字串：不可變的字元序列與 StringBuilder

Deitel & Deitel《Java How to Program: Early Objects》11e, Ch.14

§ 1　一、名詞速查表

中文	English	一句話定義	科目	章節
字串	String	一連串字元組成的物件，內容**不可變**	Java	CH14
不可變	immutable	物件建好後內容不能改；任何「修改」都產生**新物件**	Java	CH14
字串常值	string literal	程式裡用 `"…"` 寫死的字串	Java	CH14
字元	char	單一字元，用 `'A'` 單引號；`charAt` 回傳它	Java	CH14
長度	length()	字串字元個數，`s.length()`（**方法、有括號**）	Java	CH14
索引	index	字元的編號，從 `0` 到 `length()-1`	Java	CH14
相等	equals	比**內容**是否相同；不可用 `==` 比字串內容	Java	CH14
字典序比較	compareTo	依字典順序比大小，回負/0/正整數	Java	CH14
子字串	substring	取出字串的一段，`s.substring(a, b)`（含 a 不含 b）	Java	CH14
串接	concatenation	用 `+` 或 `concat` 把字串接起來，產生新字串	Java	CH14
字串建構器	StringBuilder	**可變**的字元序列，`append`/`insert`/`delete` 原地改	Java	CH14
容量	capacity	StringBuilder 內部緩衝區大小，與 `length()` 不同	Java	CH14
Character 類別	Character	包裝 `char` 的工具類，`isDigit`/`isLetter`/`toUpperCase`	Java	CH14
切分	split	依分隔符把字串切成 `String[]`，常用 regex	Java	CH14
正規表達式	regular expression	描述字串模式的字串，配 `matches`/`replaceAll`	Java	CH14

§ 2 二、核心概念

字串（String）是 Java 裡最常用的物件之一：一連串字元，用雙引號 `"Hello"` 寫出。它有一個關鍵特性——**不可變（immutable）**：物件一旦建好，內容就**永遠不會改**。`s.toUpperCase()`、`s.concat("x")`、`s.replace('a','b')` 看起來像在「改」字串，其實都是**回傳一個新字串**，原本那個 `s` 指向的物件完全沒動。

> 不可變的後果：`s.toUpperCase();` 單獨一行**沒有任何效果**（新字串被丟掉了）。要拿到結果必須接住：`s = s.toUpperCase();`。

第二個關鍵是**比較**。字串是物件、變數存的是位址，所以 `==` 比的是「兩個變數是不是指向同一個物件」，**不是內容**。要比內容一律用 `equals`（或忽略大小寫的 `equalsIgnoreCase`）。`==` 偶爾「看起來對」是因為編譯器把相同的字串常值放進共用池（string pool）讓它們共址，但只要其中一個是 `new String(...)` 或執行期算出來的，`==` 就會是 `false`。

當需要**反覆修改**字串（迴圈裡一直接、插入、刪除）時，每次都產生新 String 很浪費。改用 **StringBuilder**：它是**可變**的，`append`/`insert`/`delete` 都在**同一個物件**上原地改，最後再 `toString()` 變回 String。「組裝字串的過程用 StringBuilder，組好再轉 String」是標準作法。

§ 3 三、主要內容

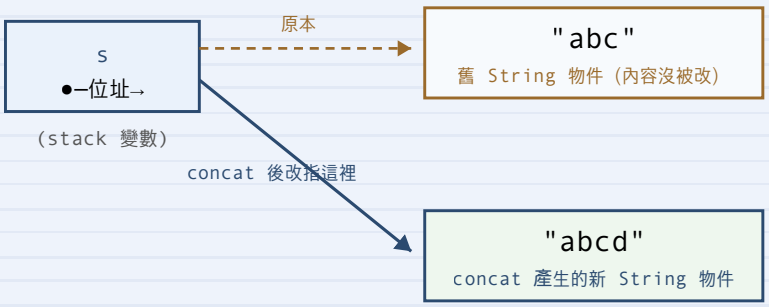
3.1 1. 建構字串與字串常值

```
String s1 = "Hello";                 // 字串常值（最常用）
String s2 = new String("Hello");     // 用 new 明確建一個新物件
char[] cs = {'J', 'a', 'v', 'a'};
String s3 = new String(cs);          // 從 char 陣列建：s3 = "Java"
String s4 = new String(cs, 1, 2);    // 從索引 1 取 2 個字元：s4 = "av"
```

- 直接寫 `"Hello"` 是最常見的建法；相同內容的常值會共用 string pool 裡的同一個物件。
- `new String("Hello")` 會**另外**配一個物件（即使內容一樣，位址不同）——這正是 `==` 陷阱的來源。

3.2 2. String 不可變：改動產生新物件

```
String s = "abc";  s = s.concat("d");  ─「修改」其實是換指向新物件
```



String 不可變：concat / replace / toUpperCase 都「不改原物件」，而是回傳新物件。
所以 s.concat("d"); 單獨一行沒效果──要寫 s = s.concat("d"); 才接得到新物件。
舊的 "abc" 若沒人再指向，交給垃圾回收（GC）。

3.3 3. 基本查詢：length、charAt、getChars

```
String s = "Hello";
s.length();           // 5（方法、有括號！跟陣列 a.length 欄位不同）
s.charAt(0);          // 'H'（第 0 個字元）
s.charAt(4);          // 'o'（最後一個 = length()-1）
// s.charAt(5);       // 越界！StringIndexOutOfBoundsException

char[] buf = new char[3];
s.getChars(0, 3, buf, 0);   // 把索引 0..2（不含 3）複製到 buf → {'H','e','l'}
```

- 字串 `length()` 是**方法（有括號）**；陣列 `length` 是**欄位（沒括號）**。兩個常被搞混。
- `charAt(i)` 的合法 `i` 是 `0` 到 `length()-1`，超出丟 `StringIndexOutOfBoundsException`。

3.4 4. 比較字串：equals、equalsIgnoreCase、compareTo、== 陷阱

```
String a = "hello";
String b = "hello";
String c = new String("hello");

a.equals(b);             // true（比內容）
a.equals(c);             // true（比內容，跟怎麼建的無關）
a == b;                  // true（常值共用 string pool，剛好同址）
a == c;                  // false！c 是 new 出來的另一個物件

a.equalsIgnoreCase("HELLO");  // true（忽略大小寫比內容）

"apple".compareTo("banana");  // 負數（a 在 b 前）
"banana".compareTo("apple");  // 正數
"apple".compareTo("apple");   // 0（相等）
```

- **比內容一律用** `equals`；`==` 比的是「是否同一個物件」，會被 string pool 誤導，不可靠。
- `compareTo` 回傳整數：`<0` 表前者較小（字典序在前）、`0` 相等、`>0` 後者較小。常用於排序。
- 想避免 `s` 為 `null` 時呼叫 `equals` 噴 `NullPointerException`，可把常值放前面：`"hello".equals(s)`。

3.5 5. 搜尋與取段：indexOf、substring、concat、replace、其他

```
String s = "programming";
s.indexOf('g');          // 3（第一個 'g' 的索引；找不到回 -1）
s.indexOf("mm");         // 6（子字串第一次出現的索引）
s.lastIndexOf('g');      // 10（最後一個 'g'）

s.substring(3);          // "gramming"（從索引 3 到尾）
s.substring(0, 4);       // "prog"（索引 0..3，含頭不含尾）

"Java".concat("Script"); // "JavaScript"（= "Java" + "Script"）
"a,b,c".replace(',', ';'); // "a;b;c"（換字元，回新字串）
"  hi  ".trim();         // "hi"（去頭尾空白）
"Hello".toUpperCase();   // "HELLO"
"Hello".toLowerCase();   // "hello"
"Hello".startsWith("He");// true
"Hello".endsWith("lo");  // true
"".isEmpty();            // true（長度 0）
```

- `substring(a, b)` **含 a、不含 b**，長度 `b - a`；`substring(a)` 取到字串尾。
- 以上方法**全部回傳新字串**（String 不可變），原字串不變；要用結果就接住。

3.6 6. 切分字串：split

```
String csv = "Amy,Ben,Cara";
String[] parts = csv.split(",");       // {"Amy", "Ben", "Cara"}
for (String p : parts)
    System.out.println(p);             // 一行一個

"one  two   three".split("\\s+");      // 用正規表達式：一個以上空白 →
{"one","two","three"}
```

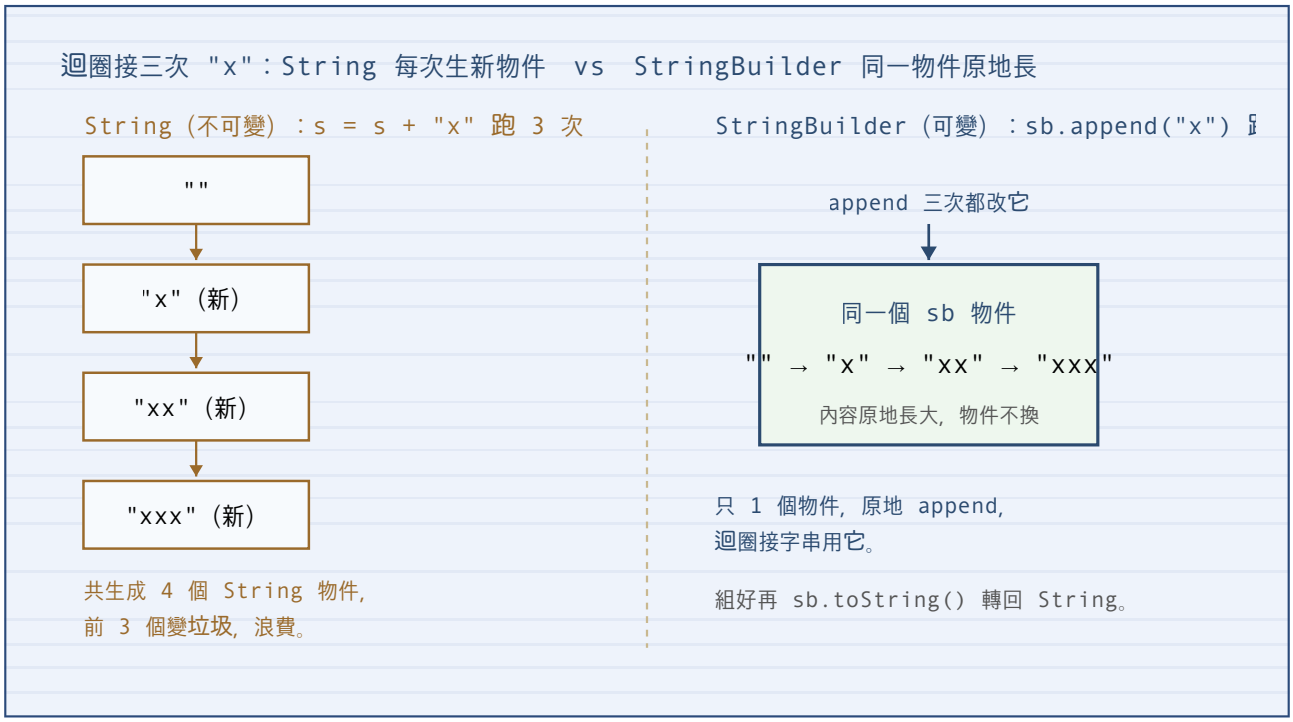
- `split` 的參數是**正規表達式（regex）**；切一般符號（逗號、空白）直接寫即可。
- 連續分隔或多種空白，用 regex `"\\s+"`（一個以上空白字元）較穩。

3.7 7. StringBuilder：可變的字元序列

```
StringBuilder sb = new StringBuilder();   // 空的
sb.append("Hello");          // "Hello"
sb.append(' ').append("Java");  // "Hello Java"（append 可串接、原地改）
sb.insert(5, ",");           // 在索引 5 插入 → "Hello, Java"
sb.delete(5, 6);             // 刪索引 5..5 → "Hello Java"
sb.reverse();                // 反轉 → "avaJ olleH"
sb.length();                 // 目前長度
String result = sb.toString();   // 轉回 String 才能當一般字串用
```

- StringBuilder **可變**：`append`/`insert`/`delete`/`reverse` 都改**同一個物件**，不像 String 每次產生新物件。
- 迴圈裡反覆接字串時用 StringBuilder（效率好）；組好用 `toString()` 轉回 String。
- `capacity()`（內部緩衝大小）和 `length()`（實際字元數）不同，一般不必管 capacity。

3.8 8. String vs StringBuilder（多次 append 的差別）



3.9 9. Character 類別方法

```
Character.isDigit('7');        // true（是數字字元嗎）
Character.isLetter('A');       // true（是字母嗎）
Character.isLetterOrDigit('_'); // false
Character.isWhitespace(' ');   // true（是空白嗎）
Character.isUpperCase('A');    // true
Character.toUpperCase('a');    // 'A'（回傳大寫 char）
Character.toLowerCase('A');    // 'a'
```

- `Character` 是包裝 `char` 的工具類，方法大多是 `static`：傳一個 `char` 進去，回傳判斷（`boolean`）或轉換後的 `char`。
- 一個字元一個字元檢查字串（如數有幾個數字）時常配 `charAt` 一起用。

3.10 10. 正規表達式（輕量帶過）

```
"12345".matches("\\d+");          // true（整串都是一個以上數字嗎）
"abc123".matches("\\d+");         // false（有字母）
"a1b2c3".replaceAll("\\d", "*");// "a*b*c*"（把每個數字換成 *）
"one  two".split("\\s+");         // {"one","two"}（一個以上空白切）
```

- 常見符號：`\\d`＝一個數字、`\\s`＝一個空白、`+`＝一個以上、`*`＝零個以上。
- `matches` 要**整串**符合才回 `true`；`replaceAll`／`split` 的第一個參數都是 regex。

- 入門階段會用 `matches`／`replaceAll`／`split` 配簡單模式即可，完整 regex 與 `Pattern`/`Matcher` 屬進階。

§ 4 四、語法與 API 速查

```
// String（不可變，方法都回傳新字串）
s.length()                      // 字元數（方法、有括號）
s.charAt(i)                     // 第 i 個字元（0..length()-1）
s.equals(t)  s.equalsIgnoreCase(t)   // 比內容（不要用 ==）
s.compareTo(t)                  // 字典序：<0 / 0 / >0
s.indexOf(x)  s.lastIndexOf(x) // 找位置（找不到 -1），x 可 char 或 String
s.substring(a)  s.substring(a, b)    // 取段（含 a 不含 b）
s.concat(t)  或  s + t          // 串接
s.replace(old, new)             // 換字元/子字串
s.toUpperCase()  s.toLowerCase()  s.trim()   // 轉大小寫、去頭尾空白
s.startsWith(p)  s.endsWith(p)  s.isEmpty()  // 前綴/後綴/空字串
s.split(regex)                  // 切成 String[]（參數是 regex）

// StringBuilder（可變，原地改）
StringBuilder sb = new StringBuilder();
sb.append(x)                    // 接到尾端（可串接 .append().append()）
sb.insert(i, x)  sb.delete(a, b)  sb.reverse()
sb.length()                     // 目前字元數
sb.toString()                   // 轉回 String

// Character（工具類，多為 static）
Character.isDigit(c)  isLetter(c)  isLetterOrDigit(c)  isWhitespace(c)
Character.isUpperCase(c)  toUpperCase(c)  toLowerCase(c)

// 正規表達式（輕量）
s.matches(regex)  s.replaceAll(regex, rep)   // \\d 數字 \\s 空白 + 一個以上
```

- **length 之爭**：字串 `s.length()`（方法）；陣列 `a.length`（欄位）；ArrayList `list.size()`（方法）。
- **比內容用 equals**；`==` 比物件位址，會被 string pool 誤導。
- **反覆改字串用 StringBuilder**；組好 `toString()` 轉回 String。

§ 5 五、常見錯誤

- **用 `==` 比字串內容**：`s == "abc"` 比的是位址，`new String("abc")` 或執行期算出的字串會 `false`；比內容一律用 `equals`。
- **以為字串方法會改原字串**：`s.toUpperCase();` 單獨一行沒效果（String 不可變）；要 `s = s.toUpperCase();` 接住新字串。
- **`length` 加不加括號搞混**：字串 `s.length()`（方法、有括號）；陣列 `a.length`（欄位、無括號）。寫反會編譯錯。

- `charAt` / `substring` **索引越界**：合法 `charAt` 索引是 `0..length()-1`；`substring(a, b)` 的 `b` 可到 `length()`（含尾），超出丟 `StringIndexOutOfBoundsException`。
- `substring(a, b)` **把 b 當「含」**：實際是**含 a 不含 b**，長度 `b - a`；`"Hello".substring(0,2)` 是 `"He"` 不是 `"Hel"`。
- `null` **字串呼叫方法**：對 `null` 呼叫 `.equals`/`.length()` 噴 `NullPointerException`；用常值在前 `"x".equals(s)` 可避免。
- **迴圈裡用** `+` **接大量字串**：每次都生新 String 很慢；改用 `StringBuilder.append`。
- `split` **忘記參數是 regex**：切 `.`（句點）要寫 `"\\."`（`.` 在 regex 是「任意字元」），直接 `split(".")` 會切成空陣列。

§ 6 六、練習題

例題 1：基本查詢 length 與 charAt

給 `String s = "Hello"`，印出長度、第 0 個字元、最後一個字元（用 `length()-1`）。

> 1. `s.length()` 取長度 2. `s.charAt(0)` 取第一個、`s.charAt(s.length()-1)` 取最後一個 3. 用 `println` 印出三項

```
```

```
public class StringBasic {
    public static void main(String[] args) {
        String s = "Hello";
        System.out.println("length = " + s.length());            // 5
        System.out.println("first  = " + s.charAt(0));            // H
        System.out.println("last   = " + s.charAt(s.length() - 1)); // o
    }
}
```

> 易錯：寫 `s.length`（少括號，字串用方法）；最後一格寫 `charAt(s.length())` 越界。

例題 2：字串比較 equals vs ==

給 `String a = "java"`、`String b = new String("java")`，分別印出 `a.equals(b)` 與 `a == b`，並說明為何不同。

> 1. `a.equals(b)`：比內容 → `true` 2. `a == b`：比位址，`b` 是 `new` 出來的另一個物件 → `false` 3. 各印一行 `boolean`

```
public class StringEquals {
    public static void main(String[] args) {
        String a = "java";
        String b = new String("java");
        System.out.println("equals = " + a.equals(b));  // true（比內容）
        System.out.println("==     = " + (a == b));      // false（比位址，b 是新物件）
    }
}
```

> 易錯：用 `==` 當作比內容（被 new 出來的物件打破）；漏掉 `(a == b)` 的括號使 `+` 先算。

例題 3：StringBuilder 組字串

用 `StringBuilder` 把 `{"Java","is","fun"}` 三個字以空白接成一句 `"Java is fun"`，再 `toString()` 印出。

> 1. `StringBuilder sb = new StringBuilder();` 2. 走訪陣列 `append(word)`，非第一個前先 `append(" ")` 3. `sb.toString()` 印出

```
public class JoinWords {
    public static void main(String[] args) {
        String[] words = {"Java", "is", "fun"};
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < words.length; i++) {
            if (i > 0) sb.append(" ");   // 不是第一個才補空白
            sb.append(words[i]);
        }
        System.out.println(sb.toString());   // Java is fun
    }
}
```

> 易錯：每個字後面都補空白導致尾端多一格；用 `String +=` 在迴圈裡接（該用 StringBuilder）。

§ 7 七、自我檢核

- [ ] 知道 String **不可變**，`toUpperCase`/`concat`/`replace` 都回傳新字串、原字串不變，要用結果得接住。
- [ ] 會用 `length()`（方法、有括號）、`charAt(i)`，知道合法索引 `0..length()-1`、越界丟例外。
- [ ] 比字串內容一律用 `equals`／`equalsIgnoreCase`，知道 `==` 比位址、會被 string pool 誤導；`compareTo` 回負/0/正。
- [ ] 會用 `indexOf`／`substring`（含頭不含尾）／`concat`／`replace`／`trim`／`toUpperCase` 等查詢與取段方法。
- [ ] 會用 `split` 把字串切成 `String[]`，知道參數是 regex（切句點要 `"\\."`）。
- [ ] 會用 `StringBuilder` 的 `append`／`insert`／`delete`／`reverse` 原地改，知道反覆接字串該用它、組好 `toString()` 轉回。
- [ ] 會用 `Character.isDigit`／`isLetter`／`toUpperCase` 等工具方法逐字元檢查。
- [ ] 知道 `matches`／`replaceAll` 配簡單 regex（`\\d`、`\\s`、`+`）的基本用法。