Laboratorium 4

W tym ćwiczeniu należy zbudować proste aplikacje graficzne, które ilustrują sposób działania kolekcji zawartych w pakiecie „*Java Collections Framework*”. Kolekcje są to obiekty tworzące abstrakcyjne struktury danych, w których możliwe jest gromadzenie innych obiektów. Na kolekcjach można wykonywać podstawowe operacje takie jak dodawanie, usuwanie oraz przeglądanie elementów . Główne cele ćwiczenia to:

- nabycie umiejętności tworzenia aplikacji wykorzystujących graficzny interfejs użytkownika,
- nabycie umiejętności implementacji interfejsów,
- nabycie umiejętności tworzenia i operowania na kolekcjach z pakietu „*Java Collections Framework*”,
- porównanie właściwości kolekcji różnych typów,

Programy przykładowe

Pliki **`PorownajKolekcje_DEMONSTRACJA.jar`** oraz **`PorownajMapy_DEMONSTRACJA.jar`** są programami demonstracyjnymi, które umożliwiają zapamiętywanie obiektów klasy `String` w różnego typu kolekcjach (listy, zbiory, mapy). Programy umożliwiają porównanie właściwości tych kolekcji. Proszę uruchomić programy i zapoznać się z ich działaniem.

Zadanie 0 (nie wymaga wysyłania do oceny)

1. Proszę uruchomić program **`PorownajKolekcje_DEMONSTRACJA.jar`** i szczegółowo zapoznać się z jego działaniem. W czasie testów proszę wprowadzać do kolekcji wiele różnych łańcuchów. Niektóre łańcuchy powinny być dodawane do kolekcji wielokrotnie. Proszę obserwować sposób zapamiętywania danych w kolekcjach różnych typów.
2. Proszę uruchomić program **`PorownajMapy_DEMONSTRACJA.jar`** i szczegółowo zapoznać się z jego działaniem. W czasie testów proszę wprowadzać do kolekcji wiele różnych par łańcuchów (klucz i odpowiadająca mu wartość). Niektóre klucze powinny być dodawane do kolekcji wielokrotnie. Proszę obserwować sposób zapamiętywania danych w kolekcjach różnych typów.
3. Proszę przeanalizować program **`PorownajMapy.java`**. Program jest wersją źródłową programu **`PorownajMapy_DEMONSTRACJA.jar`** i operuje wyłącznie na mapach, w których pamiętane są uporządkowane pary obiektów klasy `String`. Pierwszy obiekt pary stanowi unikalny klucz, który nie może się powtarzać w Mapie. Temu obiektowi jest przyporządkowany drugi obiekt klasy `String`, który może się powtarzać dla różnych kluczy. W oknie dialogowym programu tworzone są pole edycyjne do wprowadzania łańcuchów znaków (klucz i wartość) oraz przyciski umożliwiające dodawanie, usuwanie i wyświetlanie danych zapamiętywanych w kolekcji typu `Map`. Ponadto program tworzy dwa obiekty klasy `widokMapy`. Klasa ta umożliwia utworzenie na panelu otoczonej ramką listy, w której są wyświetlane łańcuchy znaków tworzących pary zapisane w mapie.

Zadanie 1 (obowiązkowe)

1. **Proszę napisać program PorownajKolekcje.java, tak by ten program tworzył i wyświetlał zawartość kolekcji typu** `Vector`, `ArrayList`, `LinkedList`, `HashSet`, `TreeSet`. **W tym celu należy utworzyć klasę** `PorownajKolekcje`, **która będzie rozszerzać klasę** `JFrame` **i implementować interfejs** `ActionListener`. **W tej klasie należy utworzyć obiekty reprezentujące kokekcje** `Vektor`, `ArrayList`, `LinkedList`, `HashSet`, `TreeSet` **np.**
   `Vector<String> vector = new Vector<String>();`
   oraz obiekty reprezentujące elementy graficznego interfejsu użytkownika i dodatkowe obiekty reprezentujące widoki poszczególnych kolekcji.
2. **W konstruktorze należy utworzyć panel klasy** `JPanel`, **na którym zostaną umieszczone wszystkie elementy graficznego interfejsu użytkownika.**
3. **Należy utworzyć metodę** `actionPerformed`, **w której będą wykonywane wszystkie operacje na elementach porównywanych kolekcji.**
   - **Ponadto należy utworzyć metodę** `main()`, **w której będzie utworzony obiekt klasy** `PorownajKolekcje`.
   - **Wyświetlanie kolekcji proszę zrealizować za pomocą pomocniczej klasy** `WidokKolekcji`, **która zawiera obiekt klasy** `JTable` **reprezentujący tabelę o jednej kolumnie w której będą wyświetlane elementy pamiętane w kolekcji oraz obiekt klasy** `DefaultTableModel`, **który będzie umożliwiał sterowanie zawartością tabeli.**
   - **W klasie** `WidokKolekcji` **należy zrobić metodę** `refresh()`, **która będzie odświeżać zawartość tabeli.**

Zadanie 2 (obowiązkowe)

1. Proszę zdefiniować nową klasę `Room`, która będzie reprezentować pokoje w pewnej firmie. Klasa powinna posiadać trzy atrybuty: pierwszy klasy String reprezentujący symbol budynku (np. "C-3"), drugi typu `int` reprezentujący numer pokoju oraz trzeci typu `String` zawierający krótki opis (np. "Laboratorium komputerowe", "Sala wykładowa", itp.). Pierwsze dwa atrybuty jednoznacznie identyfikują obiekt, trzeci ma charakter pomocniczy. Klasa `Room` ma implementować interfejs `Comparable<Room>`. W klasie `Room` proszę przedefiniować następujące metody (które były pierwotnie zdefiniowane w klasie `Object`):
   – `toString` – metoda, która zwraca reprezentację tekstową obiektu `Room` w postaci łańcucha, np. "C-3/125 : Laboratorium komputerowe",
   – `hashCode` – metoda, która zwraca wartość kodu mieszania obiektu `Room` obliczoną na podstawie wartości kodu mieszania obu atrybutów jednoznacznie identyfikujących pokój,
   – `equals` – metoda, która porównuje obiekty klasy `Room` (zwraca wartość `true` dla obiektów reprezentujących pokoje w tym samym budynku o tym samym numerze,
   – `compareTo` (implementacja interfejsu `Comparable`), – metoda, która porównuje naturalny porządek obiektów (np. uporządkowanie alfabetyczne według symboli budynku i numeru pokoju).

2. **Proszę napisać nowy program, który będzie umożliwiał przetestowanie działania różnych kolekcji na obiektach klasy** `Room`. Program powinien mieć graficzny interfejs

użytkownika zbliżony do programu **`PorownajKolekcje.jar`**, ale powinien umożliwiać wprowadzanie oddzielnie symbolu budynku, numeru pokoju oraz opisu pokoju, który ma być dodany lub usunięty z kolekcji. Program powinien porównywać działanie kolekcji utworzonych jako obiekty następujących klas: *`Vector`*, *`ArrayList`*,*`LinkedList`*, *`HashSet`*, *`TreeSet`*.

UWAGA: **W kolekcjach mają być pamiętane obiekty klasy** *`Room`* **np.** *`Vector<Room> vector = new Vector<Room>();`*
a tabelki wyświetlające widoki poszczególnych kolekcji powinny mieć tym razem trzy kolumny.

Zadanie 3 (dla ambitnych)

(Zadanie dla ambitnych !!!)
Proszę napisać program, który wczytuje kolejne słowa z pliku tekstowego i zlicza liczbę wystąpień poszczególnych słów. Do zapamiętywania danych proszę wykorzystać kolekcję typu mapa, w której będą zapisywane pary: (słowo ↔ liczba wystąpień). Po wczytaniu całego pliku program powinien wyświetlić wszystkie słowa uporządkowane alfabetycznie oraz liczby wystąpień tych słów, a następnie wyświetlić wszystkie słowa uporządkowane według liczby wystąpień (od najczęściej występujących do najrzadziej występujących). Program nie musi mieć graficznego interfejsu użytkownika.
Wskazówki:

1) Do wczytywania słów z pliku tekstowego można wykorzystać metodę parę metod *`hasNext()`* i *`next()`* z klasy *`Scanner`*.
2) Jeśli wczytane słowo nie jest jeszcze w tworzonej mapie to należy dodać to słowo do mapy z liczbą 1.
3) Jeśli wczytane słowo jest już w tworzonej mapie, to należy dodać to słowo z liczbą wystąpień powiększoną o 1.
4) Jeśli do implementacji zostanie użyta mapa typu *`TreeMap`* to słowa w tworzonej mapie zostaną uporządkowane alfabetycznie.
5) Żeby móc wypisać słowa uporządkowane według liczby wystąpień, to po zakończeniu wczytywania słów do mapy należy przepisać wszystkie słowa do zwykłej tablicy (lub do kolekcji typu *`Vector`*), a następnie posortować elementy tablicy (lub kolekcji) za pomocą metody *`sort`* z klasy *`Arrays`* (lub metody *`sort`* z klasy *`Collections`*) Uwaga: należy wcześniej zaimplementować komparator, który będzie określał porządek słów na podstawie ich liczby wystąpień odczytanej w wcześniej zbudowanej mapy. UWAGA: Dla słów o tej samej liczbie wystąpień komparator powinien uwzględniać porządek alfabetyczny.