Wariant 504

Wykonaj poniższe czynności:

- Utwórz pakiet `com.sports.team` i w nim umieść w oddzielnych plikach następujące klasy.
- Stwórz klasę `Player` z prywatnymi polami: `name` (typu `String`) oraz `number` (typu `int`).
- Zaimplementuj konstruktor, który przyjmuje dwa argumenty odpowiadające polom klasy, z tą różnicą, że nazwy parametrów konstruktora pokrywają się z nazwami pól klasy.
- Dodaj publiczne metody dostępowe (gettery i settery) dla obu pól klasy.
- Wprowadź walidację dla pola `number`, aby przyjmowało tylko wartości dodatnie. Jeżeli w konstruktorze podana zostanie wartość ujemna, ustaw wartość `number` na `10`. Setter dla `number` nie powinien pozwalać na ustawienie wartości ujemnej.
- Stwórz klasę `TestPlayer` z metodą `main`. W metodzie tej utwórz obiekt klasy `Player` i przetestuj działanie wszystkich getterów i setterów.

Zadanie należy umieścić we własnym repozytorium na Githubie.

Wariant 505

Wykonaj poniższe czynności:

- Utwórz pakiet `com.technology.device` i umieść w nim oddzielne pliki dla poniższych klas.
- Zaprojektuj klasę `Gadget` z prywatnymi polami: `model` (typu `String`) oraz `price` (typu `double`).
- Zaimplementuj konstruktor, który otrzymuje dwa argumenty odpowiadające polom klasy i mające identyczne nazwy co pola klasy.
- Utwórz publiczne metody dostępowe (gettery i settery) dla obu pól.
- Dodaj walidację dla pola `price`, tak aby nie można było ustawić ceny niższej niż 0. W sytuacji podania ujemnej ceny w konstruktorze, ustaw cenę na wartość domyślną `99.99`. Setter dla `price` powinien ignorować próby ustawienia wartości ujemnych.
- Stwórz klasę `TestGadget` z metodą `main`. W głównej metodzie utwórz instancję `Gadget` i wykorzystaj wszystkie metody dostępowe.

Zadanie należy umieścić we własnym repozytorium na Githubie.

Wariant 506

Wykonaj poniższe czynności:

- Utwórz pakiet `com.building.management` i w nim umieść w oddzielnych plikach dwie klasy.
- Zaprojektuj klasę `Building` z prywatnymi polami: `address` (typu `String`) oraz `floors` (typu `int`).
- Zaimplementuj konstruktor, który przyjmuje dwa argumenty zgodne z polami klasy i mające te same nazwy co pola.
- Dodaj publiczne metody dostępowe (gettery i settery) dla obu pól klasy.
- Wprowadź walidację dla pola `floors`, aby zapewnić, że liczba pięter jest większa od zera. Jeżeli w konstruktorze zostanie podana wartość nieprawidłowa, ustaw liczbę pięter na `1`. Setter dla `floors` również powinien odrzucać wartości ujemne i nie zmieniać wartości pola w takim przypadku.
- Stwórz klasę `TestBuilding` z metodą `main`. W tej metodzie utwórz obiekt klasy `Building` i przetestuj działanie wszystkich metod dostępowych.

Zadanie należy umieścić we własnym repozytorium na Githubie.