

Programowanie obiektowe w języku JAVA laboratorium

Opracowano na podstawie Tutorialspoint, Learn Java Programming, Java Tutorial
<https://www.tutorialspoint.com/java/index.htm> i innych źródeł
przez Jędrzeja Ułasiewicza

Wrocław 2022

Spis treści

1.	Java przygotowanie środowiska (Environment Setup)	4
1.1	Java Development Kit	4
1.2	Sublime Text 3	4
1.3	Pierwszy program	6
1.4	NetBeans IDE	7
1.4.1	Środowisko NetBeans	8
1.4.2	Tworzenie projektu	8
1.4.3	Tworzenie pliku źródłowego	9
1.4.4	Dodawanie kodu źródłowego	11
1.4.5	Uruchamianie aplikacji	12
1.5	JetBrains IntelliJ	14
1.5.1	Informacje wstępne	14
1.5.2	Pierwszy projekt	14
2.	Podstawy	21
2.1	Program HelloWorld – uruchomienie z konsoli	21
2.2	Program HelloWorld – użycie edytora SublimeText3	21
2.3	Ciąg Fibonacciego	22
2.4	Obliczanie sumy elementów tablicy	23
2.5	Tabliczka mnożenia	23
2.6	Trójkąt	23
2.7	Piramida	24
2.8	Tworzenie obiektów – przykład Puppy	24
3.	Tworzenie obiektów I	26
3.1	Wprowadzanie danych, klasa Scanner	26
3.2	Argumenty funkcji main	26
3.3	Tworzenie obiektów - materiał teoretyczny	27
3.4	Inicjalizacja obiektów – klasa Rectangle	27
3.5	Tworzenie obiektów, przeciążanie konstruktora - klasa Employee	28
3.6	Klasa Employee – dwa pliki	29
4.	Tworzenie obiektów II	30
4.1	Klasa operacji na tablicy	30
4.2	Klasa operacji na tablicy i jej test	31
4.3	Klasa operacji na tablicy - dwa pliki	32
4.4	Rozszerzona instrukcja for	32
4.5	Klasa String – konkatencja	33
4.6	Klasa String – konwersja do tablicy znaków	33
4.7	Klasa String – szukanie podłańcuchów	33
4.8	Usuwanie elementów tablicy	34
5.	Obiekty i metody	36
5.1	Operacji na tablicy sortowanie i odwracanie	36
5.2	Operacji na tablicy – szukanie elementu	37
5.3	Operacje sortowanie i przeszukiwanie tablicy – użycie funkcji bibliotecznych	37
6.	Przeciążanie metod	38
6.1	Wypisywanie danych różnego typu	38
6.2	Przeciążanie metod – klasa Array	39
6.3	Przeciążanie konstruktora - program TestArray12.java	41
7.	Tworzenie obiektów III	43
7.1	Klasa Osoba	43
7.2	Tablica danych osobowych	44
7.3	Baza danych osobowych oparta na tablicy	46
7.4	Klasa Baza danych, zabezpieczenia	47
7.5	Klasa Baza danych, optymalizacja przeszukiwania	47

8.	Hermetyzacja.....	48
8.1	Klasa Baza danych, hermetyzacja klasy Osoba.....	48
9.	Dziedziczenie.....	49
9.1	Klasa Osoba i klasy potomne.....	49
9.2	Baza danych osobowych, polimorfizm.....	50
10.	Interfejsy.....	53
10.1	Sortowanie tablicy – przykład interfejsu.....	53
10.2	Baza danych – sortowanie tablicy z użyciem interfejsu Comparable.....	54
11.	Struktury danych.....	55
11.1	Testowanie struktury Stack.....	55
11.2	Wykorzystanie struktury Vector do budowy bazy danych osobowych.....	57

1. Java przygotowanie środowiska (Environment Setup)

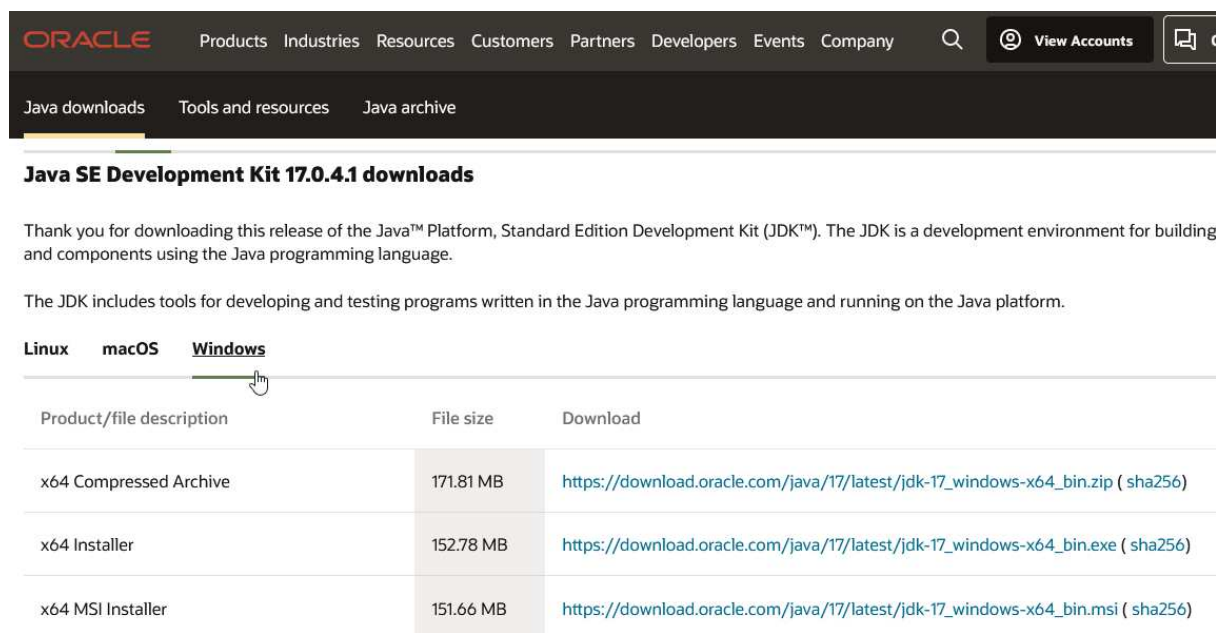
Potrzebujemy:

- JDK – Java Development Kit
- Edytor – Sublime Text3, Notepad++
- NetBeans – Zintegrowane środowisko programistyczne

1.1 Java Development Kit

Pakiet instalacyjny można pobrać ze strony:

<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>



ORACLE Products Industries Resources Customers Partners Developers Events Company View Accounts

Java downloads Tools and resources Java archive

Java SE Development Kit 17.0.4.1 downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building and components using the Java programming language.

The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform.

Linux **macOS** **Windows**

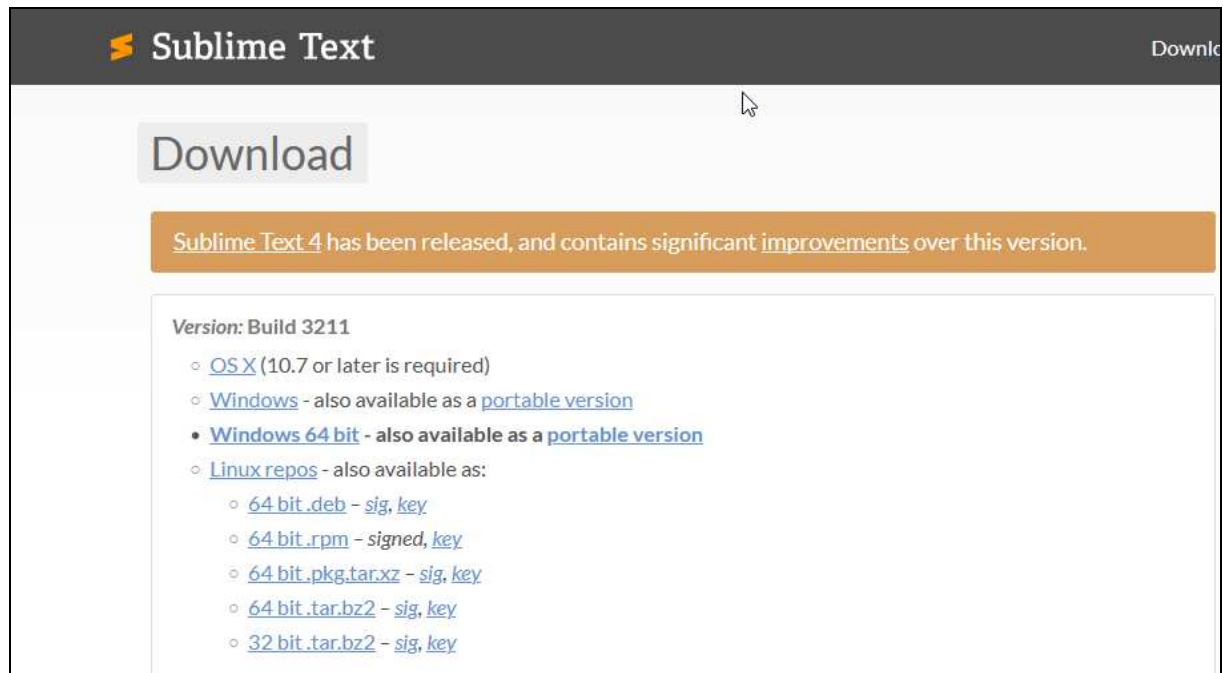
Product/file description	File size	Download
x64 Compressed Archive	171.81 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	152.78 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	151.66 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

Klikając w Windows/x64 otrzymujemy plik instalatora. Po uruchomieniu Java się zainstaluje.

1.2 Sublime Text 3

Sublime Text 3 to wygodny edytor który można pobrać ze strony:

<https://www.sublimetext.com/3>



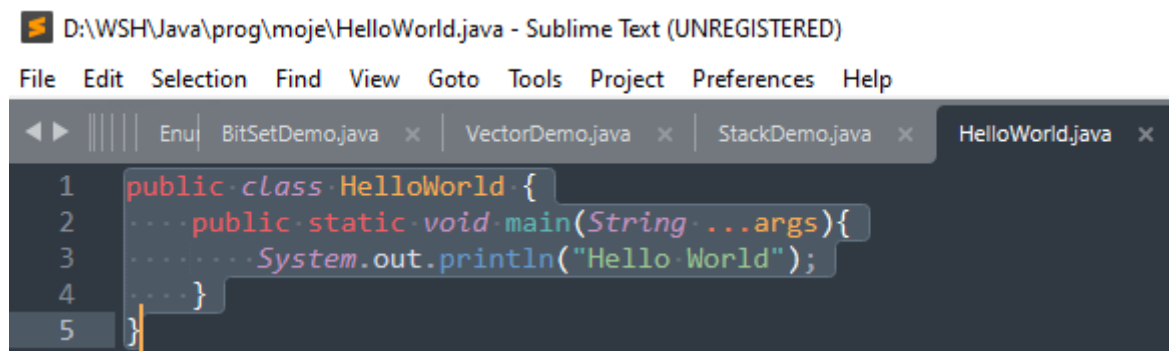
Sublime Text Download

Download

Sublime Text 4 has been released, and contains significant improvements over this version.

Version: Build 3211

- [OS X](#) (10.7 or later is required)
- [Windows](#) - also available as a [portable version](#)
- [Windows 64 bit](#) - also available as a [portable version](#)
- [Linux repos](#) - also available as:
 - [64 bit .deb](#) - [sig](#), [key](#)
 - [64 bit .rpm](#) - [signed](#), [key](#)
 - [64 bit .pkg.tar.xz](#) - [sig](#), [key](#)
 - [64 bit .tar.bz2](#) - [sig](#), [key](#)
 - [32 bit .tar.bz2](#) - [sig](#), [key](#)



D:\WSH\Java\prog\moje\HelloWorld.java - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

BitSetDemo.java x VectorDemo.java x StackDemo.java x HelloWorld.java x

```
1 public class HelloWorld {
2     ... public static void main(String... args) {
3         ..... System.out.println("Hello World");
4         ..... }
5 }
```

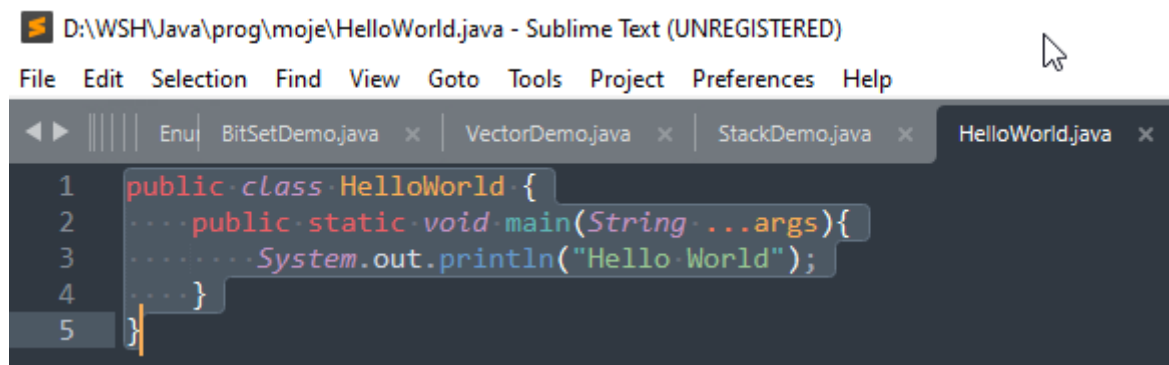
1.3 Pierwszy program

Pierwszy program HelloWorld.java wygląda następująco:

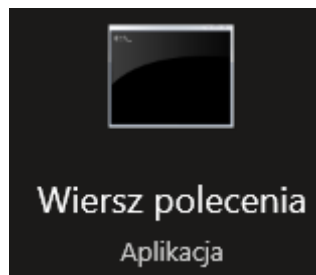
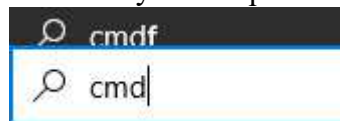
```
public class HelloWorld {
    public static void main(String ...args){
        System.out.println("Hello World");
    }
}
```

Przykład 1-1 HelloWorld.java

Wpisujemy go edytorem.



Otwieramy wiersz polecenia:



```
d:\WSH\Java\prog\moje>ls He*
Hello1.txt HelloSWT.java HelloWorld.java HelloWorld.class HelloWorldApp.java
```

Kompilacja: javac HelloWorld.java

```
d:\WSH\Java\prog\moje>javac HelloWorld.java
```

Powstaje program skompilowany: HelloWorld.class


```
d:\WSH\Java\prog\moje>ls He*
Hello1.txt HelloSWT.java HelloWorld.java HelloWorld.class HelloWorldApp.java
```

Uruchomienie: java HelloWorld

```
d:\WSH\Java\prog\moje>javac HelloWorld.java  
d:\WSH\Java\prog\moje>java HelloWorld  
Hello World
```

1.4 NetBeans IDE

Można pobrać z: <https://netbeans.apache.org/download/index.html>

 **Apache NetBeans** Community

Latest release

Apache NetBeans 12.6

Download

Apache NetBeans Releases

Apache NetBeans is released four times a year. For details, see [full release schedule](#).

Apache NetBeans 12.6

Latest version of the IDE, released on November 29, 2021.

Features Download

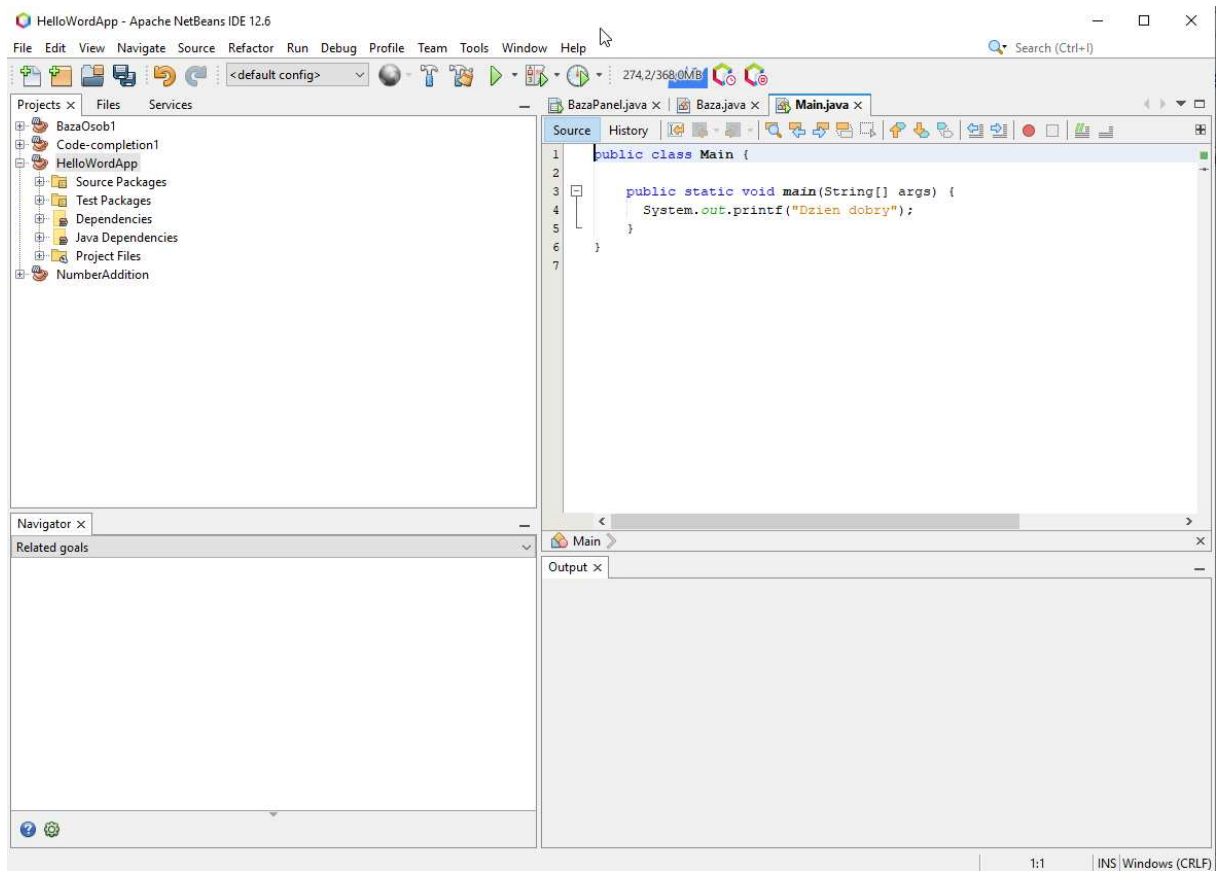
Downloading Apache NetBeans 12.6

Apache NetBeans 12.6 was released on November 29, 2021. See [Apache NetBeans 12.6 Features](#) for a full list of features.

Apache NetBeans 12.6 is available for download from your closest Apache mirror.

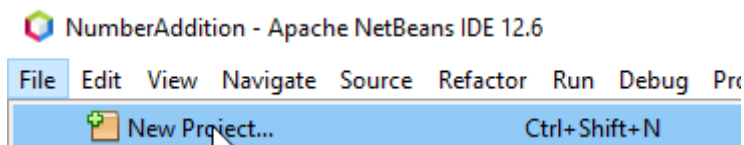
- Binaries: [netbeans-12.6-bin.zip](#) (SHA-512, PGP ASC)
- Installers:
 - [Apache-NetBeans-12.6-bin-windows-x64.exe](#) (SHA-512, PGP ASC)
 - [Apache-NetBeans-12.6-bin-linux-x64.sh](#) (SHA-512, PGP ASC)
 - [Apache-NetBeans-12.6-bin-macosx.dmg](#) (SHA-512, PGP ASC)

1.4.1 Środowisko NetBeans

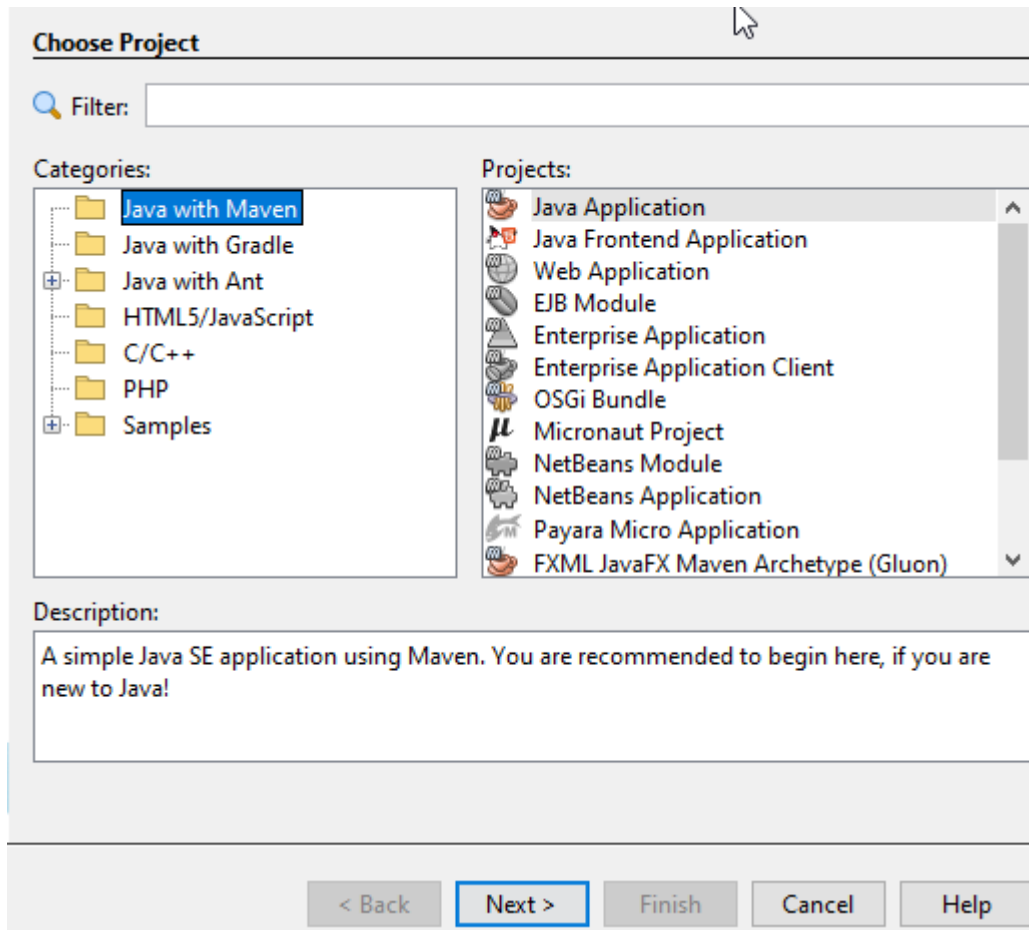


1.4.2 Tworzenie projektu

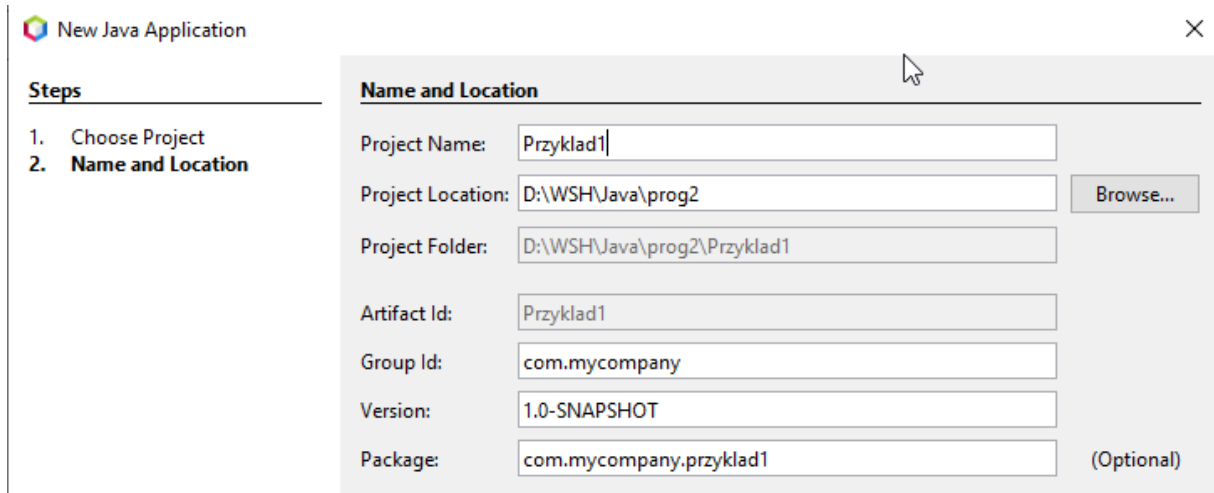
Klikamy w File > New Project



Podświetlamy Java with Maven > Java Application



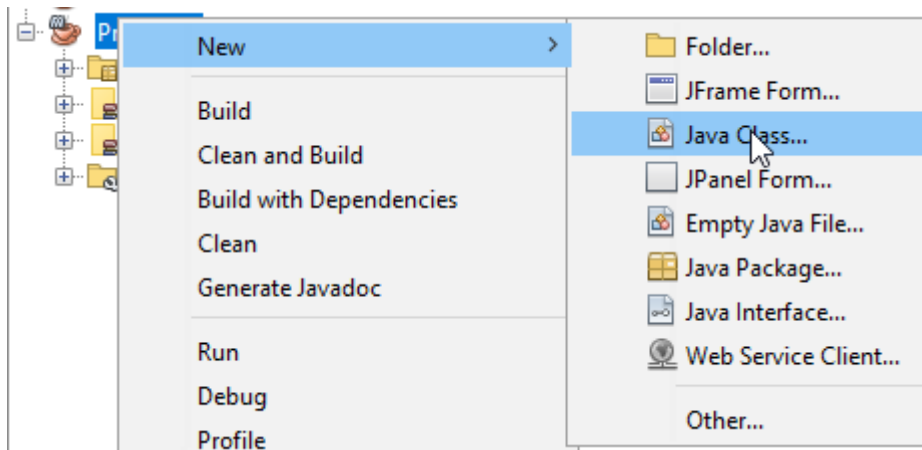
Przyciskamy Next



W polu Project Name Wpisujemy nazwę projektu (wybraną). Dalej naciskamy Finish.
Projekt jest utworzony.

1.4.3 Tworzenie pliku źródłowego

Klikamy prawym klawiszem myszki na ikonę z nazwą projektu. Wybieramy:
New > Java Class



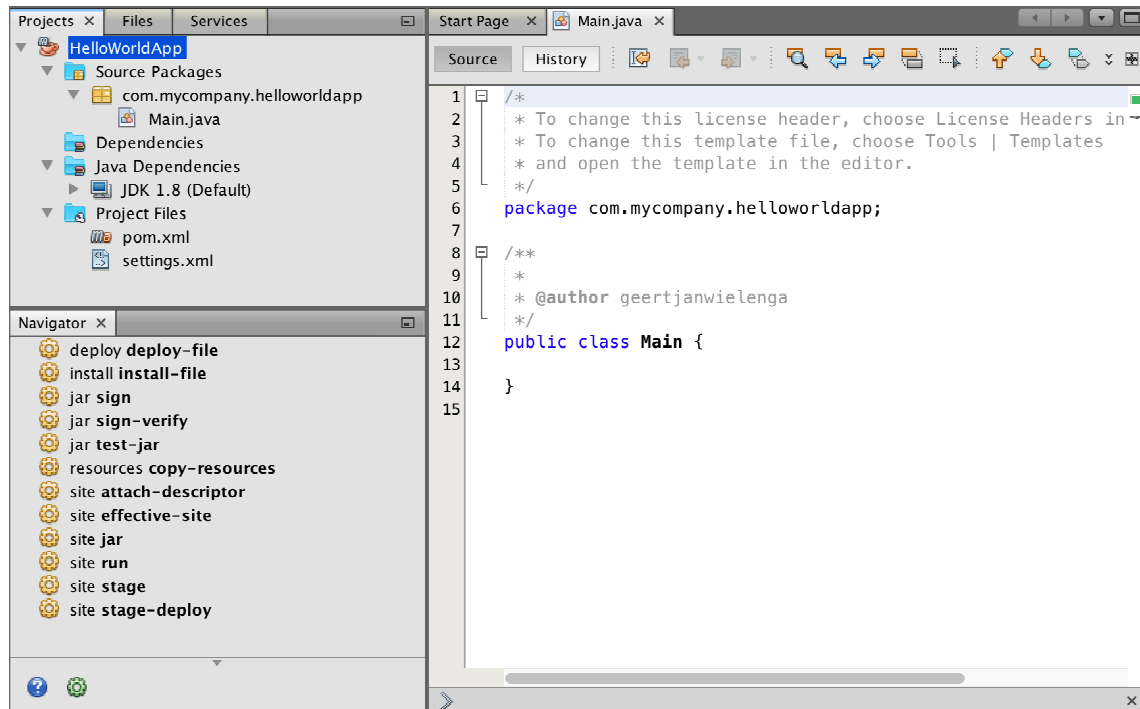
Pojawi się okno jak niżej. W oknie Class Name wpisujemy Main.

Wciskamy klawisz Finish. Zostanie utworzony szkielet aplikacji Hello Word.

Powinny być widoczne następujące komponenty:

- Okno projektu (Project Window) : Położone góra, lewa strona, zawiera drzewo komponentów projektu włączając pliki źródłowe, biblioteki od których zależy projekt, itd.
- Okno edycji kodu (Source Editor): Obszar centralny zawierający kod który musimy napisać. Obecnie plik źródłowy Java z otwartym plikiem Main.

- Nawigator (navigator): Położenie dół, lewa strona, używane do szybkiej nawigacji pomiędzy klasami.



1.4.4 Dodawanie kodu źródłowego

Szkielet kodu został dodany automatycznie. Dodamy teraz treść która umożliwia wypisanie komunikatu Hello word.

Wpisz pomiędzy nawiasami psvm i naciśnij klawisz Tab. Pojawi się fragment kodu:

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

}
```

W zakresie funkcji main napisz sout i naciśnij klawisz Tab. Pojawi się linia:

```
public static void main(String[] args) {
    System.out.println("");
}
```

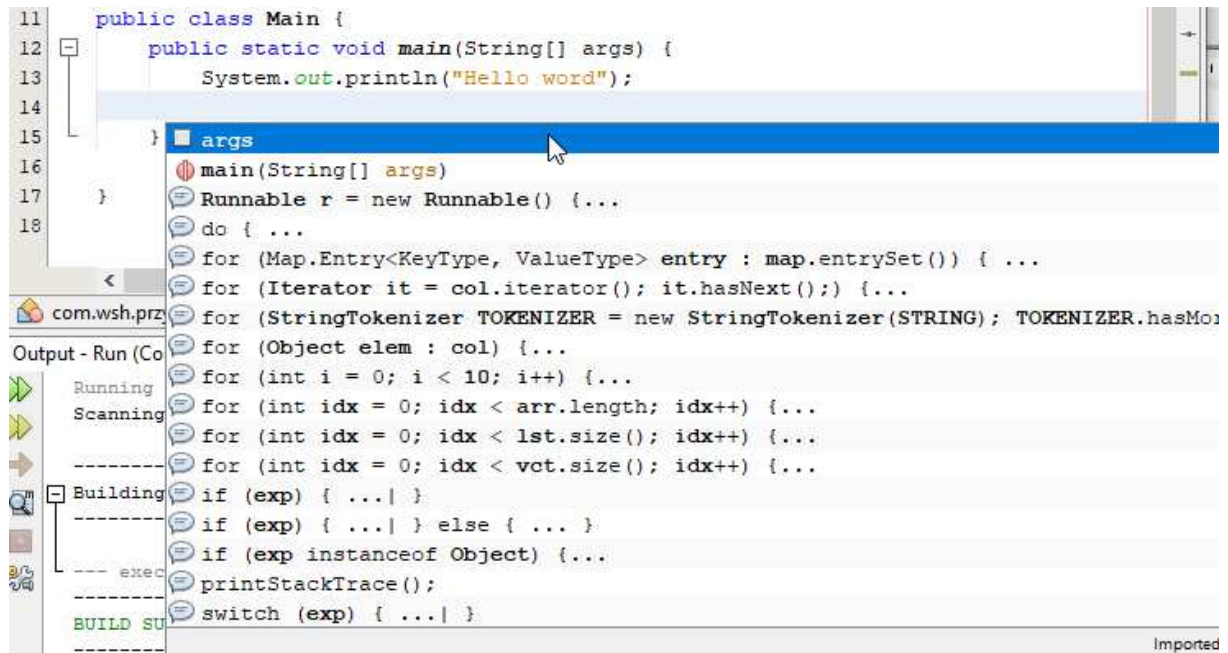
W miejscu pomiędzy cudzysłowami wpisz Hello word.

```
System.out.println("Hello word");
```

Teraz powinien się pojawić następujący kod:

```
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello word");
    }
}
```

Zauważ że gdy naciśniesz **Ctrl+Space**, edytor otworzy okno podpowiedzi z możliwymi tu konstrukcjami.

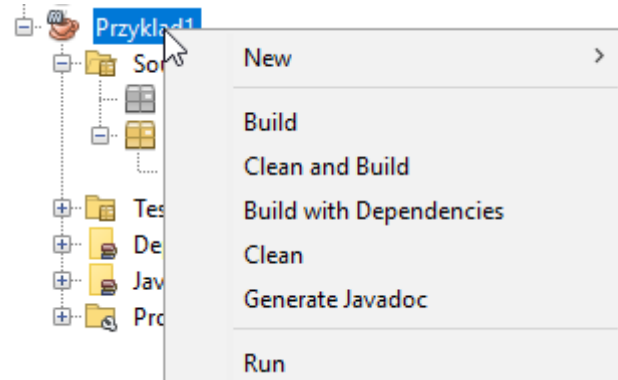


Pełny opis asystenta kodu znajduje się pod linkiem:

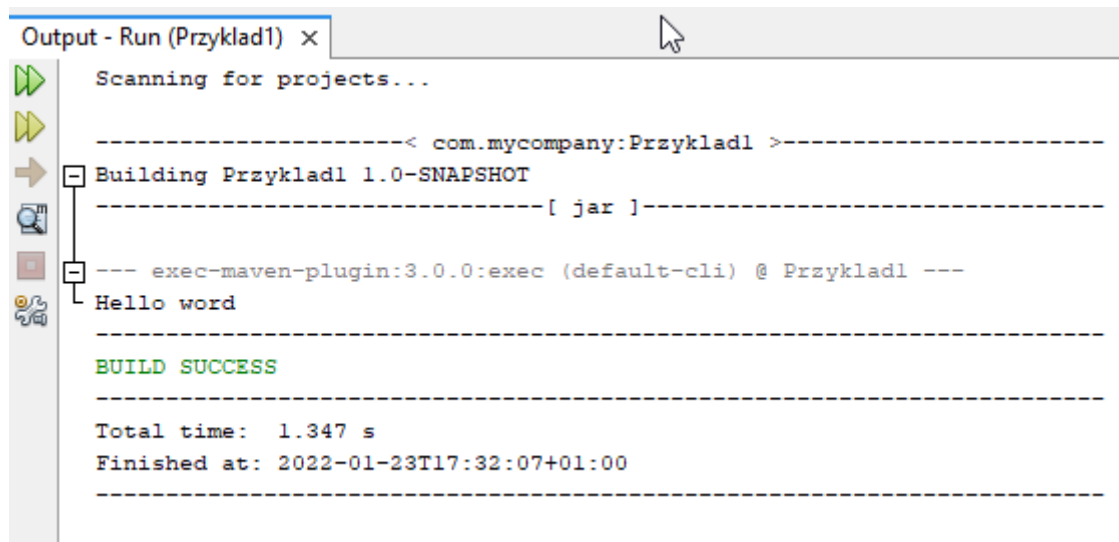
[Code Assistance in the NetBeans IDE Java Editor: A Reference Guide.](#)

1.4.5 Uruchamianie aplikacji

Zachowaj kod źródłowy przez Ctrl + S. Dalej kliknij prawym klawiszem myszy w nazwę projektu i wybierz opcję Run.



Z okna Run wybierz Main Class. W oknie Output (na dole) pojawi się raport kompilacji:



```
Output - Run (Przyklad1) x
Scanning for projects...
-----< com.mycompany:Przyklad1 >-----
Building Przyklad1 1.0-SNAPSHOT
-----[ jar ]-----
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Przyklad1 ---
Hello word
-----
BUILD SUCCESS
-----
Total time: 1.347 s
Finished at: 2022-01-23T17:32:07+01:00
-----
```

Widać tam komunikat: Hello word.

1.5 JetBrains IntelliJ

1.5.1 Informacje wstępne

JetBrains IntelliJ IDEA to, popularne środowisko do tworzenia aplikacji w Javie. Jest ono rozwijane przez czeską firmę JetBrains, która jest także autorem narzędzie dedykowanych do innych języków jak PHP, PyCharm dla Pythona. Przez wielu uważane jest za najlepsze środowisko dedykowane do Javy. Posiada bardzo dobrą integrację z innymi narzędziami i technologiami dedykowanymi dla Javy (Maven, Spring, Java EE, Android). Środowisko to występuje w dwóch wersjach:

- **Community** przeznaczona przede wszystkim do aplikacji pisanych w czystej Javie. Nie ma niestety wsparcia dla technologii, w których Java wykorzystywana jest najczęściej, czyli JEE oraz frameworka Spring. Stworzymy w nim aplikacje na system Android.
- **Ultimate** dedykowana dla programistów projektów biznesowych. Licencja jest wykupowana w systemie abonamentowym i kosztuje ok 600zł rocznie, a w przypadku licencji firmowych już ponad 1500zł. Z tego powodu z wersji tej korzystają najczęściej ludzie, którzy są już zawodowymi programistami i narzędzie to opłacane jest przez pracodawcę lub przez studentów, którzy mogą się ubiegać o bezpłatną wersję.

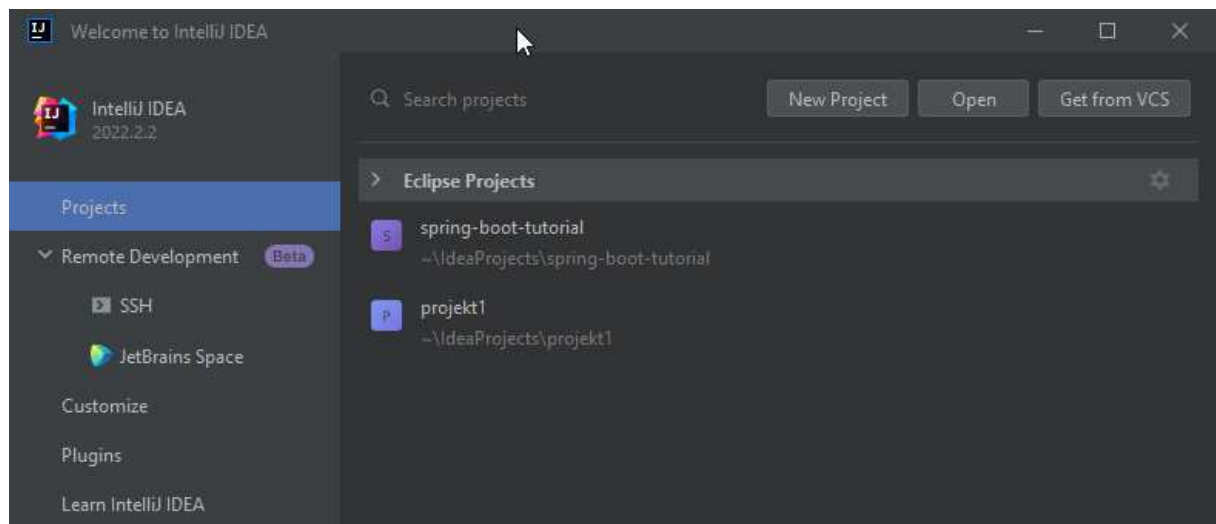
Środowisko można pobrać z oficjalnej strony JetBrains:

<https://www.jetbrains.com/idea/download/>

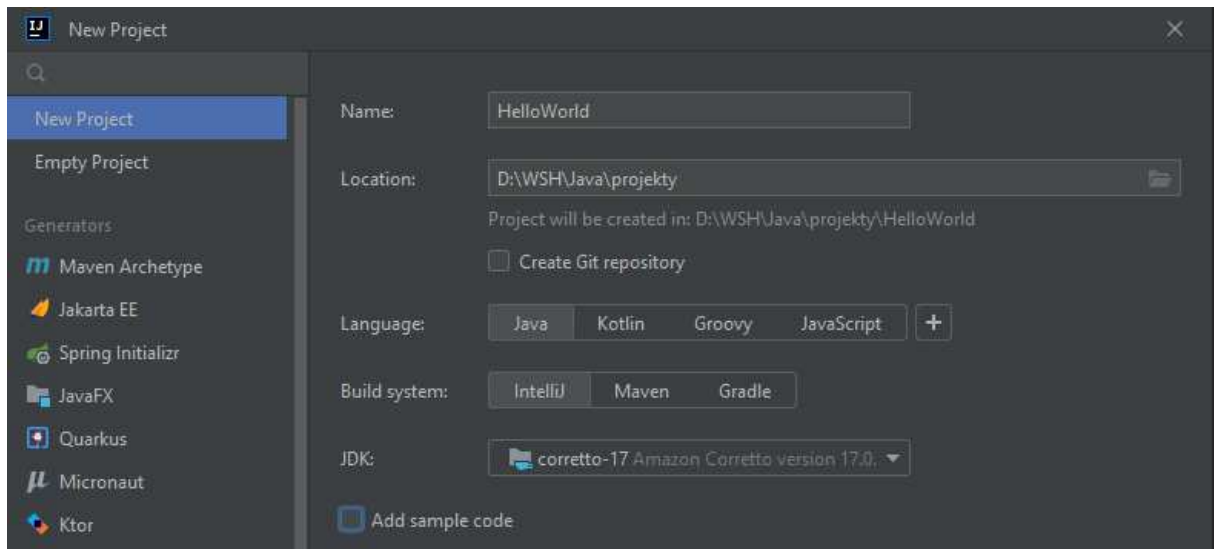
1.5.2 Pierwszy projekt

Tworzenie projektu

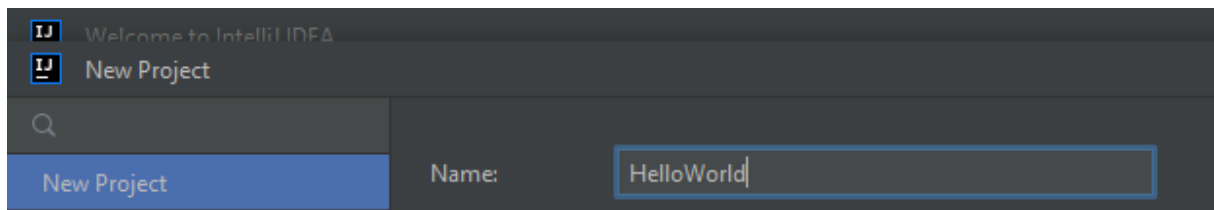
Po uruchomieniu możemy zobaczyć formatkę jak poniżej.



Po wybraniu opcji New Project pojawi się kolejna formatka. Pole Add sample code możemy ustawić jako nieaktywne. Gdy pole będzie aktywne, środowisko wstawi szkielet programu HelloWorld.

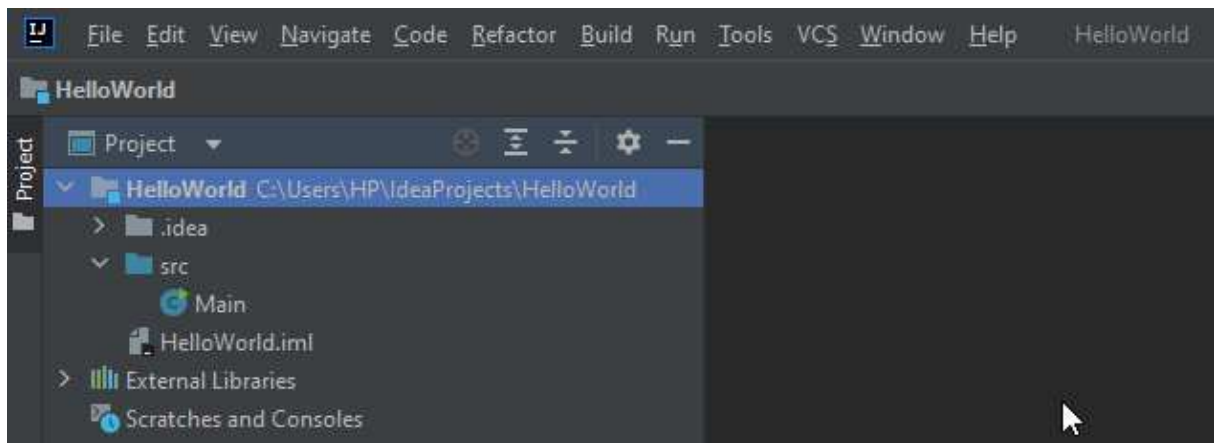


W polu Name wpisujemy wybraną nazwę projektu, w tym przypadku HelloWorld.

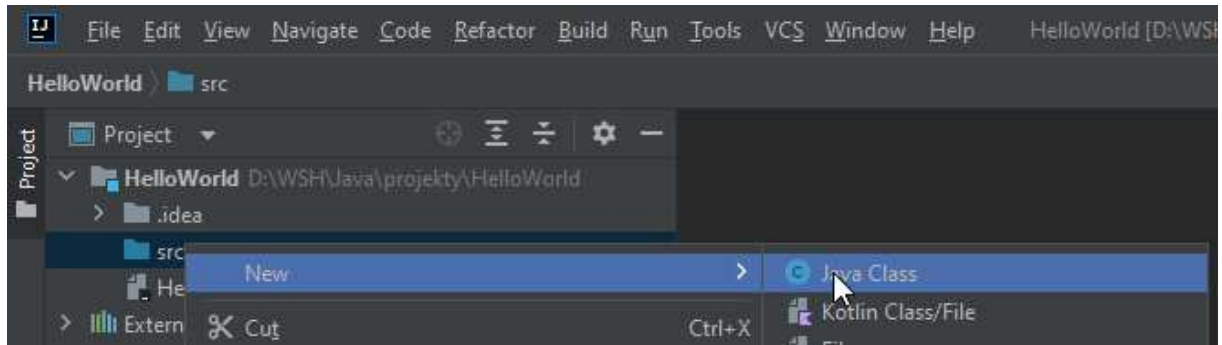


Tworzenie pakietu i klasy

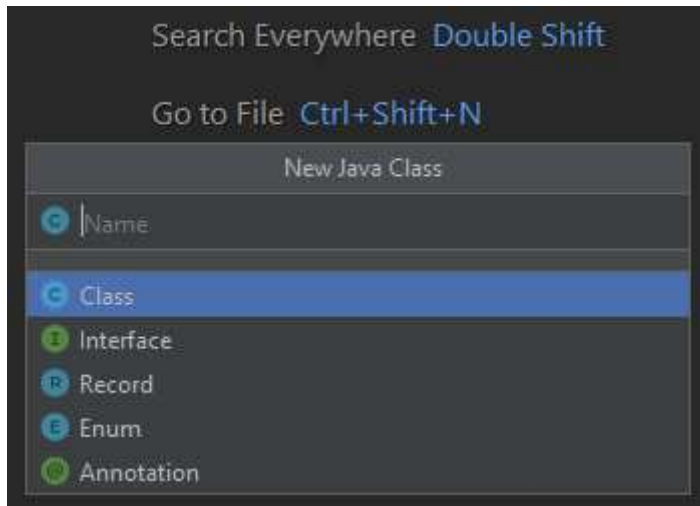
Kolejnym krokiem jest utworzenie klasy. Naciskamy przycisk **Create**. Pojawi się kolejny formularz.



Klikamy prawym klawiszem myszy na ikonę **src**. Pojawi się okno wyboru z którego wybieramy **New** i dalej **Java Class**.



Pojawi się okno wyboru:



Wpisujemy jako nazwę klasy `com.example.helloworld.HelloWorld` i klikamy OK. Oczywiście nazwa pakietu może być inna. Pojawi się szkielet kodu jak poniżej.



Ustawiamy kursor po otwierającym nawiasie i wciskamy **Shift + Enter**. Dalej wpisujemy `ma`



Środowisko uzupełni kod o metodę `main`.


```

1 package com.example.helloworld;
2
3 public class HelloWorld {
4     public static void main(String[] args) {
5
6     }
7 }

```

Uzupełnianie kodu

Środowisko zapewnia możliwość podpowiedzi i uzupełniania kodu. Pokazane to będzie na przykładzie funkcji `System.out.println()`. Po wpisaniu liter `Sy` pojawi się okno podpowiedzi jak poniżej.

```

1 public class Main {
2     public static void main(String[] args) {
3         Sy
4     }
5 }

```

Wybieramy Pakiet `system` wciskając `Enter` i pojawi się nazwa `System`. Gdy po słowie `System` wcisniemy kropkę pojawi się kolejne okno podpowiedzi.

```

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

```

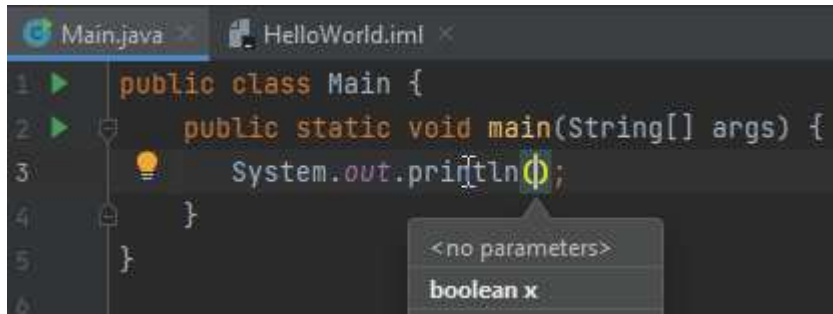
Wybieramy pakiet `out`. Po wstawieniu kropki pojawi się kolejne okno podpowiedzi,

```

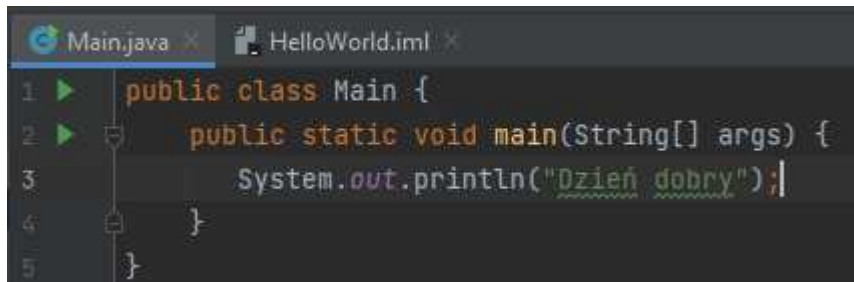
1 public class Main {
2     public static void main(String[] args) {
3         System.out.println
4     }
5 }

```

Wybieramy funkcję `println()`.



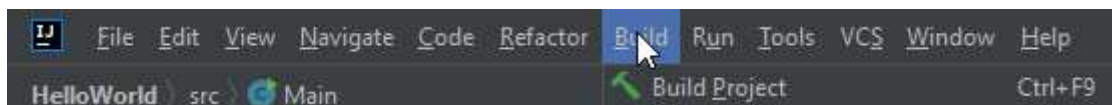
W funkcji println wpisujemy potrzebne parametry, np napis "Dzień dobry" co pokazano poniżej. Po wpisaniu pierwszego cudzysłowu drugi zostanie dodany automatycznie,



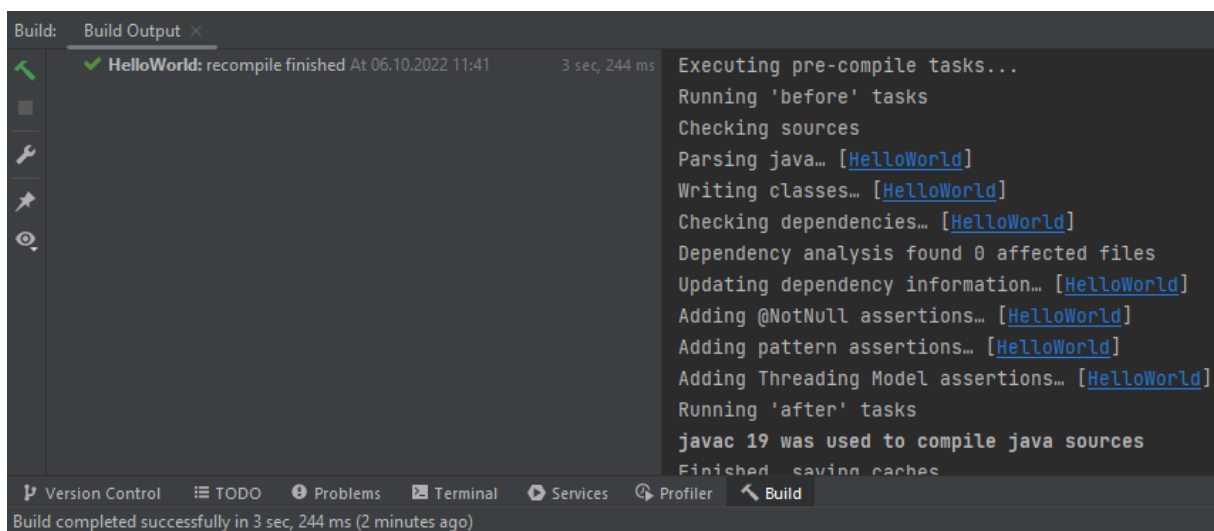
Okno podpowiedzi można aktywować za pomocą klawiszy: Ctrl + space.

Kompilacja i uruchomienie

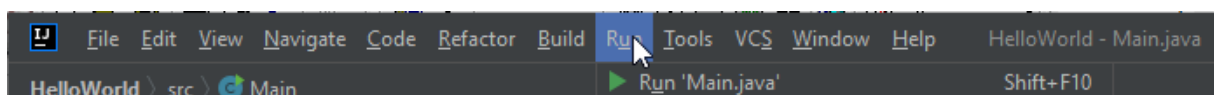
Program kompilujemy klikając w przycisk Build / Build Project



Gdy klikniemy na dolnej belce w ikonę Build pojawi się raport z kompilacji.



Program uruchamiamy klikając w ikonę Run na górnej belce

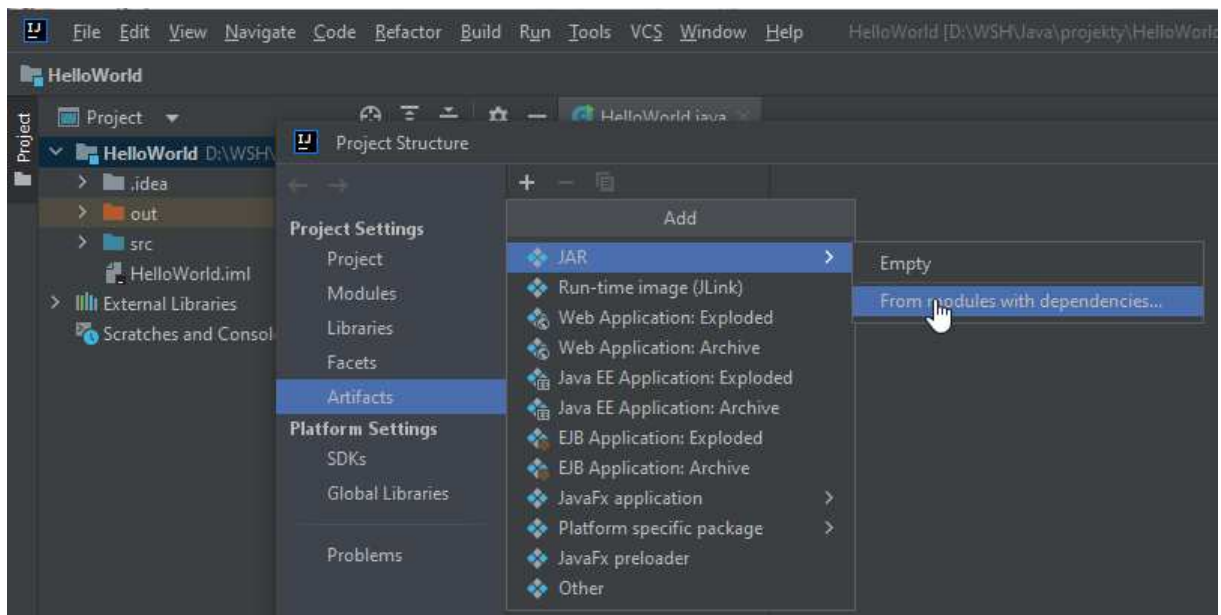


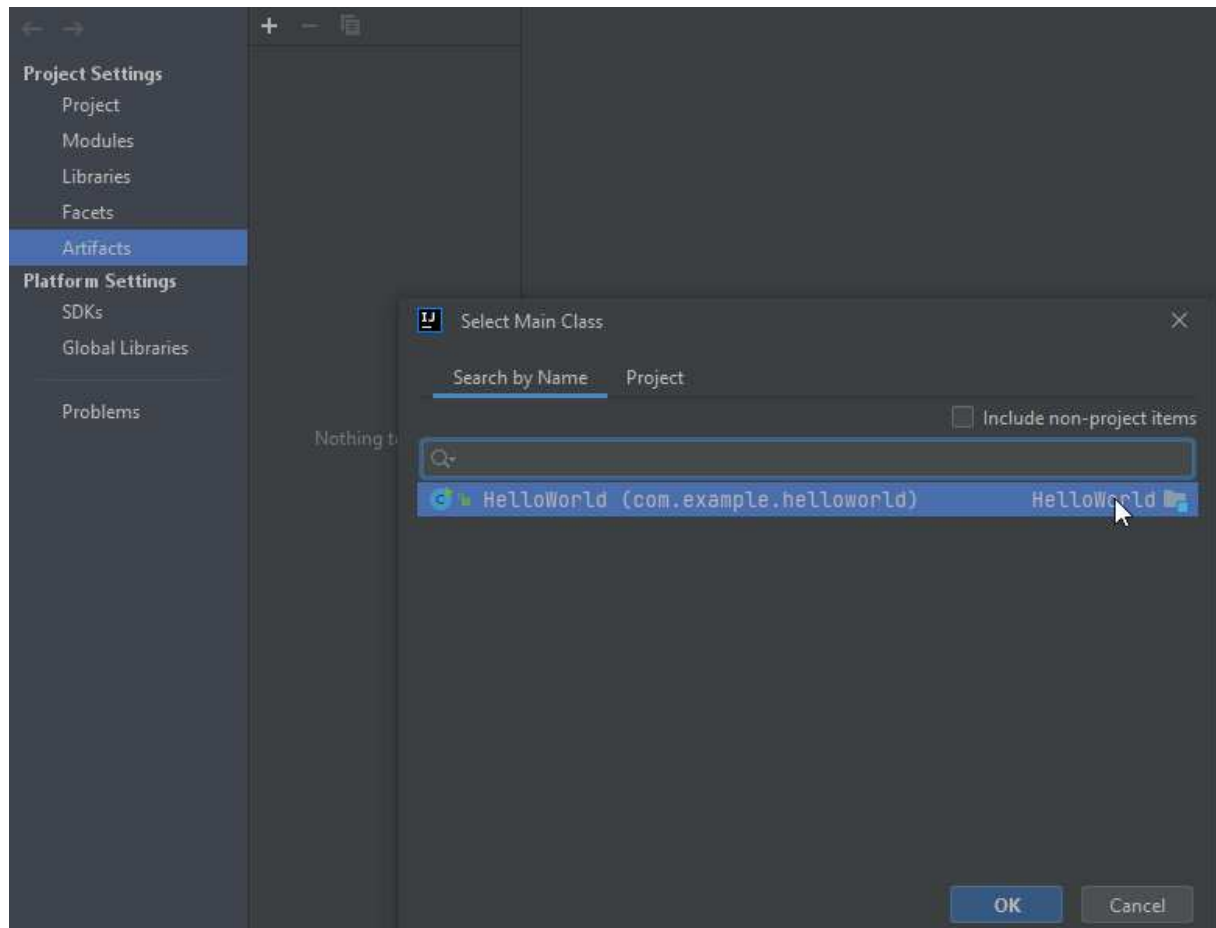
Wyniki działania programu zobaczymy w oknie wyników.



Podczas kolejnych uruchomień można się posługiwać się skrótem Shift + F10.

Tworzenie archiwum





2. Podstawy

2.1 Program HelloWorld – uruchomienie z konsoli

Napisz w Java program wypisujący napis „HelloWorld” na konsoli. Wykorzystaj Edytor tekstu SublimeText3 i okno interfejsu tekstowego cmd do kompilacji i uruchomienia programu. Użyj danych niżej poleceń do kompilacji i wykonania programu:

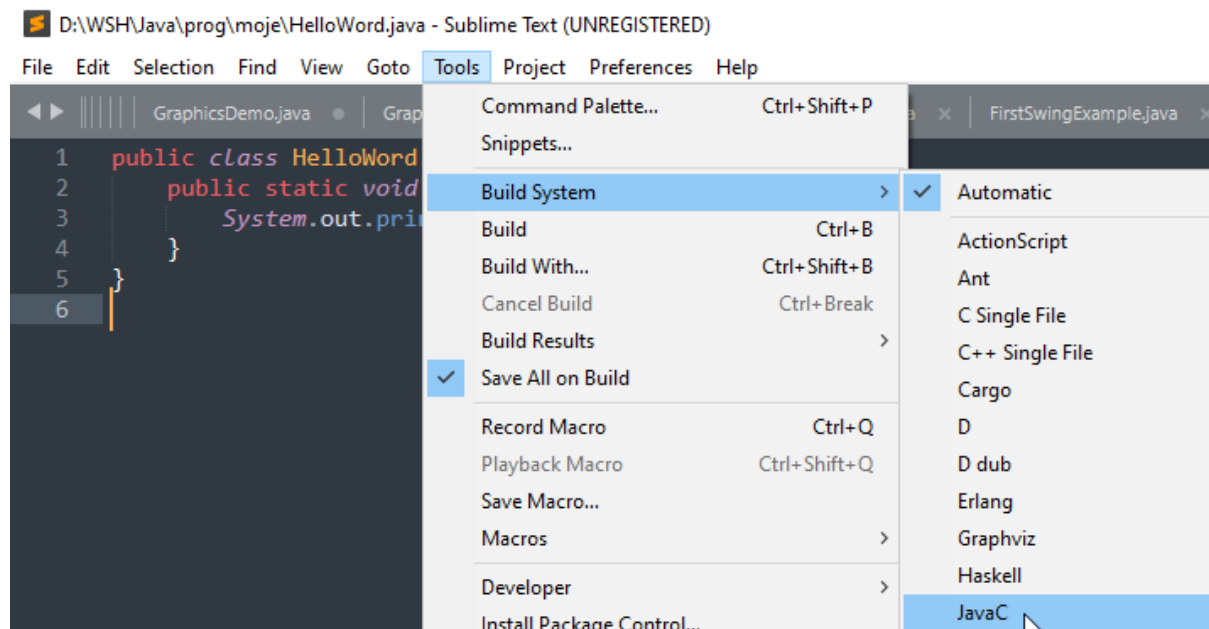
```
c:\>javac HelloWorld.java
```

```
c:\>java HelloWorld
```

2.2 Program HelloWorld – użycie edytora SublimeText3

Wykorzystaj edytor SublimeText3 do uruchomienia programu HelloWorld. W tym celu należy:

A) Ustawić język jako Java co pokazano poniżej.

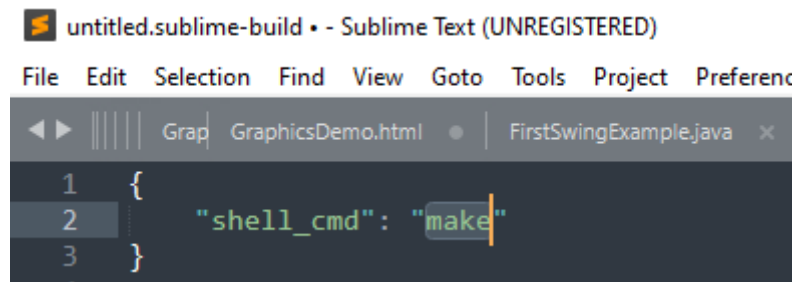


Rys. 2-1 Ustawienie języka jako Java

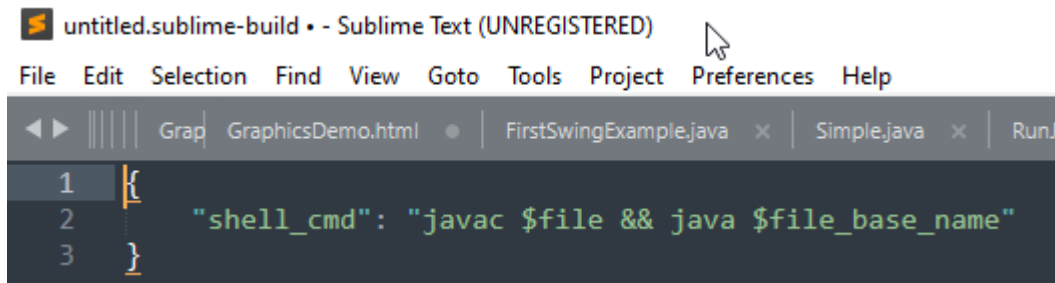
B) Skonfigurować edytor do kompilacji i wykonania programu jak poniżej.

Tools / BuildSystem / NewBuildSystem

Pojawi się okno jak poniżej.



W miejscu tekstu należy wpisać

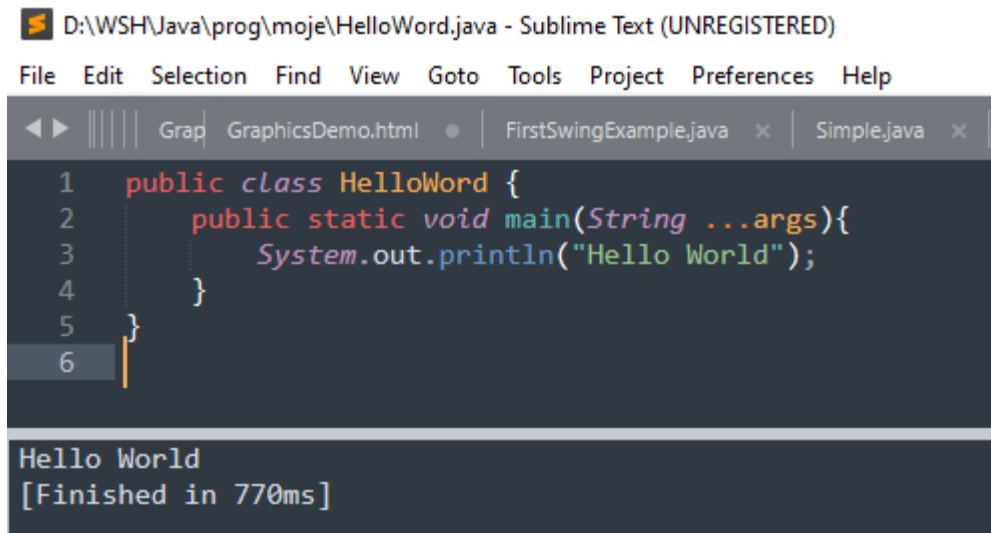


```

1 {
2     "shell_cmd": "javac $file && java $file_base_name"
3 }

```

Dalej zachowujemy plik jako: `RunJava.sublime-build` w domyślnej lokalizacji. Kompilacja i uruchomienie poprzez `Ctrl+B`.



```

1 public class HelloWord {
2     public static void main(String ...args){
3         System.out.println("Hello World");
4     }
5 }
6

```

Hello World
[Finished in 770ms]

Szczegóły konfiguracji środowiska w: <https://www.codejava.net/coding/how-to-compile-and-run-a-java-program-with-sublime-text-3>

2.3 Ciąg Fibonacciego

Napisz w Java program obliczający kolejne liczby ciągu Fibonacciego. Pierwszy wyraz ciągu jest równy 0, drugi jest równy 1, każdy następny jest sumą dwóch poprzednich.

Formalnie:

$$F_n := \begin{cases} 0 & \text{dla } n = 0, \\ 1 & \text{dla } n = 1, \\ F_{n-1} + F_{n-2} & \text{dla } n > 1. \end{cases}$$

Liczbę kroków ustal na stałe za pomocą zmiennej kroki.

Pierwsze dwadzieścia wyrazów ciągu Fibonacciego to:

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181

2.4 Obliczanie sumy elementów tablicy

Poniżej podano fragment programu zawierającego deklarację tablicy `int tab[]`. Uzupełnij podany tak aby drukował w kolejnych liniach indeks elementu tablicy, jego wartość a na końcu sumę elementów.

```
public class Array1 {

    public static void main(String[] args) {
        int size = 6;
        int tab[] = {1,2,3,4,5,6};
        ...
    }
}
```

2.5 Tabliczka mnożenia

Używając dwóch zagnieżdżonych pętli `for` napisz program który drukuje tabliczkę mnożenia taką jak poniżej. Element na przecięciu kolumny i ($i=1,2,\dots,10$) oraz wiersza j ($j=1,2,\dots,10$) jest iloczynem liczb i oraz j czyli wynosi $i * j$.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

2.6 Trójkąt

Napisz w języku Java program wypisujący na konsoli podany niżej trójkąt.

```
*
**
***
****
*****
*****
*****
*****
*****
*****
*****
```

Liczba wierszy niech będzie stała.

2.7 Piramida

Napisz w języku Java program wypisujący na konsoli podaną niżej piramidę.

```

      *
    * * *
  * * * * *
* * * * * * *
* * * * * * * * *
```

Liczba wierszy niech będzie stała.

2.8 Tworzenie obiektów – przykład Puppy

Poniżej mamy program `Puppy1.java` który posiada jeden atrybut `puppyAge`.

```

public class Puppy1 {
    int puppyAge;

    public Puppy1(String name) {
        // This constructor has one parameter, name.
        System.out.println("Name chosen is :" + name );
    }

    public void setAge( int age ) {
        puppyAge = age;
    }

    public int getAge( ) {
        System.out.println("Puppy's age is :" + puppyAge );
        return puppyAge;
    }

    public static void main(String []args) {
        /* Object creation */
        Puppy1 myPuppy = new Puppy1( "Zosia" );

        /* Call class method to set puppy's age */
        myPuppy.setAge( 2 );

        /* Call another class method to get puppy's age */
        myPuppy.getAge( );

        /* You can access instance variable as follows as well */
        System.out.println("Variable Value :" + myPuppy.puppyAge );

    }
}
```

Przykład 2-1 Program Puppy1.java

A) Napisz program `Puppy2.java` w którym dodaj atrybut `String name`. Typ `String` odpowiada napisowi, zmienne tego typu można łączyć za pomocą operatora `+`. Zmodyfikuj konstruktor aby nadać parametrowi wartość oraz dodaj metody:

`getName()` - wypisuje zmienną `name`

`setName(String newName)` - zmienia zmienną `name` na `newName`

`Print()` – wypisuje zmienne `name` i `puppyAge`.

B) Napisz program `Puppy3.java` w którym zmień konstruktor tak by nadać wartość zmiennym `puppyAge` i `name`. Utwórz dwa obiekty `myPuppy` i `myPuppy2` i wypisz ich atrybuty.

3. Tworzenie obiektów I

3.1 Wprowadzanie danych, klasa Scanner

Do wprowadzania danych z konsoli może służyć klasa `java.util.Scanner`. Obiekt tej klasy tworzy się przez konstruktor:

```
Scanner scan = new Scanner(System.in);
```

Klasa posiada następujące metody przydatne przy wprowadzaniu danych: `nextInt()`, `nextShort()`, `nextFloat()`, `nextLong()`, `nextBigDecimal()`, `nextBigInteger()`, `nextLong()`, `nextShort()`, `nextDouble()`, `nextByte()`, `nextFloat()`, `next()`.

Przykład użycia klasy `Scanner` podany jest poniżej.

```
import java.util.Scanner;

public class Fibonacci {

    public static void main(String[] args) {

        int count = 8, num1 = 0, num2 = 1;
        Scanner scan = new Scanner(System.in);

        System.out.print("Podaj liczbę wyrazów ciągu: ");
        count = scan.nextInt();

        for (int i = 1; i <= count; ++i)
        {
            System.out.print(num1+" ");
            int sumOfPrevTwo = num1 + num2;
            num1 = num2;
            num2 = sumOfPrevTwo;
        }
    }
}
```

Uruchom powyższy przykład.

3.2 Argumenty funkcji main

Program w Java można uruchomić z argumentami. Z wnętrza programu odczytujemy je jako tablicę łańcuchów `main(String[] args)`.

```
public class ArgsTest
{
    public static void main(String[] args)    {
        int index;
        for (index = 0; index < args.length; ++index) {
            System.out.println("args[" + index + "]: " + args[index]);
        }
    }
}
```

Liczbę elementów tablicy `args` możemy uzyskać przez `args.length`. Program uruchamiamy jak poniżej;

```
d:\WSH\Obiektowe\Java\prog\moje>java ArgsTest pierwszy drugi 3 4
args[0]: pierwszy
args[1]: drugi
args[2]: 3
args[3]: 4
```

Uruchom powyższy przykład.

3.3 Tworzenie obiektów - materiał teoretyczny

Przypomnij sobie materiał z rozdziału 6 – (Java – obiekty i klasy) dotyczący tworzenia obiektów podany w podręczniku “Programowanie obiektowe w języku JAVA”

3.4 Inicjalizacja obiektów – klasa Rectangle

W Java są trzy sposoby inicjalizacji obiektów (nadawania im wartości początkowych). Są to:

- Przez konstruktor
- Przez metodę
- Przez zmienną referencyjną

Poniżej podany jest program `Rectangle1.java`

```
class Rectangle1 {
    int length;
    int width;

    void calculateArea() {System.out.println(length*width); }

    public static void main(String args[]){
        Rectangle1 r1=new Rectangle1();
        Rectangle1 r2=new Rectangle1();
        r1.length = 11
        r1.width = 5;
        r2.length = 3
        r2.width = 15;
        r1.calculateArea();
        r2.calculateArea();
    }
}
```

Przykład 3-1 Program `Rectangle1.java`

Program ten tworzy dwa obiekty `r1` i `r2` które są inicjowane przez zmienną referencyjną.

- Uzupełnij program o metodę `void print(void)` która drukuje długości boków `length` i `width`.
- Napisz program `Rectangle2.java` gdzie obiekty `r1` i `r2` są inicjowane przez metodę `insert(int l, int w)`.
- Napisz program `Rectangle3.java` gdzie obiekty `r1` i `r2` są inicjowane przez konstruktor.

3.5 Tworzenie obiektów, przeciążanie konstruktora - klasa Employee

Poniżej podano fragment programu dotyczącego klasy Employee (Pracownik).

```
import java.io.*;

public class Employee {

    String name;    // Nazwisko
    int age;        // Wiek
    String department; // dział
    double salary;  // Pensja

    // This is the constructor 1 of the class Employee
    public Employee(String name) {
        // Tu należy uzupełnić
    }

    // This is the constructor 2 of the class Employee
    public Employee(String name, int age) {
        // Tu należy uzupełnić
    }

    // This is the constructor 3 of the class Employee
    public Employee(String name, int age, String department) {
        // Tu należy uzupełnić
    }

    // Podaj wiek
    public int getAge(void) {
        // Tu należy uzupełnić
    }

    // Podaj oddział
    public String getDepartment() {
        // Tu należy uzupełnić
    }

    // Ustaw oddział
    public void setDepartment(String department) {
        // Tu należy uzupełnić
    }

    // Ustaw pensje - salary
    public void setSalary(double empSalary) {
        // Tu należy uzupełnić
    }

    // Podaj pensje - salary
    public double getSalary(double empSalary) {
        // Tu należy uzupełnić
    }

    // Drukuj dane pracownika
    public void printEmployee() {
        // Tu należy uzupełnić
    }
}
```

```
public static void main(String args[]) {
    /* Create two objects using constructor */
    Employee empOne    = new Employee("James Smith");
    Employee empTwo     = new Employee("Mary Anne", 33);
    Employee empThree  = new Employee("Jan Kowalski", 44, "Testy");

    // Invoking methods for each object created
    empOne.setAge(26);
    empOne.setDepartment("Development");
    empOne.setSalary(1000);
    empOne.printEmployee();

    empTwo.setDepartment("Software");
    empTwo.setSalary(500);
    empTwo.printEmployee();

    empThree.salary = 1500;
    empThree. printEmployee();
}
}
```

Przykład 3-2 Program Employee.java

Program ilustruje przeciążanie konstruktora, który może mieć różną liczbę parametrów. Uzupełnij treść programu, skompiluj i wykonaj.

3.6 Klasa Employee – dwa pliki

Rozbij opracowany w poprzednim zadaniu program `Employee.java` na dwa pliki. W pliku `Employee.java` ma być zdefiniowana klasa `Employee` ale należy usunąć funkcję `main`. W pliku `EmployeeTest.java` należy umieścić definicję klasy `EmployeeTest` i funkcję `main` jak z poprzedniego zadania.

4. Tworzenie obiektów II

4.1 Klasa operacji na tablicy

Mamy program operujący na tablicy jak poniżej.

```
public class TestArray {

    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};

        System.out.println("Array demo");
        // Print all the array elements
        for (int i = 0; i < myList.length; i++) {
            System.out.println(myList[i] + " ");
        }

        // Summing all elements
        double total = 0;
        for (int i = 0; i < myList.length; i++) {
            total += myList[i];
        }
        System.out.println("Total is " + total);

        // Finding the largest element
        double max = myList[0];
        for (int i = 1; i < myList.length; i++) {
            if (myList[i] > max) max = myList[i];
        }
        System.out.println("Max is " + max);
    }
}
```

Przykład 4-1 Program `TestArray.java`

Program ten drukuje tablicę `myList`, oblicza sumę jej elementów i znajduje największy element. Napisz program `ArrayTest5.java` który robi to samo ale drukowanie tablicy, sumowanie elementów i znajdowanie największego elementu zawarte jest w oddzielnych metodach. Szkic programu dany jest poniżej.

```
public class TestArray5 {

    double findMax(double array[]) {
        // uzupełnić
    }

    void printArray(double array[]) {
        // uzupełnić
    }

    double sumAll(double array[]) {
        // uzupełnić
    }

    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};
        double result;
        TestArray5 arr = new TestArray5();
        // Print all the array elements
    }
}
```

```

        System.out.println("Array demo");
        arr.printArray(myList);

        // Summing all elements
        result = arr.sumAll(myList);
        System.out.println("Total is " + result);

        // Finding the largest element
        result = arr.findMax(myList);
        System.out.println("Max is " + result);
    }
}

```

Przykład 4-2 Szkielet programu TestArray5.java

Program ma dawać wyniki jak poniżej:

```

Array demo
1.9
2.9
3.4
3.5
Total is 11.7
Max is 3.5

```

4.2 Klasa operacji na tablicy i jej test

W poprzednim przykładzie metody findMax, printArray i sumAll były elementami klasy TestArray5. Dokonaj modyfikacji programu aby utworzyć oddzielną klasę TestArray zawierającą powyższe metody. Z klasy tej ma korzystać program testujący TestArray7.java jak poniżej.

```

class TestArray {

    double findMax(double array[]) {
        // uzupełnić
    }

    void printArray(double array[]) {
        // uzupełnić
    }

    double sumAll(double array[]) {
        // uzupełnić
    }
}

public class TestArray7 {
    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};
        double result;
        TestArray arr = new TestArray();

        // Print all the array elements
        System.out.println("Array demo");
        arr.printArray(myList);

        // Summing all elements
        result = arr.sumAll(myList);
        System.out.println("Total is " + result);

        // Finding the largest element
    }
}

```

```

        result = arr.findMax(myList);
        System.out.println("Max is " + result);
    }
}

```

Przykład 4-3 Szkic programu `TestArray7.java`

4.3 Klasa operacji na tablicy - dwa pliki

W poprzednim zadaniu klasy `TestArray` i `TestArray7` były w jednym pliku. Obecne zadanie polega na tym aby przenieść je do osobnych plików. Przenieś definicje klasy `TestArray` do oddzielnego pliku `TestArray.java` i skompiluj go do pliku `TestArray.class`.

Wykonaj dany poniżej mamy program do testowania klasy `TestArray`.

```

// Klasa do testowania klasy TestArray zawartej
// w pliku TestArray.java
public class TestArrayTest {

    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};
        double result;
        TestArray arr = new TestArray();

        // Print all the array elements
        System.out.println("Array demo");
        arr.printArray(myList);

        // Summing all elements
        result = arr.sumAll(myList);
        System.out.println("Total is " + result);

        // Finding the largest element
        result = arr.findMax(myList);
        System.out.println("Max is " + result);
    }
}

```

Przykład 4-4 Testowanie klasy `TestArrayTest.java`

4.4 Rozszerzona instrukcja for

W zadaniu 4.2 w metodach `findMax`, `printArray` i `sumAll` występuje instrukcja `for` np. w formie jak poniżej.

```

for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
}

```

Napisz program `TestArray8.java` w którym zastąp tę instrukcję rozszerzoną instrukcją `for` np. jak poniżej.

```

for (double x : array) {
    System.out.println(x + " ");
}

```


4.5 Klasa String – konkatencja

Uzupełnij poniższy przykład. String str3 ma być konkatencją str1 i str2. Zastosuj metodę concat i przeciążony operator dodawania.

```
import java.lang.String;

public class StringMethodsConcat {

    public static void main(String[] args) {

        String str1 = "Software";
        String str2 = "Testing";
        String str3;
        // str3 = ...   dodawanie
        System.out.println(str3);
        // str3 = ...   konkatencja
        System.out.println(str3);
    }
}
```

Przykład 4-5 Konkatencja łańcuchów

4.6 Klasa String – konwersja do tablicy znaków

Typ String w Java to nie jest tablica znaków ale klasa. String do tablicy konwertuje się za pomocą metody toCharArray(). Uzupełnij poniższy program w którym należy przekształcić String str do tablicy chars.

```
import java.lang.String;

public class StringMethodsToChar {

    public static void main(String[] args) {

        String str = "Akademia Biznesu";
        char[] chars;
        // chars = ...
        System.out.println(chars);
        for (int i= 0; i< chars.length; i++) {
            System.out.println(chars[i]);
        }
    }
}
```

Przykład 4-6 Konkatencja łańcuchów

4.7 Klasa String – szukanie podłańcuchów

Metoda str.indexOf(strToFind, Index) zwraca indeks w łańcuchu str od którego zaczyna się podłańcuch strToFind. Szukanie rozpoczyna się od indeksu Index. Uzupełnij poniższy przykład tak by znaleźć liczbę wystąpień łańcucha Java w łańcuchu str.

```
import java.lang.String;

public class StringSubstring {

    public static void main(String[] args) {
        String str = "StringJavaAndJavaStringMethodsJava";
        String strToFind = "Java";
        int count = 0, Index = 0;

        System.out.println("Liczba wystąpień jest: " + count);
    }
}
```

Przykład 4-7 Szukanie podłańcuchów

4.8 Usuwanie elementów tablicy

Poniżej podano program dodawania elementów do tablicy.

```
/ Java Program to Insert an element
// at a specific position in an Array
import java.io.*;
import java.lang.*;
import java.util.*;

class Tablice1 {
    // Function to insert x in arr at position pos
    public static int[] insertX(int n, int arr[],int x, int pos)
    {
        int i;
        // create a new array of size n+1
        int newarr[] = new int[n + 1];
        for (i = 0; i < n + 1; i++) {
            if (i < pos - 1)
                newarr[i] = arr[i];
            else if (i == pos - 1)
                newarr[i] = x;
            else
                newarr[i] = arr[i - 1];
        }
        return newarr;
    }

    //Kod do testowania
    public static void main(String[] args)
    {
        int n = 10;
        int i;
        // Początkowa tablica o wymiarze 10
        int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
        System.out.println("Initial Array:\n"
            + Arrays.toString(arr));

        int x = 50;
        int pos = 5;
        arr = insertX(n, arr, x, pos);
        System.out.println("\nArray with " + x
            + " inserted at position "
            + pos + ":\n"
            + Arrays.toString(arr));
    }
}
```

Na jego podstawie napisz program demonstrujący usuwania elementów tablicy wykorzystujący funkcję

```
public static int[] removeTheElement(int[] arr, int index)
```

5. Obiekty i metody

5.1 Operacji na tablicy sortowanie i odwracanie

Opierając się na wcześniejszych zadaniach rozszerz klasę `Array` o metodę sortowania elementów tablicy i odwracania jej kolejności.

```
class Array {

    double findMax(double array[]) {
        // Zwraca maksimum z tablicy array
    }

    void printArray(double array[]) {
        // Wypisuje zawartosci tablicy array
    }

    double sumAll(double array[]) {
        // Zwraca sume elementów tablicy array
    }

    void sortArray(double arr[]) {
        // Sortuje tablicę arr w porządku rosnącym
    }

    void odwroc(double array[]) {
        // Zamienia umieszcza elementy tablicy array w odwrotnej
        // kolejności, np. {1,2,4} zamienia na {4,2,1}
    }
}

public class TestArray10 {

    public static void main(String[] args) {
        double[] myList = {4.9, 1.9, 3.4, 3.2};
        double result;
        Array arr = new Array();

        // Print all the array elements
        System.out.println("Array demo");
        arr.printArray(myList);

        // Summing all elements
        result = arr.sumAll(myList);
        System.out.println("Total is " + result);

        // Finding the largest element
        result = arr.findMax(myList);
        System.out.println("Max is " + result);

        // Sortowanie tablicy
        System.out.println("Sorted array ");
        arr.sortArray(myList);
        arr.printArray(myList);

        // Odwracanie kolejności elementów
        System.out.println("Reversed array ");
        arr.odwroc(myList);
        arr.printArray(myList);
    }
}
```

```
}  
}
```

Przykład 5-1 Szkielet programu `TestArray10`

5.2 Operacji na tablicy – szukanie elementu

Opierając się na wcześniejszych zadaniach rozszerz klasę `Array` o metodę przeszukiwania tablicy w celu znalezienia określonego jako parametr elementu.

W tym celu napisz funkcję `szukaj`, która otrzymuje tablicę `tab` oraz wartość poszukiwaną `key` i zwraca informację czy liczba wystąpiła w podanej tablicy `tab`.

```
int szukaj(double tab[], double key)
```

Do szukania można wykorzystać algorytm szukania binarnego znany ze „Wstępu do programowania”.

5.3 Operacje sortowanie i przeszukiwanie tablicy – użycie funkcji bibliotecznych

Na podstawie wcześniejszego zadania napisz program `TestArray11.java` w którym do sortowania tablicy w metodzie `sortArray` użyj metody bibliotecznej `sortArray`.

```
void sortArray(double arr[])
```

Dodaj do klasy `Array` metodę `int Szukaj(double arr[], double key)` która w tablicy `arr` wyszukuje element `key` i zwraca jego indeks. Użyj funkcji bibliotecznej

```
public static int binarySearch(double[] a, double key)
```

6. Przeciążanie metod

6.1 Wypisywanie danych różnego typu

Dany jest poniższy program demonstrujący przeciążanie metod.

```
// Przykład przeciążania metod
class Pisanie {

    public void wypisz(int liczba) {
        System.out.println("Liczba całkowita: " + liczba);
    }

    public void wypisz(int pierwsza, int druga) {
        System.out.println("Pierwsza liczba: " + pierwsza +
            ", druga liczba: " + druga);
    }

    public void wypisz(double liczba) {
        System.out.println("Liczba rzeczywista: " + liczba);
    }

    public void wypisz(String tekst) {
        System.out.println("Tekst: " + tekst);
    }

    public void wypisz(int[] tablica) {
        // Uzupełnij - wypisz tablicę int
    }

    public void wypisz(double[] tablica) {
        // Uzupełnij - wypisz tablicę double
    }

}

public class PrzeladowanieWypisz2 {
    public static void main(String[] args) {
        int[] tabInt = {2, 40, 500} ;
        double[] tabDouble = {3.14, 5.4, 6.6, 12.3 };
        Pisanie pisz = new Pisanie();
        pisz.wypisz(10);
        pisz.wypisz(7, 128);
        pisz.wypisz(3.14);
        pisz.wypisz("Witaj!");
        pisz.wypisz(tabInt);
        pisz.wypisz(tabDouble);
    }
}
```

Przykład 6-1 Program PrzeladowanieWypisz2.java

Uzupełnij funkcje:

- `wypisz(int[] tablica)` – pisanie tablicy int
- `wypisz(double[] tablica)` – pisanie tablicy double

6.2 Przeciążanie metod – klasa Array

Poniżej dany jest program używający klasy Array.

```
class Array {

    double findMax(double array[]) {
        double max = array[0];
        for (int i = 1; i < array.length; i++) {
            if (array[i] > max) max = array[i];
        }
        return max;
    }

    void printArray(double array[]) {
        for (int i = 0; i < array.length; i++) {
            System.out.println(array[i] + " ");
        }
    }

    double sumAll(double array[]) {
        double total = 0;
        for (int i = 0; i < array.length; i++) {
            total = total + array[i];
        }
        // System.out.println("Total is " + total);
        return total;
    }
}

public class TestArray7 {
    int a;

    public static void main(String[] args) {
        double[] myList = {1.9, 2.9, 3.4, 3.5};
        double result;
        Array arr = new Array();

        // Print all the array elements
        System.out.println("Array demo");
        arr.printArray(myList);

        // Summing all elements
        result = arr.sumAll(myList);
        System.out.println("Total is " + result);

        // Finding the largest element
        result = arr.findMax(myList);
        System.out.println("Max is " + result);
    }
}
```

Przykład 6-2 Program TestArray7.java

Rozszerz program o wszystkie operacje ale wykonywane na argumentach int.

- void printArray(int array[])
- int findMax(int array[])
- int sumAll(int array[])

Przeprowadź testy wszystkich tych metod dla wersji z tablicą `double` i `int`. Zauważ że metody te mają takie same nazwy ale inne typy argumentów.

```
public class TestArray11 {  
  
    public static void main(String[] args) {  
        double[] myList = {4.9, 1.9, 3.4, 3.2};  
        int[] myList2    = {4,12,1,7,6,9,13};  
  
        // FOR myList  
        // Print all the array elements  
        // Summing all elements  
        // Finding the largest element  
  
        // FOR myList2  
        // Print all the array elements  
        // Summing all elements  
        // Finding the largest element  
    }  
}
```

Przykład 6-3 Program TestArray12.java

6.3 Przeciążanie konstruktora - program TestArray12.java

Program `TestArray7.java` zawierał klasę `Array` która wykonywała różne operacje na tablicy, ale referencja do tablicy była argumentem metod i tablica była zewnętrzną względem klasy `Array`. Napisz program `TestArray12.java` w którym tablica `arr` będzie zmienną klasową klasy `Array`.

```
// Klasa Array zawiera tablicę jako zmienna klasową
class Array {
    double[] arr; // Referencja do tablic
    int size;      // Wielkość tablicy

    Array(int n) {
        arr = new double[n];
        size = n;
    }

    Array(double[] tab) {
        // Wywołać konstruktor tworzący tablice this(wymiar_tablicy)
        // Skopiować tablice tab do arr
    }

    void printArray() {
        for (int i = 0; i < size; i++) {
            System.out.println(arr[i] + " ");
        }
    }

    double findMax() {
        // Funkcja zwraca maksymalny element tablicy arr
    }

    double sumAll() {
        // Uzupełnić sumowanie elementów tablicy arr
    }
}

public class TestArray12 {
    int a;
    public static void main(String[] args) {
        double[] myList = {4.9, 1.9, 3.4, 3.2};
        Array tab = new Array(myList);
        double result;

        // Print all the array elements
        System.out.println("Array demo");
        tab.printArray();

        // Summing all elements
        result = tab.sumAll();
        System.out.println("Total is " + result);

        // Finding the largest element
        result = tab.findMax();
        System.out.println("Max is " + result);
    }
}
```

Przykład 6-4 Program TestArray12.java

7. Tworzenie obiektów III

7.1 Klasa Osoba

Utwórz poniżej daną klasę Osoba która będzie użyta dalej w implementacji bazy danych osób.

```
class Osoba {
    String imie;      // Imie
    String nazwisko;  // Nazwisko
    int    pesel;     // Pesel

    Osoba(String imie, String nazw, int pesel) {
        // Uzupełnić konstruktor
    }

    Osoba() {
        // Uzupełnić konstruktor
    }

    void wyswietl() {
        // Wyswietl imie nazwisko i pesel
    }

    void wczytaj() {
        // Wczytaj imie nazwisko i pesel
    }

    String dajImie() {
        // Zwroc imie
    }

    String dajNazwisko() {
        // Zwroc nazwisko
    }

    int dajPesel() {
        // Zwroc pesel
    }
};
```

```

class OsobaTest1 {
    public static void main(String[] args) {    // Program testowy
        int pesel;
        int wybor;
        String nazwisko;
        String imie;

        // Tworzenie obiektów -----
        Osoba osoba1 = new Osoba();
        Osoba osoba2 = new Osoba("Jan", "Braun", 1);
        osoba2.wyswietl();
        osoba1.wyswietl();

        // Wczytywanie danych obiektu
        osoba1.wczytaj();
        osoba1.wyswietl();

        // Testowanie metod -----
        imie = osoba2.dajImie();
        nazwisko = osoba2.dajNazwisko();
        pesel = osoba2.dajPesel();
        System.out.println(imie + " " + nazwisko + " " + pesel);
    }
}

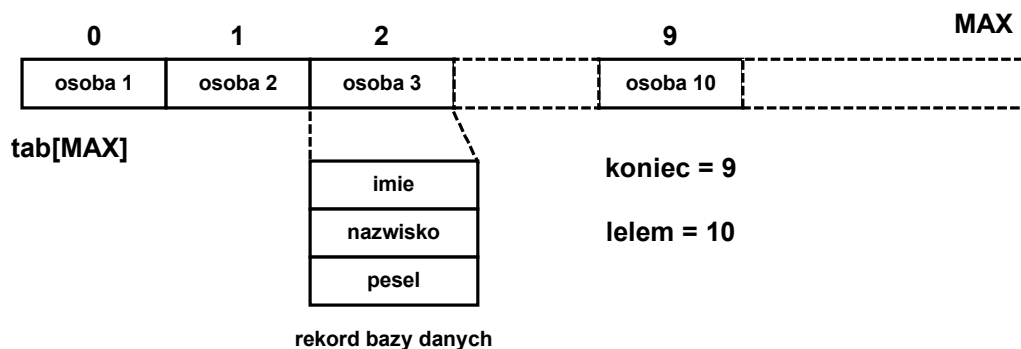
```

Przykład 7-1 Program OsobaTest1

Uzupełnij konstruktory i metody

7.2 Tablica danych osobowych

Utwórz program zarządzający bazą danych osobowych, zawierającą obiekty typu `Osoba` z poprzedniego zadania. Baza danych ma postać tablicy zawierającej obiekty typu `Osoba`.



Rys. 7-1 Baza danych osobowych jako tablica

```
// Klasa osoba jako tablica
import java.util.Scanner;

class Osoba {
    // Tak jak w poprzednim zadaniu
};

public class OsobaTest2 {
    public static void main(String[] args) { // Program testowy
        int wybor,i;
        final int MaxTab = 3;
        String nazwisko;
        String imie;

        // Utworzenie tablicy osób
        Osoba[] osobaTab = new Osoba[MaxTab];

        // Wczytywanie osób -----
        for(i=0;i<MaxTab;i++) {
            // Utworz osobaTab[i] i wprowadz dane
        }

        // Wypisywanie osób -----
        for(i=0;i<MaxTab;i++) {
            // Wypisz osobaTab[i]
        }
    }
}
```

Przykład 7-2 Program OsobaTest2 – baza danych oparta na tablicy

Zaimplementuj wprowadzanie danych i ich wypisywanie. Przykład działania programu dany jest poniżej.

```
Imie: Jan
Nazwisko: Kowal
Pesel: 1
Imie: Adam
Nazwisko: Mika
Pesel: 2
Imie: Zenon
Nazwisko: Likas
Pesel: 3
Jan Kowal 1
Adam Mika 2
Zenon Likas 3
```

7.3 Baza danych osobowych oparta na tablicy

Utwórz program zarządzający bazą danych osobowych, zawierającą obiekty typu `Osoba` z poprzedniego zadania. Baza danych ma postać tablicy zawierającej obiekty typu `Osoba`.

```
class Osoba {
    String imie;      // Imie
    String nazwisko;  // Nazwisko
    int    pesel;     // Pesel

    Osoba(String imie, String nazw, int pesel) {
        // Uzupełnić konstruktor
    }

    Osoba() {
        // Uzupełnić konstruktor
    }

    void wyswietl() {
        // Wyswietl imie nazwisko pesel
    }

    void wczytaj() {
        // Wczytaj imie nazwisko pesel
    }
};

class Baza {
    int lelem;        // aktualna liczba elementow tablicy
    int koniec;       // indeks ostatniego elementu
    int tabSize;
    Osoba[] osobaTab;

    public Baza(int size) { // Konstruktor
        System.out.println("Konstruktor baza, size: " + size);
        tabSize = size;
        osobaTab = new Osoba[size];
        lelem = 0;
        koniec = 0;
    }

    void dodaj() {
        // Dodaj nową osobę do bazy
        // Uzupełnić
    }

    int szukaj(int pesel) {
        // Szukaj osoby o peselu pesel
        // Uzupełnić
    }

    void lista () {
        // Drukuj liste osob -----
        // Uzupełnić
    }

    int usun(int pesel) {
        // Usuamy osobe o peselu pesel
    }
}
```

```

class BazaTest {

    public static void main(String[] args) {    // Program testowy
        int wybor,i,pes;
        static final int MaxTab = 10;
        Baza myBaza = new Baza(MaxTab);
        Scanner scan = new Scanner(System.in);

        // Menu -----
        do {
            System.out.println("1 dodaj, 2 lista, 3 szukaj, 4 usun, 5 koniec");
            wybor = scan.nextInt();
            System.out.println("wybor = " + wybor);
            switch(wybor) {
                case 1: System.out.println("dodaj");
                        myBaza.dodaj();
                        break;
                case 2: System.out.println("lista");
                        myBaza.lista();
                        break;
                case 3: System.out.println("szukaj");
                        System.out.println("Podaj pesel: ");
                        pes = scan.nextInt();
                        myBaza.szukaj(pes);
                        break;
                case 4: System.out.println("usun");    break;
            }

            } while(wybor != 5);
        System.out.println("Koniec");
    }
}

```

Przykład 7-3 Szkielet programu obsługi bazy osób BazaTest.java

7.4 Klasa Baza danych, zabezpieczenia

W poprzednim zadaniu wprowadź modyfikację polegającą na zabezpieczeniu, aby nie dało się wprowadzić do bazy danych więcej niż jednej osoby o danym peselu. W tym celu należy zmodyfikować funkcję `dodaj()` w której należy wykorzystać funkcję `int szukaj(int pesel)`.

7.5 Klasa Baza danych, optymalizacja przeszukiwania

W przypadku gdyby omawiana wcześniej baza danych oparta na tablicy miała bardzo dużo elementów (powiedzmy milion), przeszukiwanie jej trwałoby bardzo długo. Do poprzedniego zadania wprowadź modyfikację polegającą na optymalizacji przeszukiwania. W tym celu należy:

1. Posortować elementy tablicy według klucza którym jest pesel. Uzupełnij klasę `Baza` o funkcję `sortuj()`, która sortuje tablicę za pomocą algorytmu sortowania bąbelkowego.
2. Zmodyfikować funkcję `dodaj()` tak aby wstawiać nowy obiekt `Osoba` na właściwym miejscu w tablicy, tak aby zachować jej posortowanie.

8. Hermetyzacja

8.1 Klasa Baza danych, hermetyzacja klasy Osoba

Dokonaj modyfikacji aplikacji bazy danych osobowych hermetyzując klasę Osoba.

Wprowadź zabezpieczenia by uniemożliwić błędnego wpisania danych Imię, Nazwisko, pesel (np. liczba > 0, liczba 11 cyfrowa). Pesel nie może się powtarzać.

```
class Osoba {

    private String imie;      // Imie
    private String nazwisko;  // Nazwisko
    private int  pesel;       // Pesel

    Osoba(String imie, String nazw, int pesel) {
        // Uzupełnić
    }

    Osoba() {
        // Uzupełnić
    }

    public void setNazwisko(String name) {
        // Uzupełnić
    }

    public void setImie(String imie) {
        // Uzupełnić
    }

    public void setPesel(int pesel) {
        // Uzupełnić
    }

    public String getNazwisko() {
        // Uzupełnić
    }

    public String getImie() {
        // Uzupełnić
    }

    public int getPesel() {
        // Uzupełnić
    }

    public String toString() {
        return this.imie + " " + this.nazwisko + " " + this.pesel ;
    }

};
```

Niech klasa Baza i BazaTest używają tylko zabezpieczonych metod klasy Osoba.

9. Dziedziczenie

9.1 Klasa Osoba i klasy potomne

W poprzednich zadaniach występowała klasa Osoba jak poniżej.

```
class Osoba {
    private String imie;        // Imie
    private String nazwisko;    // Nazwisko
    private int pesel;          // Pesel

    Osoba(String imie, String nazw, int pesel) { }
    Osoba() { }
    String toString() { }
    ...
}
```

Na jej podstawie utwórz klasy:

```
Student extends Osoba {
    int indeks; // Numer indeksu
}

StudentInf extends Student {
    String jezykProg; // Jaki zna język programowania
}

Pracownik extends Osoba {
    String zakladPracy; // Gdzie pracuje
}
```

Dla każdej klasy utwórz konstruktor i metody wprowadzania i odczytu danych.

Następnie utwórz tablicę `Osoba[] pokoj = new Osoba[MAX]` i napisz funkcje wprowadzania danych do tablicy i wydruku jej zawartości. Zauważ że w tablicy mogą być obiekty różnego typu.

9.2 Baza danych osobowych, polimorfizm

Rozszerz wcześniejszy przykład 7.3 dotyczący bazy danych osobowych o możliwość wprowadzanie obiektów klasy *Osoba*, *Student*, *Pracownik* z poprzedniego przykładu.

```
class Osoba {
    // Uzupełnic
}

Student extends Osoba {
    // Uzupełnic
}

Pracownik extends Osoba {
    // Uzupełnic
}

class Baza {
    int lelem;    // aktualna liczba elementow tablicy
    int koniec;   // indeks ostatniego elementu
    int tabSize;
    Osoba[] osobaTab;

    public Baza(int size) { // Konstruktor
        // Uzupełnić
    }

    void dodaj(Osoba osb) {
        // Dodaj obiekt osb
        // Uzupełnić
    }

    int szukaj(int pesel) {
        // Szukaj osoby o peselu pesel
        // Uzupełnić
    }

    void lista () {
        // Drukuj liste osob -----
        // Uzupełnić
    }

    int usun(int pesel) {
        // Usuamy osobe o peselu pesel
    }
}

class BazaTestVirt {
    static final int MaxTab = 10;

    public static void main(String[] args) { // Program testowy
        int wybor,i,pes;
        Osoba osb;
        Baza myBaza = new Baza(MaxTab);
        Scanner scan = new Scanner(System.in);
    }
}
```

```

// Menu -----
do {
    System.out.println("1 dodaj, 2 lista, 3 szukaj, 4 usun, 5 koniec");
    wybor = scan.nextInt();

    System.out.println("wybor = " + wybor);
    switch(wybor) {
        case 1: System.out.println("dodaj");
            System.out.println("1 Student, 2 Pracownik, 3 Osoba");
            wybor = scan.nextInt();
            System.out.println("Typ wybor: " + wybor);
            switch(wybor) {
                case 1: osb = new Student();
                    osb.wczytaj();
                    break;
                case 2: osb = new Pracownik();
                    osb.wczytaj();
                    break;
                case 3: osb = new Osoba();
                    osb.wczytaj();
                    break;
                default: System.out.println("zly wybor");
                    osb = new Osoba();
            }
            myBaza.dodaj(osb); // Dodanie obiektu
            break;

        case 2: System.out.println("lista");
            myBaza.lista();
            break;
        case 3: System.out.println("szukaj");
            System.out.println("Podaj pesel: ");
            pes = scan.nextInt();
            myBaza.szukaj(pes);
            break;
        case 4: System.out.println("usun");
            System.out.println("Podaj pesel: ");
            pes = scan.nextInt();
            myBaza.usun(pes);
            break;
        default: System.out.println("Zly wybor");
    }
} while(wybor != 5);
System.out.println("Koniec");
}
}

```

Przykład 9-1 Program

Zauważ że w funkcji `myBaza.dodaj(osb)` występuje zmienna `osb` która jest referencja do klasy `Osoba` ale może wskazywać na obiekty typu `Osoba`, `Student` lub `Pracownik`. Jest to więc funkcja wirtualna. Podobnie tablica `myBaza.osobaTab[]` zawiera referencje do obiektów różnego typu `Osoba`, `Student`, `Pracownik`. Wprowadzanie danych ma być jak poniżej.

```
1 dodaj, 2 lista, 3 szukaj, 4 usun, 5 koniec
wybor = 1
dodaj
1 Student, 2 Pracownik, 3 Osoba
Typ wybor: 2
Imie: Wanda
Nazwisko: Lis
Pesel: 2
Zaklad: PWR
...
```

10. Interfejsy

10.1 Sortowanie tablicy – przykład interfejsu

Poniżej dany jest program implementujący interfejs Comparable.

```
// Ilustracja działania interfejsu
// Dostarczamy funkcji compareTo potrzebnej do
// sortowania

import java.util.*;

class Student implements Comparable<Student>
{
    private String name;
    private int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "{" + "name='" + name + '\'' +
            ", age=" + age + '}';
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    @Override
    public int compareTo(Student o) {
        if (this.age != o.getAge()) {
            return this.age - o.getAge();
        } else
            return 0;
    }
}

class StudentTest
{
    public static void main(String[] args) {
        Student[] students = { new Student("John", 15), new Student("Sam", 20),
                                new Student("Dan", 20), new Student("Joe", 10) };
        Arrays.sort(students);
        System.out.println(Arrays.toString(students));
    }
}
```

Przykład 10-1 Program StudentTest

Uruchom ten program i zaobserwuj jak zaimplementowano funkcję compareTo()

10.2 Baza danych – sortowanie tablicy z użyciem interfejsu Comparable

Poniżej dany jest szkic klasy *Osoba* implementującej interfejs *Comparable*.

```
import java.util.Scanner;
import java.util.*;

class Osoba implements Comparable<Osoba>{

    String imie;        // Imie
    String nazwisko;    // Nazwisko
    int  pesel;         // Pesel

    Osoba(String imie, String nazw, int pesel) {
        this.imie = imie;
        this.nazwisko = nazw;
        this.pesel = pesel;
    }

    Osoba() {
        this.imie = "";
        this.nazwisko = "";
        this.pesel = 0;
    }

    void wyswietl() {
        System.out.println(imie + "  " + nazwisko + "  " + pesel);
    }

    void wczytaj() {
        // Zaimplementuj
    }

    String dajImie() {
        return this.imie;
    }

    String dajNazwisko() {
        return this.nazwisko;
    }

    int dajPesel() {
        return this.pesel;
    }

    @Override
    public int compareTo(Osoba os) {
        //
    }
}
```

Zaimplementuj metodę `compareTo()` i przetestuj ją pod kątem sortowania.

11. Struktury danych

11.1 Testowanie struktury Stack

Poniżej dany jest przykład wykorzystania struktury Stack.

```
import java.util.*;
public class StackDemo {

    static void showpush(Stack st, int a) {
        st.push(new Integer(a));
        System.out.println("push(" + a + ")");
        System.out.println("stack: " + st);
    }

    static void showpop(Stack st) {
        System.out.print("pop -> ");
        Integer a = (Integer) st.pop();
        System.out.println(a);
        System.out.println("stack: " + st);
    }

    public static void main(String args[]) {
        Stack st = new Stack();
        System.out.println("stack: " + st);
        showpush(st, 42);
        showpush(st, 66);
        showpush(st, 99);
        showpop(st);
        showpop(st);
        showpop(st);
        try {
            showpop(st);
        } catch (EmptyStackException e) {
            System.out.println("empty stack");
        }
    }
}
```

Wykorzystaj przykład do budowy programu w którym na stosie pamiętane są dane osobowe z wcześniejszych przykładów.

```
// Demonstracja bazy danych Osoba -----
// Wykorzystuje kolekcję Stack

import java.util.*;
import java.util.Scanner;

class Osoba {
    // Jak w poprzednich przykładach
};
```

```

class Baza {
    final int MAX = 50;  // Początkowy rozmiar tablicy
    Stack <Osoba> tab;

    Baza() // konstruktor
    {
        tab = new Stack();
    };

    // Dodanie osoby os do bazy -----
    int dodaj(Osoba os) {
        // ...
    }

    // Wyświetlenie bazy -----
    void wyswietl() {
        //...
    }

    void usun() {
        // ...
    }
}

public class BazaOsobaStackDemo {

    public static void main(String args[]) {
        System.out.println("Start");
        Osoba os1 = new Osoba("Jan", "Kowal", 1);
        Osoba os2 = new Osoba("Wanda", "Maj", 2);
        Osoba os3 = new Osoba("Ewa", "Gaj", 3);
        Osoba os4 = new Osoba("Zenon", "Hak", 4);
        Osoba os = new Osoba();
        os1.wyswietl();
        os2.wyswietl();
        os3.wyswietl();
        // Tworzymy baze -----
        Baza mybaza = new Baza();
        // dodajemy elementy
        mybaza.dodaj(os1);
        mybaza.dodaj(os2);
        mybaza.dodaj(os3);
        mybaza.wyswietl();
        mybaza.usun();
        mybaza.wyswietl();

    }
}

```

Przykład 11-1 Program BazaOsobaStackDemo.java

Program powinien dać wyniki:

```

Start
Kowal Jan 1
Maj Wanda 2
Gaj Ewa 3
Wyswietl - elementow: 3
Gaj Ewa 3
Maj Wanda 2
usun
Kowal Jan 1
Wyswietl - elementow: 0

```


11.2 Wykorzystanie struktury Vector do budowy bazy danych osobowych

Poniżej podano szkielet bazy danych osobowych wykorzystujących strukturę Vector.

```
// Demonstracja bazy danych Osoba -----
// Wykorzystuje kolekcję vector

import java.util.*;
import java.util.Scanner;

class Osoba {
    String imie;
    String nazwisko;
    int pesel;

    Osoba(String im, String nazw, int pes) {
        imie = im;
        nazwisko = nazw;
        pesel = pes;
    };

    Osoba() {
        imie = "";
        nazwisko = "";
        pesel = 0;
    };

    void wczytaj() {
        Scanner in = new Scanner(System.in);
        System.out.print("Podaj imie: ");
        imie = in.nextLine();
        System.out.println("imie: " + imie);
        System.out.print("Podaj nazwisko: ");
        nazwisko = in.nextLine();
        System.out.println("nazwisko: " + nazwisko);
        System.out.print("Podaj pesel: ");
        pesel = in.nextInt();
        System.out.println("pesel: " + nazwisko);
    };

    void wyswietl() {
        System.out.println(nazwisko + " " + imie + " " + pesel);
    };

    int daj_pesel() { return pesel; }
    String daj_nazwisko() { return nazwisko; }
    String daj_imie() { return imie; }
};

class Baza {
    final int MAX = 50; // Początkowy rozmiar tablicy
    Vector<Osoba> tab;

    Baza() // konstruktor
    {
        tab = new Vector(MAX);
        tab.clear();
    };

    // Dodanie osoby os do bazy -----
    int dodaj(Osoba os) {
```

```
        Osoba osob;
        tab.addElement(os);
        return 1;
    }

    // Wyświetlenie bazy -----
    void wyswietl() {
        int i;
        Osoba os;
        System.out.println("Wyswietl - elementow: " + tab.size());
        // ...
    }

    // Przeszukiwanie bazy po peselu ----
    int szukaj(int pesel) {
        int i = 0;
        int pes;
        Osoba os;
        System.out.println("Szukanie pesel: " + pesel);

        // czy baza pusta ?
        if(tab.size() == 0) {
            System.out.println("Baza pusta");
            return -1;
        }

        // Przeglądamy tablice tab ---
        for(i=0;i<tab.size(); i++) {
            os = tab.get(i);
            os.wyswietl();
            pes = os.daj_pesel();
            if(pes == pesel) {
                System.out.print("znaleziono na poz: " + i + " - ");
                os.wyswietl();
                return i;
            }
        }
        return -1;
    }

    int usun(int pesel) {
        int i = 0;
        int found = -1;
        Osoba osob;
        System.out.println("usun, pesel: " + pesel);
        found = szukaj(pesel);
        // Sprawdzamy czy znaleziono ---
        if(found < 0) {
            System.out.println("Nie znaleziono osoby");
            return 0;
        }
        // Usuwanie osoby -----
        System.out.println("usuwanie z pozycji: " + found) ;
        tab.removeElementAt(found);
        return 1;
    }
}
```

```
public class BazaOsobaVectorDemo {  
  
    public static void main(String args[]) {  
        System.out.println("Start");  
        Osoba os1 = new Osoba("Jan", "Kowal", 1);  
        Osoba os2 = new Osoba("Wanda", "Maj", 2);  
        Osoba os3 = new Osoba("Ewa", "Gaj", 3);  
        Osoba os4 = new Osoba("Zenon", "Hak", 4);  
        Osoba os = new Osoba();  
        // Osoba os2 = new Osoba();  
        os1.wyswietl();  
        os2.wyswietl();  
        os3.wyswietl();  
        // Tworzymy baze -----  
        Baza mybaza = new Baza();  
        // dodajemy elementy  
        mybaza.dodaj(os1);  
        mybaza.dodaj(os2);  
        mybaza.dodaj(os3);  
        mybaza.wyswietl();  
        mybaza.szukaj(2);  
        mybaza.usun(2);  
        mybaza.wyswietl();  
    }  
}
```

Przykład 11-2 Plik BazaVect/BazaOsobaVectorDemo.java

Na podstawie tego szkieletu dokonaj implementacji bazy danych osobowych z prostym interfejsem użytkownika który umożliwi wykonanie operacji:

- Wprowadzania danych osoby
- Wyświetlania danych osoby na podstawie peselu
- Wyświetlania danych osoby na podstawie nazwiska
- Usuwanie osoby na podstawie peselu
- Wypisywanie zawartości bazy danych