

# Programowanie obiektowe

# Zadanie 13

Zdefiniować klasę Article, reprezentującą towar w sklepie  
pola klasy: identyfikator liczbowy, nazwa, cena i opis  
Utworzyć obiekt klasy i zainicjalizować jego pola  
Wypisać na ekran zawartość pól obiektu

# Zadanie 14

Do definicji klasy `Article` dodać zmienną statyczną `counter` (liczba całkowita)

Jawnie zdefiniować konstruktor domyślny klasy `Article` i konstruktor, przyjmujący jako argumenty wartości do inicjalizacji pól

Zwiększać wartość `counter` w obu konstruktorach

Utworzyć obiekty za pomocą obu konstruktorów i wypisać zawartość na ekran

Odwołać się do zmiennej statycznej poprzez nazwę klasy

# Zadanie 15

Utworzyć klasę Order o dostępie domyślnym, w innym pakiecie, niż Article

Spróbować utworzyć obiekt klasy Order z tego samego pakietu i z pakietu z klasą Article

Zmienić dostęp do klasy Order na publiczny

Ponownie odwołać się z obu pakietów do klasy

Dodać deklarację importu klasy Order

# Zadanie 16

Zmienić dostęp do pól klasy Article na prywatny  
Dodać metody publiczne, umożliwiające pobieranie i ustawienie wartości pól (getters/setters)  
Dodać metodę prywatną printChange(), wywoływaną przez settery, wypisującą na ekran "Zmieniono wartość pola"  
Utworzyć obiekt klasy, spróbować zmienić wartość pól, zmienić wartości setterami, spróbować wywołać metodę prywatną

# Zadanie 17

Dodać metodę toString do klasy Article, zwracającą reprezentację tekstową stanu obiektu, sprawdzić za pomocą println

Utworzyć klasę Book, dziedziczącą po Article, zawierającą pola: autor i rok wydania

Utworzyć gettery/settery dla pól

Utworzyć obiekt klasy Book za pomocą konstruktora domyślnego, sprawdzić zawartość pól

Wywołać metodę toString na obiekcie klasy Book wypisując poprzez System.out.println

# Zadanie 18

Zaimplementować w klasie Book konstruktor domyślny i inicjalizujący pola argumentami; w drugim wywołać explicite konstruktor klasy nadrzędnej

Przesłonić metodę toString w klasie Book, wykorzystując wywołanie metody w klasie nadrzędnej

Sprawdzić możliwość wzajemnego rzutowania klas

Wywołać explicite metodę toString na obiekcie Book, rzutowanym na klasę Article (polimorfizm)

# Zadanie 19

Spróbować wywołać metodę `printChange()` z klasy `Book`

Przenieść klasę `Book` do podpakietu, zmienić widoczność metody `printChange` na `protected`

Dodać wywołania metody `printChange` do setterów klasy `Book`

Sprawdzić skutek dodania deklaracji `final` do klasy `Article`



# Zadanie 20

Utworzyć abstrakcyjną klasę `Person`, zawierającą

pola: `id`, `name`, statyczne `counter`

konstruktor domyślny, przypisujący polu `id` kolejne wartości `counter`

metody: `getter id`, `getter i setter dla name`

abstrakcyjną metodę `getAuthorization`

Utworzyć klasę `Employee`, dziedziczącą po `Person`

metoda `getAuthorization` powinna zwracać wartość logiczną

# Zadanie 21

Zdefiniować interfejs Downloadable, zawierający metody umożliwiające sprawdzenie wielkości i formatu pliku  
Zdefiniować interfejs Streamable, zawierający metodę `byte[] nextPacket()`, dziedziczący po interfejsie Downloadable  
Zdefiniować klasę Record, dziedziczącą po klasie Article i implementującą interfejs Streamable  
Konstruktor klasy powinien umożliwiać inicjalizację tekstowego identyfikatora formatu nagrania i jego wielkości

# Zadanie 22

Zdefiniować klasę Shop, reprezentującą sklep, przechowującą w tabeli licznosc 10 najpopularniejszych produktów

Zamówienie reprezentowane jest przez wewnętrzną klasę Order

Klasa Order zawiera:

pole int[] quantities – liczby zamówionych towarów

pole Date – data zamówienia

metodę accept() – dodającą towary z zamówienia do stanu magazynowego sklepu

Utwórz obiekt reprezentujący sklep

Utwórz zamówienie dla sklepu i zaakceptuj jego przyjęcie

# Zadanie 23

Do sklepu dodaj  
pole przechowujące obiekt implementujący interfejs  
Downloadable  
getter i setter do tego pola  
Utwórz anonimową klasę, implementującą interfejs  
Downloadable bezpośrednio w wywołaniu settera pola