

# 3. Lekcja

Typy, zmienne, tablice, rzutowanie i operatory



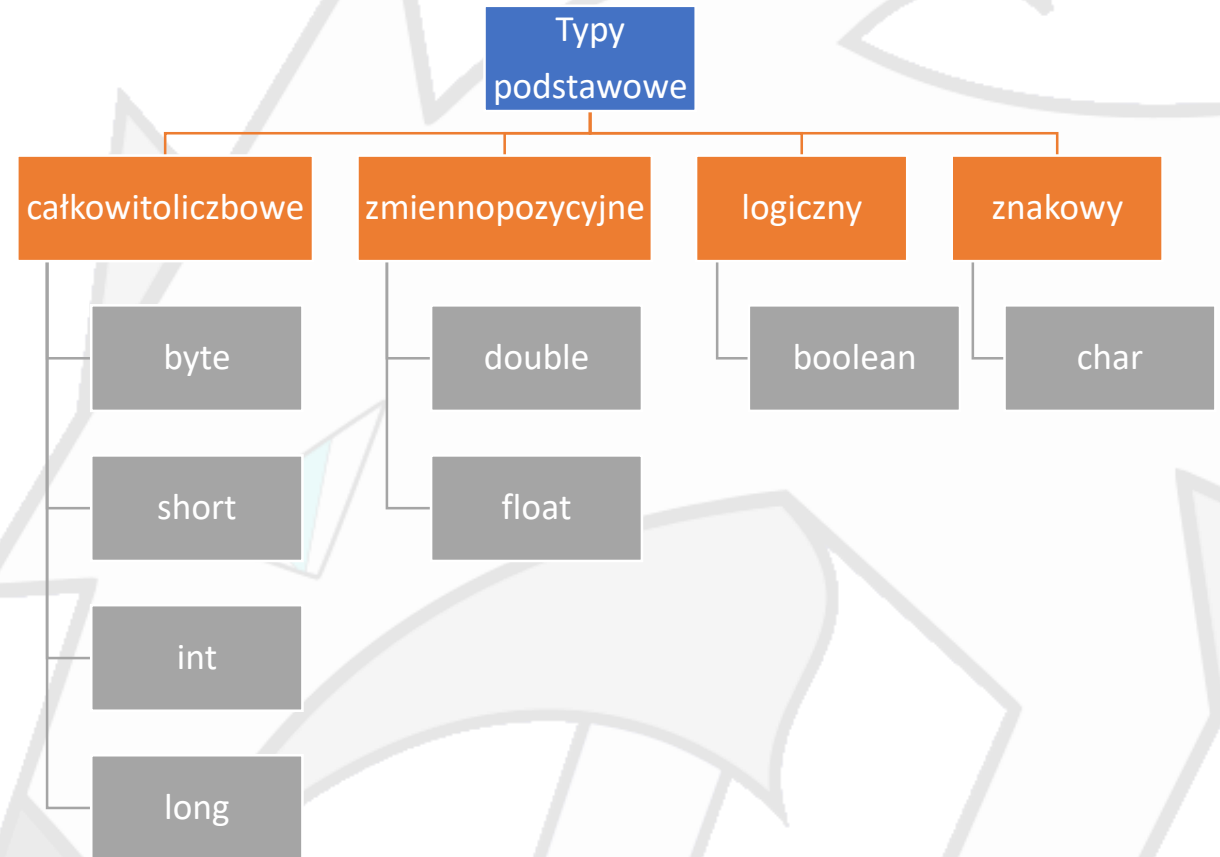
# Typy podstawowe

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

Rozróżniamy:

- typy całkowitoliczbowe
- typy zmiennopozycyjne
- typy logiczne
- typy znakowe

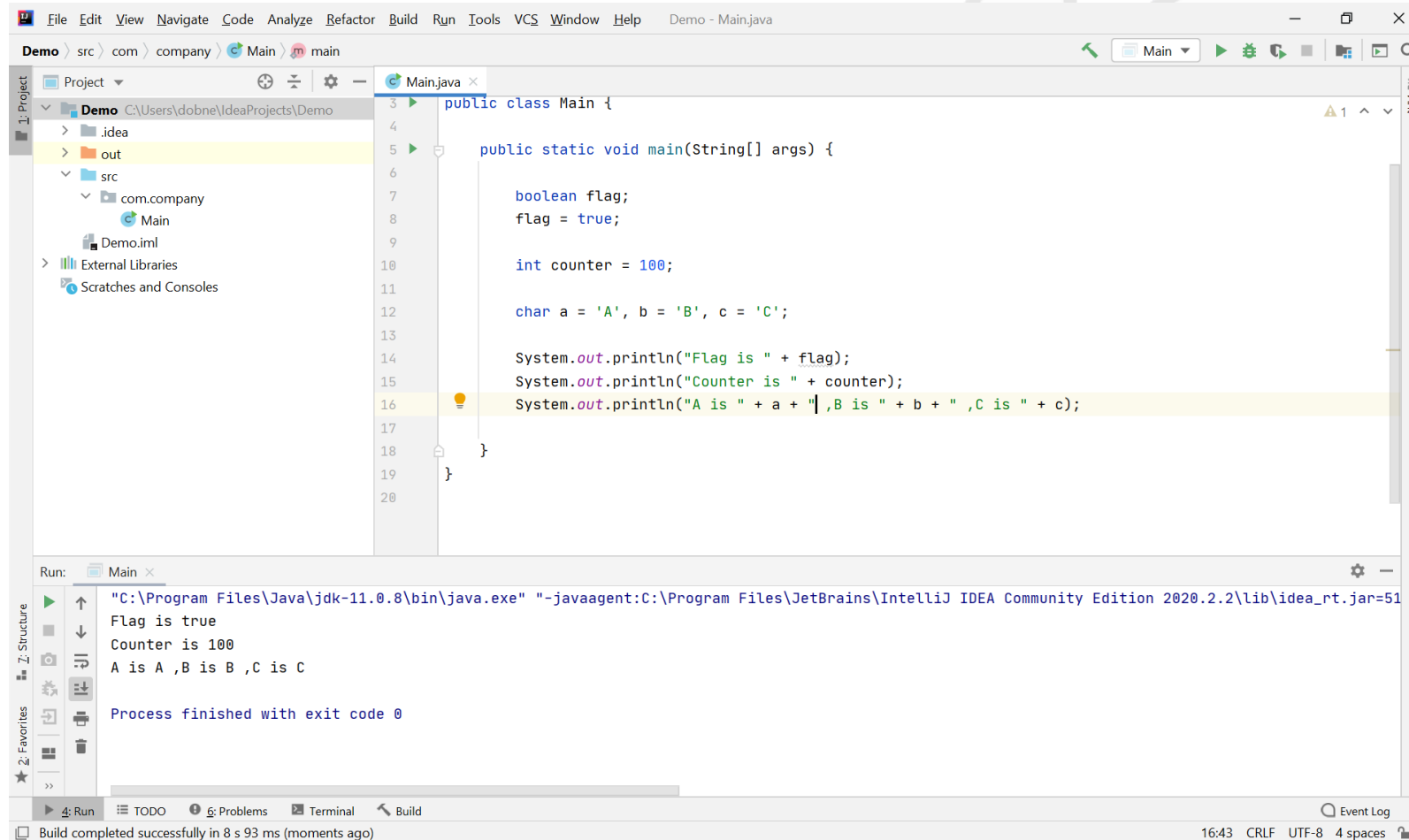
**Typy proste nie są instancjami obiektów.**





# Deklaracja i inicjalizacja zmiennych

*typ\_zmiennej nazwa\_zmiennej = wartość\_zmiennej;*



The screenshot displays the IntelliJ IDEA IDE interface. The main editor window shows a Java file named `Main.java` with the following code:

```
public class Main {  
    public static void main(String[] args) {  
        boolean flag;  
        flag = true;  
  
        int counter = 100;  
  
        char a = 'A', b = 'B', c = 'C';  
  
        System.out.println("Flag is " + flag);  
        System.out.println("Counter is " + counter);  
        System.out.println("A is " + a + ", B is " + b + ", C is " + c);  
    }  
}
```

The left sidebar shows the project structure with the following hierarchy:

- Project
- Demo (C:\Users\dobne\IdeaProjects\Demo)
  - .idea
  - out
  - src
    - com.company
      - Main
  - Demo.iml
  - External Libraries
  - Scratches and Consoles

The bottom panel shows the Run output for the `Main` class:

```
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.2\lib\idea_rt.jar=51  
Flag is true  
Counter is 100  
A is A ,B is B ,C is C  
  
Process finished with exit code 0
```

The status bar at the bottom indicates: Build completed successfully in 8 s 93 ms (moments ago). The system clock shows 16:43, and the encoding is UTF-8 with 4 spaces.

# Zasady nazewnictwa zmiennych

*//Good*

```
char firstLetterOfMyName = 'A';
```

*//Not bad, but wrong practice*

```
int Counter = 100;
```

*//Not bad, but stupid*

```
byte rAnDoMbYtEvAlUe = 123;
```

*//Wrong*

```
int_6axis;
```

- nazwa może się składać z wielkich i małych liter oraz cyfr, znaku podkreślenia i znaku dolara
- dobrą praktyką jest nazywać zmienne zgodnie z tym co przechowują
- zmienne powinny zaczynać się od małej litery, a poszczególne człony zaczynać powinny się z dużej
- nie może się jednak zaczynać od cyfry

# Zmienne tablicowe

*typ\_tablicy nazwa\_tablicy = new typ\_tablicy[liczba\_elementów];*

- zmienna specjalna do przechowywania uporządkowanych elementów jednego typu
- aby używać zmiennej tablicowej w pierwszej kolejności należy zaalokować jej wielkość.
- pierwszy element tablicy to 0 a nie 1
- niezainicjalizowany element tablicy wynosi 0
- odwołanie do elementu tablicy poza jej zakresem powoduje wyjątek.

Project

- Demo C:\Users\dobne\IdeaProjects\Demo
  - .idea
  - out
  - src
    - com.company
      - Main
  - Demo.iml
  - External Libraries
  - Scratches and Consoles

```

2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         int table[] = new int[5];
8         table[0] = 500;
9         table[1] = 1000;
10        System.out.println("Table on index 0 is " + table[0]);
11        System.out.println("Table on index 1 is " + table[1]);
12        System.out.println("Table on index 2 is " + table[2]);
13        System.out.println("Table on index 100 is " + table[100]);
14
15    }
16
17 }
18

```

Run: Main

```

"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.2\lib\idea_rt.jar=51
Table on index 0 is 500
Table on index 1 is 1000
Table on index 2 is 0
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 100 out of bounds for length 5
    at com.company.Main.main(Main.java:36)

Process finished with exit code 1

```

# Rzutowanie

*typ\_zmiennej nazwa\_zmiennej = (typ\_zmiennej)zmienna\_lub  
wartość\_innego\_typu ;*

```
int exampleInt = 100;  
double exampleDouble = 123.123;  
int intFromCharValue = 'A';  
byte byteFromAnotherIntVariable = (byte) exampleInt;  
short shortFromIntValue = (short) 40000;  
int intFromAnotherDoubleValue = (int) exampleDouble;
```

```
int exampleInt = 100;  
double exampleDouble = 123.123;  
  
long exampleLongWithSuffix = 100L;  
  
float exampleFloatWithSuffix = 123.123f;  
double exampleDoubleWithSuffix = 123.123d;
```

- konstrukcja programistyczna umożliwiająca traktowanie danej pewnego, konkretnego typu, jak daną innego typu.
- Domyślnie liczba 100 jest interpretowana jako int, a 123.123 jako double.



FileEditViewNavigateCodeAnalyzeRefactorBuildRunToolsVCSWindowHelp

Demo - Main.java

Demo > src > com > company > Main > main

Project

Demo

C:\Users\dobne\IdeaProjects\Demo

> .idea

> out

> src

> com.company

> Main

Demo.iml

> External Libraries

> Scratches and Consoles

Main.java

```
3 public class Main {
4
5     public static void main(String[] args) {
6
7         int exampleInt = 100;
8         double exampleDouble = 123.123;
9         int intFromCharValue = 'A';
10        byte byteFromAnotherIntVariable = (byte) exampleInt;
11        short shortFromIntValue = (short) 40000;
12        int intFromAnotherDoubleValue = (int) exampleDouble;
13
14
15        System.out.println("Example int = " + exampleInt);
16        System.out.println("Example double = " + exampleDouble);
17        System.out.println("Int from char = " + intFromCharValue);
18        System.out.println("Byte from int = " + byteFromAnotherIntVariable);
19        System.out.println("Short from bigger int = " + shortFromIntValue);
20        System.out.println("Int from double = " + intFromAnotherDoubleValue);
21    }
22 }
```

Run: Main

Example int = 100  
Example double = 123.123  
Int from char = 65  
Byte from int = 100  
Short from bigger int = -25536  
Int from double = 123  
  
Process finished with exit code 0

4: Run

TODO

6: Problems

Terminal

Build

Event Log

Build completed successfully in 34 s 460 ms (a minute ago)

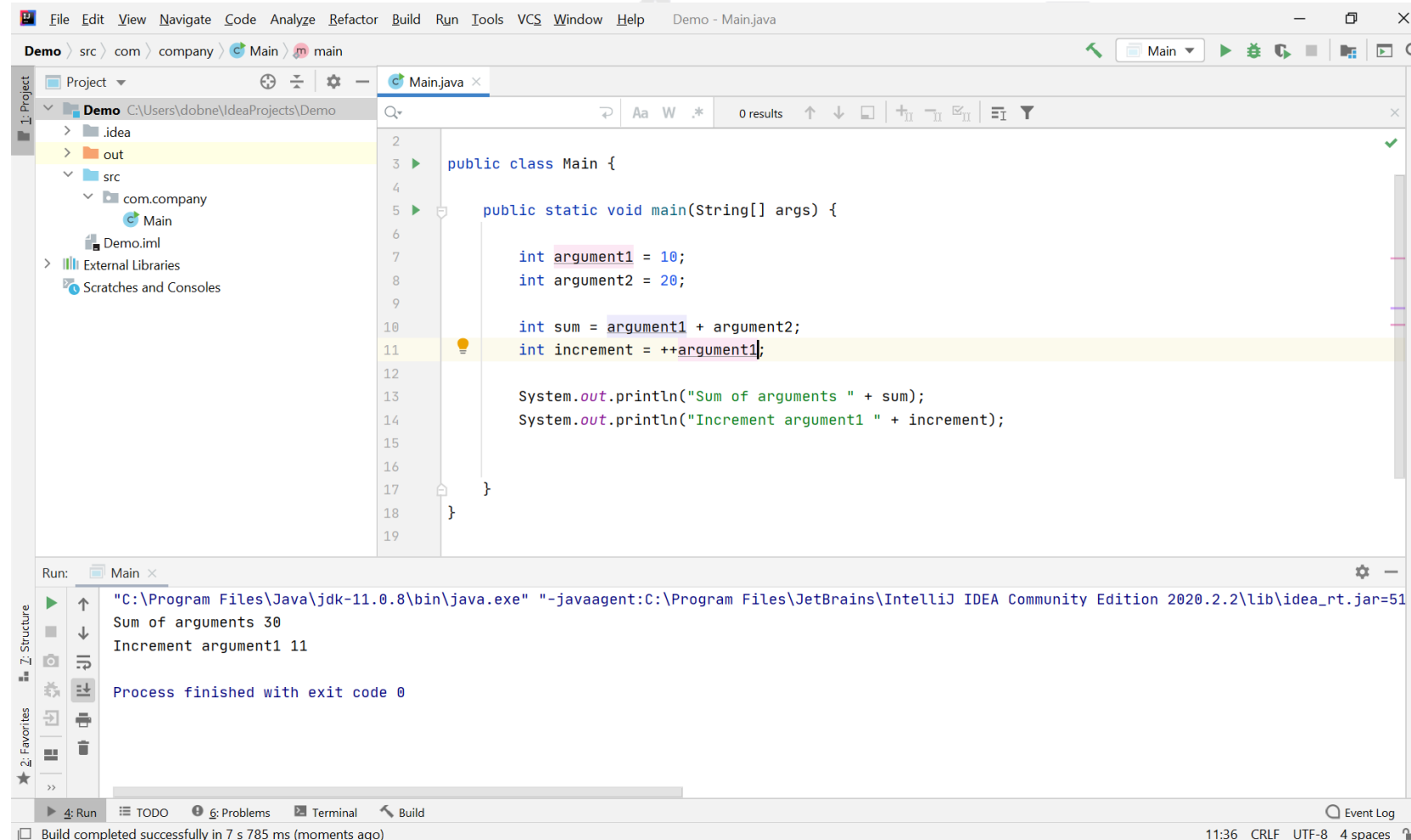
21:6 CRLF UTF-8 4 spaces

# Operator

Znaki specjalne  
wykonujące operacje  
na dostarczonych  
argumentach

Rozróżniamy

- jednoargumentowe
- dwuargumentowe



```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Demo - Main.java
Demo > src > com > company > Main > main
Project Demo C:\Users\dobne\IdeaProjects\Demo
  > .idea
  > out
  > src
    > com.company
      > Main
      Demo.iml
    External Libraries
    Scratches and Consoles
Main.java
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         int argument1 = 10;
8         int argument2 = 20;
9
10        int sum = argument1 + argument2;
11        int increment = ++argument1;
12
13        System.out.println("Sum of arguments " + sum);
14        System.out.println("Increment argument1 " + increment);
15
16    }
17
18 }
19
Run: Main
"C:\Program Files\Java\jdk-11.0.8\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.2\lib\idea_rt.jar=51
Sum of arguments 30
Increment argument1 11
Process finished with exit code 0
Build completed successfully in 7 s 785 ms (moments ago) 11:36 CRLF UTF-8 4 spaces
```

# operatory

## arytmetyczne

+

-

\*

/

%

++

--

## bitowe

&

|

~

^

<<

>>

>>>

## logiczne

&&

||

!

## przypisania

=

+=

-=

\*=

/=

%=

<<=

>>=

>>>=

&=

|=

^=

## porównania

==

!=

>

<

>=

<=

## warunkowy

?

# Priorytety operatorów

Grupa operatorów	Symbole
inkrementacja przyrostkowa	++, --
inkrementacja przedrostkowa, negacje	++, --, ~, !
mnożenie, dzielenie	*, /, %
dodawanie, odejmowanie	+, -
przesunięcia bitowe	<<, >>, >>>
porównania	<, >, <=, >=
porównania	==, !=
bitowe AND	&
bitowe XOR	^
bitowe OR	
logiczne AND	&&
logiczne OR	
warunkowe	?
przypisania	=, +=, -=, *=, /=, %=, >>=, <<=, >>>=, &=, ^=,  =

# Komentarze

```
// one line comment
```

```
/*  
multi  
line  
comment  
*/
```

```
/**  
 * multi line comment  
 * used in documentation  
 * generation  
 */
```

- fragment kodu źródłowego ignorowany przez kompilator w procesie kompilacji, nie wykorzystywany jako część instrukcji programu, którego jedynym celem istnienia jest informowanie o czymś osoby czytającej kod, a który nie ma żadnego wpływu na program.

Zadanka!

