基礎程式設計（Java）　講義

泛型集合：List、Set、Map 一次到位

Deitel & Deitel《Java How to Program: Early Objects》11e, Ch.16

§ 1　一、名詞速查表

中文	English	一句話定義	科目	章節
集合框架	collections framework	Java 內建一整套裝資料的介面與類別（`java.util`）	Java	CH16
集合	collection	把一群物件當成一個單位來存取的資料結構	Java	CH16
泛型型別參數	generic type parameter `<E>`	用 `<E>` 指定集合裝哪種元素，編譯期就檢查型別	Java	CH16
型別包裝類別	type-wrapper class	把原始型別包成物件（`int`→`Integer`、`double`→`Double`）	Java	CH16
自動裝箱	autoboxing	`int` 自動轉成 `Integer` 放進集合	Java	CH16
自動拆箱	auto-unboxing	`Integer` 自動轉回 `int` 來運算	Java	CH16
介面	interface `Collection` / `List`	規範集合「能做什麼」的型別（`add`、`size`…）	Java	CH16
串列	List	有序、可重複、有索引的集合（`ArrayList`、`LinkedList`）	Java	CH16
走訪器	Iterator	逐一取出集合元素的物件，`hasNext()`／`next()`	Java	CH16
集	Set	不允許重複元素的集合（`HashSet`、`TreeSet`）	Java	CH16
映射	Map	用「鍵→值」配對存取的集合（`HashMap`、`TreeMap`）	Java	CH16
鍵	key	Map 裡用來查找的唯一識別	Java	CH16
值	value	Map 裡與某個鍵對應的資料	Java	CH16
鍵值對	entry	Map 的一筆「鍵→值」紀錄（`Map.Entry`）	Java	CH16

中文	English	一句話定義	科目	章節
Collections 類別	class `Collections`	一堆操作集合的靜態方法（`sort`、`max`、`shuffle`…）	Java	CH16

§ 2 二、核心概念

CH7 的陣列固定長度、`ArrayList` 只是其一。Java 把「裝資料」標準化成一整套**集合框架**（collections framework），全在 `java.util`。挑集合不是挑「哪個類別」，是先挑**要哪種行為**：要**有序、可重複、能用索引**→ `List`；要**不重複**→ `Set`；要**用鍵查值**→ `Map`。

> 三選一先問三句話：要保留放入順序、允許重複嗎（List）？要自動去重嗎（Set）？要用一個鍵去查對應的值嗎（Map）？答案決定用哪族。

每個集合都用**泛型型別參數** `<E>` 標明裝什麼：`ArrayList<String>`、`HashSet<Integer>`、`HashMap<String, Integer>`。寫上去之後，編譯器在**編譯期**就擋下放錯型別，取出來也不用轉型。

集合只裝**物件**，不裝原始型別。要放 `int` 得先變成 `Integer`，這步由**自動裝箱**（autoboxing）自動做；取出來運算時再**自動拆箱**回 `int`。所以 `list.add(5)` 其實是 `list.add(Integer.valueOf(5))`。

`List` 兩個常用實作各有取捨：`ArrayList` 內部是連續陣列，**隨機存取**（`get(i)`）快、中間增刪慢；`LinkedList` 是節點鏈，**頭尾增刪**快、隨機存取慢。多數情況用 `ArrayList`。

§ 3 三、主要內容

3.1 1. 三種集合對照（List／Set／Map）

3.2 2. 型別包裝與自動裝箱／拆箱

集合只裝物件，原始型別要靠**型別包裝類別**（type-wrapper）包成物件：`int`→`Integer`、`double`→`Double`、`char`→`Character`、`boolean`→`Boolean`。

```
ArrayList<Integer> nums = new ArrayList<>();
nums.add(5);           // 自動裝箱：5 → Integer.valueOf(5)
nums.add(10);
int first = nums.get(0);    // 自動拆箱：Integer → int
int sum = nums.get(0) + nums.get(1);  // 拆箱後相加 = 15
```

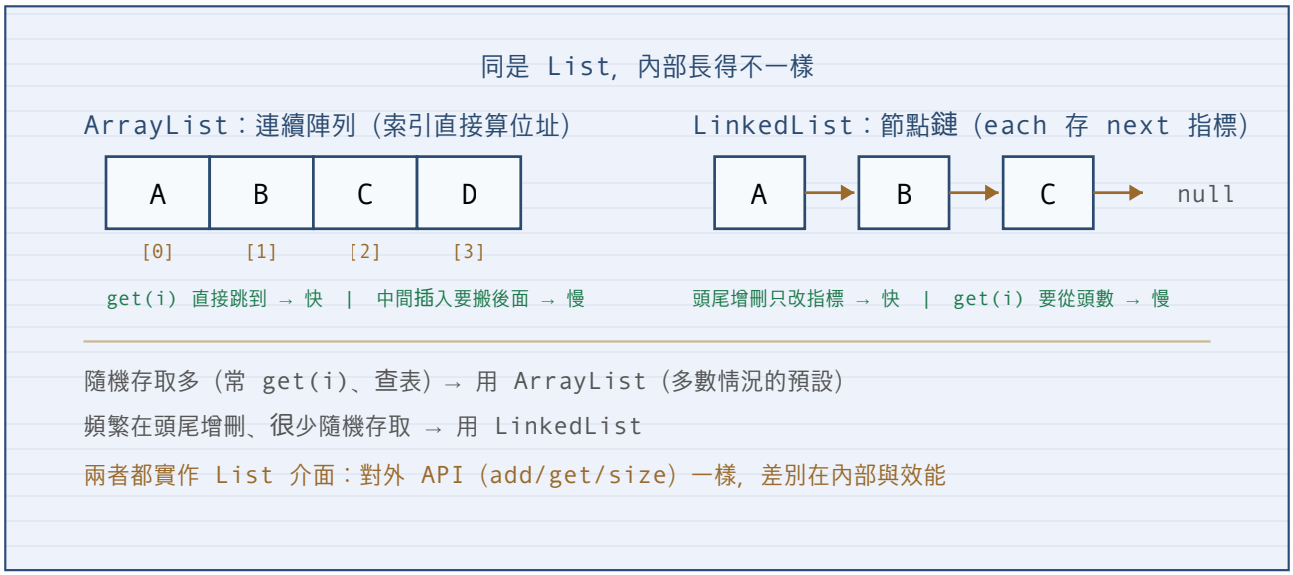
- 泛型參數一定寫**包裝類別**：`ArrayList<Integer>`，不能寫 `ArrayList<int>`。
- 裝箱、拆箱由編譯器自動補；自己不用寫 `Integer.valueOf` 或 `.intValue()`。

3.3 3. List：ArrayList 與 Iterator

```java
import java.util.*;

List<String> names = new ArrayList<>();   // 用介面型別 List 宣告
names.add("Amy");          // 加到尾端
names.add("Ben");
names.add("Amy");          // List 允許重複
names.get(0);              // 索引取 → "Amy"
names.set(1, "Bob");       // 設第 1 個 → "Bob"
names.size();              // 個數 → 3
names.contains("Amy");     // 是否含 → true
names.remove("Bob");       // 移除元素

// 走訪法一：增強 for
for (String n : names) System.out.println(n);

// 走訪法二：Iterator（可在走訪中安全移除）
Iterator<String> it = names.iterator();
while (it.hasNext()) {
    String n = it.next();
    if (n.equals("Amy")) it.remove();   // 用 Iterator 的 remove 才安全
}
```

- **用介面型別宣告**：`List<String> x = new ArrayList<>()`，之後換 `LinkedList` 不必改其他程式碼。
- **Iterator** 用 `hasNext()`／`next()` 逐一取；要在走訪途中刪元素，得用 `it.remove()`（直接 `list.remove` 會丟 `ConcurrentModificationException`）。

3.4 4. ArrayList vs LinkedList 內部結構



3.5 5. Collections 類別的靜態方法

`Collections`（複數，類別）提供一堆操作 `List` 的靜態方法：

```java
import java.util.*;

List<Integer> nums = new ArrayList<>(Arrays.asList(5, 2, 8, 1, 9));

Collections.sort(nums);              // 升冪排序 → [1, 2, 5, 8, 9]
Collections.sort(nums, Collections.reverseOrder());  // 降冪 → [9, 8, 5, 2, 1]
int hi = Collections.max(nums);      // 最大 → 9
int lo = Collections.min(nums);      // 最小 → 1
Collections.shuffle(nums);           // 隨機打亂順序
Collections.reverse(nums);           // 反轉順序

Collections.sort(nums);              // binarySearch 前要先排序！
int pos = Collections.binarySearch(nums, 8);  // 回傳索引；找不到回負數
```

- `Collections`（複數）**是類別**，方法是靜態的；`Collection`（單數）是介面。別搞混。
- `binarySearch`（二分搜尋）**前提是已排序**；沒排序結果不可靠。

3.6 6. Set：自動去重

```java
import java.util.*;

Set<String> tags = new HashSet<>();   // 無序、不重複
tags.add("java");
tags.add("c");
tags.add("java");         // 重複加入無效，集合大小不變
System.out.println(tags.size());      // 2
System.out.println(tags.contains("c"));  // true

Set<String> sorted = new TreeSet<>(tags);  // TreeSet：自動依字典序排序
// 走訪 sorted → c, java（排好序）
```

- `HashSet`：**快、無序**（不保證走訪順序）。
- `TreeSet`：自動**依排序順序**走訪（數字由小到大、字串字典序），代價是稍慢。
- 用途：去重、判斷「有沒有出現過」。`add` 重複元素回 `false`、大小不變。

3.7 7. Map：鍵值對

```
import java.util.*;

Map<String, Integer> scores = new HashMap<>();   // 鍵 String、值 Integer
scores.put("Amy", 90);        // 放入鍵值對
scores.put("Ben", 85);
scores.put("Amy", 95);        // 鍵重複 → 覆蓋舊值（Amy 變 95）
scores.get("Amy");            // 用鍵查值 → 95
scores.getOrDefault("Cara", 0);  // 鍵不存在時回預設值 0
scores.containsKey("Ben");    // 是否有此鍵 → true
scores.size();                // 鍵值對個數 → 2

// 走訪：用 entrySet 同時拿鍵與值
for (Map.Entry<String, Integer> e : scores.entrySet())
    System.out.println(e.getKey() + " = " + e.getValue());

// 只走訪鍵 / 只走訪值
for (String k : scores.keySet()) System.out.println(k);
for (int v : scores.values()) System.out.println(v);
```

字串字元／單字計次是 Map 的經典用途：

```
String text = "banana";
Map<Character, Integer> count = new HashMap<>();
for (char c : text.toCharArray())
    count.put(c, count.getOrDefault(c, 0) + 1);  // 沒有就從 0、有就 +1
// count = {b=1, a=3, n=2}
```

- **鍵唯一**：`put` 同一個鍵會**覆蓋**舊值，不會新增一筆。
- `HashMap`（快、無序）／`TreeMap`（依鍵排序）。
- 取值前用 `getOrDefault` 或 `containsKey` 避免 `null`。

§ 4 四、語法與 API 速查

```
import java.util.*;                          // List, ArrayList, Set, HashSet, Map,
HashMap, Collections...

// --- List（有序、可重複、有索引）---
List<String> li = new ArrayList<>();        // 或 new LinkedList<>()
li.add(e);  li.get(i);  li.set(i, e);
li.size();  li.contains(e);  li.remove(i 或 物件);
for (String x : li) { ... }                  // 增強 for 走訪
Iterator<String> it = li.iterator();         // it.hasNext() / it.next() / it.remove()

// --- Collections 靜態方法（操作 List）---
Collections.sort(li);                        // 升冪；binarySearch 前必先排序
Collections.sort(li, Collections.reverseOrder());  // 降冪
Collections.max(li);  Collections.min(li);
Collections.shuffle(li);  Collections.reverse(li);
Collections.binarySearch(li, key);           // 須先排序；找不到回負數

// --- Set（不重複）---
Set<Integer> s = new HashSet<>();            // 無序；或 new TreeSet<>() 自動排序
s.add(e);  s.contains(e);  s.size();         // add 重複元素回 false

// --- Map（鍵→值）---
Map<String,Integer> m = new HashMap<>();     // 無序；或 new TreeMap<>() 依鍵排序
m.put(k, v);                                 // 鍵重複 → 覆蓋
m.get(k);  m.getOrDefault(k, 預設);
m.containsKey(k);  m.size();
for (Map.Entry<String,Integer> en : m.entrySet()) { en.getKey(); en.getValue(); }
m.keySet();  m.values();                     // 只走訪鍵 / 只走訪值

Integer.valueOf(5);  someInteger.intValue();  // 裝箱/拆箱（通常自動，免寫）
```

- **選型**：要順序＋索引 → List；要去重 → Set；要查表 → Map。
- **個數一律** `size()`（集合都是方法、有括號）；只有陣列才用 `length` 欄位。
- `Collection`（介面，單數）≠ `Collections`（工具類別，複數）。

§ 5 五、常見錯誤

- **泛型寫原始型別**：`ArrayList<int>` 編譯錯誤；要寫包裝類別 `ArrayList<Integer>`。
- `Collection` **與** `Collections` **混淆**：`Collections.sort(...)`（複數工具類）；`Collection` 是單數介面，沒有那些靜態方法。
- **集合用** `length`：取個數一律 `size()`（方法、有括號），不是 `length`；`length` 只有陣列用。
- **走訪中用** `list.remove`：增強 for 或 Iterator 走訪途中直接 `list.remove(...)` 丟 `ConcurrentModificationException`；要刪用 `Iterator.remove()`。
- `binarySearch` **沒先排序**：二分搜尋前提是已排序，未排序結果不可靠；先 `Collections.sort`。

- **以為 Set 會保留順序**：`HashSet` 無序、走訪順序不保證；要排序用 `TreeSet`、要保留插入順序用 `LinkedHashSet`。
- **以為 Map 重複鍵會新增**：`put` 同一個鍵是**覆蓋**舊值，不會多一筆。
- `map.get(k)` **直接用**：鍵不存在回 `null`，拆箱成 `int` 會 `NullPointerException`；用 `getOrDefault` 或先 `containsKey`。
- **HashSet／HashMap 自訂物件當元素／鍵**：自訂類別要正確覆寫 `equals` 與 `hashCode`，否則去重與查找會失準（用內建 `String`、`Integer` 則沒問題）。

§ 6 六、練習題

例題 1：ArrayList 加入、走訪與移除

建一個 `ArrayList`，加入 `Amy`、`Ben`、`Amy` 三筆，印出個數，用增強 for 印出每一個，再移除一個 `Amy` 後印出剩餘個數。

> 1. `import java.util.*`，`List list = new ArrayList<>()` 2. `add` 三筆，`size()` 印個數，增強 for 走訪 3. `list.remove("Amy")` 後再 `size()`

```
import java.util.*;
public class ListDemo {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("Amy");
        list.add("Ben");
        list.add("Amy");
        System.out.println("Count = " + list.size());    // 3（List 允許重複）
        for (String n : list) System.out.println(n);
        list.remove("Amy");                               // 移除第一個 Amy
        System.out.println("After remove = " + list.size());  // 2
    }
}
```

> 易錯：用 `length` 取個數（應 `size()`）；漏 `import java.util.*`。

例題 2：用 Set 去重

給 `int[] a = {3, 1, 3, 2, 1, 3}`，用 `HashSet` 去掉重複，印出不重複元素的個數（`Unique = 3`）。

> 1. `Set set = new HashSet<>()` 2. 增強 for 把每個元素 `add` 進去（重複自動忽略） 3. `set.size()` 即為不重複個數

```
import java.util.*;
public class UniqueCount {
    public static void main(String[] args) {
        int[] a = {3, 1, 3, 2, 1, 3};
        Set<Integer> set = new HashSet<>();
        for (int x : a) set.add(x);          // 自動裝箱 + 自動去重
        System.out.println("Unique = " + set.size());   // 3 (1, 2, 3)
    }
}
```

> 易錯：寫 `HashSet`（要 `Integer`）；以為 `add` 重複會報錯（其實只是回 false、被忽略）。

例題 3：HashMap 計算字元次數

給字串 `"banana"`，用 `HashMap` 統計每個字元出現幾次，用 `entrySet` 走訪印出（如 `a = 3`）。

> 1. `Map count = new HashMap<>()` 2. `for (char c : s.toCharArray())`：`count.put(c, count.getOrDefault(c, 0) + 1)` 3. `for (Map.Entry e : count.entrySet())` 印 `e.getKey()` 與 `e.getValue()`

```
```

```
import java.util.*;
public class CharCount {
    public static void main(String[] args) {
        String s = "banana";
        Map<Character, Integer> count = new HashMap<>();
        for (char c : s.toCharArray())
            count.put(c, count.getOrDefault(c, 0) + 1);  // 沒有從 0、有就 +1
        for (Map.Entry<Character, Integer> e : count.entrySet())
            System.out.println(e.getKey() + " = " +
e.getValue());  // b=1 a=3 n=2
    }
}
```

> 易錯：`get` 鍵不存在回 null 直接拆箱 → NullPointerException（用 `getOrDefault`）；以為重複鍵會新增（其實覆蓋）。

§ 7 七、自我檢核

- [ ] 會用三句話（要順序/重複？要去重？要用鍵查值？）判斷該用 List、Set 還是 Map。
- [ ] 知道集合只裝物件、原始型別靠型別包裝（`Integer`）與自動裝箱／拆箱；泛型寫包裝類別不寫 `int`。
- [ ] 會用 `List`／`ArrayList` 的 `add`／`get`／`set`／`size`／`contains`／`remove`，會用增強 for 與 Iterator 走訪。
- [ ] 說得出 `ArrayList`（隨機存取快）與 `LinkedList`（頭尾增刪快）內部差異與選用時機。
- [ ] 會用 `Collections` 的 `sort`／`max`／`min`／`shuffle`／`binarySearch`，知道 `binarySearch` 前要先排序。
- [ ] 會用 `Set`／`HashSet` 去重，分得清 `HashSet`（無序）與 `TreeSet`（自動排序）。
- [ ] 會用 `Map`／`HashMap` 的 `put`／`get`／`getOrDefault`／`containsKey`，用 `entrySet` 走訪鍵值對；知道重複鍵是覆蓋。
- [ ] 分得清 `Collection`（介面，單數）與 `Collections`（工具類別，複數）；知道走訪中刪元素要用 `Iterator.remove()`。