Laboratorium 1

Kolekcje

Studenci w ramach zajęć laboratoryjnych zapoznają się z podstawowymi operacjami na kolekcjach danych oraz mechanizmami porównywania i sortowania obiektów.

Zadanie powinno być realizowane jako projekt oparty na narzędziu Apache Maven skonfigurowanym tak aby wykorzystywał platformę Java w wersji 11 lub wyższej. Należy zwrócić uwagę na poprawną identyfikację projektu oraz pakiet wykorzystane w aplikacji.

Należy zaimplementować aplikację w trybie tekstowym pozwalającą na wczytanie rekurencyjnego zbioru obiektów, a następnie wyświetlenie go z wykorzystaniem sortowania lub bez. Aplikacja powinna udostępniać możliwość sortowania na podstawie naturalnego porządku jak i dodatkowego kryterium. Specyfikację obiektu, kryteria sortowania oraz pożądany format wyjścia podaje prowadzący na zajęciach.

Należy zrealizować następujące zadania:

1. Klasa modelowa przedstawiającą strukturę rekurencyjną. Klasa powinna zawierać przynajmniej 3 pola o różnych typach prostych (lub typie `String`). Specyfikację klasy podaje prowadzący (przykład znajduje się na końcu instrukcji). Klasa powinna zawierać zbiór (instancję klasy `Set`) obiektów tego samego typu. Należy zadbać o poprawne porównywanie obiektów (implementacje metod `equals` i `hashCode`), sortowanie zgodnie z naturalnym porządkiem (implementacja interfejsu `Comparable`) oraz reprezentację tekstową (implementacja metody `toString`). Pola klasy, naturalny porządek oraz reprezentację tekstową definiuje prowadzący. (1 pkt)
2. Alternatywne kryterium sortowania poprzez implementację interfejsu `Comparator`. Specyfikację kryterium podaje prowadzący. (1 pkt)
3. Tworzenie zbioru testowego. Tworzenie kolejnych elementów zbioru może być zaimplementowane na sztywno w kodzie. Jedynym parametrem przekazywanym jako parametr startowy aplikacji (parametr metody `main`) jest informacja o sposobie sortowania elementów. Wymagane tryby to: brak sortownia, sortowanie zgodnie z naturalnym porządkiem, sortowanie zgodnie z alternatywnym kryterium. W zależności od parametru należy wybrać odpowiednią implementację interfejsu `Set`: `HashSet` dla braku sortownia i `TreeSet` dla sortowania. W przypadku alternatywnego kryterium należy przekazać instancję komparatora do konstruktora obiektu zbioru. Zbiór testowy powinien zawierać przynajmniej 10 elementów i 3 poziomy. (1 pkt)
4. Wypisywanie zbioru. Wszystkie elementy zbioru należy wypisać na standardowe wyjście zachowując strukturę rekurencyjną. Sposób przedstawienia struktury podaje prowadzący (przykład na końcu instrukcji). (1 pkt)

5. Statystyki liczby potomków. Generowanie statystyki powinno być zrealizowane w postaci metody zwracającej mapę odwzorowująca element ze zbioru na liczbę wszystkich potomków (obiektów nie tylko zawierających się bezpośrednio w nim ale i obiektów znajdujących się na kolejnych poziomach struktury). Po wygenerowaniu odpowiedniego obiektu mapy należy wypisać jego zawartość na standardowe wyjście. Dobór implementacji interfejsu `Map` zależy od parametru programu, `HashMap` dla braku sortowania i `TreeMap` dla sortowania. (1 pkt)

Przykład:

Klasa modelowa:

```
public class Mage {

    private String name;

    private int level;

    private double power;

    private Set<Mage> apprentices;

}
```

Naturalny porządek:

```
name, level, power
```

Alternatywne kryterium:

```
level, name, power
```

Reprezentacja tekstowa:

```
Mage{name='', level=, power=}
```

Reprezentacja struktury rekurencyjnej:

```
-Mage{name='', level=, power=}
--Mage{name='', level=, power=}
---Mage{name='', level=, power=}
--Mage{name='', level=, power=}
```