

Java 程式設計進階

常用類別

鄭安翔

ansel_cheng@hotmail.com

課程大綱

- 1) **String** 類別
 - 2) **StringBuffer**及**StringBuilder**類別
 - 3) 包裹類別 **Wrapper Class**
 - 4) **ArrayList**
-

Java SE 8 API 文件

Overview (Java Platform SE 8) x +

docs.oracle.com/javase/8/docs/api/

Java™ Platform Standard Ed. 8

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV NEXT FRAMES NO FRAMES

Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.

String v.s. char

比較項目	String	char
資料型別 (Data type)	參考資料型別 Reference type	基本資料型別 Primitive type
Literal 資料的符號 Literal enclosed	使用雙引號 (")	使用單引號 (')
抽象表現 Represent	類別 (class)	Unicode
比較運算 Operator Compared	equals() 或 (==)	雙等號 (==)
記憶體內容值是否可變？	不可變	可變

String 建構

■ 建立字串物件

□ 標準物件建構語法

`String str1 = new String("Hello World!");` //一般物件建構

建構子	說明
<code>String()</code>	建立一個空字元序列的 String 物件
<code>String(byte[] bytes)</code>	使用指定的 byte 陣列內容，建構一個新的 String
<code>String(char[] value)</code>	使用指定的 char 陣列內容，建構一個新的 String
<code>String(String original)</code>	建立一個與字串參數相同的新 String 物件
<code>String(StringBuffer buffer)</code>	建立一個與字元緩衝參數相同的新 String 物件
<code>String(StringBuilder builder)</code>	建立一個與字元緩衝參數相同的新 String 物件

□ 非標準物件建構語法

`String str2 = "Hello World!";`

//literal方式賦值

■ 字串池重複使用

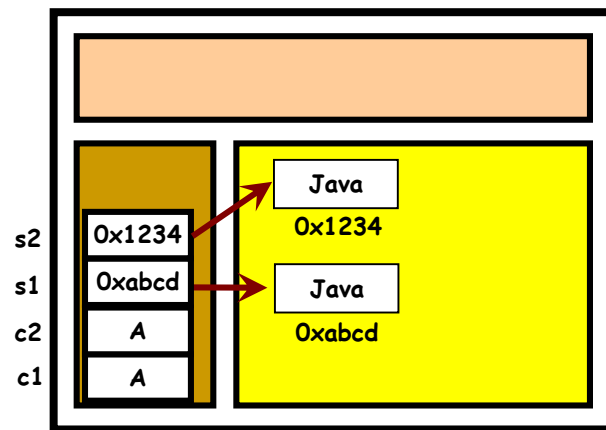
比較運算

■ “==”：

□ 用來比較在 **stack** 中的變數內容是否相等

- 基本資料型別：變數內容即為其值
- 參考資料型別：物件是不是同一個

```
char c1 = 'A';  
char c2 = 'A';  
String s1 = new String("Java");  
String s2 = new String("Java");  
  
c1 == c2 ? true !  
s1 == s2 ? 0xabcd == 0x1234 ? false !
```

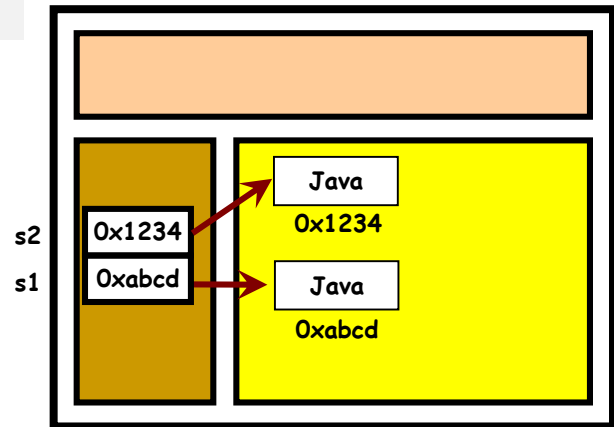


equals() 方法

■ equals() 方法

- String 所提供的 equals() 來比較字串的實際內容

```
String s1 = new String("Java");  
String s2 = new String("Java");  
s1.equals( s2 ) ? "Java" == "Java" true !
```

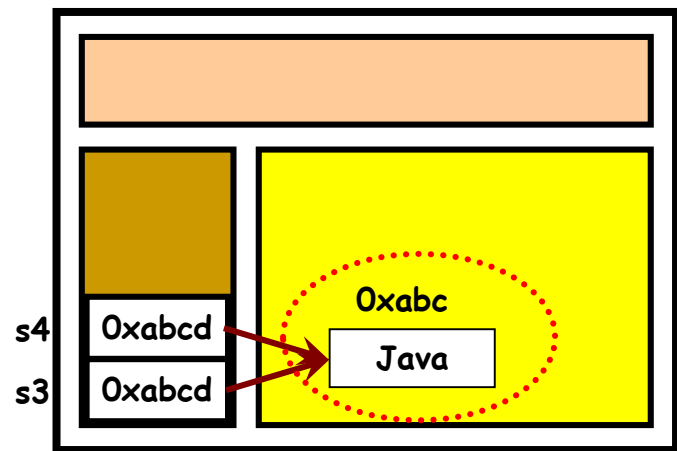


String Literal Pool

- JVM中存放字串的共用區域
- 以 **String identifier = string_literal;** 方式宣告的字串物件儲存到string pool中
- 新增字串物件到 **String pool** 前會先檢查是否有相同樣的物件, 如果有的話直接傳回參考值, 不會新增字串物件

```
String s3 = "Java";  
String s4 = "Java";
```

```
s3 == s4 ? 0xabcd == 0xabcd ? true !
```



String 內容值比較結論

```
01 public class TestString {  
02     public static void main(String[] args) {  
03         String s1 = new String("Java");  
04         String s2 = new String("Java");  
05         String s3 = "Java";  
06         String s4 = "Java";  
07         System.out.println(s1 == s2);           → false  
08         System.out.println(s3 == s4);           → True // String pool  
09         System.out.println(s1.equals(s2));       → true // 比較實際內容  
10         System.out.println(s3.equals(s4));  
11     }  
12 }
```

String 常用方法

方法名稱	傳回值	說明
charAt(int index)	char	返回指定索引處的 char 值。
equals(Object anObject)	boolean	將此字串與指定的物件比較。
equalsIgnoreCase(String anotherString)	boolean	將此字串與另一個字串 anotherString 比較，不考慮大小寫。
compareTo(String anotherString)	int	按字典順序比較兩個字串。
compareToIgnoreCase(String str)	int	按字典順序比較兩個字串，不考慮大小寫。
hashCode()	int	返回此字串的雜湊碼。
contains(CharSequence s)	boolean	當此字串包含指定的 char 序列時，返回 true 。
startsWith(String prefix)	boolean	測試此字串是否以指定的前綴開始。
endsWith(String suffix)	boolean	測試此字串是否以指定的後綴結束。
indexOf(String str)	int	傳回指定子字串在此字串中第一次出現處的索引。
lastIndexOf(String str)	int	傳回指定子字串在此字串中最後一次出現處的索引。
isEmpty()	boolean	當 length() 為 0 時傳回 true 。
length()	int	傳回此字串的長度。
intern()	String	傳回字串物件的規範化表示形式

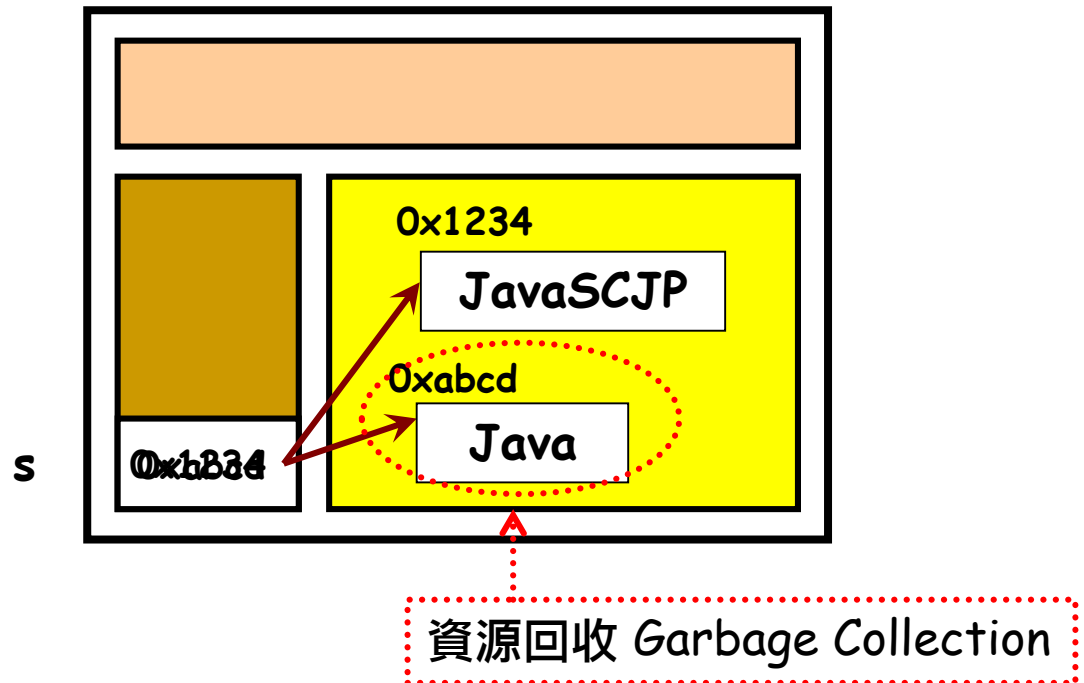
String 常用方法

方法名稱	傳回值	說明
concat(String str)	String	將指定字串連接到此字串的結尾。
replace(char oldChar, char newChar)	String	傳回一個新的字串，以 newChar 替換此字串中出現的所有 oldChar
replaceAll(String regex, String replacement)	String	使用給定的 replacement 替換此字串所有符合給定的正則表達式的子字串。
toLowerCase()	String	使用預設語言環境的規則將此字串中的所有字元都轉換為小寫。
toUpperCase()	String	使用預設語言環境的規則將此字串中的所有字元都轉換為大寫。
trim()	String	返回字串的副本，忽略前導空白和尾部空白。
substring(int beginIndex)	String	返回一個內容為此字串指定位置開始到結尾位置的子字串。
substring(int beginIndex, int endIndex)	String	返回一個內容為此字串指定位置開始到指定位置結尾的子字串。
split(String regex)	String[]	以符合給定正則表達式的元素切割此字串。
format(String format, Object... args)	static String	使用指定的格式字串和參數傳回一個格式化字串。
matches(String regex)	boolean	驗證此字串是否符合給定的正則表達式。

字串內容永遠不變！？

- 字串內容是永不變的（immutable）
 - 當字串已在記憶體中被建立出來，該記憶體中的字串內容就不允許被變更。

```
String s = "Java";  
s = s + "SCJP";  
System.out.println(s);  
➔ JavaSCJP
```



String 內容永遠不變

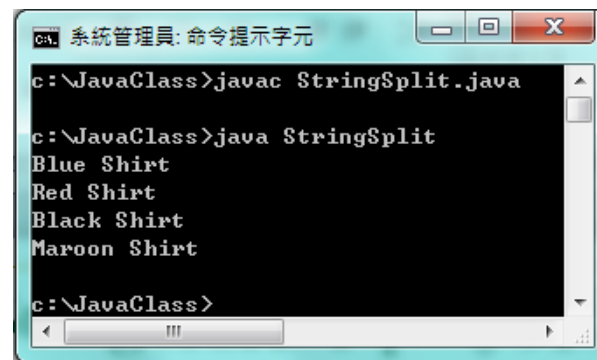
```
01 public class TestImmutable {  
02     public static void main(String[] args) {  
03         String s1 = "Java";  
04         s1 = s1.concat("SCJP");  
05         System.out.println(s1);    → JavaSCJP  
06         String s2 = "Java";  
07         s2.concat("SCJP");  
08         System.out.println(s2);    → Java  
09         String s3 = "Java";  
10         s3 = s3.replace('J', 'I');  
11         System.out.println(s3);    → Iava  
12         String s4 = "Java";  
13         s4.replace('J', 'I');  
14         System.out.println(s4);    → Java  
15     }  
16 }
```

String 類別的split() 方法

- String 類別也提供 split() 方法來切割字串
 - 事實上是呼叫 Pattern 類別的 split() 方法

常用方法	傳回值	說明
split(String regex)	String[]	以指定正則表示式之比對樣式 regex 來切割字串

```
public class StringSplit {  
    public static void main(String[ ] args) {  
        String shirts = "Blue Shirt, Red Shirt,  
                        Black Shirt, Maroon Shirt";  
  
        String[ ] results = shirts.split(", ");  
        for(int i=0; i<4; i++){  
            System.out.println(results[i]);  
        }  
    }  
}
```



```
ca. 系統管理員: 命令提示字元  
c:\JavaClass>javac StringSplit.java  
c:\JavaClass>java StringSplit  
Blue Shirt  
Red Shirt  
Black Shirt  
Maroon Shirt  
c:\JavaClass>
```

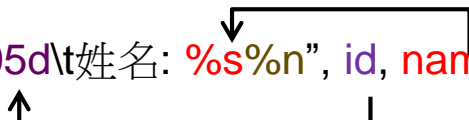
字串簡易格式化

■ String 類別 format() 方法

□ Java 5.0 新增，程式可簡易的將輸出的資料格式化

格式化輸出方法	傳回值	說明
format(String format, Object... args)	String	根據指定的格式化規則 (format) 及參數內容 (args)，傳回一個格式化字串

```
int id = 25;  
String name = "王小明";  
System.out.printf("學號: %05d\t姓名: %s%n", id, name);
```



■ PrintStream 類別

格式化輸出方法	傳回值	說明
printf(String format, Object... args)	PrintStream	根據指定的格式化規則 (format) 及參數內容 (args) 列印輸出

常用格式化輸出表

資料類型	說明
%b, %B	格式化成布林類型(小、大寫)
%d, %o, %x, %X	整數類型 (十、八進位、十六進位)
%f, %g, %e, %E	浮點數類型 (浮點數、科學符號, 自動)
%s, %S, %c, %C	格式化成字串、字元類型
%n	代表換行
%%	代表%

特殊符號	說明
\$	指定要格式化的來源(1\$,2\$)
<	使用前一個格式化資料來源
0	使用0來當填充字元
#	不使填充字元
,	代表會加上千分位數
.	代表小數點

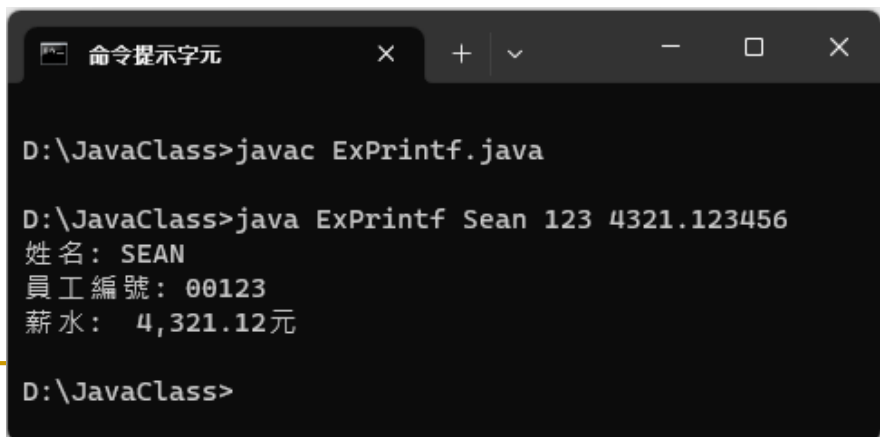
日期	說明
%td	日，共2位，不足部分補0
%tm	月，共2位，不足部分補0
%ty	年，共2位，不足部分補0
%tY	年，共4位，不足部分補0

時間	說明
%tH	時（24時制），共2位，不足部分補0
%tI	時（12時制），共2位，不足部分補0
%tM	分，共2位，不足部分補0
%tS	秒，共2位，不足部分補0

日期/時間	說明
%tF	相當於「%tY-%tm-%td」
%tT	相當於「%tH:%tM:%tS」

printf() & format() 範例

```
01 public class ExPrintf{
02     public static void main(String[] args) {
03         String name = args[0];
04         int id = Integer.parseInt(args[1]);
05         double salary = Double.parseDouble(args[2]);
06         String s = String.format("姓名: %S%n員工編號: %05d%n薪水: %,9.2f元%n",
07                                   name, id, salary);
08         System.out.println(s);
09     }
10 }
```



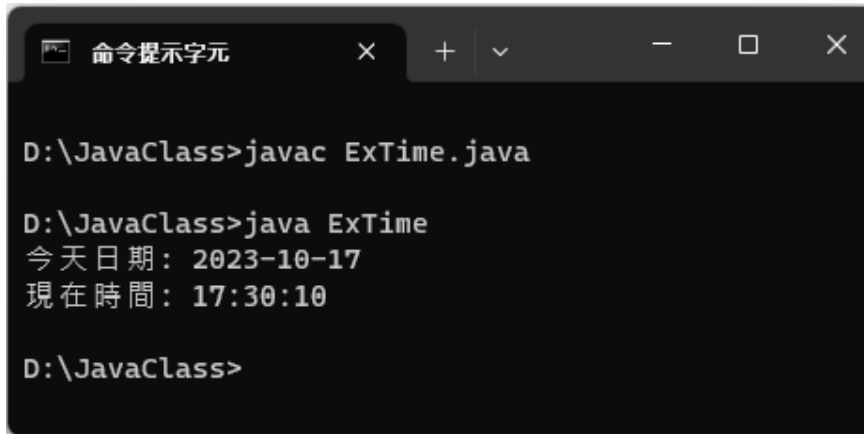
```
D:\JavaClass>javac ExPrintf.java

D:\JavaClass>java ExPrintf Sean 123 4321.123456
姓名: SEAN
員工編號: 00123
薪水: 4,321.12元

D:\JavaClass>
```

printf() & format() 範例

```
01 public class ExTime{  
02     public static void main(String[] args) {  
03         java.util.Date date = new java.util.Date();  
04         System.out.printf("今天日期: %tF%n現在時間: %<tT%n", date);  
05     }  
06 }
```



The screenshot shows a Windows Command Prompt window titled "命令提示字元". The user has entered the following commands and received the corresponding output:

```
D:\JavaClass>javac ExTime.java  
  
D:\JavaClass>java ExTime  
今天日期: 2023-10-17  
現在時間: 17:30:10  
  
D:\JavaClass>
```

課程大綱

- 1) String 類別
- 2) **StringBuffer及StringBuilder**類別
- 3) 包裹類別 Wrapper Class
- 4) ArrayList

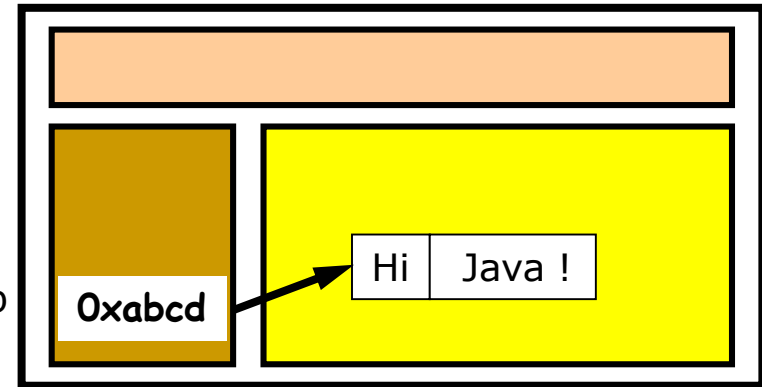
StringBuffer / StringBuilder類別

■ StringBuffer 及 StringBuilder類別

- ❑ 可變動內容(Mutable)的字串型態
- ❑ 提供進階處理字串的方法
- ❑ 利用 **append()** 來改變字串內容

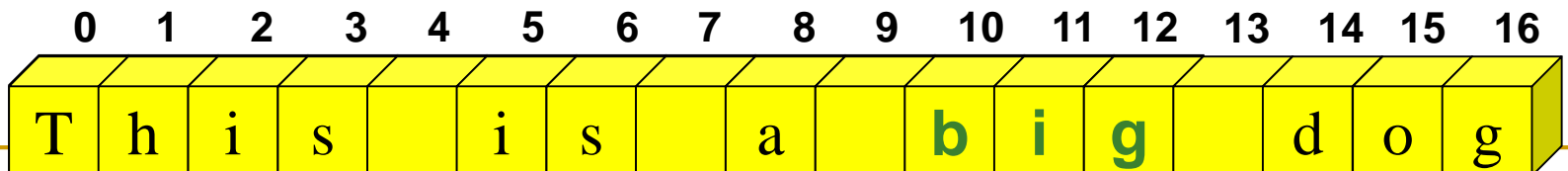
```
StringBuffer sb = new StringBuffer("Hi");  
sb.append(" Java!");  
System.out.println(sb.toString());
```

sb



- ❑ 利用 **insert()** 方法來選擇指定加入的位置

```
StringBuffer sb = new StringBuffer("This is a dog");  
sb.insert(10, "big ");  
System.out.println(sb.toString());
```

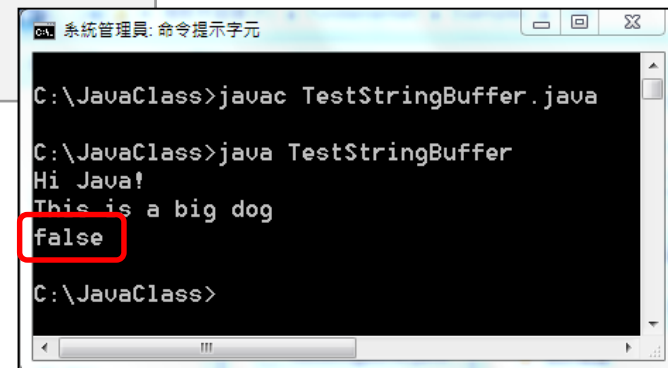


StringBuffer / StringBuilder常用方法

方法名稱	傳回值	說明
append(String str)	StringBuilder	將指定的字串加到此StringBuilder之後
append(StringBuffer sb)	StringBuilder	將指定的StringBuilder加到此StringBuilder之後
length()	int	傳回此StringBuilder的字元個數
capacity()	int	傳回此StringBuilder的容量
ensureCapacity(int minimumCapacity)	void	確保StringBuilder容量至少等於指定的最小值
charAt(int index)	char	傳回此StringBuilder中指定位置的字元值
indexOf(String str)	int	傳回此StringBuilder中,首次出現指定子字串的位置
lastIndexOf(String str)	int	傳回此StringBuilder中,最後出現指定子字串的位置
insert(int offset, String str)	StringBuilder	在StringBuilder指定位置之後插入指定字串
delete(int start, int end)	StringBuilder	將StringBuilder中,start到end位置中的字元移除
replace(int start, int end, String str)	StringBuilder	將StringBuilder中,start到end位置中的內容以指定字串取代
reverse()	StringBuilder	將StringBuilder字元順序顛倒
toString()	String	將StringBuilder變為一個內容不可變更的新字串傳回
substring(int start, int end)	String	將StringBuilder中指定位置變為一個內容不可變更的新字串傳回

StringBuffer 範例

```
01 public class TestStringBuffer {  
02     public static void main(String[] args) {  
03         StringBuffer sb1 = new StringBuffer("Hi");  
04         sb1.append(" Java!");  
05         System.out.println(sb1.toString( ));  
06  
07         StringBuffer sb2 = new StringBuffer("This is a dog");  
08         sb2.insert(10, "big ");  
09         System.out.println(sb2.toString( ));  
10  
11         StringBuffer sb3 = new StringBuffer("Hi Java!");  
12         System.out.println(sb1.equals(sb3));  
13     }  
14 }
```

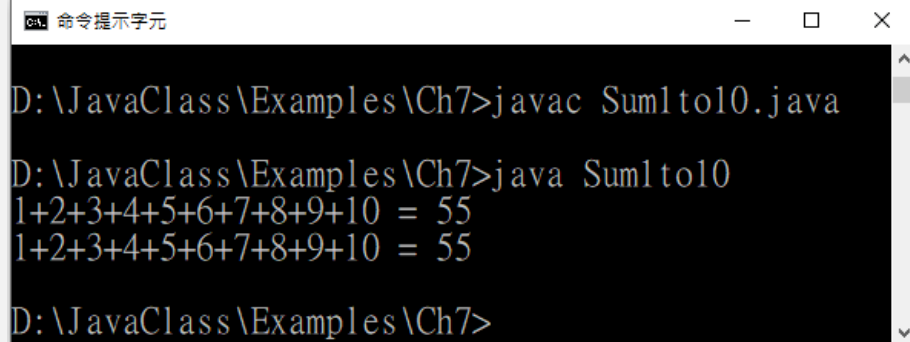


系統管理員: 命令提示字元

```
C:\JavaClass>javac TestStringBuffer.java  
  
C:\JavaClass>java TestStringBuffer  
Hi Java!  
This is a big dog  
false  
  
C:\JavaClass>
```

StringBuilder 範例

```
01 public class Sum1to10 {
02     public static void main(String[] args) {
03         String op = "1";
04         int sum1 = 1;
05         for (int i=2; i<=10; i++){
06             op += "+"+i;
07             sum1 += i;
08         }
09         System.out.println(op + " = " + sum1);
10
11         var sb = new StringBuilder("1");
12         var sum2 = 1;
13         for (int j=2; j<=10; j++){
14             sb.append("+"+j);
15             sum2 += j;
16         }
17         System.out.println(sb.append(" = ").append(sum1));
18     }
19 }
```



```
命令提示字元
D:\JavaClass\Examples\Ch7>javac Sum1to10.java
D:\JavaClass\Examples\Ch7>java Sum1to10
1+2+3+4+5+6+7+8+9+10 = 55
1+2+3+4+5+6+7+8+9+10 = 55
D:\JavaClass\Examples\Ch7>
```

課程大綱

- 1) String 類別
- 2) StringBuffer及StringBuilder類別
- 3) 包裹類別 **Wrapper Class**
- 4) ArrayList

包覆類別 Wrapper Class

- 基本資料型別 (primitive type) 有一個對應包裹類別 (Wrapper Class)
 - 將基本型態資料包裹成物件，以物件方式操作
 - 物件包含更多資訊及操作
 - 集合架構只接受物件型態
 - 裝箱 boxing
 - 基本型別轉成包裹物件
 - 拆箱 unboxing
 - 包裹物件中取出基本型別資料

基本資料型別 Primitive type	包裹類別 Wrapper Class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

包裹類別物件建構

■ 包裹類別物件建構

- ❑ Java 9 之後包裹類別建構子標示為過時(depreated)
- ❑ 類別方法 `Integer.valueOf(...)`

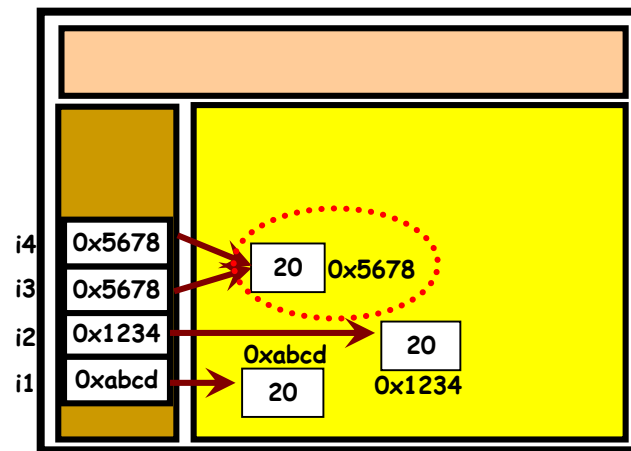
static valueOf (int i)	Returns an Integer instance representing the specified int value.
static valueOf (String s)	Returns an Integer object holding the value of the specified String.
static valueOf (String s, int radix)	Returns an Integer object holding the value extracted from the specified String when parsed with the radix given by the second argument.

The screenshot shows the Oracle Java SE 11 & JDK 11 API documentation for the `Integer` class. The page is titled "Integer (Java SE 11 & JDK 11)" and includes navigation tabs for OVERVIEW, MODULE, PACKAGE, CLASS, USE, TREE, DEPRECATED, INDEX, and HELP. The "CLASS" tab is selected. The page shows the "Constructor Detail" section for the `Integer` class. It lists two constructors, both marked as deprecated with a yellow warning icon. The first constructor is `public Integer(int value)`, and the second is `public Integer(String s) throws NumberFormatException`. Both constructors have a "Deprecated" note explaining that they are rarely appropriate and that `valueOf` or `parseInt` should be used instead. The page also includes a "Parameters" section for each constructor and a "Throws" section for the `Integer(String s)` constructor.

Integer 物件建構及常用方法

■ 建構 Integer 物件

```
Integer i1 = new Integer(20);  
Integer i2 = new Integer("20");  
Integer i3 = Integer.valueOf(20);  
Integer i4 = Integer.valueOf("20");
```



□ Integer Cache 整數快取

- Integer.valueOf() 取得的 Integer 物件，節省記憶體使用

- -128~127的Integer 物件會被暫存，供下次相同 Integer 重複使用

□ 包裹類別物件是不可變更的 (immutable)

- 物件建立後，其包裝的數值就不可改變

Integer 內容值比較

```
01 public class TestInteger {
02     public static void main(String[] args) {
03         Integer i1 = new Integer(20);
04         Integer i2 = new Integer("20");
05         Integer i3 = Integer.valueOf(20);
06         Integer i4 = Integer.valueOf("20");
07         System.out.println(i1 == i2);      → false
08         System.out.println(i3 == i4);      → true   // Integer Cache
09         System.out.println(i1.equals(i2));
10         System.out.println(i3.equals(i4)); → true   // 比較實際內容
11     }
12 }
```

```
01 public class TestInteger {
02     public static void main(String[] args) {
03         Integer i1 = new Integer(200);
04         Integer i2 = new Integer("200");
05         Integer i3 = Integer.valueOf(200);
06         Integer i4 = Integer.valueOf("200");
07         System.out.println(i1 == i2);      → false
08         System.out.println(i3 == i4);      → false  // 超過Integer Cache範圍
09         System.out.println(i1.equals(i2));
10         System.out.println(i3.equals(i4)); → true   // 比較實際內容
11     }
12 }
```

Integer 類別物件常用方法

■ 常用方法

方法名稱	傳回值	說明
<code>equals(Object obj)</code>	<code>boolean</code>	比較此 <code>Integer</code> 物件與傳入物件內容是否相等
<code>intValue()</code>	<code>int</code>	以 <code>int</code> 型別傳回 <code>Integer</code> 物件的值
<code>byteValue()</code>	<code>byte</code>	以 <code>byte</code> 型別傳回 <code>Integer</code> 物件的值
<code>longValue()</code>	<code>long</code>	以 <code>long</code> 型別傳回 <code>Integer</code> 物件的值

■ 工具方法

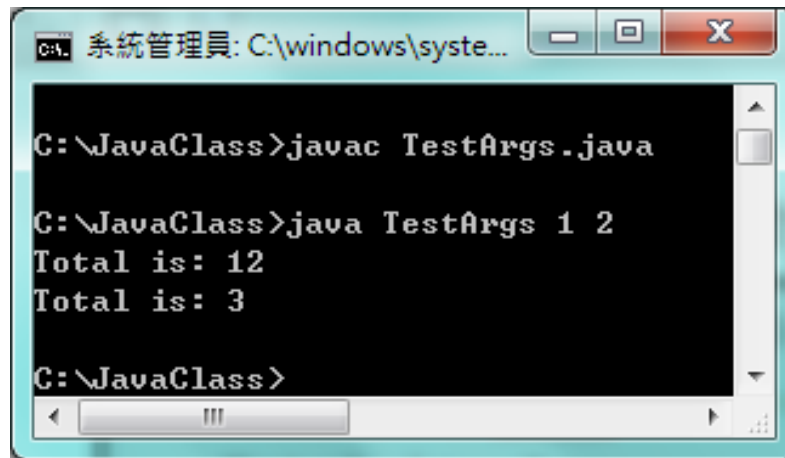
□ **Ex**：將表示基本型別的字串轉為基本型別

方法名稱	傳回值	說明
<code>parseInt(String s)</code>	<code>int</code>	將表示 <code>int</code> 的字串轉換成 <code>int</code> 型別
<code>getInteger(String nm)</code>	<code>Integer</code>	取得指定名稱對應的整數型態系統屬性值

```
int i = Integer.parseInt ("20");
```

數字字串轉型

```
1 public class TestArgs {  
2     public static void main(String[] args) {  
3         System.out.println("Total is: " + (args[0]+args[1]));  
4         int arg0 = Integer.parseInt(args[0]);  
5         int arg1 = Integer.parseInt(args[1]);  
6         System.out.println("Total is: " + (arg0+arg1));  
7     }  
8 }
```



The screenshot shows a Windows command prompt window titled "系統管理員: C:\windows\system...". The prompt is at "C:\JavaClass>". The user has entered the command "javac TestArgs.java" and pressed Enter. The prompt is now "C:\JavaClass>". The user has entered the command "java TestArgs 1 2" and pressed Enter. The output shows "Total is: 12" on the first line and "Total is: 3" on the second line. The prompt is now "C:\JavaClass>".

```
C:\JavaClass>javac TestArgs.java  
  
C:\JavaClass>java TestArgs 1 2  
Total is: 12  
Total is: 3  
  
C:\JavaClass>
```

課程大綱

- 1) String 類別
- 2) StringBuffer及StringBuilder類別
- 3) 包裹類別 Wrapper Class
- 4) **ArrayList**

java.util.ArrayList類別

- java.util.ArrayList 類別
 - Java提供之類別函式庫

	陣列 array	java.util.ArrayList 集合
容量大小	需預先給定 無法於執行時期動態增加元素	可動態增加集合元素,不需事先指定
資料型別	陣列內只能置入同型別資料	集合內可置入不同型別的資料
內容	可置入基本資料型別或物件	只能置入物件 基本型別需用包裝器類別
元素處理	基本讀取	提供進階的方法處理集合內元素

ArrayList常用方法

方法名稱	傳回值	說明
add(Object o)	boolean	從 ArrayList 的末端(end of this list)加入指定物件 element 。
add(int index, Object element)	void	沒有傳回值。將指定物件 element 加到 ArrayList 所指定的索引(index)位置，並傳回此集合物件。
set(int index, Object element)	Object	將 ArrayList 中指定索引位置的元素以指定物件 element 取代
addAll(Collection c)	boolean	從 ArrayList 的末端(end of this list)加入集合 c
size()	int	傳回此集合所有元素個數。
isEmpty()	boolean	判斷此集合是否為空。若此集合是空的，將傳回true；反之則會傳回false。
contains(Object o)	boolean	用以判斷指定物件o是否屬於該集合物件的成員之一，屬於傳回true，不屬於則傳回false。
indexOf(Object o)	int	回傳指定物件 o 在 ArrayList 上的索引位置。傳回值若為 -1 表示 ArrayList 集合物件元素中並沒有包含指定物件 o 。
get(int index)	Object	取得 ArrayList 中指定索引位置的元素
equals(Object o)	boolean	比較 ArrayList 物件與指定的物件o內容是否相同，指定物件o需亦為 ArrayList 集合，且各元素內容均相同時傳回true。
clear()	void	清除集合中的所有元素
remove(int index)	Object	移除 List 中指定索引位置的元素，並傳回被移除的元素。
toString()	String	將 ArrayList 集合中各元素以字串表示

ArrayList 範例

```
01 public class ArrayListExample {  
02     public static void main(String[] args) {  
03         java.util.ArrayList myList;  
04         myList = new java.util.ArrayList();  
05  
06         myList.add("One");  
07         myList.add("Second");  
08         myList.add("3rd");  
09         myList.add(new Integer(4));  
10         myList.add(new Float(5.0));  
11  
12         //列印ArrayList,利用toString()方法  
13         System.out.println(myList.toString());  
14  
15         //移除元素  
16         myList.remove(0);  
17         myList.remove(myList.size()-1);  
18         myList.remove("3rd");  
19         System.out.println(myList);  
20     }  
21 }  
22 }
```

宣告及建立 **ArrayList** 集合

利用 **add()** 將元素資料加到 **ArrayList** 集合中

ArrayList 的 **toString()**

利用 **remove()** 將指定元素由 **ArrayList** 中移除

```
C:\JavaClass>java ArrayListExample  
[One, Second, 3rd, 4, 5.0]  
[Second, 4]
```

裝箱、拆箱

■ 基本型別與包覆類別 wrapper class 轉換

- Java SE 5 之前資料型態不一樣，不可混用

~~Integer i1 = 10;~~

~~int i2 = new Integer(10);~~

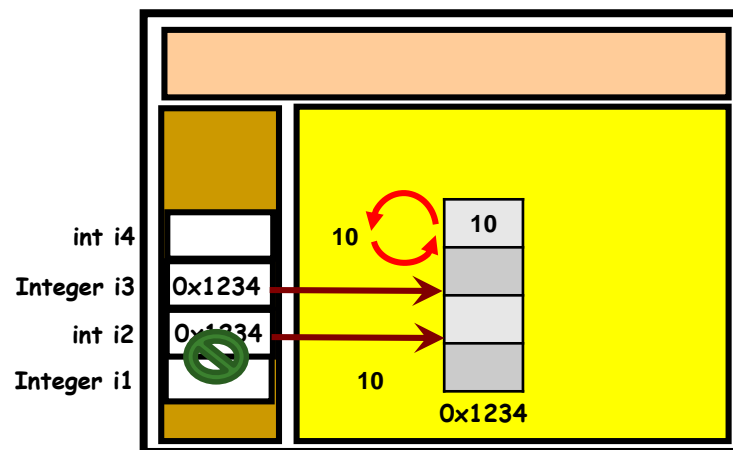
- 裝箱 boxing :

- 基本型別轉成包裹物件
 - 將int包裝為Integer物件
- Integer i3 = new Integer(10);**

- 拆箱 unboxing :

- 包裹物件中取出基本型別資料
- 將Integer物件的int值取出

int i4 = i3.intValue();



自動裝箱、自動拆箱

- J2SE 5.0 提供自動裝箱 (Auto-boxing) 及自動拆箱 (Auto-unboxing)功能

自動裝箱：

```
Integer i1 = 10;
```

```
int i = 10;  
Integer j = i;
```

```
Number number = 3.14f;
```

自動拆箱：

```
int i2= new Integer(10);
```

```
Integer i = new Integer(10);  
int j = i;
```

自動裝箱、拆箱

■ 運算時之自動裝箱與拆箱

Integer i = 10;

自動裝箱

System.out.println(i + 10);

自動拆箱再運算

System.out.println(i++);

自動拆箱再運算

Boolean boo = true;

自動裝箱

System.out.println(boo && false);

自動拆箱再運算