Ćwiczenie 9
Typy generyczne. Kolekcje

1 Wprowadzenie

1.1 Typy generyczne

- Klasy (typy) generyczne są to klasy o parametryzowanych typach danych.
- Klasy generyczne posiadają kompletną implementację, jednak nie definiują typów danych wykorzystanych w tej implementacji.
- Od wersji Java 1.5, kolekcje są klasami generycznymi (tzw. kolekcje uogólnione).

1.1.1 Kolekcje

- Kolekcja - obiekt grupujący elementy danych (inne obiekty), pozwalający traktować je jako jeden zestaw danych oraz umożliwiający wykonywanie operacji na zestawie danych (np. dodawanie, usuwanie).
- **Java Collections Framework** - zestaw bibliotek zawierających gotowe do wykorzystania kolekcje (abstrakcyjne struktury danych).
- Interfejsy i klasy służące do tworzenia i posługiwania się różnego rodzaju kolekcjami zdefiniowane są w pakiecie **java.util**.
- Rodzaje kolekcji:
  - listy,
  - zbiory,
  - mapy (tablice asocjacyjne).

1.1.2 Listy

- Lista – zestaw elementów, z których każdy znajduje się na określonej pozycji.
- Implementowany interfejs: **List**.
- Przykładowe implementacje (klasy):

```
ArrayList<E>
LinkedList<E>
```

1.1.3 Zbiory

- Zbiór - zestaw niepowtarzających się elementów.
- Implementowany interfejs: **Set**.
- Przykładowe implementacje (klasy):

```
HashSet<E>
TreeSet<E>
LinkedHashSet<E>
```

1.1.4 Mapy

- Mapa - zestaw par "klucz-wartość", przy czym odwzorowanie kluczy w wartości jest jednoznaczne.
- Implementowany nterfejs: **Map**.
- Przykładowe implementacje (klasy):

```
HashMap<K,V>
TreeMap<K,V>
LinkedHashMap<K,V>
```

1.1.5 Interfejs Collection

- Interfejs **Collection** definiuje wspólne metody dla list i zbiorów. Interfejsami pochodnymi są interfejsy **List** i **Set**.
- Wybrane metody interfejsu **Collection**:

Metoda	Opis
add(E)	dodawanie obiektu do kolekcji
boolean remove(E)	usuwanie obiektu z kolekcji
boolean contains(E)	sprawdzanie czy obiekt znajduje się w kolekcji
int size()	określenie liczby obiektów znajdujących się w kolekcji
boolean isEmpty()	sprawdzanie czy kolekcja jest pusta
void clear()	usuwanie wszystkich obiektów z kolekcji
Iterator<E> iterator()	pobranie iteratora dla kolekcji

1.1.6 Interfejs Map

- Interfejs **Map** definiuje wspólne metody dla map.
- Wybrane metody interfejsu **Map**:

Metoda	Opis
V put(K,V)	dodawanie pary wartość-klucz do mapy
V get(K)	pobranie wartości dla podanego klucza
V remove(K)	usunięcie wartości dla podanego klucza
int size()	określenie liczby par wartość-klucz znajdujących się w mapie
boolean isEmpty()	sprawdzanie czy mapa jest pusta
void clear()	usuwanie wszystkich par klucz-wartość z mapy
boolean containsKey(K)	sprawdzenie czy mapa zawiera podany klucz
boolean containsValue(V)	sprawdzenie czy mapa zawiera podaną wartość
Set<K> keySet()	pobranie zbioru wszystkich występujących kluczy w mapie

1.1.7 Iteratory

- Iterator - obiekt klasy implementującej interfejs **Iterator**, służący do przeglądania kolekcji.
- Metody interfejsu **Iterator**:

Metoda	Opis
boolean hasNext()	sprawdzenie czy istnieje następny obiekt w kolekcji
Object next()	pobranie następnego obiektu z kolekcji
void remove()	usunięcie z kolekcji ostatniego obiektu zwróconego przez iterator

1.1.8 Algorytmy kolekcyjne

- Klasa **Collections** dostarcza statyczne metody operujące na kolekcjach lub zwracające kolekcje.
- Wybrane algorytmy kolekcyjne:

Metoda	Opis
int binarySearch(List<E>,E)	wyszukiwanie binarne na liście
int binarySearch(List<E>,E,Comparator<E>)	wyszukiwanie binarne na liście wg. podanego komparatora
void sort(List<E>)	sortowanie listy (w porządku niemalejącym)
void sort(List<E>,Comparator<E>)	sortowanie listy wg. podanego komparatora
E max(Collection<E>,	wyszukiwanie elementu maksymalnego w kolekcji
E max(Collection<E>,Comparator<E>)	wyszukiwanie elementu maksymalnego w kolekcji wg. podanego komparatora
E min(Collection<E>)	wyszukiwanie elementu minimalnego w kolekcji
E min(Collection<E>,Comparator<E>)	wyszukiwanie elementu minimalnego w kolekcji wg. podanego komparatora

1.1.9 Komparatory

- Komparator jest obiektem służącym do porównywania innych obiektów.
- Klasa komparatora implementuje interfejs **Comparator**.
- Metody interfejsu **Comparator**:

Metoda	Opis
int compare(E,E)	Porównanie dwóch obiektów. Wynik porównania zwracany jest jako liczba całkowita: -1 gdy obiekt pierwszy jest mniejszy, 0 gdy oba obiekty są równe, 1 gdy obiekt pierwszy jest większy.

2 Zadania

2.1

Stwórz aplikację pozwalającą przechowywać na liście informacje o osobach (imię, nazwisko, wiek, wzrost, waga). Aplikacja powinna ponadto umożliwiać:

- wyszukiwanie osób według różnych kryteriów: nazwiska, wieku, wzrostu, wagi.
- wyszukiwanie osoby najstarszej, najwyższej, najcięższej, najmłodszej, najmniejszej, najlżejszej.
- sortowanie osób według różnych kryteriów: wieku, wzrostu, wagi.

2.2

Stwórz aplikację pozwalającą przechowywać na liście informacje o wektorach w przestrzeni dwuwymiarowej (współrzędna x, współrzędna y). Aplikacja powinna ponadto umożliwiać sortowanie wektorów względem ich długości oraz wyszukiwanie wektorów najdłuższego i najkrótszego.

2.3

Zaproponuj odpowiednią kolekcję pozwalającą przechowywać informację o stawkach wynagrodzenia dla poszczególnych osób identyfikowanych przez nazwisko. Następnie stwórz aplikację, która umożliwia:

- wyświetlanie informacji o stawce dla każdej osoby (do tego celu wykorzystaj iterator odpowiedniej kolekcji),
- wyświetlanie informacji o wszystkich występujących stawkach (do tego celu wykorzystaj odpowiednią kolekcję),
- znajdowanie stawki przyporządkowanej danemu nazwisku.

2.4

Zaproponuj odpowiednią kolekcję pozwalającą przechowywać informację o liczbie dni dla poszczególnych miesięcy identyfikowanych przez nazwę. Następnie stwórz aplikację, która umożliwia:

- wyświetlanie informacji o liczbie dni dla każdego miesiąca (do tego celu wykorzystaj iterator odpowiedniej kolekcji),
- znajdowanie liczby dni dla podanego miesiąca.