す（リスト 3-7 と実行結果を参照）。

リスト 3-7　Main.java

```
1  import java.util.*;
2
3  public class Main {
4    public static void main(String[] args) {
5      Map<String, Integer> prefs = new HashMap<String, Integer>();
6      prefs.put("京都府", 255);
7      prefs.put("東京都", 1261);
8      prefs.put("熊本県", 181);
9      int tokyo = prefs.get("東京都");
10     System.out.println("東京都の人口は、" + tokyo);
11     prefs.remove("京都府");
12     prefs.put("熊本県", 182);
13     int kumamoto = prefs.get("熊本県");
14     System.out.println("熊本県の人口は、" + kumamoto);
15   }
16 }
```

ペアで値を格納

キーを指定し値を取得

値（182）を上書き

実行結果

```
東京都の人口は、1261
熊本県の人口は、182
```

値が上書きされた

3.5.3　HashMap の中身を 1 つずつ取り出す

HashMap の中の情報を拡張 for 文で取り出そうとしたんですけど、うまくいかなくって…。

マップの中身を取り出すには、ちょっとしたコツが必要なんだよ。

HashMap に格納されたペアを順に取り出す場合、List や Set と同様の方法で取り出そうとするとエラーが出てしまいます。

```
Map<String, Integer> prefs = new HashMap<String, Integer>();
for(String pref : prefs){ … }
```

構文エラー

マップに格納されたデータを順に取り出す方法はいくつかありますが、「キーの一覧を取得 → 各キーについて、対応する値を取得」の 2 段階の手続きを踏む方法が比較的理解しやすいでしょう（リスト 3-8 と実行結果を参照）。

マップに格納された情報を 1 つずつ取り出す

```
for( キーの型 key : マップ変数 .keySet()){
  値の型 value = マップ変数 .get(key);
  /* key と value を用いて何らかの処理を行う */
}
```

リスト 3-8　Main.java

```
1  import java.util.*
2
3  public class Main {
4    public static void main(String[] args) {
5      Map<String, Integer> prefs = new HashMap<String, Integer>();
6      prefs.put("京都府", 255);
```

```
7       prefs.put("東京都", 1261);
8       prefs.put("熊本県", 182);
9       for(String key : prefs.keySet()) {          ← 県名一覧を取得し繰り返す
10        int value = prefs.get(key);               ← 県名を指定し人口を取得
11        System.out.println(key + "の人口は、" + value);
12      }
13    }
14  }
```

実行結果

```
東京都の人口は、1261
京都府の人口は、255
熊本県の人口は、182
```

なお、HashMap は格納したペア同士の順序保証をしないコレクションなので、毎回同じ順序で取り出せるとは限りません。よって、格納順に取り出したい場合は LinkedHashMap を、自然順序で取り出したい場合は TreeMap を利用してください。

Collection インタフェース

3.2 ～ 3.3 節で学んだ List 関連と 3.4 節で学んだ Set 関連のクラスは、共通の親インタフェースとして java.util.Collection を持っています。このインタフェースは「(重複しているか否かを問わず) 何らかの単独データの集まり」を表す、とても抽象度が高いインタフェースです。

3.5 節で紹介した Map は、「ペアデータの集まり」であるため、Collection インタフェースとは継承関係にありません。

3.6 コレクションの応用

3.6.1 コレクションのネスト

これまで、List、Set、Map に関するさまざまなコレクションクラスを学びました。これらのコレクションクラスは、要素として別のコレクションを格納することも可能です。

たとえば、「都道府県別特産物ランキング」を考えてみましょう。各都道府県に対して、順位付けされた複数の特産物データを保持しなければなりません。結果として、以下のような Map の中に List がネストした構造になるでしょう (図 3-18)。

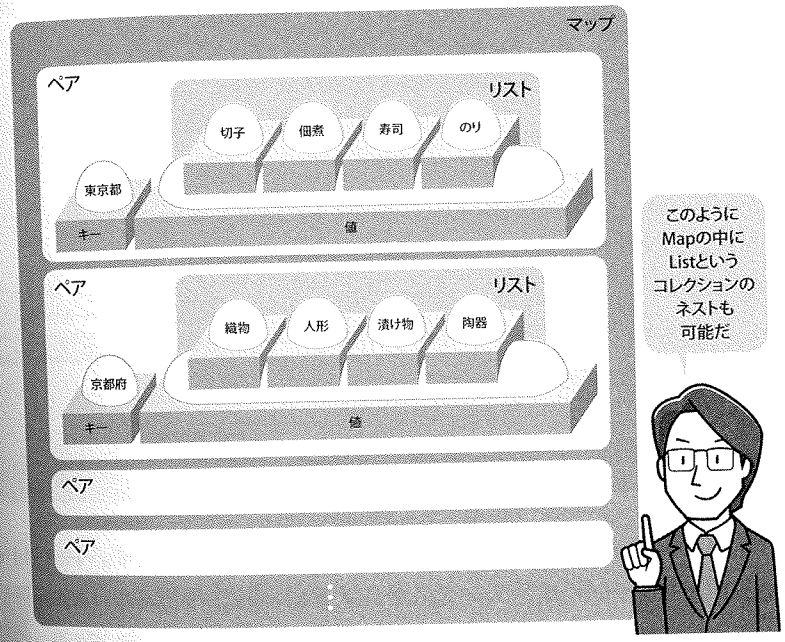

図 3-18　ネストしたコレクション

図 3-18 の例は、「文字列」と「文字列リスト」のペアを格納するので、Map<String, List<String>> という型を使います。ほかにも、Map の中に Map を入れたり、List の中に Map といった構造や、3 重以上のネストも可能です。

3.6.2 要素の参照に注意

コレクションクラスは、さまざまなインスタンスを格納することができるとても便利なクラスですが、いくつか注意すべき点があります。まずは最初の落とし穴を理解するために、次のリスト 3-9 を見てください。

リスト 3-9　Main.java

```
1  import java.util.*;
2
3  class Hero {
4    public String name;
5  }
6
7  public class Main {
8    public static void main(String[] args) {
9      Hero h = new Hero();
10     h.name = "ミナト";
11     List<Hero> list = new ArrayList<Hero>();
12     list.add(h);
13     h.name = "スガワラ";
14     System.out.println(list.get(0).name);
15   }
16 }
```

（11行目：h をリストに格納　／　13行目：格納後に h を書き換え　／　14行目：中身は…？）

このコードのポイントは 13 行目の「h.name=" スガワラ ";」です。12 行目でミナトの名前を持つ Hero インスタンスがリストに格納された後に、13 行目でリストの中身（list.get(0)）ではなく、h を書き換えてしまっています。

手遅れですよ。リストには僕の名前で入っちゃった後ですから。

そうそう。「list.get(0).name = " スガワラ ";」にしないとダメでしょ。

ところが、実際にこのコードを動かすと予想外の実行結果になるんだ。

実行結果

```
スガワラ
```

この不思議な現象を理解するためには、**変数 h やリストに格納された値が実際には参照である**ということを思い出す必要があります。

9 行目で Hero インスタンスが生まれた時点で、そのメモリ上の番地（仮に 1234 番地とします）が h に代入されています。12 行目で h をリストに格納していますが、リストに格納されるのは、この変数 h の 1234 というアドレス情報です。よって、list.get(0) と h はまったく同一のインスタンスを指す 2 つの変数になります。したがって、list.get(0).name と h.name は同一のものですから、13 行目で h.name を書き換えれば当然リスト内の要素も書き換わってしまうのです。

このように「**コレクションへ格納が終わった変数のインスタンスの中身を書き換えると、コレクションに格納済みの要素の中身も書き換わってしまう**」という現象が起こります。同様に、get() で別変数に取り出したインスタンスに対する内容変更も、格納中のインスタンスに影響を与えてしまいます。このように、思わぬ不具合の原因となることがあるため、格納済み要素を参照する際には注意してください。

格納しているのは参照

格納前や取得後の変数のインスタンスを書き換えると、コレクションに格納中の要素も書き換わるおそれがある。

3.6.3 equals() と hashCode() のオーバーライド

Hero のように自分で作るクラスをコレクションに格納する場合、とても重要なことがあるんだ。

Integer や String そして Date のような API が提供するクラスではなく、自分が開発したクラスのインスタンスをコレクションに格納する場合には特別な注意が必要です。**実は equals() や hashCode() を正しくオーバーライドしていないと、コレクションは正しく動作せず、原因の特定が困難な不具合につながることがあるの**です。詳細については第 4 章で解説します。

古参のコレクションクラス

この章で紹介した ArrayList や HashMap といったクラスたちは、Java が誕生した頃にはまだありませんでした。当時からあった「古参のコレクションクラス」として、Vector (ArrayList の原型) や Hashtable (HashMap の原型) というクラスが今も残っていますが、性能などの面で劣るため、特別な理由がない限り利用しないようにしましょう。

Collections クラスと Arrays クラス

JavaAPI には、コレクションや配列を便利に利用するための命令を集めたクラスが用意されています。

- java.util.Collections：**コレクション操作関連の便利なメソッド集**
- java.util.Arrays：**配列操作関連の便利なメソッド集**

両クラスに備わるすべてのメソッドは static ですので、インスタンス化することなく利用できます。第 4 章の 4.5 節で Collections.sort() を紹介しますが、ほかにも興味がある方は API リファレンスで調べてみてください。

3.7 この章のまとめ

コレクション

- おもなコレクションとして List、Set、Map の 3 種類がある。
- コレクションの容量(格納できる要素の数)は、必要に応じて自動的に増加する。
- 使われ方に応じて適切なコレクションクラスを選択する必要がある(図 3-19)。

リスト、セットおよび マップ

- 重複を許し順序性があるデータの格納には List を用いる。
- 重複を許さず順序性がないデータの格納には Set を用いる。
- イテレータを用いることで要素を 1 つずつ取り出すことができる。
- 2 つのインスタンスをペアにして格納するには Map を用いる。
- Map では値の重複は許されるが、キーの重複は許されない。

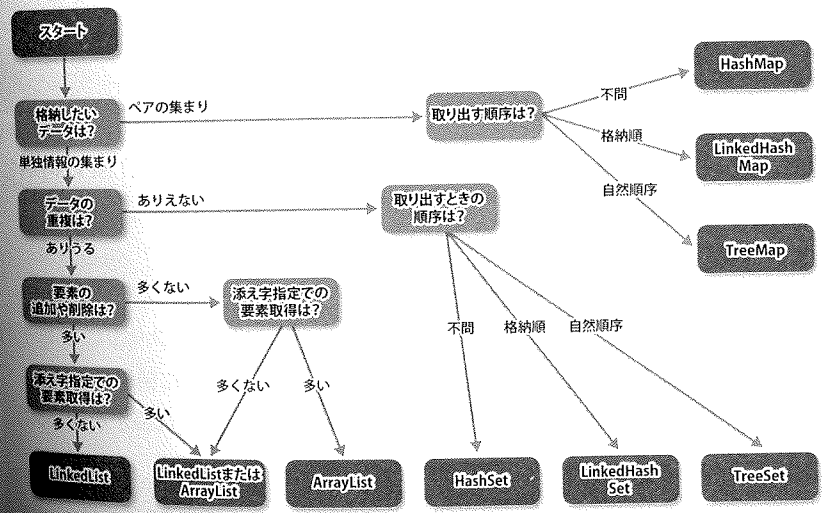

図 3-19　利用すべきコレクションの選び方

3.8 練習問題

練習 3-1

次の各情報を格納する適切なコレクションを List、Set、Map の中から選んでください。

(1)日本の 47 都道府県の名前(順序は不問)
(2)5 人の生徒のテストの点数
(3)過去の総理大臣の名前と任期(順序は不問)

練習 3-2

次のような Hero クラスがあります。

Hero.java

```
1  public class Hero {
2    private String name;
3    public Hero(String name) { this.name = name; }
4    public String getName() { return this.name; }
5  }
```

勇者 2 名(斎藤、鈴木)を Hero としてインスタンス化して ArrayList に格納し、1 つずつ順番に取り出して名前を表示するプログラムを作成してください。

練習 3-3

上の Hero クラスを元に勇者 2 名(斎藤、鈴木)をインスタンス化し、それぞれの勇者が倒した敵の数(3、7)と勇者をペアでコレクションに格納してください。次に 1 つずつ取り出し、次のような画面表示を行ってください(表示順は不問)。

```
斎藤が倒した敵＝3
鈴木が倒した敵＝7
```


3.9 練習問題の解答

練習 3-1 の解答

(1)Set　(2)List　(3)Map

練習 3-2 の解答

(java.util.* を import 済みとします。また、main メソッド内のみ記述します)

```
1  Hero h1 = new Hero("斎藤");
2  Hero h2 = new Hero("鈴木");
3  List<Hero> heroes = new ArrayList<Hero>();
4  heroes.add(h1);
5  heroes.add(h2);
6  for(Hero h : heroes) {
7    System.out.println(h.getName());
8  }
```

練習 3-3 の解答

(java.util.* を import 済みとします。また、main メソッド内のみ記述します)

```
1  Hero h1 = new Hero("斎藤");
2  Hero h2 = new Hero("鈴木");
3  Map<Hero, Integer> heroes = new HashMap<Hero, Integer>();
4  heroes.put(h1, 3);
5  heroes.put(h2, 7);
6  for(Hero key : heroes.keySet()) {
7    int value = heroes.get(key);
8    System.out.println(key.getName() + "が倒した敵=" + value);
9  }
```