Programowanie obiektowe w Javie (POJ)

Laboratorium 7

Każde z poniższych zadań wykonaj w osobnym pakiecie wewnątrz jednego projektu. Pamiętaj o właściwym stosowaniu modyfikatorów dostępu i enkapsulacji detali implementacji, które nie powinny być widoczne dla innych klas.

Zadanie 1.

Utwórz klasę Osoba o polach: imie, nazwisko, rok urodzenia. W metodzie konstruktora sprawdź czy imię bądź nazwisko jest przekazane jako pusty String – jeśli tak jest to zwróć wyjątek Exception. Dodaj klasę Main, gdzie poprzez Scanner odpytujesz użytkownika o dane potrzebne do utworzenia obiektu a następnie ten obiekt tworzysz. Następnie:

a) Nadpisz metodę toString w klasie Osoba aby zwracała dane osoby
b) Dodaj obsługę wyjątku – poinformuj użytkownika gdy nie udało się utworzyć nowego obiektu, z względu na błędne dane
c) Zapętl tworzenie użytkownika – dopóki nie zostanie utworzony poprawnie
d) W klasie Osoba utwórz metodę sprawdzDaneOsoby(String imie, String nazwisko, int rokUrodzenia):void która sprawdzi poprawność zadanych danych
    a. Wywołaj metodę już w pierwszej linii konstruktora klasy Osoba
    b. Niech metoda zwróci wyjątek wraz z odpowiednią wiadomością gdy:
        i. Imię jest pustym stringiem -> „Podano niewłaściwe imię!”
        ii. Nazwisko jest pustym stringiem -> „Podano niewłaściwe nazwisko!”
        iii. Rok jest mniejszy od 1900 i większy od 2020 -> „Podano niewłaściwy rok urodzenia!”
    c. W bloku catch wyświetl otrzymany błąd

Zadanie 2.

Pobierz klasy Account.java oraz MainBankException dostępne w zakładce ‘Pliki’ i dodaj je do projektu.
Następnie :

a) Zidentyfikuj miejsca gdzie może dojść do pojawienia się wyjątku i w rezultacie awarii programu. (Sprawdź kod i/lub samodzielnie uruchom program)
b) Obsłuż te zidentyfikowane miejsca poprzez :
    a. dodanie odpowiedniej klauzuli try-catch
        i. Wyświetlenie wiadomości o niepoprawnie wprowadzonej wartości
    b. Rzucenie wyjątkiem gdy użytkownik probuje wypłacić większą kwotę niż stan konta
        i. Obsługa rzuconego wyjątkiem odpowiednim komunikatem
c) Utwórz nowy wyjątek InsufficientFundsException, które będzie rzucane zamiast wcześniej zdefiniowanego wyjątku

Zadanie 3.

Pobierz klasę Divider.java dostępne w zakładce ‘Pliki’ i dodaj do projektu.
Następnie:

a) Obsłuż sytuację, gdy użytkownik próbuje wprowadzić nie-liczbę np. tekst
b) Obsłuż sytuację, gdy użytkownik próbuje podzielić przez zero (dodaj obsługę ArithmeticException)
c) Zapewnij wykonanie podstawowego scenariusza tj. zebrania przez program 2 liczb i wykonania akcji dzielenia, w sytuacji gdy użytkownik dzielił przez zero wyświetl „Błąd, nie można dzielić przez zero”

Zadanie 4.

Napisz metodę analizująca wprowadzony przez użytkownika tekst i drukującą statystyki w następującej formie :
Podany tekst „Tekst" zawiera:
X liter, w tym Y samogłosek oraz Z spółgłosek
R białych znaków
P liczb
S innych znaków
Gdzie P,R,S,X,Y,Z oznaczają liczby całkowite

Zadanie 5.
Utwórz klasę Lokal, która w konstruktorze przyjmuje 2 parametry typu String : nazwa lokalu oraz adres z Google Maps np. „Jana Kilińskiego 4, 80-452 Gdańsk". Zdefiniuj następujące pola klasy:

a) nazwaLokalu:String
b) miejscowość:String
c) kodPocztowy:String
d) ulica:String
e) numerDomu:Integer
f) numberLokalu:Integer

Za pomocą wyrażen regularnych uzupełnij pola b-f posługując się przesłanym adresie w konstruktorze klasy.
Nadpisz metodę toString() tak aby zwracała następującą postać:
{NAZWA LOKALU}
Miasto: {Miasto}
Ulica: {Ulica}
Numer domu/lokalu: {numerDomu/numerLokalu}
Kod pocztowy : {kodPocztowy}

Np.
Mąka i Kawa
Miasto: Gdańsk
Ulica: Jana Kilińskiego
Numer domu/lokalu: 4
Kod pocztowy: 80-452

Zadanie 6.
Utwórz typ enum o nazwie Znak, przyporządkuj mu następujące możliwe wartości :
SPOLGLOSKA, SAMOGLOSKA, SPACJA, LICZBA, ZNAK_SPECJALNY.
Utwórz klasę TransformacjaString z następującymi metodami statycznymi:

a) usunZnaki(String tekst, Znak znak):String – zwraca przesłany tekst pozbawiony wybranego rodzaju znaków np. *usunZnaki(„123Test", Znak.LICZBA)* → *„Test"*
b) podmienZnaki(String tekst, Znak znak, String podmien):String – zwraca przesłany tekst z podmienionymi wybranymi znakami np. *podmienZnaki(„123Test", Znak.LICZBA, „$")* → *„$$$Test"*
c) pozostawZnaki(String tekst, Znak znak):String – usuwa wszystkie znaki oprócz tych zdefiniowanych w argumencie, np. *pozostawZnaki(„123Test", Znak.LICZBA*) → *„123"*

Rozwiązanie zadań powinno być udostępnione w na indywidualnym repozytorium github studenta maksymalnie do końca czwartego dnia po zajęciach:

- zajęcia w poniedziałek -> termin do piątku 23:59:59
- zajęcia w środę -> termin do niedzieli 23:59:59
- zajęcia w czwartek -> termin do poniedziałku 23:59:59

Zadania powinny być zrealizowane w osobnych klasach w ramach jednego projektu, a umieszczony w prywatnym repozytorium, które jest udostępnione prowadzącemu (https://github.com/plucins)