

Podstawy języka JAVA

Wykład nr 1



Podstawy języka JAVA – informacje wstępne

- Prowadzący: mgr inż. Łukasz Paciorek
- Adres e-mail: l.paciorek@wit.edu.pl
- Liczba laboratoriów komputerowych: 24 godziny
- Zaliczenie na ocenę:
 - Aktywność na zajęciach – 60% oceny,
 - miniprojekt – 40% oceny.



Podstawy języka JAVA – informacje wstępne

Lp.	Liczba godzin	Temat zajęć
1	3	Wprowadzenie do języka Java. Kluczowe zagadnienia związane z rozwojem języka Java. Konfiguracja i wykorzystywanie środowiska programistycznego Eclipse. Tworzenie i kompilowanie projektów Java. Typy danych, zmienne, operatory, tablice oraz pętle w języku Java. Podstawowe elementy wejścia/wyjścia w języku Java.
2	3	Programowanie strukturalne, a programowanie obiektowe. Operacje na plikach. Podstawowe zasady obiektowości w języku Java. Rodzaje relacji pomiędzy obiektami w języku Java. Obiekty, zmienne obiektowe oraz referencje w języku Java.
3	3	Hermetyzacja, konstruktory i metody w języku Java. Interfejsy i kolekcje w języku Java.
4	3	Napisy w języku Java. Klasy: String oraz StringBuilder. Typ wyliczeniowy w języku Java. Rodzaje wyjątków w języku Java oraz własne klasy wyjątków. Obsługa wyjątków w języku Java.
5	3	Pola statyczne, stałe statyczne i metody statyczne w języku Java. Parametry metod. Sposoby przekazywanie parametrów do metod. Dzienniki w języku Java.
6	3	Deklaracje i nadpisywanie metod w języku Java. Komentarze dokumentacyjne w języku Java. Generowanie dokumentacji za pomocą javadoc. Debugowanie aplikacji w języku Java.
7	3	Konstruktory i ich przeciążanie w języku Java. Konstruktory bezargumentowe. Bloki inicjalizujące i niszczenie obiektów. Dziedziczenie w języku Java.
8	3	Hierarchia klas w języku Java. Wykorzystanie pakietów w projektach, zasięg pakietów. Polimorfizm w języku Java. Wprowadzenie do wzorców projektowych.
Razem	24	

JAVA - Wstęp

- JAVA została wydana w roku 1995
- Zyskała popularność i uznanie wśród wielu firm. Tylko jeden gigant nie zaakceptował Javy – Microsoft.
- Twórcy Javy napisali tzw. "białą księgę", w której przedstawili cele i osiągnięcia.
- Dodatkowo opublikowali streszczenie, które zorganizowane zostało według 11 słów kluczowych opisujących język Java.



11 SŁÓW KLUCZOWYCH O JAVA

1. Prostota – prostota budowania aplikacji bez konieczności kończenia „tajemniczych szkoleń” jak w przypadku języka C++. Składnia Javy jednak jest bardzo podobna do języka C++.
2. Obiektowość – skupienie się na danych (obiektach) oraz interfejsach dających dostęp do tych obiektów.
3. Sieciowość – Java ma bogatą bibliotekę sieciową, umożliwiającą wykorzystanie takich protokołów jak HTTP czy FTP.
4. Niezawodność – aplikacje w języku Java mają być niezawodne i uniemożliwić np. nadpisanie pamięci za pomocą wskaźników jak w C++. Dodatkowo wiele potencjalnych błędów wykrywanych jest w trakcie kompilacji.



11 SŁÓW KLUCZOWYCH O JAVA

5. Bezpieczeństwo – w Java postawiono silny nacisk na bezpieczeństwo aplikacji, a szczególnie aplikacji sieciowych. Twierdzono, że aplikacje Java są odseparowane od systemu użytkownika.
6. Niezależność od architektury – kompilator Java generuje kod bajtowy niezależny od architektury procesora. Aplikację można uruchomić na dowolnym komputerze pod warunkiem, że posiada zainstalowany Java Runtime System.
7. Przenośność – Java nie jest uzależniona od konkretnej implementacji tak jak język C++ np. int zawsze ma 32 bity.



11 SŁÓW KLUCZOWYCH O JAVA

8. Interpretacja – interpreter Java jest w stanie wykonać każdy kod bajtowy Javy bezpośrednio na urządzeniu gdzie ten interpreter jest zainstalowany. Dodatkowo środowisko Javy ma umożliwiać programowanie „szybkie i odkrywczе” – możliwość szybkiego sprawdzania wyników.....
9. Wysoka wydajność – kompilatory JIT (Just In Time) mogą osiągać lepszą wydajność od kompilatorów tradycyjnych.
10. Wielowątkowość – wykorzystanie procesorów wielordzeniowych.
11. Dynamiczność – podatność Javy na zmiany oraz rozbudowę klas i bibliotek.



Historia JAVY



2009



Środowisko programistyczne JAVA

Pakiety Java na stronie firmy Oracle:

Nazwa	Akronim	Objaśnienie
Java Development Kit	JDK	Oprogramowanie dla programistów, którzy chcą pisać aplikacje JAVA
Java Runtime Environment	JRE	Oprogramowanie do uruchamiania aplikacji napisanych w Java
Server JRE	-	Oprogramowanie do uruchamiania aplikacji napisanych w Java na serwerach
Standard Edition	SE	Platforma Javy do zastosowań na komputerach biurowych oraz prostych aplikacji serwerowych
Enterprise Edition	EE	Platforma Javy przeznaczona do skompilowanych zastosowań serwerowych
Micro Edition	ME	Platforma Javy dla telefonów komórkowych oraz innych małych urządzeń wbudowanych.

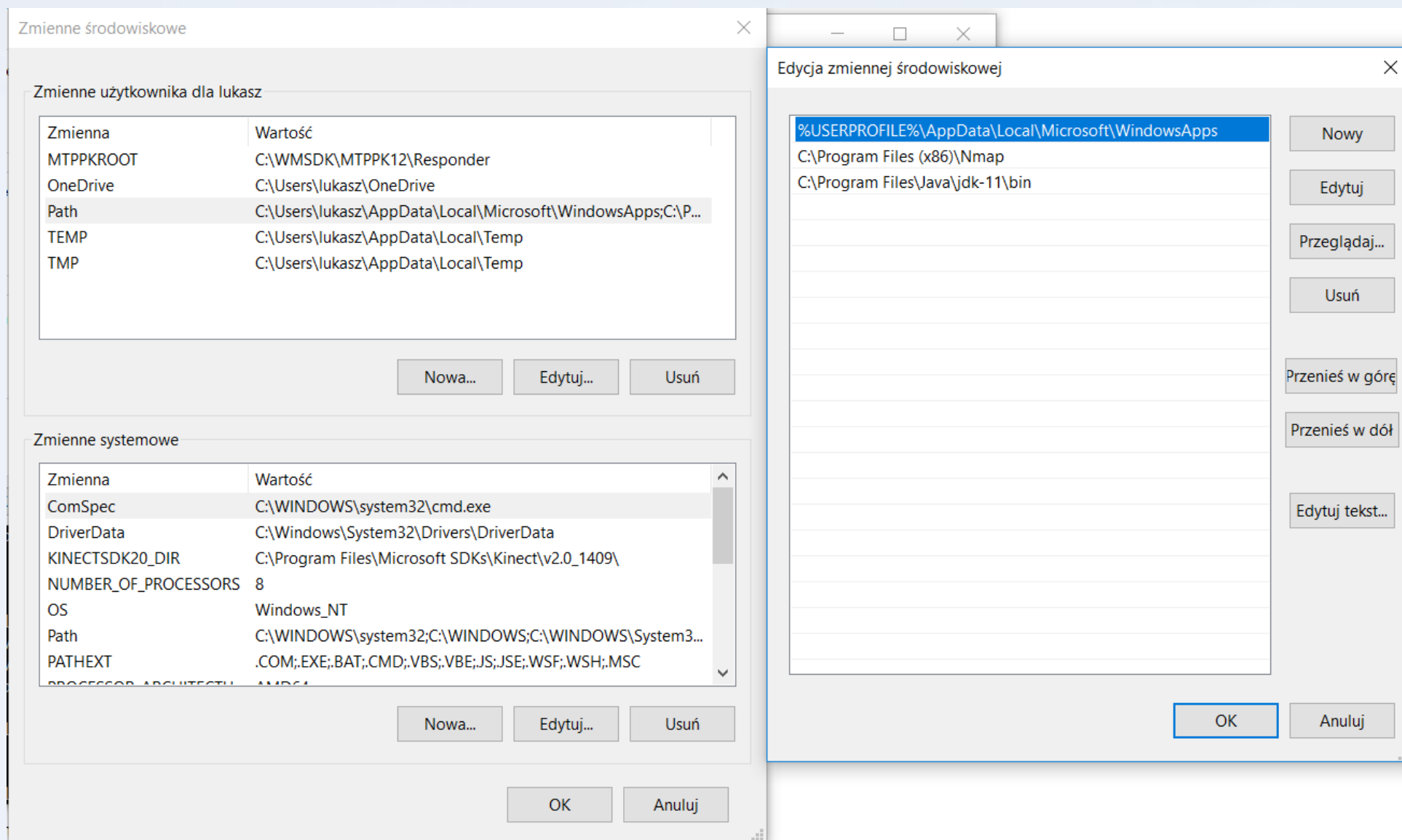


Środowisko programistyczne JAVA

- Obecna wersja Java SE to wersja 15.
- W celu pisania aplikacji w Java należy pobrać JDK odpowiednie do wersji naszego procesora tzn. x64 lub x86.
- Następnie przechodzimy przez proces instalacji.
- Należy uruchomić wiersz poleceń (cmd.exe) i sprawdzić czy rozpoznawane jest polecenie javac.
- Jeżeli polecenie nie jest rozpoznawalne należy dodać ścieżkę do jdk_wersja/bin do zmiennej PATH.



Środowisko programistyczne JAVA



Środowisko programistyczne JAVA

- Uruchamiamy ponownie linię komend i wpisujemy: `javac --version`
- W przypadku poprawnego wykonania otrzymamy numer wersji zainstalowanego JDK

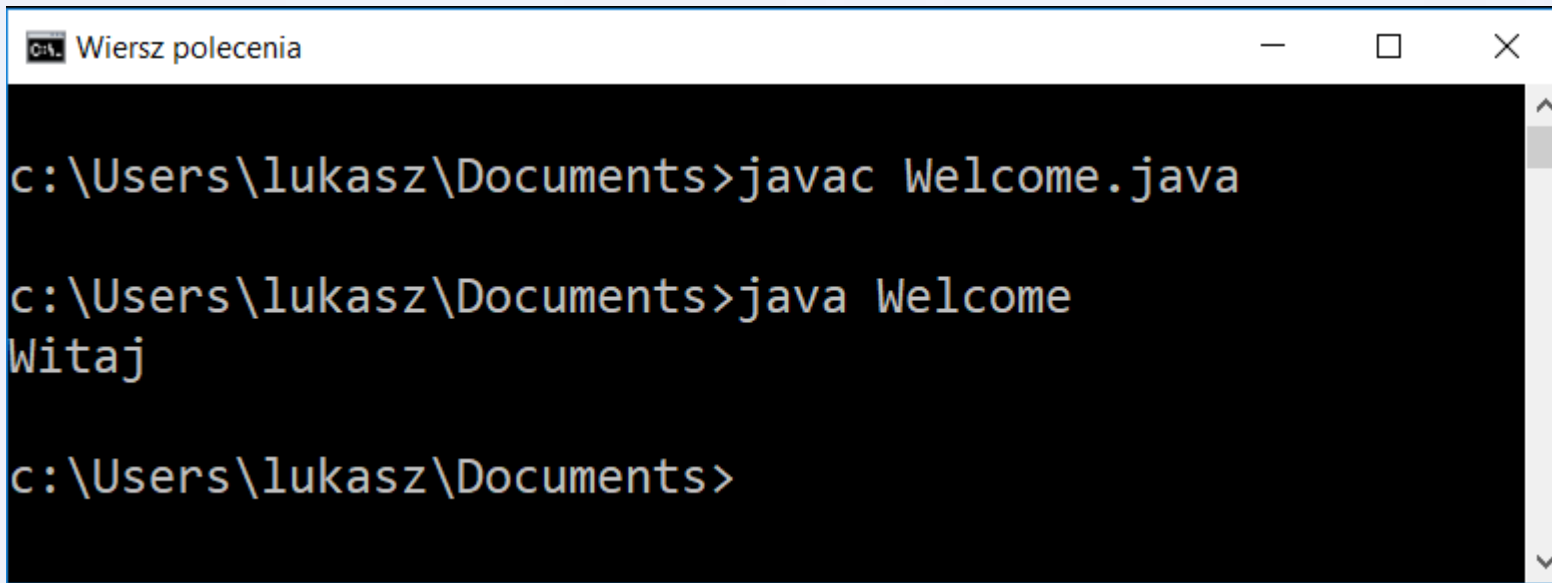
```
C:\Users\lukasz>javac --version
javac 13

C:\Users\lukasz>
```



Środowisko programistyczne JAVA

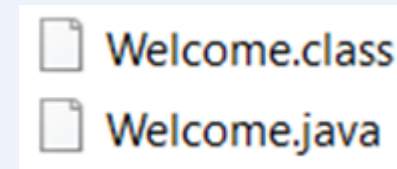
- Zainstalowane JDK umożliwia kompilowanie i uruchamianie kodu z wiersza poleceń.
- W celu skompilowania aplikacji należy wykonać polecenie `javac nazwa.java`
- W celu uruchomienia aplikacji należy wykonać polecenie `java nazwa`



```
C:\Users\lukasz\Documents>javac Welcome.java

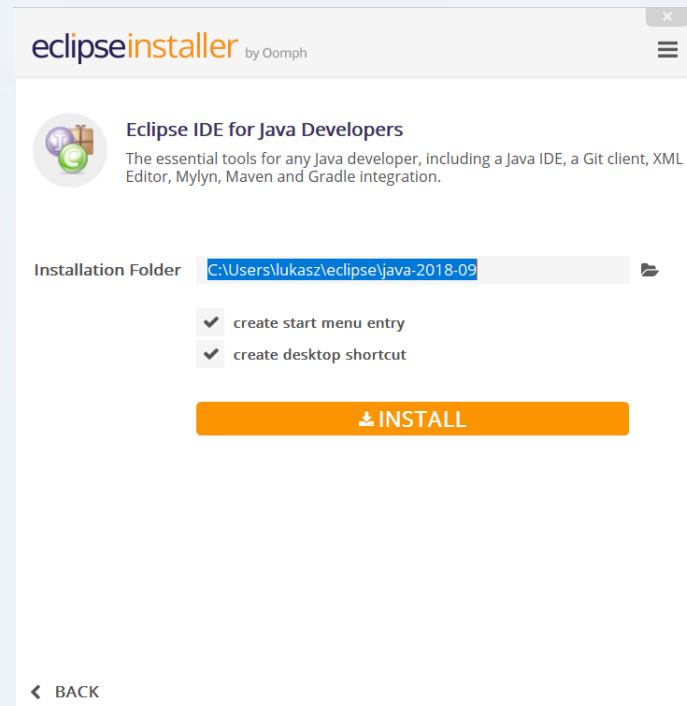
C:\Users\lukasz\Documents>java Welcome
Witaj

C:\Users\lukasz\Documents>
```



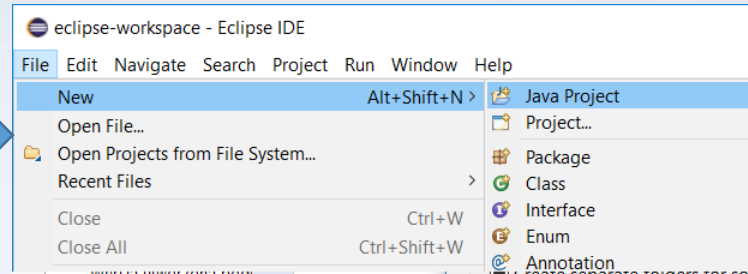
Środowisko programistyczne JAVA

- Najpopularniejszym środowiskiem graficznym do pisania aplikacji Java jest Eclipse.
- Eclipse dostępny jest pod adresem: www.eclipse.org
- Ze strony pobierany jest instalator, w którym należy wskazać Eclipse IDE for Java Developers

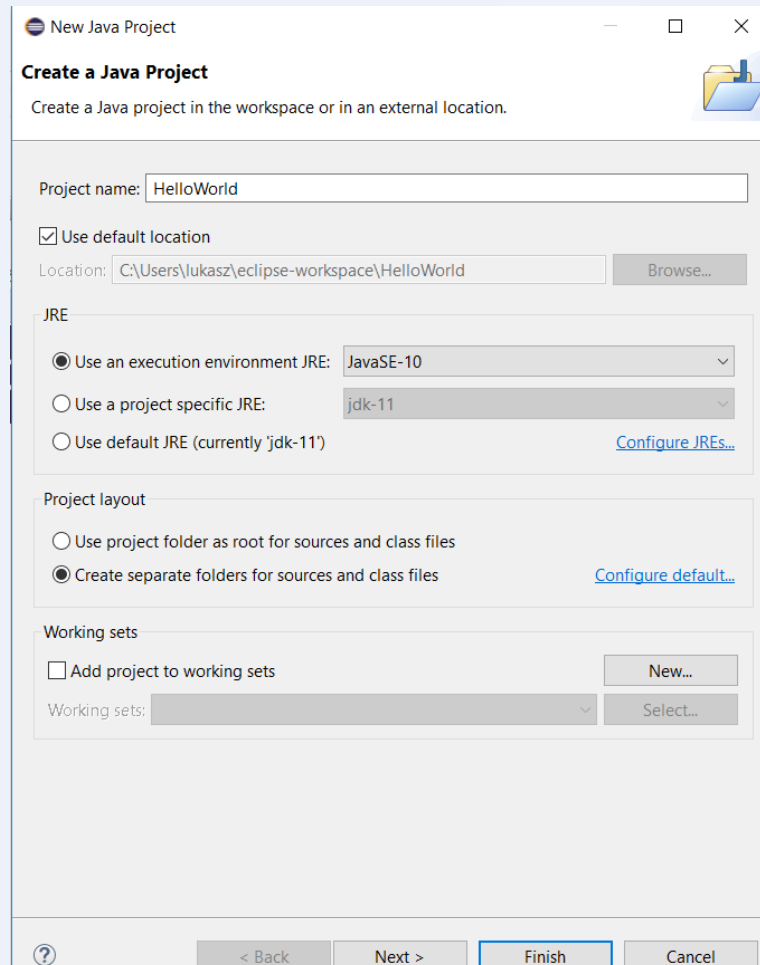


Eclipse – „Hello World”

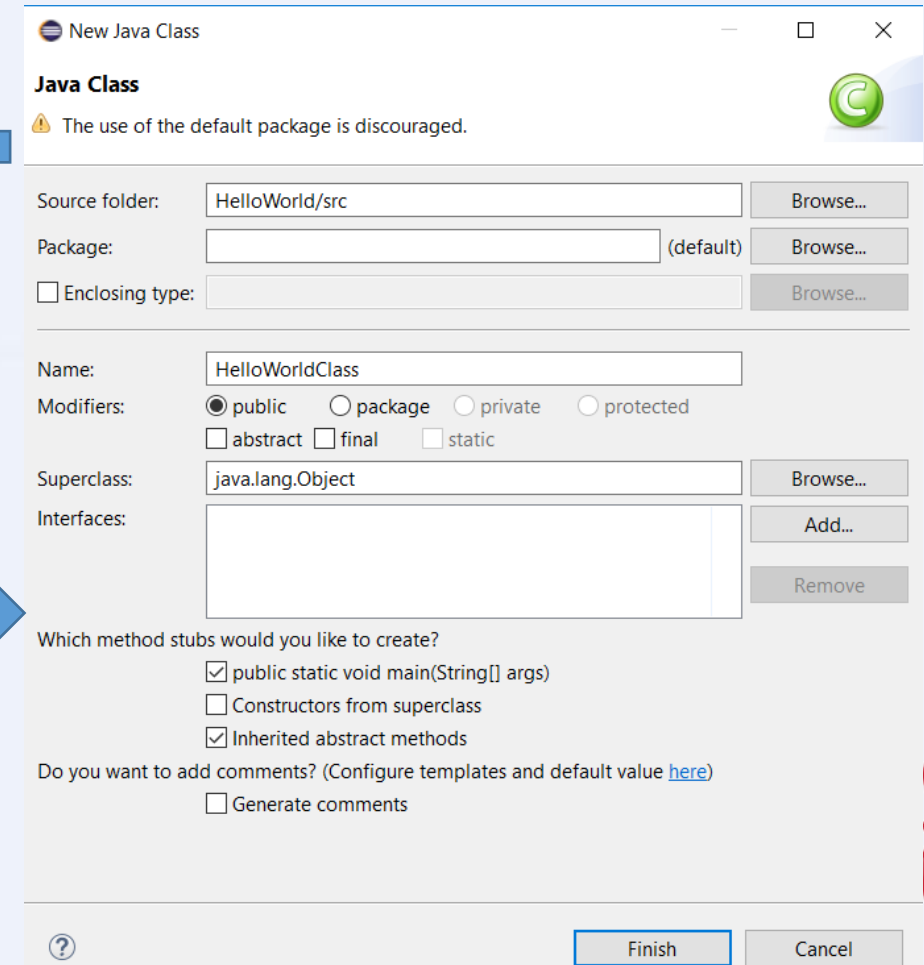
1. Tworzymy nowy projekt
Java



2. Wprowadzamy nazwę
projektu oraz opcje



3. Tworzymy nową klasę
z metodą main



Java – typy danych

- Java jest językiem programowania o ścisłej kontroli typów.
- Oznacza to, że każda zmienna musi mieć określony typ.
- Java posiada osiem podstawowych typów.
- Cztery z nich reprezentują liczby całkowite, dwa – liczby rzeczywiste, jeden reprezentuje symbole Unicode, a ostatni wartość logiczną (true lub false).



Java – typy całkowite oraz zmiennoprzecinkowe

Typ	Liczba bajtów	Zakres
int	4	od -2 147 483 648 do 2 147 483 647
short	2	od -32 768 do 32 767
long	8	od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807
byte	1	od -128 do 127

Typ	Liczba bajtów	Zakres
float	4	Około $\pm 3,40282347E+38F$ (6-7 znaczących cyfr dziesiętnych)
double	8	Około $\pm 1,79769313486231570E+308$ (15 znaczących cyfr dziesiętnych)

Java – typy całkowite oraz zmiennoprzecinkowe

- Istnieją trzy szczególne wartości umożliwiające określenie wartości, które wykraczają poza dozwolony zakres błędu:
- Dodatnia nieskończoność
- Ujemna nieskończoność
- NaN – Not a Number (0/0 lub pierwiastek kwadratowy z liczby ujemnej)



Java – typ char oraz boolean

- Początkowo każdy znak reprezentowano za pomocą jednej zmiennej typu char, jednak obecnie niektóre znaki wymagają dwóch zmiennych.
- W Java typ char reprezentuje jedną jednostkę kodową UTF-16.
- Zmienne typu char umieszcza się w cudzysłowie pojedynczym np. ' A ' to zmienna typu char o wartości 65 i nie jest tym samym co np. „A” czyli łańcuchem znaków zawierającym jeden znak.
- Zalecane jest jednak używanie ciągów znaków, a nie typu char.
- Typ boolean może przyjmować dwie wartości (true lub false) i wykorzystywany jest do sprawdzania warunków logicznych.



Java – zmienne

- W Java każda zmienna musi mieć określony typ.
- Deklaracja zmiennej polega na napisaniu jej typu, a następnie nazwy tej zmiennej.

```
int licznik;  
double wynik;  
long licznikLudnosci;
```

- Nazwa zmiennej musi zaczynać się od litery i składać się musi z liter oraz cyfr.
- Po zadeklarowaniu zmiennej należy zainicjować ją wartością.

```
int licznik;  
licznik=10;  
double wynik = 10.0094;
```



Java – stałe

- W Java stałe oznaczamy słowem kluczowym final.
- Słowo kluczowe final oznacza, że można tylko raz przypisać wartość do zmiennej i nie można jej później zmieniać.
- Zwyczajowo nazwy stałych pisze się samymi wielkimi literami.

```
final int COUNTER_JAVA = 10;
```

- W Java wykorzystuje się również stałe klasowe – dostępne w obrębie jednej klasy.
- Stałe klasowe określa się słowem kluczowym static final

```
public class HelloWorldClass {  
  
    public static final int COUNTER_JAVA = 10;  
    public static void main(String[] args) {  
    }  
  
}
```



Java – operatory

Operator	Działanie
+	Operacja dodawania
-	Operacja odejmowania
*	Operacja mnożenia
/	Dzielenie całkowitoliczbowe (jeżeli obie wartości są całkowitoliczbowe) oraz zmiennoprzecinkowe (w przeciwnym przypadku)
%	Dzielenie modulo
++	Operator inkrementacji
--	Operator dekrementacji



Java – operatory

- Jakie wartości będą miały zmienne a oraz b?

```
public static void main(String[] args) {  
    int a=10;  
    int b=10;  
    int c=0;  
    int d=0;  
    a = a+(c++);  
    b = b+(++d);  
    System.out.println("a = "+a);  
    System.out.println("b = "+b);  
}
```



Java – operatory

- Jakie wartości będą miały zmienne a oraz b?

```
public static void main(String[] args) {  
    int a=10;  
    int b=10;  
    int c=0;  
    int d=0;  
    a = a+(c++);  
    b = b+(++d);  
    System.out.println("a = "+a);  
    System.out.println("b = "+b);  
}
```

```
a = 10  
b = 11
```



Java – operatory relacyjne i logiczne

Operator	Działanie
==	Operator równości. Sprawdzenie równości dwóch argumentów (zwraca true lub false)
!=	Operator nierówności.
>	Operator większości
<	Operator mniejszości
>=	Operator większe lub równe
<=	Operator mniejsze lub równe
&&	Koniunkcja logiczna
	Alternatywa logiczna
!	Negacja logiczna

Java – operatory bitowe

Operator	Działanie
&	Koniunkcja bitowa
	Alternatywa bitowa
~	Bitowa negacja
^	Lub wykluczające
>>	Przesunięcie bitowe w prawo
<<	Przesunięcie bitowe w lewo



Java – klasa Math

- Klasa Math zawiera zestaw funkcji matematycznych, użytecznych w trakcie pisania programów.

Funkcja	Opis
Math.sqrt(x)	Obliczenie pierwiastka kwadratowego
Math.pow(x,a)	Podniesienie zmiennej x do potęgi a
Math.floor(x)	Zwraca „podłogę” dla wartości x
Math.abs(x)	Zwraca wartość bezwzględna z x
Math.cos(x), Math.sin(x), Math.tan(x)	Funkcje trygonometryczne
Math.exp(x), Math.log(a), Math.log10(a)	Funkcje wykładnicza oraz logarytmy
Math.PI, Math.E	Stałe matematyczne



Java – instrukcja warunkowa

- Instrukcja warunkowa (if) pozwala na sprawdzenie warunku i w przypadku jego spełnienia (wartość true) wykonanie odpowiedniego bloku programu.

```
if(salary > target)
{
    //Blok 1 do wykonania
}
else if (salary < target)
{
    //Blok 2 do wykonania
}
else
{
    //Blok 3 do wykonania
}
```



Java – pętla while

- Pętla while wykonuje instrukcję lub blok instrukcji tak długo, jak warunek ma wartość true.
- Instrukcje pętli while nie zostaną nigdy wykonane, jeżeli warunek ma wartość false na początku.

```
int counter=0;
while(counter<100)
{
    System.out.println(counter);
    counter++;
}
while(counter>100)
{
    System.out.println(counter);
    counter++;
}
```



Java – pętla do while

- Pętla while nie gwarantuje nawet jednokrotnego wykonania instrukcji.
- Pętla do while pozwala na wykonanie instrukcji (lub bloku instrukcji), a dopiero później na sprawdzenie warunku.
- Instrukcje w bloku do będą wykonywane dopóki warunek while ma wartość true.

```
int counter=0;
do
{
    System.out.println(counter);
    counter++;
}
while(counter<100);
```



Java – pętla for

- Pętla for pozwala dokładnie określić liczbę powtórzeń pętli.
- Za pomocą licznika (lub innej zmiennej) określamy liczbę powtórzeń (wartość licznika zmienia się z każdym wykonaniem pętli).

```
for(counter=0;counter<=10;counter++)  
{  
    System.out.println(counter);  
}
```

- Pętla for jest wygodnym „narzędziem” do przechodzenia po elementach tablicy.
- Pętla for jest krótszym zapisem pętli while.



Java – pętla for

```
for(int counter=0;counter<=10;counter++)  
{  
    System.out.println(counter);  
}  
for(int counter=10;counter<15;counter++)  
{  
    System.out.println(counter);  
}  
for(int counter=10;counter>=0;counter--)  
{  
    System.out.println(counter);  
}
```



Java – instrukcja switch

- W przypadku kiedy jest wiele możliwości do wyboru, wykorzystanie funkcji if oraz if-else może być mało efektywne.
- W Java dostępna jest instrukcja switch (w takiej samej postaci jak w językach C/C++).

```
int counter=1;
switch (counter)
{
    case 1:
        counter=2;
        System.out.println(counter);
    case 2:
        counter=3;
        System.out.println(counter);
    case 3:
        counter=4;
        System.out.println(counter);
}
```



Java – instrukcja switch

- W przypadku kiedy jest wiele możliwości do wyboru, wykorzystanie funkcji if oraz if-else może być mało efektywne.
- W Java dostępna jest instrukcja switch (w takiej samej postaci jak w językach C/C++).

```
int counter=1;
switch (counter)
{
case 1:
    counter=2;
    System.out.println(counter);
case 2:
    counter=3;
    System.out.println(counter);
case 3:
    counter=4;
    System.out.println(counter);
}
```

```
int counter=1;
switch (counter)
{
case 1:
    counter=2;
    System.out.println(counter);
    break;
case 2:
    counter=3;
    System.out.println(counter);
    break;
case 3:
    counter=4;
    System.out.println(counter);
    break;
}
```



Java – instrukcja switch

- W instrukcji switch można dodać blok default, który zostanie wykonany jeżeli żaden case nie został spełniony.

```
int counter=1;
switch (counter)
{
    case 1:
        counter=2;
        System.out.println(counter);
        break;
    case 2:
        counter=3;
        System.out.println(counter);
        break;
    case 3:
        counter=4;
        System.out.println(counter);
        break;
    default:
        System.out.println("Nie udało się");
}
```



Java – tablice

- Tablica to rodzaj struktury danych będąca zestawem elementów tego samego typu
- Dostęp do elementów tablicy odbywa się za pomocą indeksu typu int.
- Deklaracja tablicy polega na określeniu typu i nazwy tablicy.

```
int[] tablica;
```

- Utworzenie tablicy realizuje się poprzez wykorzystanie operatora new.

```
tablica = new int[100];
```

- Elementy tablicy są indeksowane od 0.



Java – tablice – pętla for each

- Pętla for może zostać wykorzystana do przeglądania elementów tablicy.
- Programista może wykorzystać standardową formę pętli for lub formę będącą odpowiednikiem for each.

```
int[] tablica= {10,50,39,34,43};  
for(int i=0;i<5;i++) {  
    System.out.println(tablica[i]);  
}  
for(int i:tablica)  
{  
    System.out.println(i);  
}
```



Java – tablice - kopiowanie

- Jakie wartości wypisze poniższy kod?

```
int[] tablica= {10,20,30,40,50};  
int[] tablicaPom = tablica;  
tablicaPom[0]=0;  
System.out.println(tablicaPom[0]);  
System.out.println(tablica[0]);
```



Java – tablice - kopiowanie

- Jakie wartości wypisze poniższy kod?

```
int[] tablica= {10,20,30,40,50};  
int[] tablicaPom = tablica;  
tablicaPom[0]=0;  
System.out.println(tablicaPom[0]);  
System.out.println(tablica[0]);
```

- Wyświetlone zostaną dwie wartości 0, ponieważ tablica oraz tablicaPom wskazują na ten sam obszar pamięci.
- W celu skopiowania tablicy, należy wykorzystać metodę copyOf z klasy Arrays.
- Metoda copyOf przyjmuje dwa parametry: nazwę tablicy oraz jej długość.



Java – tablice - kopiowanie

- Metoda copyOf może zostać wykorzystana do zwiększenia długości tablicy, bądź jej skrócenia.
- W przypadku zwiększenia długości, nowe elementy dostaną wartości domyślne tj. 0 / false / null, w zależności od typu tablicy.
- W przypadku skrócenia tablicy, skopiowane zostaną tylko pierwsze elementy.



Java – tablice - kopiowanie

```
import java.util.Arrays;

public class HelloWorldClass {
    public static void main(String[] args) {
        int[] tablica = {30,40,50,60,70,80,90};
        int[] tab = Arrays.copyOf(tablica, 3);
        int[] tab2 = Arrays.copyOf(tablica, 15);
        for(int i:tab)
        {
            System.out.println(i);
        }
        for(int i:tab2)
        {
            System.out.println(i);
        }
    }
}
```



Java – tablice wielowymiarowe

- Do reprezentowania tabel lub innych złożonych struktur danych w Java można wykorzystać tablice wielowymiarowe.
- W celu uzyskania dostępu do tablicy wielowymiarowej należy wykorzystać więcej niż jeden indeks.

```
int[][] tabela = {{10,20,30,40},{50,60,70,80},{100,200,300,400}};  
for(int[] i:tabela)  
{  
    for(int j:i)  
    {  
        System.out.println(j);  
    }  
}
```



Java –IO

- Wypisywanie danych do standardowego wyjścia (konsoli) odbywa się za pomocą System.out.println.
- Odczytywanie danych ze standardowego wejścia (konsoli) odbywa się przy wykorzystaniu klasy Scanner.

```
import java.util.Scanner;

public class HelloWorldClass {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
    }
}
```



Java – IO – klasa Scanner

Metoda	Opis
Scanner (InputStream in)	Tworzy obiekt klasy Scanner przy pomocy podanego strumienia wejściowego.
String NextLine()	Wczytuje kolejny wiersz danych.
String next()	Wczytuje kolejne słowo (do spacji)
int nextInt()	Wczytuje kolejną wartość całkowitoliczbową.
double nextDouble()	Wczytuje kolejną wartość zmiennoprzecinkową.
Boolean hasNext()	Sprawdza czy jest kolejne słowo
int hasNextInt()	Sprawdza czy jest kolejna wartość całkowitoliczbową.
double hasNextDouble()	Sprawdza czy jest kolejna wartość zmiennoprzecinkowa.



Java – IO – klasa Scanner

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        System.out.println("Podaj swój wiek");
        int wiek = in.nextInt();
        int rok = 2020-wiek;
        System.out.println("Urodziles sie w roku: "+rok);
    }
}
```

Podaj rok swojego urodzenia.

20

Urodziłeś się w roku: 1998



Java – IO – wyjście

- Wartość zmiennej do konsoli można wypisać za pomocą `System.out.print(zmienna)`.
- W Java SE5 wprowadzono znaną z języków C/C++ metodę `printf`.

```
public static void main(String[] args) {  
    int year = 2018;  
    int month = 10;  
    int day = 6;  
    System.out.printf("Rok %d, miesiąc %d, dzień %d", year, month, day);  
}
```

- Każdy specyfikator formatu, który zaczyna się od znaku % jest zastępowany odpowiadającym mu argumentem.
- Znak konwersji znajdujący się za specyfikatorem formatu, określa typ wartości do sformatowania.



Java – IO – wyjście

Znak konwersji	Typ
d	Liczba całkowita dziesiętna.
x	Liczba całkowita szesnastkowa.
o	Liczba całkowita ósemkowa.
f	Liczba zmiennoprzecinkowa.
e	Liczba zmiennoprzecinkowa w notacji wykładniczej.
a	Liczba zmiennoprzecinkowa szesnastkowa.
s	Łańcuch znaków
c	Znak
b	Wartość logiczna
h	Wartość skrótu
%	Symbol procenta
n	Separator wiersza



Kolejne zajęcia

Programowanie strukturalne, a programowanie obiektowe.
Operacje na plikach. Podstawowe zasady obiektowości w języku Java. Rodzaje relacji pomiędzy obiektami w języku Java. Obiekty, zmienne obiektowe oraz referencje w języku Java.



DZIĘKUJĘ ZA UWAGĘ

