

# Wyższa Szkoła Informatyki Stosowanej i Zarządzania

pod auspicjami Polskiej Akademii Nauk

## WYDZIAŁ INFORMATYKI

Kierunek INFORMATYKA

Studia I stopnia (dyplom inżyniera)

---



### Język Java

dr inż. Łukasz Sosnowski  
[lukasz.sosnowski@wit.edu.pl](mailto:lukasz.sosnowski@wit.edu.pl)  
[sosnowsl@ibspan.waw.pl](mailto:sosnowsl@ibspan.waw.pl)  
[l.sosnowski@dituel.pl](mailto:l.sosnowski@dituel.pl)

[www.lsosnowski.pl](http://www.lsosnowski.pl)



## **Część 1 - administracyjna**



## Informacje o przedmiocie

**Cel główny:** Zdobycie podstawowej wiedzy o języku JAVA oraz umiejętności zaprojektowania i zakodowania programów obiektowych.

**Cele szczegółowe:** Zdobycie wiedzy o:  
Klasach, Metodach, Typach danych, Instrukcjach sterujących,  
Operatorach, Interfejsach, Dziedziczeniu, Polimorfizmie,  
Wielowątkowości, Obsłudze wyjątków, Obsłudze wejścia/wyjścia,  
Bibliotece Swing i aplikacjach okienkowych

**Liczba zajęć (zaoczne):**  
8 wykładów na 4 zjazdach (po 2 wykłady na co drugim zjeździe),  
8 laboratoriów



## Forma zaliczenia przedmiotu

- 2 kolokwia (4 i 7 zajęcia)
- 1 projekt do samodzielnego wykonania (tematy rozdane na 5)
- Egzamin pisemny w sesji egzaminacyjnej

### Punktacja:

Kolokwium	20 pkt (x2)
Projekt	10 pkt
Egzamin	40 pkt
Praca na zajęciach	5 pkt
Praca domowa	5 pkt
Punkty dodatkowe	*

### Skala ocen:

< 0,50>	pkt → 2.0
<51,60>	pkt → 3.0
<61,70>	pkt → 3.5
<71,80>	pkt → 4.0
<81,90>	pkt → 4.5
<91,100>	pkt → 5.0

Certyfikat Java SE Programmer I lub Java SE Programmer II \*\*



## Literatura do przedmiotu

Podstawowa:

Herbert Schildt, Java Przewodnik dla początkujących, Wydanie VIII, ,  
ORACLE PRESS, HELION 2020

Uzupełniająca:

Marcin Lis, Java. Ćwiczenia praktyczne. Wydanie IV, Helion, 2014



## Zasady:

**Wykład** – obecność nieobowiązkowa, sprawdzana lista obecności

**Laboratorium** – obecność obowiązkowa, sprawdzana lista obecności

**TEAMS** – zajęcia prowadzone w ramach konferencji w aplikacji

- obowiązkowe kamery + mikrofon,
- udostępnienie ekranu
- brak mikrofonu i/lub kamery = nieobecność

Kolokwia będą obejmować materiał z wykładów i laboratoriów

Egzamin obejmuje cały zakres materiału z wykładów i laboratoriów

Bloki zajęć 3h 15min posiadają 1 przerwę pomiędzy wykładami



## **Część 2 - wprowadzająca**



## Informacje wstępne o języku JAVA

- Rok 1991 – stworzenie języka OAK (Object Application Kernel) w firmie Sun Microsystems przez zespół programistów James'a Gosling'a
- Rok 1995 – zmiana nazwy języka OAK na JAVA ze względu na zastrzeżenie nazwy
- Rok 1996 – pojawienie się pierwszej wersji przeglądarki Netscape z obsługą Java 1.0
- Początkowo język dedykowany dla urządzeń konsumenckich (mikrofalówki, piloty RTV, tostery, etc.), które cechowały się różnorodnością sprzętową procesorów
- Wraz z rozwojem internetu język znalazł powszechne zastosowanie, wywierając jednocześnie wpływ na kierunek jego rozwoju





## Hasła związane z JAVA, wersje, etc.

- JDK (Java Development Kit)
- JRE (Java Runtime Environment)
- JVM (Java Virtual Machine)

### Ważne daty w historii JAVA:

- 1996 – Wydanie Java 1.0
- 1998 – Wydanie Java 1.2 (istotne zmiany)
- 1999 – Wydanie Java EE
- 2004 – Wydanie Java 5 (zmiana numeracji)
- 2010 – Przejęcie firmy Sun Microsystems przez ORACLE
- 2011 – Wydanie Java 7
- 2014 – Wydanie Java 8 (LTS) → 12.2030
- 2018 – Wydanie Java 11 (LTS) → 09.2026
- 2021 – Wydanie Java 17 (LTS) → 09.2029
- ...
- 2023 – Wydanie Java 21(LTS) → 09.2031



## Główne założenia języka JAVA

- 1) **Prostota** – naturalne podobieństwo składni z C++, zwiezłość i spójność cech i mechanizmów
- 2) **Bezpieczeństwo** – zapewnia tworzenie bezpiecznych aplikacji, zwłaszcza do zastosowania w internecie, posiada mechanizmy zabezpieczające kod, np. podpis cyfrowy klas
- 3) **Przenośność** – program Java może być wykonywany w dowolnym środowisku w którym istnieje JVM
- 4) **Obiektowość** – program jako relacja obiektów, które posiadają swój stan i zachowanie (wyjątek typy proste ;))
- 5) **Niezawodność** – silne „otypowanie” oraz kontrola kodu podczas wykonywania
- 6) **Wielowątkowość** – gotowe wsparcie dla programowania wielu wątków i zarządzania nimi.



## Główne założenia języka JAVA c.d.

- 7) **Niezależność** – brak zależności od architektury lub systemu operacyjnego
- 8) **Interpretowalność** – program JAVA kompilowany jest do kodu bajtowego, dzięki któremu może być wykonywany różnych platformach systemowych (interpretowanie kodu bajtowego)
- 9) **Wysoka efektywność** – kod bajtowy został zoptymalizowany z myślą o efektywności wykonania. Uwaga: duże różnice w wydajności pomiędzy wersjami!
- 10) **Rozproszoność** - zaprojektowana z myślą o działaniu w rozproszonym środowisku internetu
- 11) **Dynamiczność** – silne typowanie pozwala dodatkowo weryfikować poprawność dostępu podczas wykonywania programu



# Tworzenie i wykonywanie programów w JAVA



kompilacja  
→  
(javac)

KOD  
BAJTOWY  
(BYTE  
CODE)

uruchomienie  
→

JAVA  
VIRTUAL  
MACHINE

Kod JAVA  
w postaci  
Klas,  
zapisanych  
w plikach  
\*.java

Wynik w  
plikach  
\*.class

Weryfikacja i  
interpretacja kodu  
bajtowego,  
współpraca z OS



## **Część 3 - Klasy, obiekty i metody**



# Hermetyzacja

ukrywanie wybranych danych składowych lub metod obiektów danej klasy w taki sposób, aby były one dostępne jedynie metodom wewnętrznym danej klasy. Funkcjonuje również pod pojęciem „enkapsulacja” z angielskiego „encapsulation”. W języku JAVA hermetyzacja jest ściśle związana z klasą, w której wykorzystywane są odpowiednie *modyfikatory dostępu*. Pozwala na pełną kontrolę nad zachowaniem i stanem danego obiektu. Przyjmuje się, iż dane powinny mieć najbardziej restrykcyjny poziom dostępu jaki jest możliwy dla danego przypadku.



## Klasy w języku JAVA

- Klasa stanowi szablon definiujący postać obiektu. Określa dane obiektu oraz zachowanie obiektu poprzez metody działające na jego danych.
- Klasa stanowi pewien wspólny opis zbioru *elementów* świata rzeczywistego o wspólnych cechach i grupujących je w *byt*.
- Klasa może definiować dane dla obiektu (dane składowe)
- Klasa może definiować zachowanie dla obiektów poprzez metody

Obiekty są **instancjami** klas. Klasa jest jedynie „instrukcją” do utworzenia obiektu, wypełnienia go danymi. Pamięć komputera jest zasilana reprezentacją dopiero w chwili utworzenia obiektu danej klasy!



## Ogólna definicja klasy

- Słowo kluczowe „**class**” i następująca po niej nazwa klasy. Nazwy klasy zaczynają się wielką literą. Stosujemy notację *PascalCase*.

```
class NazwaKlasy{  
    //Deklaracje zmiennych składowych  
    typ zmienna1;  
    typ zmienna2;  
    ...  
    //Deklaracje metod  
    typ metoda1(parametry){  
        //ciało metody  
    }  
}
```





## Przykłady definicji klas

```
class Example1{  
}
```

```
class Example2{  
    public static void main(String args[]){  
    }  
}
```

```
class Example3{  
    private boolean isBusy;  
  
    public void setBusy(){  
        this.isBusy=true;  
    }  
}
```



## Typy danych – typy proste

- 8 wbudowanych typów nieobiektowych, zwanych również podstawowymi, prymitywnymi, elementarnymi. Stanowią bezpośrednią reprezentację wartości binarnych. Wszystkie inne typy danych w JAVA tworzone są z typów prostych.

***boolean*** – wartość logiczna, dziedzina wartości {true, false}

***byte*** - 8 bitowa wartość całkowita

***char*** - znak unicode, 2 bajtowy

***double*** - wartość zmiennoprzecinkowa o podwójnej precyzji

***float*** - wartość zmiennoprzecinkowa o pojedynczej precyzji

***int*** - *wartość całkowita*

***long*** - *długa wartość całkowita*

***short*** - *krótka wartość całkowita*



## Typy danych – typy całkowite (proste)

- 4 podstawowe typy całkowite do reprezentowania zmiennych z określonych dziedzin wartości

Typ	Liczba bitów reprezentacji	Zakres
byte	8	od -128 do 127
short	16	od -32 768 do 32 767
int	32	od -2 147 483 648 do 2 147 483 647
long	64	od -9 223 372 036 854 775 808 do 9 223 372 036 854 775 807

- Wartość domyślna niezainicjowanej zmiennej: 0 i 0L



## Typy danych – typy zmiennoprzecinkowe (proste)

- 2 podstawowe typy zmiennoprzecinkowe różniące się precyzją

Typ	Liczba bitów reprezentacji	Zakres
float	32	od 1.40129846432481707e-45 do 3.40282346638528860e+38 (dodatnie lub ujemne)
double	64	od 4.94065645841246544e-324d do 1.79769313486231570e+308d (dodatnie lub ujemne)

- Wartość domyślna niezainicjowanej zmiennej, odpowiednio: 0.0f, 0.0d



## Typy danych – typ znakowy (prosty)

- Java używa UNICODE, umożliwiając proste kodowanie dowolnego znaku w dowolnym języku. Znak kodowany jest 2 bajtowo

Typ	Liczba bitów reprezentacji	Zakres
char	16	Znaki kodowane od 0 do 65535

- Wartość domyślna niezainicjowanej zmiennej: `'\u0000'`



## Typy danych – typ logiczny (prosty)

- Typ kodowany na 2 wartościach: *true* i *false* stanowiącymi zarezerwowane słowa języka Java
- Wartość domyślna niezainicjowanej zmiennej: *false*
- Przypisywanie wartości: *zmienna=true; zmienna=false;*

```
class Example{  
    public static void main(String args[]){  
        boolean isSet=false;  
        System.out.println(„isSet=” + isSet);  
        isSet = true;  
        System.out.println(„isSet=” + isSet);  
    }  
}
```



## Zmienne, inicjalizacja

- Wyróżniamy dwa typy zmiennych: lokalne i globalne
- Zmienne globalne, to składowe klasy, widoczne w obrębie całej klasy, jej wszystkich metod.
- Zmienne lokalne to widoczne jedynie w lokalnym kontekście metody lub bloku kodu.
- Zmienne definiujemy poprzez wskazanie typu oraz nazwę:  
typ nazwaZmiennej;
- Dla nazw zmiennych stosujemy notację wielbłądzą
- Inicjalizacja zmiennej może być wykonana przy deklaracji lub poprzez operator przypisania (omówiony później).

```
boolean isBusy;  
boolean isBusy=true, isBusy2=false, isBusy3;
```



## Składowe klasy

- Zmienne globalne klasy służące do przechowywania danych obiektu
- Klasyczna definicja zmiennej lecz poprzedzona modyfikatorem dostępu
- Brak modyfikatora dostępu oznacza przyjęcie domyślnej wartości modyfikatora
- Składowe klasy to zmienne zarówno typów prostych jak i obiektowych

```
class Example3{  
    private boolean isBusy; → <modyfikator> <typ> <nazwa zmiennej>  
  
    public void setBusy(){  
        this.isBusy=true;  
    }  
}
```





## Modyfikatory dostępu

Modyfikatory dostępu regulują widoczność elementu (klasy, zmiennej, metody) który jest nim poprzedzony w deklaracji. Wyróżniamy 3 słowa kluczowe reprezentujące modyfikatory dostępu:

***private*** - Najbardziej restrykcyjny modyfikator. Definiuje dla danego elementu którego dotyczy (klasa, metoda lub zmienna składowa) widoczność jedynie wewnątrz klasy.

***protected*** - Elementy oznaczone tym modyfikatorem dostępu są dostępne w danej klasie i jej podklasach (dziedziczących z danej klasy). Ponadto elementy oznaczone jako *protected* są udostępnione dla innych klas z tego samego pakietu.

***public*** – Oznacza dostęp publiczny do elementu, czyli pełny bez ograniczeń.



## Modyfikatory dostępu c.d.

- Brak zdefiniowania modyfikatora dostępu oznacza dostęp prywatny jednakże z udostępnieniem dla innych klas pakietu.

## Przykład klasy z modyfikatorami dostępu

```
class Example3{  
    private boolean isBusy=false;  
    protected int quantity =0;  
    private void calculate(){  
        //ciało  
    }  
    public void setBusy(){  
        this.isBusy=true;  
    }  
    public boolean isBusy(){  
        quantity++;  
        return isBusy;  
    }  
}
```



## Konstruktor

- Zarezerwowany element klasy, którego zadaniem jest inicjalizacja obiektu podczas jego tworzenia.
- Konstruktor ma nazwę identyczną z nazwą klasy
- Konstruktor przeważnie jest publiczny, ale może być również prywatny oraz chroniony (np. przy implementacji wzorca Singleton używamy konstruktora prywatnego)
- Każda klasa posiada konstruktor jawny bądź domyślny.
- Konstruktor domyślny inicjuje zmienne składowe wartościami domyślnymi typów.
- Jeśli zdefiniowany zostanie konstruktor jawny, domyślny nie będzie wywoływany
- Wyróżniamy dodatkowo konstruktory parametryczne, czyli takie do których przekazywane są parametry



## Konstruktor c.d.

- Klasa może zawierać wiele konstruktorów

```
class Example4{  
    private boolean isBusy;  
    protected int quantity;  
    public Example4(){this.quantity=0; this.isBusy=false;}  
    public Example4(int quantity){this.quantity=quantity;}  
    public Example4(boolean isBusy){this.isBusy=isBusy;}  
    public Example4(int quantity,boolean isBusy){  
        this(isBusy);  
        this.quantity=quantity;  
    }  
}
```

- Konstruktory muszą różnić się parametrami, ich liczbą lub typami.  
Nazwa parametru nie ma znaczenia
- Kompilator nie pozwoli zadeklarować 2 takich konstruktorów:  
`public Example4(int quantity){}` `public Example4(int value){}`



## Metody

- Metoda jest podprogramem działającym na danych przechowywanych w klasie
- Metoda zawiera jedną lub więcej instrukcji
- Metoda powinna wykonywać jedno ściśle określone zadanie
- Metody mogą służyć do przekazywania danych do obiektu klasy (setters) oraz odczytywania danych (getters)
- Każda metoda ma swoją nazwę
- Metoda może zawierać parametry
- Kombinacja nazwy i parametrów z uwzględnieniem typów oraz liczby parametrów musi być w klasie unikalna.
- Istnieje metoda specjalna o nazwie „main” od której rozpoczyna się działanie programu. Nie jest jednak obowiązkowa.



## Metody

- Metoda nie zwracająca wartości deklarowana jest ze słowem kluczowym **void**
- Metody mogą zwracać wartości dowolnych obsługiwanych typów prostych jak i obiektowych

```
class Example4{  
    private boolean isBusy;  
    public void setIsBusy(boolean isBusy){this.isBusy=isBusy;}  
    public boolean getIsBusy(){return isBusy;}  
}
```

- Instrukcja **return** pozwala na zakończenie wykonywania kodu metody (return;) oraz dodatkowo daje możliwość zwrócenia wartości (return wartość; lub return zmienna;)
- Ogólna postać metody: 

```
zwracany-typ nazwa(lista-parametrów){  
    //ciało metody  
}
```



## Przykład 1 – hermetyzacja

```
public class Car {  
    protected int bodyType;  
    protected int engineType;  
    protected int engineCapacity;  
    public Car(int bodyType, int engineType, int engineCapacity){  
        this.bodyType = bodyType; this.engineType=engineType; this.engineCapacity=engineCapacity;  
    }  
    public double calculateFuelConsumption(){  
        double result=0.0d;  
        //ciało metody  
        return result;  
    }  
    public int getEngineCapacity() {  
        return engineCapacity;  
    }  
    public void setEngineCapacity(int engineCapacity) {  
        this.engineCapacity = engineCapacity;  
    }  
}
```

- Brak dostępu do danych składowych obiektu
- Dostępny jedynie wynik zwracany przez metodę
- Klasa posiada konstruktor parametryczny
- Konstruktor domyślny nie jest tu dostępny



## Podsumowanie

- Historia powstania i rozwoju języka
- Sposób kompilacji i uruchamiania programu
- Podstawowa wiedza o klasach
- Podstawowa wiedza o konstruktorach
- Podstawowa wiedza o metodach
- Podstawowa wiedza o modyfikatorach dostępu
- Podstawowa wiedza o składowych klasy



# Wyższa Szkoła Informatyki Stosowanej i Zarządzania

pod auspicjami Polskiej Akademii Nauk

## WYDZIAŁ INFORMATYKI

Kierunek INFORMATYKA

Studia I stopnia (dyplom inżyniera)

---



**Dziękuję za uwagę!**