Laboratorium 3

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,

Program przykładowy

Pliki **`KolekcjeDemo.jar`** oraz Mapy **`Demo.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 1

1. Proszę uruchomić program **`KolekcjeDemo.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 przeanalizować program **`Kolekcje.java`**. Program jest uproszczoną wersją programu **`KolekcjeDemo.jar`** i operuje wyłącznie na kolekcji klasy `Vector`. W oknie dialogowym programu tworzone są pole edycyjne do wprowadzania tekstów oraz przyciski umożliwiające zapisywanie, usuwanie i wyświetlanie danych w kolekcji `vector`, która jest obiektem klasy `Vektor`. Ponadto program tworzy obiekt `widokVector,` który jest obiektem klasy `WidokKolekcji`. Klasa ta umożliwia utworzenie na panelu otoczonej ramką listy wyświetlającej elementy zapamiętane w kolekcji.
    - **Proszę rozbudować program `Kolekcje.java`, tak by tworzył i wyświetlał zawartość kolekcji typu `ArrayList`, `LinkedList`, `HashSet`, `TreeSet`.** W celu dodania innego typu kolekcji należy:
    - w klasie `Kolekcje` dodać dodatkowe atrybuty reprezentujące tą kolekcję i jej widok,
    - w konstruktorze utworzyć i obiekt będący widokiem kolekcji i dodać go do panelu,
    - uzupełnić metodę `actionPerformed`, tak by wszystkie operacje były wykonywane na dodanej kolekcji oraz by wszystkie elementy kolekcji były wyświetlane w widoku kolekcji.
3. Proszę uruchomić program **`MapyDemo.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.

4. Proszę przeanalizować program **`Mapy.java`**. Program jest uproszczoną wersją programu **`Mapy.jar`** i operuje wyłącznie na kolekcji klasy *`HashMap`*. W oknie dialogowym programu tworzone są pola edycyjne do wprowadzania tekstów oraz przyciski umożliwiające zapisywanie, usuwanie i wyświetlanie danych. w kolekcji *`hashMap`*, która jest obiektem klasy *`HashMap`*. Ponadto program tworzy obiekt *`widokHashMap,`* który jest obiektem klasy *`WidokMapy`*. Klasa ta umożliwia utworzenie na panelu otoczonej ramką listy wyświetlającej elementy zapamiętane w kolekcji.
5. **Proszę rozbudować program `Mapy.java`, tak by tworzył wyświetlał zawartość kolekcji typu *`TreeMap`*.** W tym celu należy wykonać analogiczne zmiany jak w punkcie nr 3.

Zadanie 2

1. Proszę zdefiniować nową klasę *`Osoba`*, która będzie implementować interfejs *`Comparable`* i będzie zawierać dwa pola typu *`String`* reprezentujące imię i nazwisko osoby. W klasie *`Osoba`* należy przedefiniować następujące metody (które były pierwotnie zdefiniowane w klasie *`Object`*):
   – *`toString`* która zwraca reprezentację tekstową obiektu *`Osoba`* w postaci łańcucha ”Imię Nazwisko”,
   – *`hashCode`* która zwraca wartość kodu mieszania obiektu *`Osoba`* obliczoną na podstawie wartości kodu mieszania atrybutu nazwisko i atrybutu imie,
   – *`equals`* która porównuje obiekty klasy *`Osoba`* (zwraca wartość *`true`* dla obiektów reprezentujących osoby o tym samym nazwisku i imieniu).

   oraz zdefiniować metodę *`compareTo`* (implementacja interfejsu *`Comparable`*), która porównuje naturalny porządek obiektów (np. uporządkowanie alfabetyczne osób według nazwiska i imienia).
2. **Proszę napisać nowy program, który będzie umożliwiał przetestowanie działania różnych kolekcji na obiektach klasy *`Osoba`***. Program powinien mieć graficzny interfejs użytkownika zbliżony do programu **`KolekcjeDemo.jar`**, ale powinien umożliwiać wprowadzanie oddzielnie imienia i nazwiska osoby, która ma być dodana lub usunięta z kolekcji.

Zadanie 3 (dla ambitnych)

(Zadanie dla ambitnych !!!)
Proszę napisać własny program z graficznym interfejsem użytkownika, który będzie tworzył dwa zbiory A i B, w których będą zapamiętywane nazwy jakiś „skarbów”. Program powinien umożliwiać:

- dodanie skarbu do zbioru A,
- dodanie skarbu do zbioru B,
- usunięcie skarbu ze zbioru A,
- usunięcie skarbu ze zbioru B.

Ponadto program powinien wyliczać i wyświetlać sumę, iloczyn, różnicę oraz różnicę symetryczną zbiorów A i B.