2-19 頁頂：

當類別原始碼開始使用 package 進行分類時，就會具有四種管理上的意義：

- 原始碼檔案要放置在與 package 所定義名稱階層相同的資料夾階層。
- package 所定義名稱與 class 所定義名稱，會結合而成類別的**完全吻合名稱（Fully qualified name）**。

格式化: 字型: (英文) New

格式化: 字型: 粗體

2-22 頁中：

偷懶也是有個限度，如果你自己寫了一個 Arrays：

格式化: 字型: (英文) New

```
package cc.openhome;
public class Arrays {
...
}
```

2-34 頁底：

圖 2.36 中可看到，如果只指定-source 與-target 進行編譯，會出現警示訊息，這是因為編譯時預設的 Bootstrap 類別載入器（Class loader，沒有改變，第 15 章會說明 Bootstrap 類別載入器是什麼。簡單來說，系統預設的類別載入器仍參考至 1.7 的 rt.jar（也就是 Java SE 7 API 的 JAR 檔案），如果引用

3-3 頁頂：

- 浮點數

主要用來儲存小數數值，可分為 **float** 浮點數（佔 4 個位元組）與 double 浮點數（佔 8 個位元組），**double** 浮點數使用的記憶體空間比 float 浮點數來得多，可表示的精確度也比較大。

格式化: 字型: 粗體

格式化: 字型: 粗體

4-40 頁頂：

字串編碼

先前字串都用英文示範，OK！相信現在或未來的日子，在 Java 中你不會只處理英文，所以你要瞭解 Java 中如何處理中文！在正式介紹 Java 如何處理字串編碼前，容我問一句：你寫的.java 原始碼檔案是什麼編碼？

5-22 頁底：

va 程式設計領域，早就有許多良好命名慣例，沒有遵守慣例並不是錯，但會　成溝通與維護的麻煩。以類別命名實例來說，首字是大寫，以 static 使用慣例來說，是透過類別名稱與.運算子來存取。在大家都遵守命名慣例的情況下，看到首字大寫就知道它是類別，透過類別名稱與.運算子來存取，就會知道

刪除: 剛

6-7 頁頂：

來看實際的例子，以下的程式碼片段，相信你現在可以沒有問題地看懂，而且知道可以通過編譯：

刪除: 1

```
SwordsMan swordsMan = new SwordsMan();
Magician magician = new Magician();
```

6-15 頁底：

Game4_Role.java

```
package cc.openhome;

public abstract class Role {
    ...略
    public abstract void fight();
}
```

格式化: 字型: 粗體

6-19 頁底：

如果在 SwordsMan 子類別中重新定義 toString()的內容時，可以執行 Role 中的 toString()方法取得字串結果，再串接"劍士"字樣，不就是你想要的描述了嗎？在 Java 中，如果想取得父類別中的方法定義，可以於呼叫方法前，加上 **super** 關鍵字。例如：

刪除: 鍵

6-30 頁頂（「運」的字型不對）：

這個程式片段示範了 equals()實作的基本概念，相關說明都以註解方式呈現了，這邊也看到了 **instanceof** 運算子，它可以用來判斷物件是否由某個類別建構，左運算元是物件，右運算元是類別，在使用 instanceof 時，編譯器還會

格式化: 預設段落字型, 字型: 非粗體

7-12 頁底：

當然需求是無止盡的，原有程式架構也許確實可滿足一些需求，但有些需求也可能超過了原有架構預留之彈性，一開始要如何設計才會有彈性，是必須

7-26 頁中：

提示 »» 要瞭解為什麼，必須涉及一些底層機制。區域變數的生命週期往往比 牛短，像是方法呼叫後傳回物件，區域變數生命週期就結束了，此時再透過 牛嘗試存取區域變數會發生錯誤，Java 的作法是採用傳值，以上例而言，實際上會在匿名內部類別的實例中，建立新的變數參考原本的物件：

```
int ai[] = {10, 20};
Object obj = new Object(ai) {
    public String toString() {
        return (new StringBuilder())
                    .append("example:")
                    .append(x[0]).toString();
    }
    final int x[];
    {
        x = ai;
    }
};
```

刪除:

格式化: 縮排: 第一行: 12.29

8-10 頁底（最後一個 { 括號的字型是 Courier New）：

```
        return builder.toString();
    }
}
```

格式化: 命令列

雖然還沒正式介紹到 Java 中如何存取檔案，不過 4.1.2 曾經談過，Scanner

8-15 頁中：

要善用堆疊追蹤，前題是程式碼中不可有私吞例外的行為，例如在捕　列外後什麼都不作：

格式化: 字型: 粗體

9-10 頁頂：

程式中使用 Set 收集了 Student 物件，其中故意重複加入了相同的　生資料，然而在執行結果中看到，Set 並沒有將重複的學生資料排除：

```
[(Monica, B835032), (Justin, B835031), (Justin, B835031)]
```

格式化: 命令列

9-12 頁頂：

```
    @Override
    public int hashCode() {
        int hash = 5;
        hash = 13 * hash + (this.name != null ? this.name.hashCode() : 0);
        hash = 13 * hash + (this.number != null ? this.number.hashCode() : 0);
        return hash;
    }
```

格式化: 字型: 粗體

9-18 頁中：

實際上增強式 **for** 迴圈是編譯器蜜糖，當運用在 Iterable 物件時，會展開為：

```
private static void forEach(Iterable iterable) {
    Object o;
    for(Iterator i$ = iterable.iterator();
                i$.hasNext();
                    System.out.println(o)) {
        o = i$.next();
    }
}
```

格式化: 縮排: 字元

11-58 頁頂：

使用 Collections.synchronizedList()並非就不用如此操作，它傳回的實例保證是 List 操作時的執行緒安全，而非保證傳回的 Iterator 操作時的執行緒安全，所以使用迭代器操作時，仍得類似以下實作：

刪除: 下的

12-27 頁頂：

如果在貪婪量詞表示法後加上+，將會成為獨吐量詞（Possessive quantifier），比對器看到獨吐量詞時，會先將剩餘文字全部吃掉，然後看看獨吐量詞部份是否可符合吃下的文字，如果符合就不會再吐出來了。

12-41 頁底：

如果想要複製來源 Path 的檔案或目錄至目的地 Path，可以使用 **Files.copy()** 方法，這個方法的第三個選項可以指定 **CopyOption** 介面的實作物件，CopyOption 實作類別有 Enum 型態的 **StandardCopyOption** 與 **LinkOption**。例如指定

刪除: 以

13-16 頁底：

以選單項目按下時的事件處理為例，必須實作 **jav** 介面。例如：

```
// 開啟舊檔選單項目的事件處理
menuOpen.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
```

13-17 頁頂：

```
        openFile();
    }
  }
);
```

格式化: 縮排: 字元

13-24 頁底：

```
public class JNotePad extends JFrame {
    ...略
    private JFileChooser fileChooser;
    ...略
    private void initComponents() {
        ...略
        fileChooser = new JFileChooser();
    }
```

格式化: 縮排 字元

格式化: 縮排 字元

格式化: 縮排

14-50 頁底：

如果在查詢之後，想要離線進行操作，則可以使用 CachedRowSet 或其子介面實作物件，查詢資料之後可以直接使用 **close()** 關閉連線，若在相關更新操作之後，想再與資料來源進行同步，則可以呼叫 **acceptChanges()** 方法，例如：

刪除: 與

15-11 頁中：

這個程式片段會拋出 java.lang.ClassCastException，告訴你不可以將 Object[] 當作 String[] 來使用，為何？回顧一下程式片段中粗體字部份，你建立的物件確實是 Object[]，而不是 String[]，可以如下解決：

刪除: 生

16-21 頁中：

```
public enum Priority2 {
    MAX(10) {
        public String toString() {
            return String.format("(%2d) - 最大權限", value);
        }
    },
    NORM(5) {
        public String toString() {
            return String.format("(%2d) - 普通權限", value);
        }
    },

    MIN(1) {
        public String toString() {
            return String.format("(%2d) - 最小權限", value);
        }
    };
```

16-26 頁頂（n 的字型不對）：

自訂標註型態

每個標註都會有個標註型態（Annotation type），所有標註型態其實都是 **java.lang.annotation.Annotation** 子介面，@Override 的標註型態為 **java.lang.Override**，@Deprecated 的標註型態為 **java.lang.Deprecated** 等，之前

16-26 頁中：