Zadanie 13

Zapoznaj się ze schematem klas kontenerów Javy. Znajdź w dokumentacji interfejsy **List**, **Set**, **Map**. Znajdź metody służące do wstawiania i usuwania elementów.

```
*pliki => Kolekcje-ściągawka.pdf,  Kolekcje-wykład.pdf
*dokumentacja => http://docs.oracle.com/javase/7/docs/api/index.html
```

Zadanie 14

Przyjrzyj się przykładom wypełnienia i wyświetlenia zawartości kontenerów w **PrintingContainers.java**. Stwórz własne wersje metody wypełniającej **fill()** dla **Map** i **Collection**. Wypełnij kontenery postaciami z kreskówek. Spróbuj wstawić do kontenera obiekt innego typu niż **String** i zaobserwuj komunikaty kompilatora.

Zadanie 15

Napisz nową klasę o nazwie **Gerbil** (gryzoń myszopodobny):

- dodaj pole typu **int gerbil_numer**
- dodaj metodę o nazwie **hop()**, wypisującą numer gryzonia i to, że podskakuje.
- stwórz **ArrayList** i wypełnij tę listę kilkoma obiektami klasy **Gerbil**.
- wykorzystaj metodę get() do poruszania się po liście i wywołaj hop() dla każdego z obiektów. Wykorzystaj nową składnię pętli for dla kolekcji.

```
for (deklaracja zmiennej pętli : kolekcja){
        //...
}
```

Zadanie 16

Zmień kod w zadaniu 15 tak, aby podczas wywoływania **hop()** wykorzystać iterator do poruszania się po liście.

Uwaga: Iteratory pozyskujemy metodą **iterator()** z interfejsu **Collection**. Typowe użycie iteratora jest następujące:

- mamy kolekcję (np. **List**, **Set**) przechowującą obiekty typu **Wartosc**,
- najpierw pozyskujemy iterator, a następnie używamy go do przeglądania elementów:

```
Iterator<Wartosc> e = kolekcja.iterator();
while(e.hasNext()) {
        Wartosci t = e.next() ;
        //...
}
```

lub

```
for (Iterator<Wartosc> i = kolekcja.iterator(); i.hasNtext(); ) {
        Wartosc w = i.next();
        //...
 }
```

Zadanie 17

Klasę **Gerbil** zamieść tym razem w odwzorowaniu **Map**, kojarząc imię gryzonia jako **String** (klucz) dla każdego obiektu **Gerbil** (wartość). Przeglądaj odwzorowanie dla każdego klucza (klucze pobierz metodą **keySet()**, następnie użyj nowej składni pętli **for** do pobierania elemetów zbioru) i dla każdej wartości (metoda **get(klucz)**) **Gerbil** wywołaj **hop().**

Zadanie 18

Modyfikacja zadania 17. Pobierz iterator dla **keySet()** i zastosuj go do poruszania się po odwzorowaniu, przeglądając obiekt **Gerbil** dla każdego klucza i wypisując go przy użyciu **hop()**.

Zadanie 19

Zmień kod klasy **Hamster** z przykładu **HamsterMaze.java** w ten sposób, aby wartość pola **hamsterNumber** była inicjalizowana losową wartością z przedziału [0,10]. Utwórz kolekcję typu **List** 20 obiektów klasy **Hamster**. Wypisz kolekcję na ekran. Celem jest posortowanie listy względem pola **hamsterNumber**.

1. Zaimplementuj w klasie **Hamster** interfejs **Comparable<Hamster>**. Posortuj listę za pomocą metody **sort(List<Hamster>)** z klasy usługowej **Collections** i sprawdź rezultat posortowania.

2. Stwórz oddzielną klasę implementującą interfejs **Comparator<Hamster>**. Posortuj listę za pomocą metody **sort(List<Hamster>, Comparator<Hamster>)** z klasy usługowej **Collections** i sprawdź rezultat posortowania.

Zadania zaczerpnięte z książki: Bruce Eckel "Thinking in Java".