Zadanie 1: System Zwierząt w ZOO

Cel:

Stwórz system zarządzania zwierzętami w ZOO z wykorzystaniem polimorfizmu, enkapsulacji, dziedziczenia i interfejsów.

Szczegółowy opis:

Część 1: Abstrakcyjna klasa bazowa

Stwórz abstrakcyjną klasę Animal z:

- prywatnymi polami: name (String), age (int)
- konstruktorem inicjującym wszystkie pola
- getterami i setterami do pól (enkapsulacja)
- abstrakcyjną metodą void makeSound()
- metodą void displayInfo() – wyświetla name i age

Część 2: Interfejs

Stwórz interfejs Feedable z:

- void feed()
- String getFoodType()

Część 3: Klasy pochodne

Stwórz trzy klasy dziedziczące po Animal i implementujące Feedable:

1. Klasa Lion:
    - Pole: maneLength (double)
    - Implementacja makeSound()
    - Implementacja feed() – "Feeding meat to lion"
    - Implementacja getFoodType() – "Meat"
2. Klasa Elephant:
    - Pole: trunkLength (double)
    - Implementacja makeSound()
    - Implementacja feed() – "Feeding vegetables to elephant"
    - Implementacja getFoodType() – "Vegetables"
3. Klasa Monkey:
    - Pole: tailLength (double)

    - Implementacja makeSound()
    - Implementacja feed() – "Feeding fruits to monkey"
    - Implementacja getFoodType() – "Fruits"

Część 4: Klasa zarządzająca

Stwórz klasę Zoo z:

- polem: animals (tablica Animal[])
- metodami:
    - addAnimal(Animal animal) – dodaje zwierzę do tablicy
    - makeAllSounds() – wywołuje makeSound() dla wszystkich zwierząt
    - feedAllAnimals() – wywołuje feed() dla wszystkich (wymaga rzutowania na Feedable)
    - displayAllAnimals() – wyświetla informacje o wszystkich zwierzętach

Część 5: Program główny

Stwórz klasę ZooManagement z metodą main:

- Utwórz przynajmniej po jednym obiekcie każdego typu zwierzęcia
- Dodaj je do obiektu Zoo
- Zademonstruj działanie wszystkich metod
- Pokaż polimorfizm w działaniu

Zadanie 2: System Książek w Bibliotece

Cel:

Stwórz system zarządzania różnymi typami książek z walidacją danych i funkcją rezerwacji.

Szczegółowy opis:

Część 1: Abstrakcyjna klasa bazowa

Stwórz abstrakcyjną klasę Book z:

- prywatnymi polami: title (String), author (String), year (int)
- konstruktorem z walidacją: year > 1450 (wynalazek druku)
- getterami i setterami z walidacją (tytuł i autor nie mogą być puste)
- abstrakcyjną metodą double calculateRentalFee()

Część 2: Interfejs

Stwórz interfejs Reservable z:

- void reserve()
- boolean isReserved()

Część 3: Klasy pochodne

Stwórz trzy klasy dziedziczące po Book:

1. Klasa FictionBook:
    - Pole: genre (String)
    - Implementacja calculateRentalFee() – stała opłata 10.0
2. Klasa TextBook (implementuje Reservable):
    - Pole: subject (String), isReserved (boolean)
    - Implementacja calculateRentalFee() – 5.0
    - Implementacja metod z Reservable
3. Klasa RareBook (implementuje Reservable):
    - Pole: condition (String), isReserved (boolean)
    - Implementacja calculateRentalFee() – 25.0 + dodatkowa opłata za stan
    - Implementacja metod z Reservable

Część 4: Klasa zarządzająca

Stwórz klasę Library z:

- polem: books (tablica Book[])
- metodami:
    - addBook(Book book) – dodaje książkę do tablicy
    - displayAllBooks() – wyświetla informacje o wszystkich książkach
    - calculateTotalRentalFees() – oblicza sumę opłat za wszystkie książki
    - reserveBook(String title) – wyszukuje książkę i rezerwuje jeśli implementuje Reservable

Część 5: Program główny z menu

Stwórz klasę LibrarySystem z metodą main i interaktywnym menu:

java

Menu:

1. Dodaj książkę

2. Wyświetl wszystkie książki

3. Oblicz całkowite opłaty

4. Zarezerwuj książkę

5. Wyjście

Wymagania dla menu:

- Użyj Scanner do wprowadzania danych
- Pozwól użytkownikowi wybrać typ książki przy dodawaniu
- Waliduj wprowadzane dane
- Obsłuż wszystkie możliwe błędy