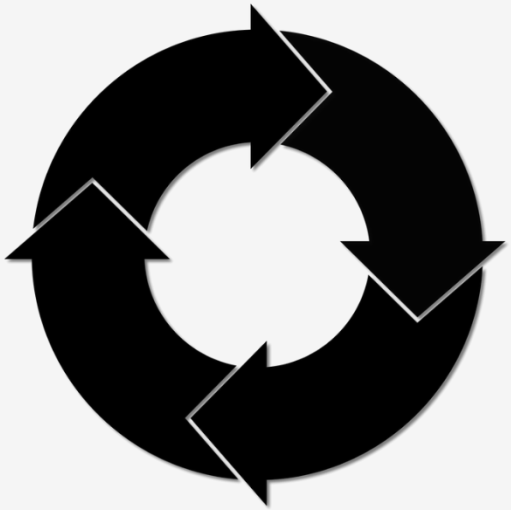
Co to jest pętla w programie?

Pętla - umożliwia cykliczne wykonywanie ciągu instrukcji określoną liczbę razy, do momentu zajścia pewnych warunków, dla każdego elementu kolekcji lub w nieskończoność

Pętle

Cechy Pętli:

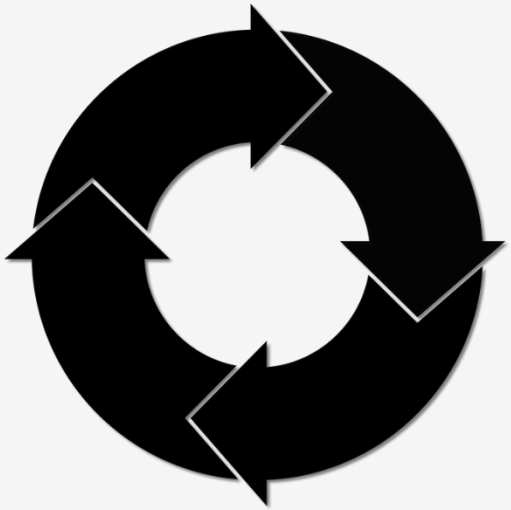
- Wykonują umieszczony blok kodu dopóki warunki są spełnione
- Kod wykonywany jest przez skończoną liczbę obiegów (pętli)
- Pętle które nie mają końca nazywamy pętlami nieskończonymi



Pętle – Pętla for

Pętla for:

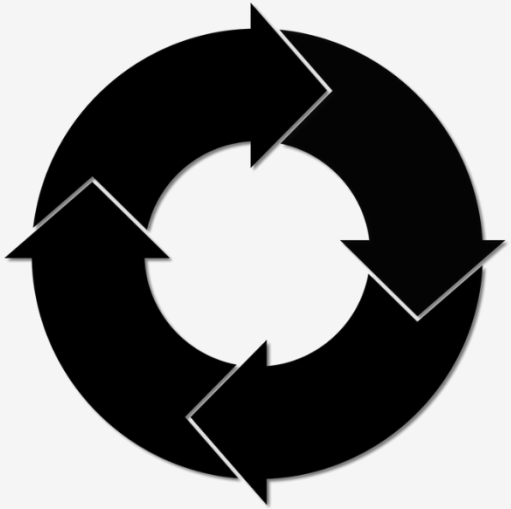
```
for(inicjalizacja; sprawdzenieWarunku; aktualizacja){
    //instrukcje wewnątrz pętli
}
```



Pętle – Pętla for

Pętla for:

```
int warunekWyjsciaZPetli =10;
int warunekPoczatkowy = 0;

for(int i= warunekPoczatkowy; i<warunekWyjsciaZPetli; i++){
    //instrukcje wewnątrz pętli
}
```

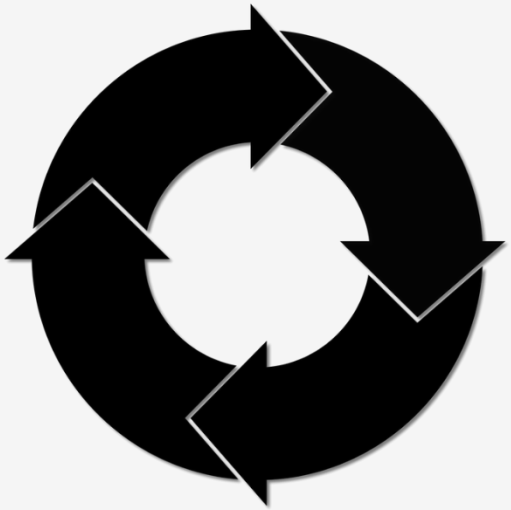


Pętle – Pętla for

Pętla for:

```
int warunekWyjsciaZPetli =10;
int warunekPoczatkowy = 0;

for(int i= warunekPoczatkowy; i<warunekWyjsciaZPetli; i++){
    //instfukcje wewnątrz pętli
}
```

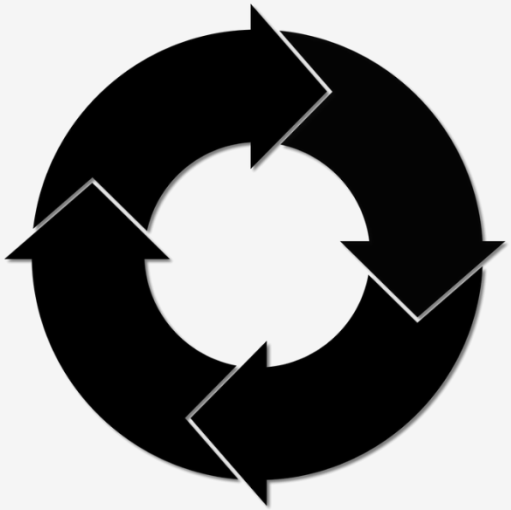
Wykonywane raz przy inicjalizacji pętli



Pętle – Pętla for

Pętla for:

```
int warunekWyjsciaZPetli =10;
int warunekPoczatkowy = 0;

for(int i= warunekPoczatkowy; i<warunekWyjsciaZPetli; i++){
    //instfukcje wewnątrz pętli
}
```



Wykonywane raz przy inicjalizacji pętli

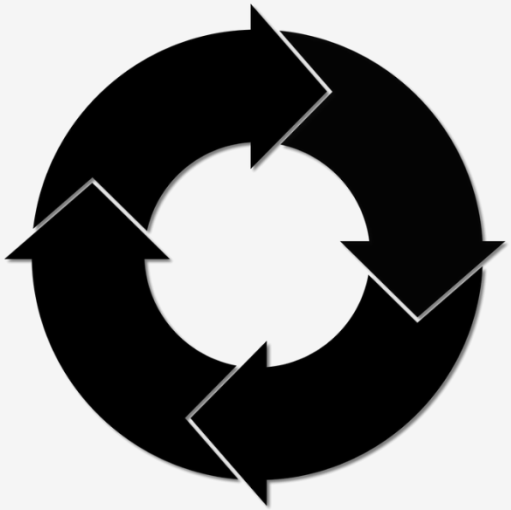
Sprawdzenie warunku:
dla wartości ‚true' kod wewnątrz pętli jest wykonywany
dla wartości ‚false' kod wewnątrz pętli nie jest wykonywany

Pętle – Pętla for

Pętla for:

Po wykonaniu pętli, aktualizacja licznika

```
int warunekWyjsciaZPetli =10;
int warunekPoczatkowy = 0;

for(int i= warunekPoczatkowy; i<warunekWyjsciaZPetli; i++){
    //instfukcje wewnątrz pętli
}
```



Wykonywane raz przy inicjalizacji pętli

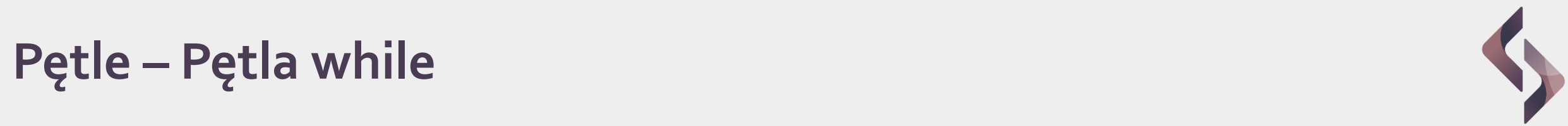
Sprawdzenie warunku:
dla wartości ‚true' kod wewnątrz pętli jest wykonywany
dla wartości ‚false' kod wewnątrz pętli nie jest wykonywany

Pętle - Zadanie

Pętla for – zadanie do wykonania

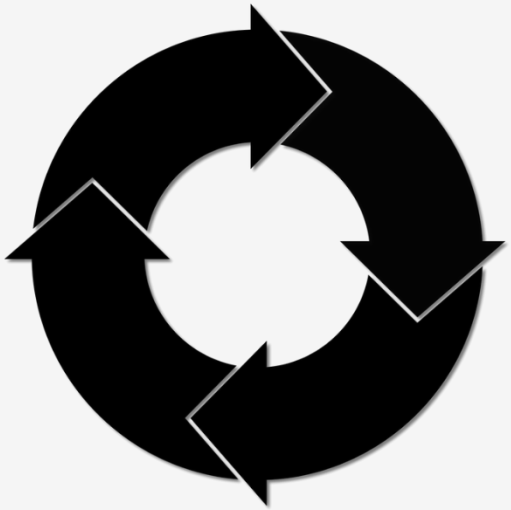
1. Napisz program, który przyjmuje od użytkownika dzielnik oraz liczbę, a następnie drukuje na ekranie wszystkie liczby mniejsze od zadanej liczby podzielne przez zadany dzielnik
2. * Napisz program wyznaczający potęgę liczby n i m – obie zadane przez użytkownika i drukujący w czytelny sposób wynik na ekranie konsoli

Pętle – Pętla while



Pętla while:

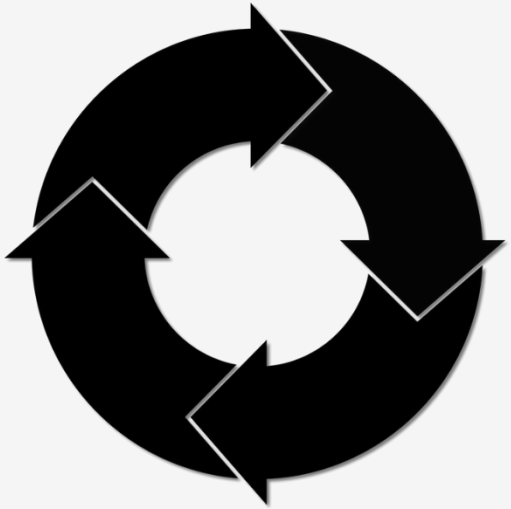
```
while(warunek){
    //instrukcje
}
```



Pętle – Pętla while

Pętla while:

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



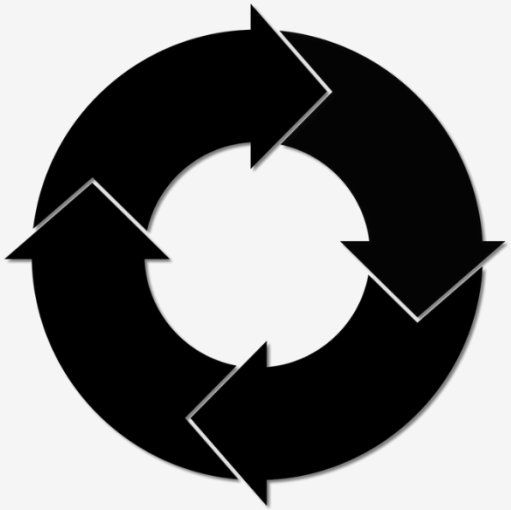
Instrukcje są wykonywane dopóki warunek logiczny ma wartość ‚true’

Pętle – Pętla while

Pętla while:

Aby wyjść z pętli należy :
A. sprawić aby warunek zmienił wartośc na false LUB
B. dodać słowo kluczowe ‚break’ wewnątrz instrukcji

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



Instrukcje są wykonywane dopóki warunek logiczny ma wartość ‚true’

Pętle - Zadanie

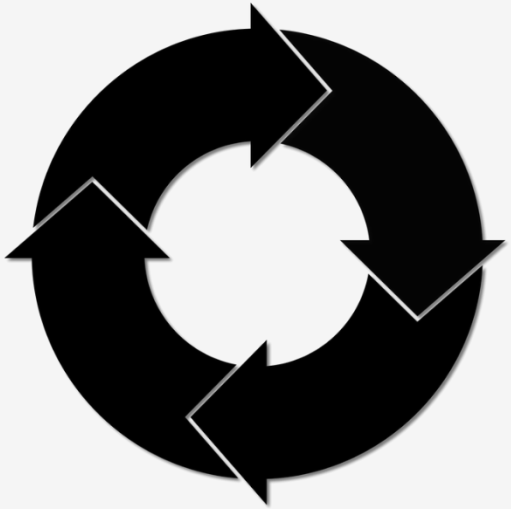
Pętla while – zadanie do wykonania

1. Napisz program, który oblicza sumę wszystkich liczb poprzedzających zadaną przez użytkownika liczbę – dla liczby 100 będzie to suma liczb od 0 do 100 czyli 5050
2. *Napisz program, który oblicza silnię dla zadanej liczby przez użytkownika (do n=12) korzystając z pętli while https://pl.wikipedia.org/wiki/Silnia

Pętle – Pętla do while

Pętla do while:

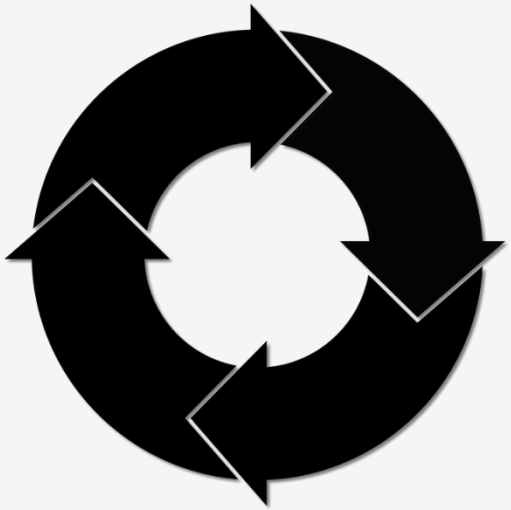
```
boolean warunek = true;

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



Pętle – Pętla do while

Pętla do while:

```
boolean warunek = true;

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



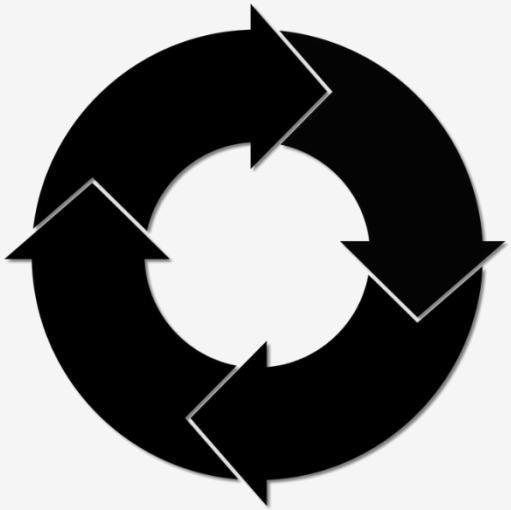
Blok do wykonywany jest w 1 kroku,
niezależnie od wartości warunku

Pętle – Pętla do while

Pętla do while:

```
boolean warunek = true;

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

Następnie sprawdzany jest warunek logiczny:
- W przypadku braku spełnienia pętla jest przerywana



Blok do wykonywany jest w 1 kroku, niezależnie od wartości warunku

Pętle - Zadanie

Pętla do while – zadanie do wykonania

1. Napisz program, który wypisuje wszystkie liczby mniejsze od zadanej od użytkownika dopóki użytkownik wpisuje liczby większe od 0
2. *Napisz program, który oblicza wartość pierwiastka z wprowadzonej przez użytkownika liczby, dopóki ta przyjmuje wartości większe od 0 (dla uproszczenia przyjmij że użytkownik wprowadza liczby całkowite)

TEST WIEDZY

Powtórzenie wiedzy z poprzednich zajęć

Wykonaj poniższe zadania:

1. Utwórz nowy projekt o nazwie ‚PetlePowtorka’
2. Utwórz nowy pakiet pamiętając o konwencji Javy
3. Zrealizuj każdy z poniższych programów w osobnej klasie:
    1. Przy użyciu dowolnej pętli wyświetl liczby od 100 do 0
    2. Przy użyciu dowolnej pętli wyświetl wszystkie liczby większe od zadanej przez użytkownika liczby, ale mniejsze od 100
    3. Program sumujący dwie liczby wprowadzone przez użytkownika i prezentujący wynik działania na ekranie konsoli
    4. * Program sumuje liczby wprowadzane przez użytkownika i wyświetla sumę wszystkich wprowadzonych dotychczas liczb. Aby przerwać wprowadzanie liczb użytkownik wpisuje:
        1. 0
        2. **Dowolną nie-liczbę

Tablice

Co to jest tablica w Javie?

Tablica - **kontener** uporządkowanych danych takiego samego typu, w którym poszczególne elementy dostępne są za pomocą kluczy (indeksu). Rozmiar tablicy ustalany jest z góry.

Tablice

Cechy Tablic:

- Przechowują elementy tego samego typu w sposób uporządkowany
- Wielkość tablicy jest deklarowana przy jej inicjalizacji
- Wielkość tablicy pozostaje niezmienna
- Do elementów tablicy odwołujemy się przez indeks, liczony od **0** do (**n-1**), gdzie **n** - to ilość elementów w tablicy

Tablice

Deklaracja tablicy:

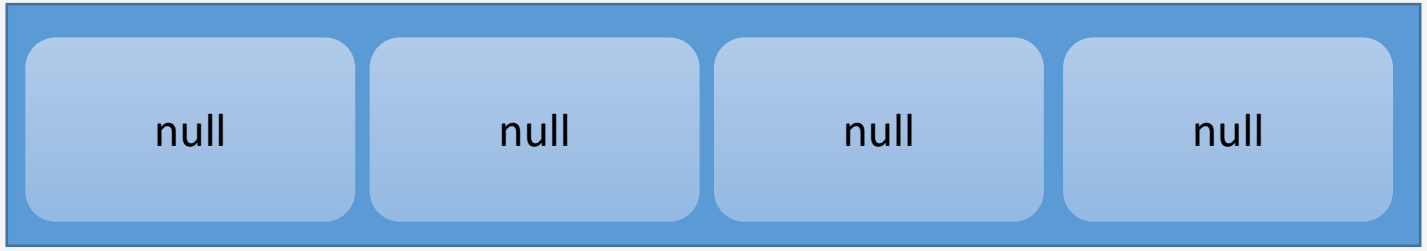
```
String[] pasazerowie;
```

Pusta tablica znaków

Tablice

Inicjalizacja tablicy:

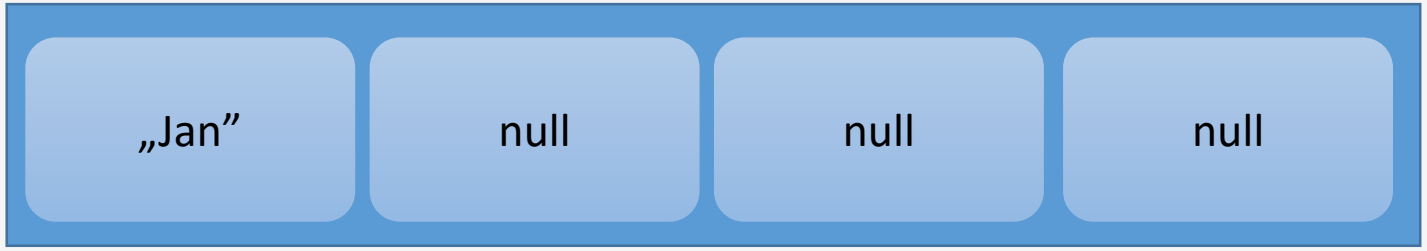
```
pasazerowie = new String[4];
```



Tablice

Przypisywanie elementów do tablicy:

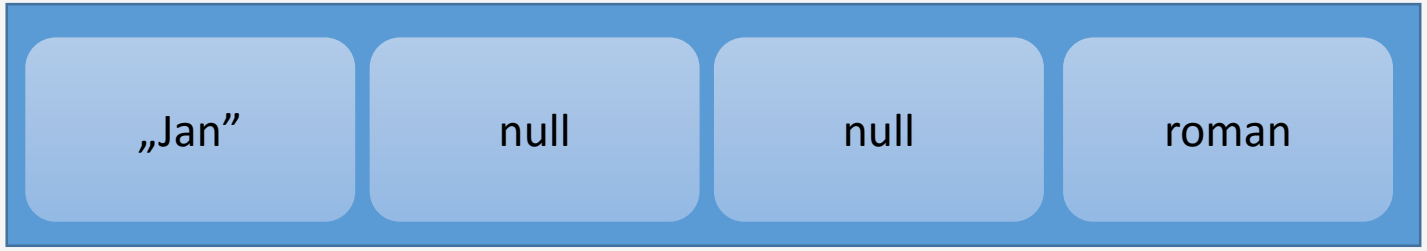
```
pasazerowie[0] = "Jan";
```



Tablice

Przypisywanie elementów do tablicy:

```
String roman = "Roman";

pasazerowie[3] = roman;
```



Tablice

Używanie elementów z tablicy:

```
System.out.println(pasazerowie[0]);
```

→

```
Jan
```

„Jan”	null	null	roman

Tablice

Używanie elementów z tablicy:

```
String drugiPasazer = pasazerowie[1];
System.out.println(drugiPasazer);
```

→ null

„Jan”	null	null	roman

Tablice

Określanie wielkości tablicy:

```
System.out.println(pasazerowie.length);
```

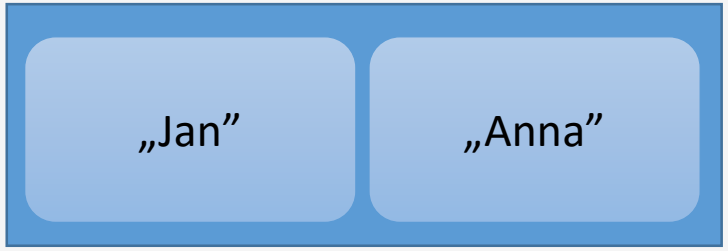
→ 4

„Jan”	null	null	roman

Tablice

Inicjalizacja tabeli wraz z elementami:

```
String[] pasazerowie = new String[]{"Jan", "Anna"};
```

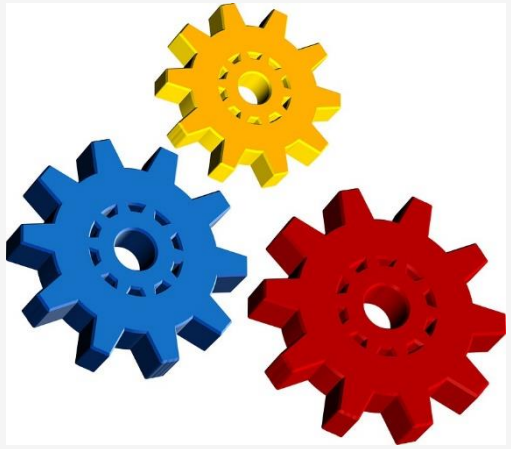


Tablice - Zadanie

Tablice – zadanie do wykonania

1. Utwórz tablicę liczb {1,3,5,10}
2. Wypisz wszystkie elementy po kolei
3. Wypisz elementy w pętli
4. Wypisz tylko liczby o parzystym indeksie
5. Wypisz tylko liczby parzyste
6. *Wypisz elementy w odwróconej kolejności

Metody



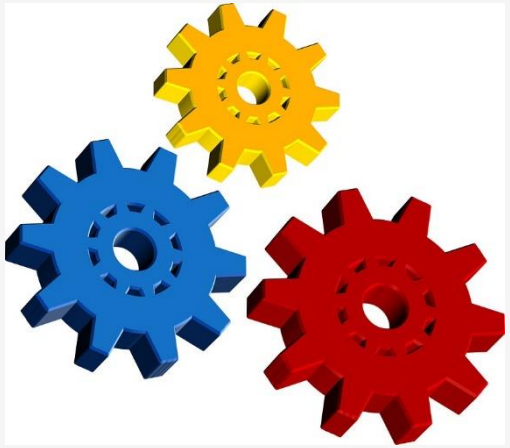
Czym jest metoda w Javie?

Metoda- **podprogram** składowy **klasy**, którego zadaniem jest działanie na rzecz określonych elementów danej klasy lub klas z nią spokrewnionych.

Metody

Cechy metod:

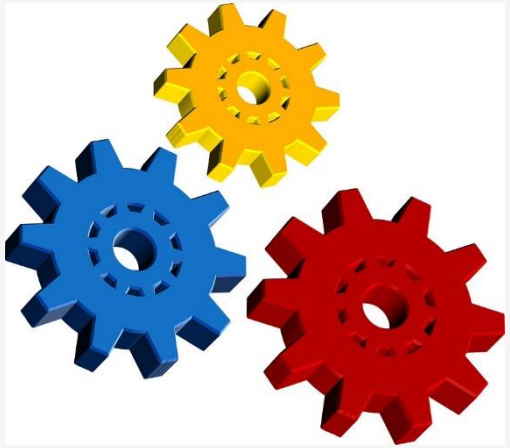
- Opisywane jako „sposób na wykonanie czegoś”
- Mogą być tworzone tylko jako części składowe klasy
- Mogą być wywoływane przez samą klasę lub klasy powiązane
- Ogólna składnia metody:
  - [akcesor*] TypZwracany nazwaMetody([parametry]){
    //ciało metody
    }



Metody

Po co używać metod?:

- Dzielą duże bloki kodu na mniejsze
- Pozwalają na lepszą organizację programu
- Zwiększają czytelność oraz łatwość utrzymania programu
- Wspierają ponowne użycie kodu i unikanie powtórzeń (DRY)

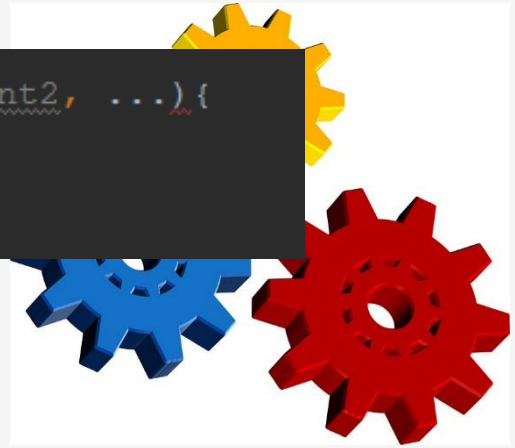


Metody

Nazwa metody

- Używana do jej wywołania
- Powinna określać działanie funkcji
- Zgodnie z konwencją nazewniczą powinna być pisana camelCasem

```
private static TypZwracany nazwaMetody(TypArgumentu1 argument1, TypArgumentu2 argument2, ...){
    //ciało metody
    return TypZwracany;
}
```

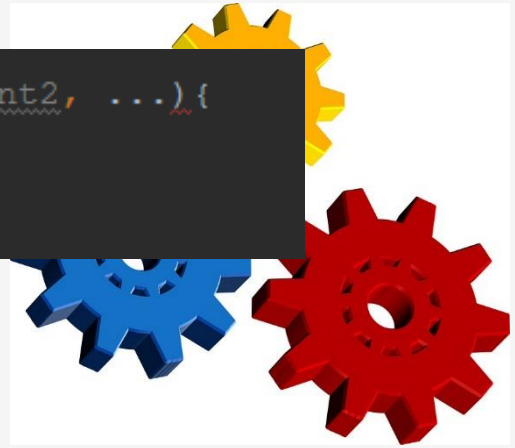


Metody

Typ zwracany

- Określa jaki typ danych zwraca metoda np. int, String, itp.
- W przypadku gdy metoda nie zwraca żadnego typu danych TypZwracany określamy jako ‚void' a słowo kluczowe ‚return' możemy pominąć

```
private static TypZwracany nazwaMetody(TypArgumentu1 argument1, TypArgumentu2 argument2, ...){
    //ciało metody
    return TypZwracany;
}
```

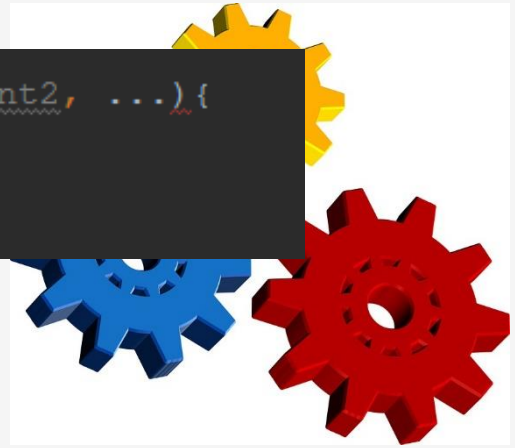


Metody

Argumenty:

- Metoda może przyjmować od 0 do n argumentów
- Każdy argument może być różnego typu
- Im więcej argumentów, tym mniej czytelna staje się metoda

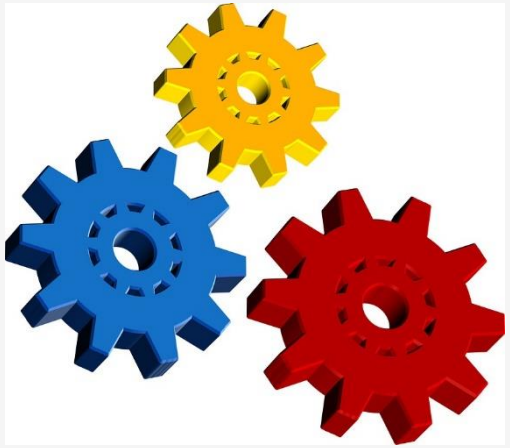
```
private static TypZwracany nazwaMetody(TypArgumentu1 argument1, TypArgumentu2 argument2, ...){
    //ciało metody
    return TypZwracany;
}
```



Metody

Metody statyczne:

- Wywoływane na rzecz klasy
- Nie wymagają utworzenia obiektu danej klasy
- Ogólna składnia metody:
  - [akcesor*] **static** TypZwracany nazwaMetody([parametry]){
    //ciało metody
    }



Metody - Zadanie

Metody statyczne - zadanie

1. Napisz program, który będzie posiadał następujące metody statyczne:
    1. drukujParzystoscLiczby
        1. Przyjmuje pojedynczy parametr w postaci liczby całkowitej
        2. Zwraca nic
        3. Drukuje na ekranie wynik „liczba jest parzysta” gdy liczba jest parzysta i „liczba jest nieparzysta” dla nieparzystej liczby
    2. czyLiczbaJestParzysta
        1. Przyjmuje pojedynczy parametr w postaci liczby całkowitej
        2. Zwraca typ logiczny w postaci wyniku obliczenia parzystości liczb
    3. * Zastanów się w jaki sposób mógłbyś ułatwić sobie życie i zminimalizować liczbę zduplikowanego kodu

Metody - Zadanie

Metody statyczne - zadanie

1. Napisz program, który sprawdza czy liczba podana przez użytkownika jest podzielna przez 3 lub 5.
2. * Napisz program który pozwala użytkownikowi samemu zdecydować podzielność przez jaką liczbę będzie badana

Metody - Zadanie

Metody statyczne - zadanie

1. Utwórz program ‚Kalkulator’
    1. Nadpisz punkt wejścia, tak aby wyświetlał następującą instrukcję po uruchomieniu:
       „Podaj rodzaj działania : *,+,-,/”
       „Podaj pierwszą liczbę”
       „Podaj druga liczbę”
       „Otrzymany wynik {działanie} wynosi: {wynik}”
    2. Użyj switcha do wybrania rodzaju metody
    3. Użyj switcha do wydrukowania końcowego wyniku (np. ‚+’ → ’dodawania’) np. po wpisaniu przez użytkownika ‚+’, ‚2’, ‚3’ wyświetlony zostanie tekst:
       Wynik dodawania liczb 2 i 3 wynosi: 5

Bibliografia

1. https://pixabay.com/pl/kodowanie-komputer-1294361/ (dostęp 21.07.2017)
2. https://pl.wikipedia.org/wiki/Programowanie_komputer%C3%B3w (dostęp 22.07.2017)
3. *Wprowadzenie do Języka Java* (Łukasz Ognan, Marzec 2017)
4. https://pl.wikipedia.org/wiki/Java (dostęp 22.07.2017)
5. https://pl.wikipedia.org/wiki/Interpreter_(program_komputerowy) (dostęp 22.07.2017)
6. https://pixabay.com/pl/m%C5%82otek-narz%C4%99dzia-metalowe-celuj-w-33617/ (dostęp 22.07.2017)
7. https://pixabay.com/pl/matematyka-sub-zmienna-matematyczne-27893/ (dostęp 24.07.2017)
8. https://pixabay.com/pl/nagrobek-protokół-rip-martwe-śmierć-159792/ (dostęp 24.07.2017)
9. https://pixabay.com/pl/typu-mainframe-sprz%C4%99tu-serwer-mail-35647/ (dostęp 24.07.2017)
10. https://pixabay.com/pl/pole-przypadku-otwarte-pakiet-1295176/ (dostęp 24.07.2017)
11. *Podstawy Java* (Rafał Roppel)
12. https://pixabay.com/pl/puchar-kubek-kawa-pi%C4%87-herbata-42427/ (dostęp 24.07.2017)
13. https://pixabay.com/pl/zarejestruj-stanowisko-wy%C5%BCywienie-48703/ (dostęp 24.07.2017)
14. https://pixabay.com/pl/narzędzi-połowowych-funkcja-razem-818456/ (dostęp 24.07.2017)
15. https://pixabay.com/pl/skaner-palmtop-kod%C3%B3w-kreskowych-36385/ (dostęp 24.07.2017)
16. https://pl.wikipedia.org/wiki/P%C4%99tla_(informatyka) (dostęp 24.07.2017)
17. https://pixabay.com/pl/cykl-obieg-proces-zmiana-zmiany-2019535/ (dostęp 25.07.2017)
18. https://pixabay.com/pl/monitora-monitor-wall-big-screen-1054600/ (dostęp 25.07.2017)
19. https://pl.wikipedia.org/wiki/Tablica_(informatyka) (dostęp 25.07.2017)