**Programowanie obiektowe w Javie (POJ)**

Laboratorium 12 – Test Wiedzy

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.

**Przygotowanie do realizacji zadań**

1. Utwórz nowy plik typu .doc lub .docx
2. Na górze zapisz swój numer indeksu
3. Rozwiązanie każdego zadania umieść pod punktem ‘Zadanie X.’, gdzie X – oznacza numer realizowanego zadania
4. W przypadku umieszczania kodu w rozwiązaniu można zarówno umieścić zrzut ekranu kodu jak i też kopiować kod z zachowaniem formatowania i czytelności. Nieczytelny kod nie będzie podlegał ocenie.
5. Na koniec przekonwertuj plik do postaci .pdf

**Zadanie 1. (1 pkt)**

Opisz krótko czym różni się interfejs Comparator od interfejsu Comparable

**Zadanie 2. (1 pkt)**

Na podstawie poniższego kodu wymień linijki, które mają niepotrzebne słowa kluczowe w swoich deklaracjach (w sensie, że program zadziała dokładnie tak samo z nimi jak i bez nich)

package pl.pjatk.kolokwium; //1

import java.util.Collection; //2

import java.util.List; //3

import java.util.Set; //4

public interface KlasaAbstrakcyjna { //5

public abstract void foo(); //6

List<String> getList(); //7

abstract Collection<Object> update(Set<String> set); //8

String foo2(); //9

}

**Zadanie 3. (4 pkt)**

Utwórz klasę Osoba z atrybutami:  
imie:String

Nazwisko:String

oraz konstruktorem przyjmującym oba teksty.

Następnie nadpisz metodę toString() dla czytelnego wyświetlenia osoby

*np. {%imie%, %nazwisko%}*

Niech klasa Osoba implementuje interfejs Comparable w taki sposób, aby obiekty klasy Osoba były sortowane po nazwisku w kolejności odwrotnej do porządku alfabetycznego   
(z->a).

Utwórz klasę Main oraz blok PSVM i tam dodaj osoby do listy i wyświetl stan listy przed i po sortowaniu za pomocą wyżej wspomnianego interfejsu.

**Zadanie 4. (4 pkt)**

Napisz metodę, który przyjmie dowolny tekst, a następnie zwróci mapę wystąpień samogłosek w przesłanym tekście. Niech metoda ignoruje wielkość liter. Następnie dodaj blok PSVM i pozwól użytkownikowi na przesłanie własnego tekstu do tej metody. Wynik (mapę) wyświetl za pomocą polecenia SOUT.

*zliczSamogloski(String tekst): Map<String, Integer> -> “Ala ma kota” -> [„A” -> 4, „O” -> 1]*