Module VIII
陣列與字串

1. 陣列(Array)的定義
2. 陣列的使用
3. 一維陣列
4. 多維陣列
5. 陣列length屬性
6. 陣列的指定敘述
7. String類別
8. String類別常用方法
9. 命令列引數
10. 不固定引數

陣列的定義

- 陣列(Array)的定義：

  - 陣列是由**一群相同資料型態的變數所組成**的一種資料結構
  - 比較一個變數與一個陣列
    - 一個變數：int x = 0;　　Pen myPen = **new** Pen();
    - 一個陣列：int x[] = **new** int[3];　Pen myPen[] = **new** Pen[3];
    - 程式進入點main方法可以接受零至多個字串當作參數(String args[])傳入，String args[]其實就是一個字串陣列
  - 陣列必須用**new**關鍵字來分配陣列的儲存空間，所以：
    - 陣列也是**一種Reference資料型態**
    - 陣列的指定運算，也是傳遞陣列的**記憶體位址**(memory address)
    - 注意：在new宣告的同時**必須指定長度且不可再更改**
  - 陣列宣告時，中括號可置於陣列參考的前面或後面
    - int x[]; 或　int[] x;
    - int x[], y[]; 或　int[] x, y;

陣列的使用

- 陣列(Array)的使用：

  - 取得陣列的長度：

    · 語法：陣列名稱.length (如myArray.length)

    · 注意1：一維陣列為元素**個數**

    · 注意2：二維陣列為**列數**

    · 注意3：length後面不可以加上小括弧，因為此處的length並不是方法，而且陣列的一個屬性(跟String類別的length()不同)

  - 取得陣列的元素

    · 可藉由索引值(index)存取陣列中儲存的資料值

    · 注意：**索引值從0開始**

  - 陣列使用new關鍵字分配好儲存空間後，**所有元素都會自動賦予初始值**

一維陣列

- 一維陣列

```
int x[] = new int[3];
x[0] = 10;
x[1] = 20;
x[2] = 30;
```

```
int x[] = {10, 20, 30}
```

```
String s[] = new String[3];
s[0] = “one”;
s[1] = “two”;
s[2] = “three”;
```

```
String s[] = {“one”, “two”, “three”}
```

```
Pen p[] = new Pen[3];
p[0] = new Pen();
p[1] = new Pen();
p[2] = new Pen();
```

```
Pen p[] = {new Pen(), new Pen(), new Pen()}
```

變數初始值

- 陣列宣告(沒給自訂初始值時)：

變數型態	值
byte	0
short	0
int	0
long	0L
float	0.0F
double	0.0(D)
boolean	False
char	‘\u0000’
物件	null

多維陣列

- 多維陣列(Java的多維陣列是**陣列的陣列**)

  - int xx[][] = new int [4][5];

- 多維陣列也可以如一維陣列般，一邊宣告一邊給初值

2x3陣列：

```
int xx[][] = {
        {1,2,3},
        {4,5,6}
    };
```

3x2陣列：

```
int xx[][] = {
        {1,2}
        {3,4}
        {5,6}
    };
```

- 非矩形(non-rectangular)的多維陣列

```
- int xx[][] = new int[4][];
    xx[0] = new int [3];
    xx[1] = new int [4];
    xx[2] = new int [5];
    xx[3] = new int [5];
```

一維陣列範例 – length屬性

- 以下示範計算**一維陣列**中所有整數的總合

```
public class TestOneDimArray {
    public static void main(String[] args) {
        int[] intArray = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        int sum = 0;
        for (int i = 0; i < intArray.length; i++)
            sum += intArray[i];
        System.out.println("總合為： " + sum);
    }
}
```

範例：TestOneDimArray.java

二維陣列範例 – length屬性

- 以下示範計算**二維陣列**中所有整數的總合

```
public class TestTwoDimArray {
    public static void main(String[] args) {
        int[][] intArray = {
                            {1,2,3,4,5},
                            {6,7,8,9,10}
                           };
        int sum = 0;
        for (int i = 0; i < intArray.length; i++) {
            for (int j = 0; j < intArray[i].length; j++)
                sum += intArray[i][j];
        }
        System.out.println("總合為： " + sum);
    }
}
```

範例：TestTwoDimArray.java

陣列的指定敘述 (1/2)

- 以下示範陣列的**指定運算(=)**

```
public class TestAssignArray {
    public static void main(String[] args) {
        int[] intArray1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        //將intArray1指定給intArray2
        int[] intArray2 = intArray1;

        //將intArray2的所有元素改成0
        for (int i = 0; i < intArray2.length; i++)
            intArray2[i] = 0;

        //列印原來intArray1所有元素，也都跟著變成0
        for (int i = 0; i < intArray1.length; i++)
            System.out.println(intArray1[i]);
} }
```

範例：TestAssignArray1.java

陣列的指定敘述 (2/2)

- 以下示範傳遞陣列的**記憶體位址(memory address)**

```
public class TestAssignArray2 {

  static void passReference(int[] intArray) {
      for(int i = 0; i < intArray.length; i++)
          intArray[i] = 0;
  }
  public static void main(String[] args) {
      int[] intArray = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
      passReference(intArray);
      for (int i = 0; i < intArray.length; i++)
          System.out.println(intArray[i]);
  }
}
```

範例：TestAssignArray2.java

課堂練習

- 請分別建立x, y, z三個3x3的int陣列，然後再將x和y陣列的加總存放到z陣列裡，再將結果顯示於螢幕上

- x和y陣列中的數字：
  - 請用亂數產生介於0 ~ 30之間的整數
  - 亂數之取得可參考 java.lang.Math的靜態方法 random()
  - public static double random()：傳回亂數值其範圍為0.0 ~ 1.0

進階陣列操作

- 陣列的排序：
  - static void **Arrays**.sort(欲排序的陣列名稱)
  - 可讓該陣列元素由小到大排序

- 陣列的搜尋：
  - static int **Arrays**.binarySearch(陣列名稱, 欲搜索的值)
  - 使用二分搜索法，搜索陣列內某個值的位置 (回傳int)
  - 執行搜索前，必須先將該陣列排序
  - 如果欲搜尋的值不在該陣列裡，則回傳負值

- 陣列的複製：【**JDK 6新增方法：copyOf()**】
  - static 新的陣列名稱 **Arrays**.copyOf(欲複製的陣列名稱，複製的長度)
  - 複製出的新陣列可以不用預先初始化(不用new)，直接回傳(複製出)一個新的陣列

範例：TestArray.java

String類別 (1/2)

- 在HelloWorld.java裡出現的「Hello World!」時，Java用String物件進行處理的
- String物件的幾個特性如下：

- 不可變的 (immutable)字串：
  - **String一旦宣告後，即不能在原所在記憶體位置改變字串內容**
  - **使用String類別任何方法時，傳回的字串都會放在新的記憶體空間**
- String s1 = new String(“Hello”);
  - 有自己的獨立記憶體空間
- String s1 = “Hello”;
  - 為加快程式執行，Java會把此類字串放在**字串池(String Pool)**裡
  - 程式裡若有多個變數，都使用相同的字串常數(如=“Hello”)，則均會使用相同的記憶體空間(即字串池String Pool)

String類別 (2/2)

- String的比較：
  - 比較字串內容時，應該使用String物件本身提供的一個方法，叫public boolean **equals**(Object anObject)的方法
  - 比較字串內容時，**並非**使用==，因為==在Java字串中，比較的是記憶體位址(指是否佔用相同的記憶體空間)而不是內容
- 比較如下：

```
String s1 = “Hello”;
String s2 = “Hello”;
System.out.println(s1 == s2);        // true
System.out.println(s1.equals(s2)); // true
```

```
String s1 = “Hello”;
String s2 = new String(“Hello”);
System.out.println(s1 == s2);        // false
System.out.println(s1.equals(s2));  // true
```

範例：TestString.java

String類別常用方法

- **public char charAt(int index)**：透過索引值取得字串內某一個字元(注意字元位置是從0開始算起)
- **public int length()**：傳回字串長度(注意空白也算進去)
- **public boolean isEmpty()**：如果字串長度為0，則回傳true，否則回傳false

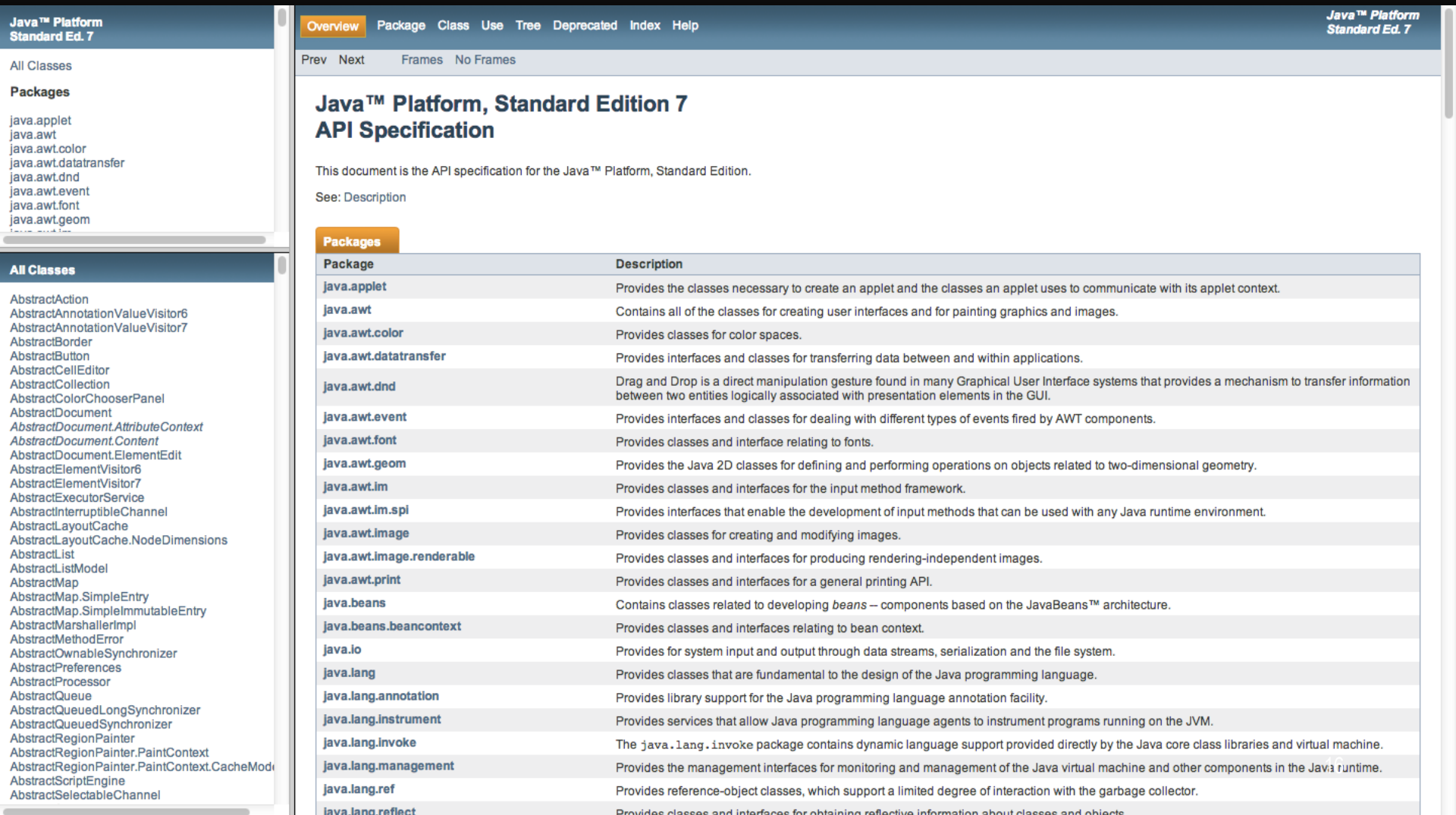
  (JDK 6.0新增方法)
- **public String substring(int beginIndex)**：擷取從開始索引值的字元至結尾字元的字串
- **public String substring(int beginIndex, int endIndex)**：擷取從開始索引值的字元至結束索引值的字元之間的字串 (注意結束索引值的字元不取)
- **public int compareTo(String anotherString)**：比較的方式是由左至右，依照字元ASCII值比較大小

  - 若回傳值=0，表示兩個字串相等
  - 若回傳值>0，表示左邊字串大於右邊字串
  - 若回傳值<0，表示左邊字串小於右邊字串

- 註：JDK6以前，測試是否為空字串(String s = “”)時，須使用if(s.length() == 0)；JDK6以後可使用if(s.isEmpty())來測試，方便又美觀！

Java類別函式庫與Java API

- 說明如何下載與使用



Java™ Platform Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutableEntry
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronizer
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchronizer
AbstractQueuedSynchronizer
AbstractRegionPainter
AbstractRegionPainter.PaintContext
AbstractRegionPainter.PaintContext.CacheMod
AbstractScriptEngine
AbstractSelectableChannel

Overview Package Class Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 7

Prev Next Frames No Frames

Java™ Platform, Standard Edition 7 API Specification

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

See: Description

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.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.invoke	The `java.lang.invoke` package contains dynamic language support provided directly by the Java core class libraries and virtual machine.
java.lang.management	Provides the management interfaces for monitoring and management of the Java virtual machine and other components in the Java runtime.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.

命令列參數

- 程式進入點main方法可以接受零至多個字串當作參數傳入，String[] args 其實就是一個字串陣列，如下所示：

  - java TestMainArray c a t

- 程式中各參數值分別以字串陣列型態 **args[0]**, **args[1]**, args[2]…存取

```
public class TestMainArray {
    public static void main(String[] args) {
        System.out.println("貓的英文是：" + args[0] + args[1] + args[2]);
    }
}
```

範例：TestMainArray.java

Varargs (不固定參數個數)

- Varargs(不固定參數個數 / 可變參數個數)：
  - 方法內可使用【...】點號，宣告【可變數目的參數】
  - 可變參數必須放在參數列的最後面
  - 方法中最多只能有1個不固參數的宣告，不能有2個或2個以上的不固定參數

- 例如：
  - void methodTest1(int x, String… args) {…}
  - 或 void methodTest2(String… args) {…}
    - 呼叫methodTest2方法時就可變化如：

      methodTest2(“xx”); 或 methodTest2(“xx”, “yy”);等

範例：AddInt.java

章節整理

- String類別注意事項與字串池(String Pool)的觀念建立
- String常用方法練習，特別注意索引值是由0開始
- 陣列為一組有相同資料型態的資料集合
- 陣列需用new關鍵字進行初始化，同時也代表著陣列就是一種物件
- 宣告陣列的同時，一定要指派長度，且不得再更改
- 我們利用陣列儲存相同資料型態，更方便的是搭配迴圈，可以輕鬆取得每個元素並加以利用
- 注意陣列的索引值是從0開始
- 熟悉並瞭解陣列元素排序與相關函式功能