# Konstruktory i destruktory w języku Java


- Wprowadzenie
- Przeczytaj
- Prezentacja multimedialna
- Sprawdź się
- Dla nauczyciela


Źródło: Matthew Hamilton, domena publiczna.

Wiesz już, na czym polega programowanie obiektowe. Umiesz wyjaśnić pojęcia takie jak klasa, obiekt, konstruktor albo destruktor. Są to fundamentalne elementy, występujące w tym paradygmacie programowania.

Ten e-materiał poświęcimy konstruktorom w języku Java. Dlaczego nie wspominamy o destruktorach? Sprawdźmy.

Więcej o programowaniu obiektowym znajdziesz w e-materiale Programowanie obiektowe – projekt, etap I.

Więcej teorii na temat konstruktorów i destruktorów znajduje się w: Konstruktory i destruktory.

Ciekawi cię, jak wyglądają implementacje w innych językach programowania? Możesz się z nimi zapoznać w dwóch pozostałych materiałach z tej serii:

- Konstruktory i destruktory w języku C++,
- Konstruktory i destruktory w języku Python.

> **Twoje cele**
>
> - Prześledzisz informacje o programowaniu obiektowym w języku Java.

- Przeanalizujesz, jak działają konstruktory w języku Java.
- Skonstruujesz klasy opisujące obiekty znane z życia codziennego i dodasz do nich konstruktory.

# Przeczytaj

## Konstruktory w języku Java

W języku Java konstruktory pełnią funkcję zgodną z założeniami programowania obiektowego. Konstruktor to metoda wywoływana w trakcie tworzenia obiektu, najczęściej służąca do nadania początkowych wartości polom w klasie. Każda klasa ma swój domyślny, pusty konstruktor, który nie zmienia żadnych pól. W przypadku, kiedy programista definiuje własny konstruktor, ten domyślny znika – może jednak zostać dodany ręcznie. Konstruktory mają dwie cechy specjalne, które wyróżniają je na tle innych metod:

- nazwy konstruktorów są zawsze takie same, jak nazwa klasy, w której się znajdują,
- konstruktory nie zwracają żadnej wartości.

Zdefiniowany w języku Java konstruktor należący do klasy `Czlowiek`:

```
class Czlowiek
{
    /* Pola */

    String imie;
    String nazwisko;
    int wiek;


    /* Konstruktor */

    Czlowiek (String kon_imie, String kon_nazwisko, int kon_wiek)
    {
        imie = kon_imie;
        nazwisko = kon_nazwisko;
        wiek = kon_wiek;
    }
}
```

> **Ważne!**
>
> Zwróć uwagę, że w języku Java definicji klasy nie kończy się znakiem średnika (`;`) po ostatnim prawym nawiasie klamrowym.

Samodzielnie zdefiniowane konstruktory ułatwiają przypisywanie wartości polom obiektu. Przedstawiony konstruktor `Czlowiek()` można dodatkowo ulepszyć.

Obecnie musimy nadać nazwy argumentom, których wartości zostaną przypisane odpowiednim polom (przykładowo: argument o nazwie `kon_imie` odpowiada polu `imie`). Wprowadza to bałagan w kodzie.

Usuniemy tę niedogodność, używając w definicji konstruktora specjalnej referencji `this`. Wskazuje ona obiekt, dla którego wywołana została metoda (w przypadku wywołania konstruktora będzie to powstająca właśnie instancja klasy `Czlowiek`).

Zmodyfikujmy zatem klasę `Czlowiek`:

```
class Czlowiek
{
    /* Pola */

    String imie;
    String nazwisko;
    int wiek;


    /* Konstruktor */

    Czlowiek (String imie, String nazwisko, int wiek)
    {
        this.imie = imie; // zasada jest cały czas taka sama -> '
        this.nazwisko = nazwisko;
        this.wiek = wiek;
    }
}
```

Konstruktor wygląda bardziej czytelnie niż poprzednio, a działa tak samo.

Warto wspomnieć, że definicje klas w języku Java mogą zawierać wiele konstruktorów. Muszą się one jednak różnić albo liczbą przyjmowanych argumentów, albo ich typem. Właśnie na podstawie liczby lub typu argumentów program dobiera odpowiedni konstruktor podczas tworzenia obiektu.

Pamiętajmy, że konstruktor domyślny tworzony automatycznie (chociaż może on zostać zdefiniowany manualnie), nie ma żadnych argumentów. Oto jego budowa:

```
Czlowiek() {} // konstruktor domyślny klasy 'Czlowiek'
```

## Destruktory

W języku Java nie używa się destruktorów, czyli metod wywoływanych przez program w celu usunięcia obiektu z pamięci. Wynika to z faktu, że stosuje się w nim szczególną metodę zarządzania pamięcią.

W przypadku języków niskopoziomowych zarządzaniem pamięcią zajmuje się programista. Właśnie on jest odpowiedzialny między innymi za usuwanie niepotrzebnych obiektów z pamięci.

Natomiast po uruchomieniu programu napisanego w języku **Java** zarządzaniem pamięcią zajmuje się wirtualna maszyna Javy (JVM, ang. *Java Virtual Machine*), a ściślej mówiąc tzw. *garbage collector*. Wirtualna maszyna zapobiega wyciekom pamięci, czyli niezamierzonemu zajmowaniu pamięci komputera.

Mechanizm ten jest bardzo wygodny, ponieważ przejmuje część zadań, które musiałby wykonać programista. Z drugiej strony ceną za to jest spowolnienie działania programów (w porównaniu z ich odpowiednikami napisanymi w językach z ręcznym zarządzaniem pamięcią). Obecnie to się zmienia: aplikacje Javy coraz mniej ustępują w szybkości programom utworzonym w językach, w których nie stosuje się automatycznego zarządzania pamięcią.

Mimo wszystko usuwanie niepotrzebnych obiektów jest uważane za dobry nawyk programisty.

## Przykład

Zdefiniujmy w języku Java klasę, która będzie zawierać wszystkie omówione elementy. Załóżmy, że jej obiekty mają służyć do opisania smartfonów w sklepie ze sprzętem elektronicznym.

Tworzymy puste ciało klasy `Smartfon`:

```
public class Smartfon  // znaczenie operatora 'public' zostanie w
{

}
```

Definiujemy pola, które będą wchodzić w skład obiektów powstającej klasy (przykładowo: marka, model, cena i opis).

```
public class Smartfon
{
    /* Pola */

    public String marka;
    public String model;
    public double cena;
    public String opis;
}
```

> **Ważne!**
>
> Słowo kluczowe `public` oznacza, że pola klasy `Smartfon` są publiczne i dostępne z innych klas i pakietów. Można je modyfikować i odczytywać bezpośrednio z innych miejsc w programie.

W przyszłości będziemy tworzyć obiekty mające wstępnie przypisane wartości pól, dlatego przygotujemy odpowiedni konstruktor.

```
public class Smartfon
{
    /* Pola */

    public String marka;
    public String model;
    public double cena;
    public String opis;


    /* Konstruktor */

    public Smartfon (String marka, String model, double cena, Str
    {
        this.marka = marka;
        this.model = model;
        this.cena = cena;
        this.opis = opis;
    }
}
```

Na koniec definiujemy metodę pozwalającą wyświetlać informacje na temat każdego smartfona:

```
public class Smartfon
{
    /* Pola */

    public String marka;
    public String model;
    public double cena;
    public String opis;


    /* Konstruktor */

    public Smartfon (String marka, String model, double cena, Str
    {
        this.marka = marka;
        this.model = model;
        this.cena = cena;
        this.opis = opis;
    }

    /* Metody */

    public void wypisz_info()
    {
        System.out.println ("Marka: " + marka);
        System.out.println ("Model: " + model);
        System.out.println ("Cena: " + cena);
        System.out.println ("Opis: " + opis);
    }
}
```

## Słownik

> ***garbage collector***
>
> jedna z metod automatycznego zarządzania dynamicznie przydzielaną pamięcią; za proces jej zwalniania odpowiedzialny jest programowy zarządca zwany w skrócie GC
> **klasa**

zdefiniowany przez programistę złożony typ zmiennej, która może zawierać metody oraz pola danych

**konstruktor**

specjalna metoda danej klasy, którą wywołuje się podczas tworzenia jej instancji; zadaniem konstruktora jest zainicjowanie obiektu, a w niektórych językach programowania także utworzenie obiektu

**destruktor**

przeciwieństwo konstruktora; specjalna metoda, wywoływana przez program przed usunięciem obiektu

**obiekt**

pojedyncza instancja pewnej klasy, czyli zmienna o jej typie; struktura zawierająca dane oraz metody (funkcje służące do wykonywania na tych danych określonych zadań)

# Prezentacja multimedialna

**Polecenie 1**

Mamy napisać program, który ma służyć do zarządzania książkami w bibliotece. Zdefiniujemy klasę, której obiekty będą reprezentować poszczególne książki. Chcemy ponadto, aby w skład klasy (oprócz zestawu danych i konstruktora) wchodziła metoda pozwalająca wyświetlić dane na temat książki.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Rozpocznijmy od utworzenia klasy `Ksiazka`:

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Dodajmy odpowiednie pola klasy, czyli pole `tytul` przechowujące łańcuch znaków, pole `autor` również przechowujące łańcuch znaków oraz pole `rokWydania` przechowujące liczbę naturalną. Dodajemy również konstruktor tworzący obiekt klasy `Ksiazki` z wykorzystaniem danych podanych w konstruktorze.

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Do naszego kodu dodamy również destruktor. Wyświetli informację na temat usuwanego obiektu.

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Ostatnim elementem klasy będzie metoda `wypiszInfo()`, prezentująca informacje na temat książki:

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Klasa `Ksiazka` jest już gotowa. Spróbujmy ją przetestować. W tym celu zdefiniujemy kolejną klasę (przykładowo, `Main`), zawierającą metodę startową `main()`. Metoda `main` może być dodana do klasy.

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

W ciele metody `main()` zapisujemy polecenie utworzenia obiektu klasy `Ksiazka` (noszącego

nazwę `ksiazka`), a następnie wywołujemy metodę `wypiszInfo()`:

Na ekranie powinien pojawić się napis:

```
Tytuł: Algorytmy + Struktury
danych = Programy

Autor: Niklaus Wirth

Rok wydania: 1976

Usunięcie obiektu typu Ksiazka
```

**Polecenie 2**

Napiszmy program przypominający nieco ten z poprzedniego przykładu. Tym razem program ma obsługiwać komis samochodowy. Tworzymy właśnie klasę, której obiekty będą reprezentować pojazdy wystawione na sprzedaż.

Także i w tym przypadku klasa ma składać się z odpowiednich pól, konstruktora oraz z metody pozwalającej wyświetlać dane na temat poszczególnych samochodów.

Będziemy pisać kod stopniowo. Spróbuj równolegle tworzyć jego własną wersję w języku Java.

**Ważne!**

Przeciążenie konstruktorów polega na zdefiniowaniu wielu metod konstruktorów dla klasy, każdy z innymi parametrami. Pozwala to na tworzenie obiektów danej klasy z różnymi zestawami parametrów bez konieczności definiowania oddzielnych metod. Podczas tworzenia nowego obiektu, kompilator wybierze odpowiedni konstruktor do użycia na podstawie liczby i typów przekazanych argumentów. Dzięki przeciążeniu konstruktorów możemy tworzyć bardziej elastyczne klasy.
**Ważne!**

Informację na temat tego, dlaczego pola nie są oznaczone jako `public`, znajdziesz w e-materiałach Zasady programowania obiektowego oraz Zasady programowania obiektowego w języku Java.

1

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Zacznijmy od zapisania szkieletu klasy `Samochod`:

2

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Dodajmy pola klasy `Samochod`. Te pola to pole `marka` przechowujące łańcuchj znaków, pole `model` również przechowujące łańcuch znaków oraz dwa pola przechowujące liczby naturalne – `rocznik` i `przebieg`. Dodajemy również konstruktor służący do tworzenia nowych obiektów klasy `Samochod`:

3

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Ostatnim elementem klasy będzie metoda `wypiszInfo ()`, pokazująca informacje na temat samochodu:

4

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Klasa `Samochod` jest gotowa. Przetestujemy ją, definiując kolejną klasę (`Main`), zawierającą metodę startową `main ()`:

5

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Wewnątrz metody startowej tworzymy obiekt klasy `Samochod` o nazwie `samochod` i wywołujemy metodę `wypiszInfo ()`:

Na ekranie pojawią się komunikaty:

Marka: Tesla

Model: Model X

Rocznik: 2019

Przebieg: 50 km

6

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Teraz zmodyfikujemy klasę `Samochod`. Pozwolimy tworzyć obiekty zawierające informację o tym, czy pojazd uległ wcześniej wypadkowi.

Zdefiniujemy drugi konstruktor, który będzie odpowiadał za tworzenie obiektów zawierających dane o wypadkach z udziałem samochodu wystawianego na sprzedaż.

Wewnątrz drugiego konstruktora użyjemy dodatkowego argumentu. Pozwoli to programowi dobrać odpowiednią metodę podczas tworzenia obiektu klasy `Samochod`.

7

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Definiujemy nowe pole typu logicznego (`boolean`). Nosi ono nazwę `czyBezwypadkowy`. Wykorzystujemy je w drugim konstruktorze należącym do klasy `Samochod`:

8

Materiał audio dostępny pod adresem:

https://zpe.gov.pl/b/P6UeNy7ZF

Dodajmy destruktor.

# Sprawdź się

Pokaż ćwiczenia: 

**Ćwiczenie 1**

Do przedstawionego kodu dodaj konstruktor, dzięki któremu będzie można utworzyć obiekt klasy `Kot` mający pole `imie`, a następnie wyświetlić zawartość pola `imie` na ekranie.

Swój program przetestuj dla pola `imie` o wartości `Ryfka`.

**Specyfikacja problemu:**

*Dane:*

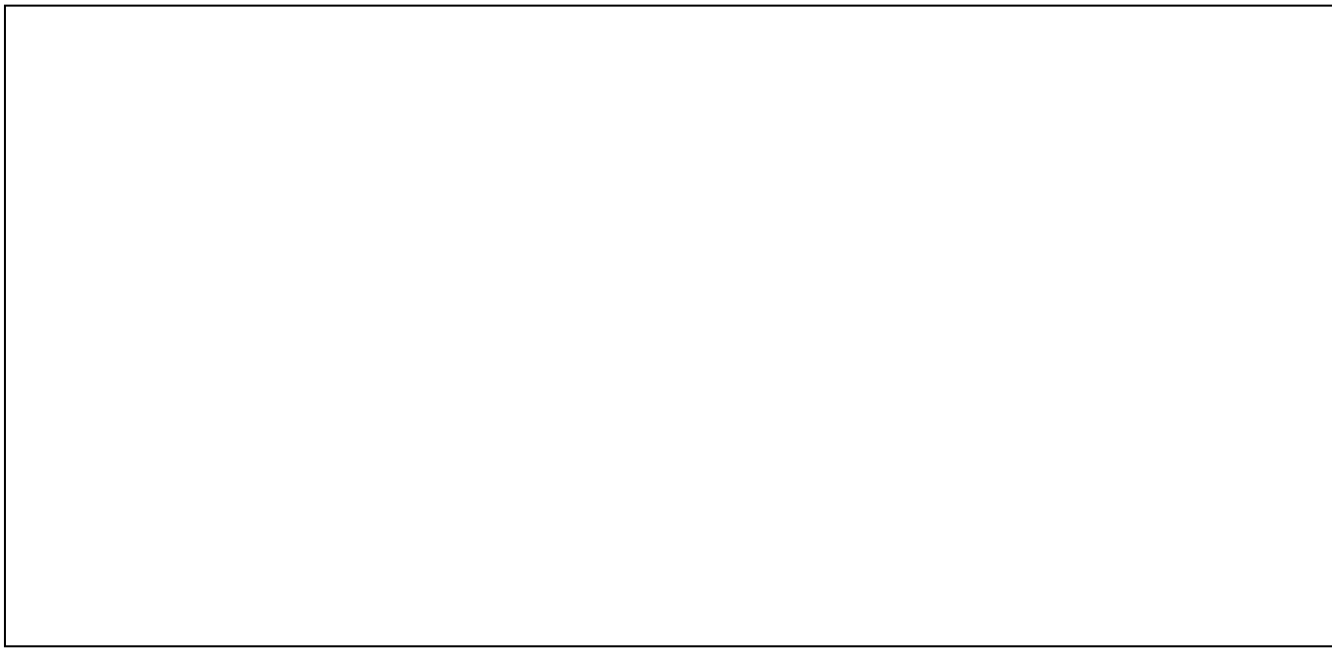
- `imie` – pole klasy `Kot`

*Wynik:*

- program wyświetla wartość pola `imie` dla obiektu `kot` klasy `Kot`

## Twoje zadania

1. Program wyświetla zawartość pola `imie` obiektu klasy `Kot`.



**Ćwiczenie 2**

Uzupełnij kod tworzący obiekt o nazwie `czlowiek` klasy `Czlowiek` o polach `imie` i `nazwisko`. Program powinien wyświetlić zawartość pól.

Swój program przetestuj dla pola `imie` o wartości `Adam` i pola `nazwisko` o wartości `Nowak`.

## Twoje zadania

1. Program powinien wyświetlać imię i nazwisko, zapisane jako pola obiektu klasy `Czlowiek`.

**Ćwiczenie 3**

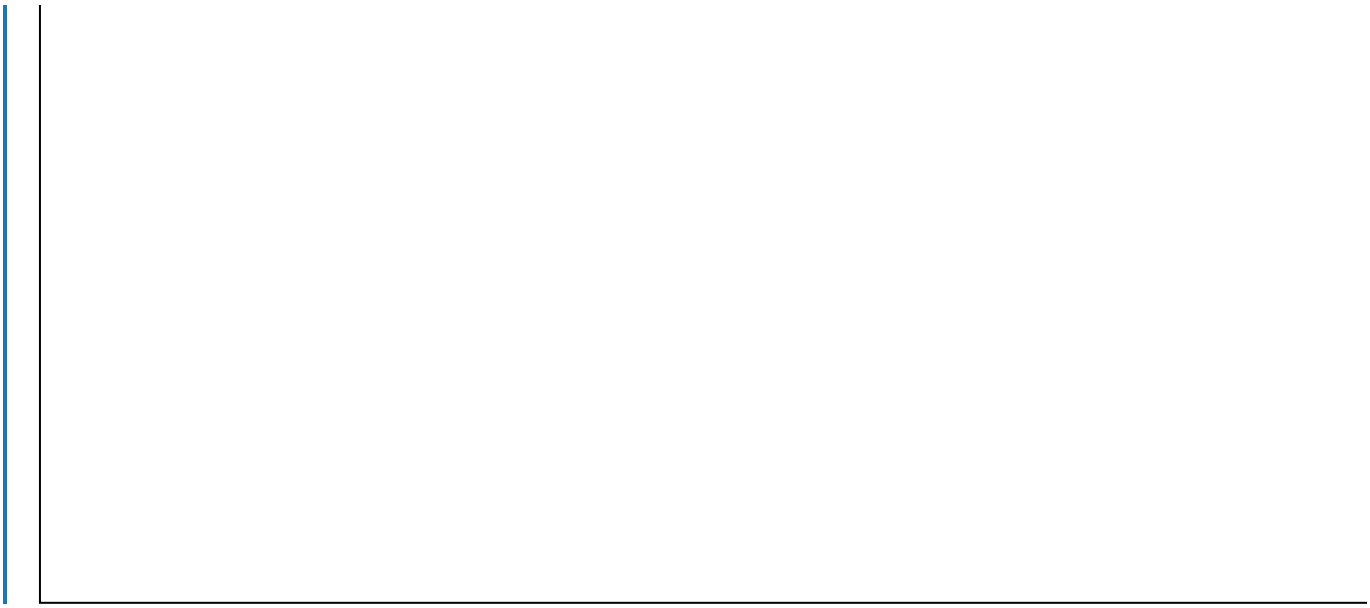
Zmodyfikuj kod, dodając metodę, dzięki której program będzie wyświetlał pola `imie` oraz `nazwisko` obiektu `czlowiek` klasy `Człowiek`. Dodaj również destruktor klasy `Człowiek`, który wyświetli informację o usuwaniu obiektu.

*Przykładowe wyjście:*

```
Adam Nowak
Jan Kowalski
Usuwanie obiektu klasy Czlowiek
```

## Twoje zadania

1. Program powinien wyświetlić imię i nazwisko, zapisane w polach obiektu klasy `Czlowiek`.



# Dla nauczyciela

**Autor:** Maurycy Gast
**Przedmiot:** Informatyka

**Temat: Konstruktory i destruktory w języku Java**

**Grupa docelowa:**

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres rozszerzony

**Podstawa programowa:**

Cele kształcenia – wymagania ogólne

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

Treści nauczania – wymagania szczegółowe

> II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.
>
> > Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:
> >
> > > 1) projektuje i tworzy rozbudowane programy w procesie rozwiązywania problemów, wykorzystuje w programach dobrane do algorytmów struktury danych, w tym struktury dynamiczne i korzysta z dostępnych bibliotek dla tych struktur;
> > >
> > > 2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów;
> > >
> > > 3) sprawnie posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

**Kształtowane kompetencje kluczowe:**

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

**Cele operacyjne (językiem ucznia):**

- Prześledzisz informacje o programowaniu obiektowym w języku Java.
- Przeanalizujesz, jak działają konstruktory w języku Java.
- Skonstruujesz klasy opisujące obiekty znane z życia codziennego i dodasz do nich konstruktory.

**Strategie nauczania:**

- konstruktywizm;
- konektywizm.

**Metody i techniki nauczania:**

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

**Formy pracy:**

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

**Środki dydaktyczne:**

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda;
- oprogramowanie dla języka Java SE 8 (lub nowszej wersji), w tym Eclipse 4.4 (lub nowszej wersji).

**Przebieg lekcji**

**Przed lekcją:**

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Konstruktory i destruktory w języku Java". Uczniowie mają zapoznać się z treściami w sekcji „Przeczytaj" i wykonać obliczenia na podstawie dołączonych danych.

**Faza wstępna:**

1. Nauczyciel wyświetla temat i cele zajęć zawarte w sekcji „Wprowadzenie". Prosi uczniów, by na podstawie wiadomości zdobytych przed lekcją zaproponowali kryteria sukcesu.
2. **Rozpoznanie wiedzy uczniów.** Nauczyciel zadaje uczniom pytanie dotyczące ich aktualnego stanu wiedzy w obszarze poruszanego tematu, opartego o programowanie.

**Faza realizacyjna:**

1. **Praca z tekstem.** Nauczyciel ocenia, na podstawie informacji na platformie, stan przygotowania uczniów do zajęć. Jeżeli jest ono niewystarczające, prosi wybraną osobę o przedstawienie najważniejszych informacji z sekcji „Przeczytaj". Następnie uczniowie wspólnie analizują rozwiązanie przykładu: „Zdefiniujmy w języku Java klasę, która będzie zawierać wszystkie omówione elementy. Załóżmy, że jej obiekty mają służyć do opisania smartfonów w sklepie ze sprzętem elektronicznym".
2. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Prezentacja multimedialna". Uczniowie zapoznają się z prezentacją, która pokazuje w jaki sposób można napisać program, który ma służyć do zarządzania książkami w bibliotece. Pisanie kodu pokazane jest stopniowo, uczniowie próbują równolegle tworzyć jego własną wersję w języku Java.
3. **Ćwiczenie umiejętności.** Uczniowie wykonują indywidualnie ćwiczenie nr 1 z sekcji „Sprawdź się" , a następnie porównują swoje odpowiedzi z kolegą lub koleżanką.
4. W kolejnym etapie uczniowie dobierają się w pary i wykonują ćwiczenia nr 2 i 3 z sekcji „Sprawdź się" . Następnie konsultują swoje rozwiązania z inną parą uczniów i ustalają jedną wersję odpowiedzi.

**Faza podsumowująca:**

1. Nauczyciel ponownie wyświetla na tablicy temat lekcji zawarty w sekcji „Wprowadzenie" i inicjuje krótką rozmowę na temat zrealizowanych celów (czego uczniowie się nauczyli).
2. Nauczyciel prosi uczniów o podsumowanie zgromadzonej wiedzy w zakresie programowania w języku Java.

**Praca domowa:**

1. Uczniowie wykonują polecenie 2 z sekcji „Prezentacja multimedialna". Ich zadaniem jest napisanie programu, który ma obsługiwać komis samochodowy. Tworzą klasę, której obiekty będą reprezentować pojazdy wystawione na sprzedaż.

**Materiały pomocnicze:**

- Oficjalna dokumentacja techniczna dla języka Java SE 8 (lub nowszej wersji).
- Oficjalna dokumentacja techniczna dla oprogramowania Eclipse 4.4 (lub nowszej wersji).

**Wskazówki metodyczne:**

- Uczniowie mogą wykorzystać multimedium w sekcji „Prezentacja multimedialna" do przygotowania się do lekcji powtórkowej.