Java 程式設計概念整理

1. Java 物件導向程式設計基礎

三大特性

- **封裝 (Encapsulation)**
  - 將資料和對資料的操作方法包裝在一起
  - 透過存取修飾符控制存取權限
  - 確保資料安全性和隱私性

- **繼承 (Inheritance)**
  - 允許新類別建立在既有類別之上
  - 促進代碼重用
  - 建立類別層次結構

- **多型 (Polymorphism)**
  - 方法重載 (Overloading)：同名方法，不同參數
  - 方法覆寫 (Overriding)：子類別重新實作父類別方法
  - 執行期多型：透過父類別參考指向子類別物件

2. 抽象類別與介面比較

抽象類別 (Abstract Class)

- 可以包含具體方法和抽象方法
- 只能單一繼承
- 可以有建構子
- 可以有成員變數
- 可以有 static 方法

介面 (Interface)

- 只能包含抽象方法（Java 8+ 可有預設方法）
- 可以多重實作
- 不能有建構子

- 只能有常數（public static final）
- 方法預設為 public abstract

3. 重要關鍵字與概念

this 關鍵字

- 參考當前物件實例
- 用於區分局部變數和實例變數
- 在建構子中呼叫其他建構子

super 關鍵字

- 參考父類別
- 呼叫父類別建構子
- 存取父類別方法和欄位

static 關鍵字

- 屬於類別而非實例
- 不需要創建物件即可使用
- 常用於工具類別和常數定義

執行緒 (Thread)

- 程式執行的最小單位
- 可通過繼承 Thread 類別或實作 Runnable 介面創建
- 提供並行處理能力

4. Web 技術

JSP vs HTML

- JSP 可包含 Java 程式碼
- JSP 會編譯成 Servlet
- JSP 可動態產生內容
- HTML 是靜態網頁

JSP 四大範圍變數

1. page：當前頁面
2. request：單次請求

3. session：使用者會話
4. application：整個應用程式

JSP vs Servlet

- JSP 專注於表現層
- Servlet 專注於處理邏輯
- JSP 最終會轉換成 Servlet

5. 前端技術

EL 與 JSTL

- **EL (Expression Language)**
  - 簡化 JSP 頁面中的 Java 程式碼
  - 存取 JavaBean 屬性
- **JSTL (JSP Standard Tag Library)**
  - 提供常用標籤庫
  - 簡化頁面邏輯處理

Form vs Ajax

- **傳統 Form 提交**
  - 整頁重新載入
  - 同步處理
- **Ajax 提交**
  - 局部更新
  - 非同步處理
  - 提供更好的用戶體驗

6. 集合框架

List vs Set

- **List**
  - 有序集合
  - 允許重複元素

    - 支援索引存取
    - 常用實現：ArrayList, LinkedList

- **Set**
    - 不允許重複元素
    - 不保證順序（除 LinkedHashSet）
    - 常用實現：HashSet, TreeSet

Map

- 鍵值對映射
- 不允許重複鍵
- 常用實現：HashMap, TreeMap
- 支援各種資料類型作為鍵和值

7. 日誌層級

- FATAL：嚴重錯誤
- ERROR：一般錯誤
- WARN：警告訊息
- INFO：一般資訊
- DEBUG：除錯資訊
- TRACE：追蹤資訊

8. MVC 架構

組成

- **Model**: 資料和業務邏輯
- **View**: 使用者介面
- **Controller**: 處理使用者輸入

優點

- 關注點分離
- 代碼重用性高
- 便於團隊協作
- 維護性好

缺點

- 學習曲線較陡
- 小型專案可能過於複雜
- 需要仔細規劃

9. JDBC 資料庫連接步驟

```
// 1. 註冊驅動程式
Class.forName("com.mysql.jdbc.Driver");

// 2. 建立連接
Connection conn = DriverManager.getConnection(url, username, password);

// 3. 建立陳述式
Statement stmt = conn.createStatement();

// 4. 執行查詢
ResultSet rs = stmt.executeQuery(sql);

// 5. 關閉連接
conn.close();
```

10. 九九乘法表函式優化

```
private void multiplyNum(int n) {
    if (n <= 0) {
        System.out.println("請輸入正整數");
        return;
    }

    System.out.printf("%d 的乘法表：%n", n);
    for (int i = 1; i <= 9; i++) {
        System.out.printf("%d × %d = %d%n", n, i, n * i);
    }
}
```

改進重點：

- 加入輸入驗證
- 使用 printf 格式化輸出
- 增加可讀性
- 統一輸出格式