PROGRAMOWANIE OBIEKTOWE JAVA – LABORATORIUM

MAPY

MAPY przechowują pary klucz - wartość w postaci obiektów o nazwie entry. Zwyczajowo mówimy zatem, że mapa jest kolekcją entries, gdzie każde entry składa się z pary:

```
Klucz - Wartość (Key - Value)
```

Przykład dla tworzenia mapy filmów w wypożyczalni, ustawiamy klucz w postaci nr identyfikujących, a wartością są nazwy filmów.

849, Avatar 123,

Szeregowiec Ryan

543, Mission Impossible

CECHY MAP:

- Nie są dozwolone duplikaty kluczy.
- Wartości mogą się duplikować.
- Mogą być sortowalne lub nie - zależy od konkretnej implementacji interfejsu.
- Klucze mogą być null-ami lub nie - zależy od konkretnej implementacji interfejsu (TreeMapy nie pozwalają na nulle).

IMPLEMENTACJE MAP:

- HashMap z pakietu java.util - Bardzo często stosowana implementacja. Elementy są nieposortowane. Ich kolejność nie odpowiada również kolejności wkładania do zbioru. Może przechowywać jednego null-a wśród kluczy.
- LinkedHashMap z pakietu java.util - Implementacja przechowująca elementy w kolejności ich dodawania. Rozszerza klasę HashMap. Zatem może być przydatna jeśli zależy nam zarówno na unikalności kluczy jak i na tworzeniu historii unikalnych wpisów. Może przechowywać jednego null-a wśród kluczy.
- TreeMap z pakietu java.util - Nie pozwala na przechowywanie nulla w miejscu klucza. Elementy są przechowywane pod postacią drzewa. Elementy są poukładane według kluczy w sposób posortowany (rosnąco). Przydaje się gdy chcemy zapewnić unikalność elementów oraz podstawowe sortowanie.

TWORZENIE MAPY

```
Map<Object, Object> mapOfAnything = new HashMap<Object,Object>();
Map<Integer, String> linkedWordsWithIds = new
LinkedHashMap<Integer,String>();
Map<Integer, String> sortedWordsWithIds = new TreeMap<Integer,String>();
```

PODSTAWOWE OPERACJE NA MAPACH

Podstawowymi operacjami jakie możemy wykonywać na mapach jest dodawanie, pobieranie oraz usuwanie elementów. Operacje te są realizowane przez następujące metody:

- put(<obiekt>, <obiekt>) Umożliwia ona dodanie elementu do mapy. Co ważne - wymagane jest dodanie elementów tych samych typów (lub podtypów) co parametry typów zadeklarowanych przez mapę. Metoda ta umożliwia również aktualizację obiektu w mapie. Wprowadzając do mapy parę o tym samym kluczu, ale innej wartości, podmieniamy de facto tę wartość.

- get(<obiekt>) - Metoda pobiera element z mapy poprzez podanie wybranego obiektu klucza. Poniższy zapis zwróci obiekt z tekstem "Joker".
- remove(<obiekt>) - Metoda usuwa element z mapy poprzez podanie wybranego obiektu klucza. Poniższy zapis usunie film "Psy 3", który wyżej dodaliśmy z kluczem 3

PRZEGLĄDANIE ZAWARTOŚCI MAPY

Mapy nie implementują interfejsu Iterable, ale i tak można je w łatwy sposób przeglądać za pomocą iteratora. Pobranie bieżącego elementu i przejście do następnego wykonywane jest za pomocą metody next. W przypadku map - aby użyć iteratora - najpierw pobieramy z mapy zbiór entries. Wszystkie entries tworzą zbiór (czyli znany Wam już Set), a jak wiemy zbiory możemy iterować i właśnie tak można przeglądać mapy:

```
public static void przegladaj(){
    Map<Integer, String> movies = new HashMap<Integer, String>();
    movies.put(1, "Joker");
    movies.put(2, "Jurassic World");
    movies.put(3, "Psy 3");

    Set<Map.Entry<Integer, String>> entries = movies.entrySet();
    Iterator<Map.Entry<Integer, String>> moviesIterator =
entries.iterator();

    while(moviesIterator.hasNext()) {
        Map.Entry<Integer, String> entry = moviesIterator.next();
        System.out.println(entry.getKey());
        System.out.println(entry.getValue());
    }
}
```

Zadania do samodzielnego rozwiązania:

Zadanie 1.

1. Utwórz nową klasę Kangur ze składową int nrKangura, inicjalizowaną z poziomu konstruktora. Wyposaż klasę w metodę skok(), wypisującą wartość tej składowej i sygnalizującej wykonywanie podskoków. Utwórz kontener ArrayList i wstaw do niego obiekty Kangur (minimum 10). Teraz skorzystaj z metody get() kontenera w celu przejrzenia jego zawartości i wywołania metody skok() dla każdego umieszczonego w nim kangura.
2. Zmodyfikuj uzyskany kod tak, aby przeglądało listę (i wywoływało metodę skok()) za pomocą iteratora.
3. Weź klasę Kangur z podpunktu 1 i umieść jej elementy w kontenerze HashMap, kojarząc każdy egzemplarz Kangur (wartość) z nazwą ("Jacek", "Marta" itd.) w postaci obiektu String (klucz). Pozyskaj iterator zbioru zwracanego przez keySet() i wykorzystaj go do przejrzenia kontenera HashMap. Wypisz w konsoli imiona kangurów oraz odpowiadające im numery, zwracane poprzez metodę skok().
4. Wyodrębnij z kontenera HashMap (utworzonego w poprzednim podpunkcie) pary, posortuj je według kluczy i umieść całość w kontenerze LinkedHashMap.

Zadanie 2.

Napisz klasę o nazwie Command, która zawiera ciąg znaków String i metodę operation(), która go wypisuje. Napisz drugą klasę, z metodą wypełniającą kolejkę Queue obiektami klasy Command i zwracającą wypełniony kontener. Przekaż kontener do metody z trzeciej klasy: metoda ma konsumować obiekty z kolejki Queue, wywołując dla każdego z nich metodę operation().

Zadanie 3.

W poniższym zadaniu napiszemy własną klasę odnośnie stosu, czyli co zostanie włożone na stos jako ostatnie jest pierwszym elementem, który można z niego zdjąć (LIFO).

1. Utwórz klasę Stos<T> zawierającą:
   - zainicjowane prywatne pole LinkedList<T> stos,
   - publiczne metody:
   - void push(T v) - wkłada element na stos,
   - T peek() - zwraca pierwszy element stosu, ale go nie usuwa,
   - T pop() - zwraca pierwszy element stosu i usuwa go,
   - boolean empty() - sprawdza, czy stos jest pusty,
   - String toString() - wypisuje elementy naszego stosu.
2. Stosy są często używane do obliczania wyrażeń w językach programowania. Za pomocą utworzonej klasy Stos oblicz poniższe wyrażenie, w którym '+' oznacza "umieszczenie następnej litery na stosie", a '-' "zdjęcie szczytowego elementu stosu i wypisanie go na wyjściu".

Wyrażenie do wyliczenia: " +B+a+ł---+a+g+a---+n-+w-+l+i+t---+e-+r+k--+a+c+h---"

Zadanie 4.

Napisz prostą symulację pójścia na zakupy:

- utwórz tablicę String zawierającą 10 nazw produktów dostępnych w sklepie,
- po wejściu do sklepu zapełnij kosz (będący stosem - użyj klasy z poprzedniego zadania) losową ilością losowych towarów (koszyk pomieści maksymalnie 15 przedmiotów),
- umieść siebie w kolejce (Queue) na losową pozycję (maksymalna ilość osób w kolejce wynosi 10),
- gdy dojdziesz do kasy wypisz produkty znajdujące się w koszyku.

Zadanie 5.

1. Napisz klasę Film zawierającą prywatne pole czasTrwania, tytul oraz czyObejrzany (wartość true gdy film został obejrzany). Dodaj konstruktor i odpowiednie metody obsługujące pola.
2. Stwórz dwie klasy dziedziczące z klasy Film, np. Horror i Komedia. Każda z nich ma zawierać pole finalne pole typ (wskazujące na typ filmu). Dodaj konstruktory i odpowiednie metody obsługujące pola. Dodaj metody wypisujące wszystkie dane odnośnie filmu.
3. W głównej klasie stwórz mapę prywatnaKolekcja, gdzie kluczem będzie numer egzemplarza, a wartością obiekt Film. Dodaj kilka elementów do utworzonej mapy.
4. Za pomocą iteratora wypisz posiadane filmy w kolekcji.

Przykład:

"Nr (nr w kolekcji), Tytul: (tytul filmu), Czas trwania: (czas), Obejrzany: (tak/nie), Typ: (gatunek filmu)"